Semi-Supervised Fine-Tuning of Vision Foundation Models with Content-Style Decomposition

Mariia Drozdova
Department of Computer Science
University of Geneva
Switzerland
mariia.drozdova@unige.ch

Vitaliy Kinakh
Department of Computer Science
University of Geneva
Switzerland
vitaliy.kinakh@unige.ch

Yury Belousov
Department of Computer Science
University of Geneva
Switzerland
yury.belousov@unige.ch

Erica Lastufka
Department of Computer Science
University of Geneva
Switzerland
erica.lastufka@unige.ch

Slava Voloshynovskiy[*]
Department of Computer Science
University of Geneva
Switzerland
svolos@unige.ch

Abstract

In this paper, we present a semi-supervised fine-tuning approach designed to improve the performance of pre-trained foundation models on downstream tasks with limited labeled data. By leveraging content-style decomposition within an information-theoretic framework, our method enhances the latent representations of pre-trained vision foundation models, aligning them more effectively with specific task objectives and addressing the problem of distribution shift. We evaluate our approach on multiple datasets, including MNIST, its augmented variations (with yellow and white stripes), CIFAR-10, SVHN, and GalaxyMNIST. The experiments show improvements over supervised finetuning baseline of pre-trained models, particularly in low-labeled data regimes, across both frozen and trainable backbones for the majority of the tested datasets.

1 Introduction

In recent years, vision foundation models have gained significant popularity across a wide range of applications. These models, which are typically pre-trained on vast publicly available datasets, have demonstrated their utility in numerous domains, including multimedia applications and scientific fields such as astronomy[5], biology[12], medical imaging[17], and remote sensing[7]. The common practice in developing these vision foundation models involves pre-training them in a self-supervised (DINOv2 [9]) or weakly supervised (CLIP [3, 10], RADIOv2[11]) manner. These training methodologies do not target any specific downstream tasks; instead, they aim to find a representation that can be adapted to a variety of tasks. This characteristic supports the versatility of foundation models

[*]Slava Voloshynovskiy (svolos@unige.ch) is the corresponding author.

but introduces challenges when applying them to downstream tasks (DSTs), particularly when there is a distribution mismatch between the pre-training data and the DST data.

1.1 Challenges in Fine-Tuning Foundation Models for Downstream Tasks

One of the primary challenges in utilizing vision foundation models for specific DSTs is the issue of **distribution shift**. The distribution of data $p_{\mathbf{x}}^{\mathrm{tr}}(\mathbf{x})$ used during the pre-training phase of vision foundation models does not necessarily match the distribution of the data $p_{\mathbf{x}}(\mathbf{x})$ associated with a particular DST. This mismatch can lead to suboptimal performance when the foundation model is directly applied to DSTs without appropriate adjustments. Our proposed semi-supervised learning approach aims not only to adapt the foundation model to the task objective but also to address this distribution shift by better aligning the learned representations with DST data distributions. Consequently, fine-tuning becomes essential to adapt the model's learned representations to align with the characteristics of the DST data.

Furthermore, the pre-training of vision foundation models often aligns with the **principles of the information bottleneck theory**[13], which suggests that a model should retain only the information relevant to a particular task while discarding irrelevant details. However, self- or weakly-supervised learning, by its nature, might either retain too much or too little information relevant to a specific DST since DST is unknown at the pre-training stage. When faced with distribution mismatches between DST and pre-training, the model may struggle to retain the appropriate information for downstream tasks. In cases where essential information is overly suppressed, fine-tuning might not fully recover the necessary details. Conversely, if too much information is retained, targeted fine-tuning can help prune and specialize the model's representations for the DST.

1.2 Fine-Tuning Challenges in Scientific Applications

In this study, we focus on fine-tuning vision foundation models for simple datasets, simulating scenarios often found in scientific applications where labeled data is limited. In these situations, we typically encounter a moderate number of unlabeled examples, ranging from 10,000 to 70,000, while the number of labeled examples may vary from just 1-2 to all available samples per class. Our approach demonstrates how leveraging both labeled and unlabeled data can help address these issues by improving the representations of foundation models in downstream tasks, even when distribution shift is present.

2 Information-Theoretic Framework for Fine-Tuning of VFMs

As an alternative to conventional fine-tuning approaches, we propose leveraging methods grounded in semi-supervised learning, a well-established field that has demonstrated its effectiveness in various contexts. Specifically, we extend an information-theoretic framework to address the fine-tuning challenge [15]. This framework enables the model to adapt to the specifics of the downstream task by optimizing the retention of task-relevant information while discarding irrelevant details.

Rather than applying this approach directly to image data, we focus on the latent representations of vision foundation models. By improving these representations, our method aims to overcome the distribution shift that occurs when applying models to downstream datasets by leveraging available unlabeled data and a limited number of labeled examples.

We demonstrate the applicability of this approach by evaluating its performance across datasets that are simpler and distinct from large-scale, high-resolution natural image distributions, such as MNIST dataset variations[6], SVHN[8], CIFAR-10[4], and GalaxyMNIST[16]. More details about these datasets can be found in Section 2.2. Our results indicate that while certain vision foundation models struggle with these simpler datasets—particularly in low-labeled sample cases with purely supervised approaches—others are better suited to these specialized applications. This highlights the importance of model selection and fine-tuning strategies. This work showcases the advantages of our proposed approach in enhancing the adaptability and effectiveness of foundation models for handling diverse data.

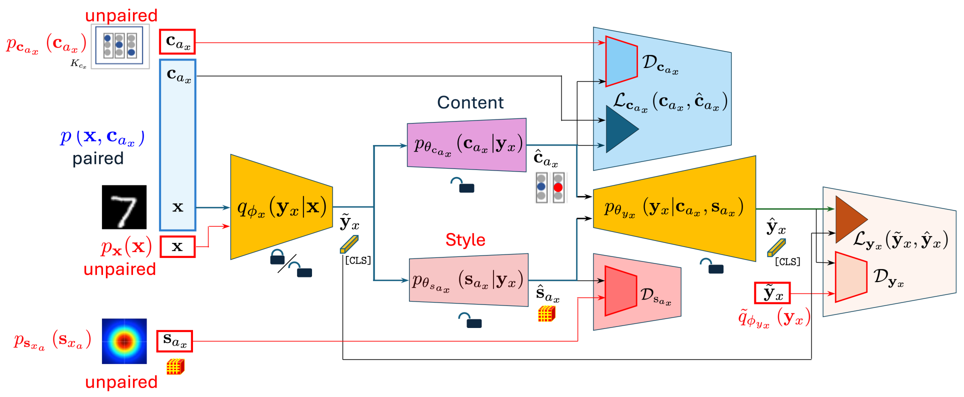
unpaired
$p_{\mathbf{c}_{a_x}}(\mathbf{c}_{a_x})$
K_{c_x}
$\mathbf{c}_{a_x}$
$\mathbf{c}_{a_x}$
$p(\mathbf{x}, \mathbf{c}_{a_x})$
paired
$\mathbf{x}$
$\mathbf{x}$
$p_{\mathbf{x}}(\mathbf{x})$
unpaired
$q_{\phi_x}(\mathbf{y}_x|\mathbf{x})$
$\tilde{\mathbf{y}}_x$
[CLS]
Content
$p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x}|\mathbf{y}_x)$
$\hat{\mathbf{c}}_{a_x}$
Style
$p_{\theta_{s_{a_x}}}(\mathbf{s}_{a_x}|\mathbf{y}_x)$
$\hat{\mathbf{s}}_{a_x}$
$\mathcal{D}_{\mathbf{c}_{a_x}}$
$\mathcal{L}_{\mathbf{c}_{a_x}}(\mathbf{c}_{a_x}, \hat{\mathbf{c}}_{a_x})$
$p_{\theta_{y_x}}(\mathbf{y}_x|\mathbf{c}_{a_x}, \mathbf{s}_{a_x})$
$\mathcal{D}_{\mathbf{s}_{a_x}}$
$\hat{\mathbf{y}}_x$
[CLS]
$\tilde{\mathbf{y}}_x$
$\tilde{q}_{\phi_{y_x}}(\mathbf{y}_x)$
$\mathcal{L}_{\mathbf{y}_x}(\tilde{\mathbf{y}}_x, \hat{\mathbf{y}}_x)$
$\mathcal{D}_{\mathbf{y}_x}$
$p_{\mathbf{s}_{x_a}}(\mathbf{s}_{x_a})$
$\mathbf{s}_{a_x}$
unpaired

Figure 1: Architecture of the proposed semi-supervised fine-tuning scheme. The vision foundation model generates a representation in the form of a [CLS] token $\tilde{\mathbf{y}}_x$, which is decomposed into content attribute label $\hat{\mathbf{c}}_{a_x}$ and generic style $\hat{\mathbf{s}}_{a_x}$. These are then used for targeted reconstruction of the [CLS] token $\hat{\mathbf{y}}_x$.

2.1 Proposed Semi-Supervised Fine-Tuning Scheme

The proposed semi-supervised fine-tuning method for the vision foundation model leverages the principle of content-style decomposition. The foundation model, denoted as an encoder $q_{\phi_x}(\mathbf{y}_x|\mathbf{x})$, pre-trained on a large dataset with the distribution $p^{\mathrm{tr}}_{\mathbf{x}}(\mathbf{x})$, maps an input $\mathbf{x}$ to a latent representation $\tilde{\mathbf{y}}_x$. The latent representation can be in the form of patch tokens and a [CLS] token for transformer-based models, aggregation or summary tokens, or just a vector output of CNN-based foundation models. Since in this work DST task is a classification, we proceed with the [CLS] token representation. One might use patch tokens or CNN latent tensors for more complex tasks such as segmentation or depth estimation. Therefore, the latent representation $\tilde{\mathbf{y}}_x$ in the form of a [CLS] token is then decomposed into content $\mathbf{c}_{a_x}$ and style $\mathbf{s}_{a_x}$ representations. As this work focuses on classification, the content $\mathbf{c}_{a_x}$ is represented as a one-hot encoding of the class. The content and style are then subsequently combined and passed through a decoder to reconstruct the [CLS] token $\hat{\mathbf{y}}_x$.

Each element of the model, namely content representation, style representation, and CLS token reconstruction, is associated with specific blocks and regularizers. We employ the concept of **adversarial mutual information decomposition** [15], which leads to the formulation where mutual information is decomposed into: (a) a conditional cross-entropy term, which serves as a metric of similarity between content, style, and the CLS token for the paired data, and (b) a discriminator term, which represents the Kullback-Leibler divergence for unpaired data.

In the proposed semi-supervised model, the vision foundation model that generates the initial representations can be kept either **frozen** or **trainable**, depending on the setup.

- In the **frozen setup**, only the content prediction block $p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x}|\mathbf{y}_x)$, style prediction block $p_{\theta_{s_{a_x}}}(\mathbf{s}_{a_x}|\mathbf{y}_x)$, and CLS token reconstruction $p_{\theta_{y_x}}(\mathbf{y}_x|\mathbf{c}_{a_x}, \mathbf{s}_{a_x})$ components are trained, while the foundation model remains unchanged.
- In the **trainable setup**, both the foundation model $q_{\phi_x}(\mathbf{y}_x|\mathbf{x})$ and the components $p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x}|\mathbf{y}_x)$, $p_{\theta_{s_{a_x}}}(\mathbf{s}_{a_x}|\mathbf{y}_x)$ and $p_{\theta_{y_x}}(\mathbf{y}_x|\mathbf{c}_{a_x}, \mathbf{s}_{a_x})$ are trained, but with different learning rates: the backbone learning rate is set to be 100 times smaller than that of the classifier and other components to avoid excessive disruption of the foundation model's pre-trained features.

The core idea of the fine-tuning process in this semi-supervised model revolves around the availability of a certain number of paired examples. Let $\mathcal{X}$ represent the input data, and $\mathcal{C}_A$ denote the content attributes. We define $N_{\mathcal{X},\mathcal{C}}$ as the total number of paired examples $\{\mathbf{x}_i, \mathbf{c}_{a_x i}\}_{i=1}^{N_{\mathcal{X},\mathcal{C}}}$, which is assumed to be limited. Additionally, let $N_{\mathcal{X}}$ denote the number of unpaired samples $\mathbf{x}$, derived from the downstream task distribution $p_{\mathbf{x}}(\mathbf{x})$. Here, the downstream task is focused on content attribute prediction.

The main mechanism of our system is as follows:

- The content predictor, or content attribute estimator, is trained to predict the correct content attribute $\mathbf{c}_{a_x}$ for paired data $\{\mathbf{x}_i, \mathbf{c}_{a_{x}i}\}_{i=1}^{N_{\mathcal{X}},c}$. This attribute is represented as a one-hot encoded class label.
- For unpaired data $\{\mathbf{x}_i\}_{i=1}^{N_{\mathcal{X}}}$, the system ensures that the predicted content attribute belongs to one of the predefined classes. The distribution of these attributes is modeled as a categorical distribution, regulated by the discriminator and its corresponding categorical representation of content attributes.

In this model, the style distribution is not explicitly constrained and is assumed to follow a Gaussian distribution. This approach allows the model to adapt flexibly to different styles while maintaining robustness in content attribute prediction.

As shown in Figure 1, for paired data $\{\mathbf{x}_i, \mathbf{c}_{a_{x}i}\}_{i=1}^{N_{\mathcal{X}},c}$, we apply a cross-entropy loss $\mathcal{L}_{c_{a_x}}(\tilde{\mathbf{c}}_{a_x}, \hat{\mathbf{c}}_{a_x})$, which represents the supervised learning step. In contrast, for unpaired data $\{\mathbf{x}_i\}_{i=1}^{N_{\mathcal{X}}}$ the model performs content-style disentanglement and reconstruction, regulated by discriminators operating on different spaces. These discriminators, $\mathcal{D}_{\mathbf{c}_{a_x}}$ for content, $\mathcal{D}_{\mathbf{s}_{a_x}}$ for style, and $\mathcal{D}_{\mathbf{y}_{a_x}}$ for the reconstructed `CLS` token, enforce Kullback–Leibler divergences on the corresponding spaces. Additionally, a reconstruction loss for the `CLS` token, denoted as $\mathcal{L}_{y_{a_x}}(\tilde{\mathbf{y}}_{a_x}, \hat{\mathbf{y}}_{a_x})$, is calculated using cosine similarity between the true `CLS` token $\tilde{\mathbf{y}}_{a_x}$ and the predicted one $\hat{\mathbf{y}}_{a_x}$. These losses are integrated into an unsupervised learning step. Further architectural details and mathematical definitions can be found in Appendices A, B.

This formulation follows the approach proposed in [15], where detailed ablation studies were conducted. Based on these results, we selected the best setup for our work: learnable priors and enabling all proposed losses.

In the trainable setup, the backbone is updated only during the supervised step and not during the unsupervised steps. This is because we reconstruct the `CLS` token, and allowing gradients through both components and the backbone would lead to degenerate results. The learning process involves alternating between supervised and unsupervised updates: for the first 20 iterations, we use only supervised losses. Afterward, we introduce unsupervised losses, performing 1 unsupervised update for every 2 supervised updates. This configuration was selected after a brief hyperparameter search, where it showed optimal balance between leveraging labeled data and making effective use of the unlabeled data.

This semi-supervised fine-tuning scheme provides a robust framework for leveraging the expressive power of pre-trained vision foundation models while efficiently utilizing limited labeled data and a larger pool of unlabeled data to adapt to specific downstream tasks.

2.2 Datasets

We conducted experiments on six datasets, including variations of MNIST [6] and other widely-used image classification datasets. Samples from each dataset are visualized in Figure 2. For all models, the images were resized to 224x224 using bilinear interpolation with the Python Image Library (PIL) [14].

MNIST: The MNIST dataset consists of 60,000 training images and 10,000 test images of handwritten digits, each of size 28x28. These grayscale images were expanded into three-channel images by treating them as L-mode images and converting them to RGB using PIL.

MNIST with Yellow Stripes: This dataset is a modification of the original MNIST, where yellow stripes were added across the top of each digit. These stripes can be seen as a type of augmentation technique or even as a form of adversarial attack, designed to make the images deviate from the original training set. The yellow stripes partially obscure the digits but remain easily recognizable.

MNIST with White Stripes: This MNIST variation includes white stripes across the top of each image, making the digits harder to recognize as the stripes overlap with the digits, often blending in due to the same color. This modification presents a greater challenge than the yellow stripes.

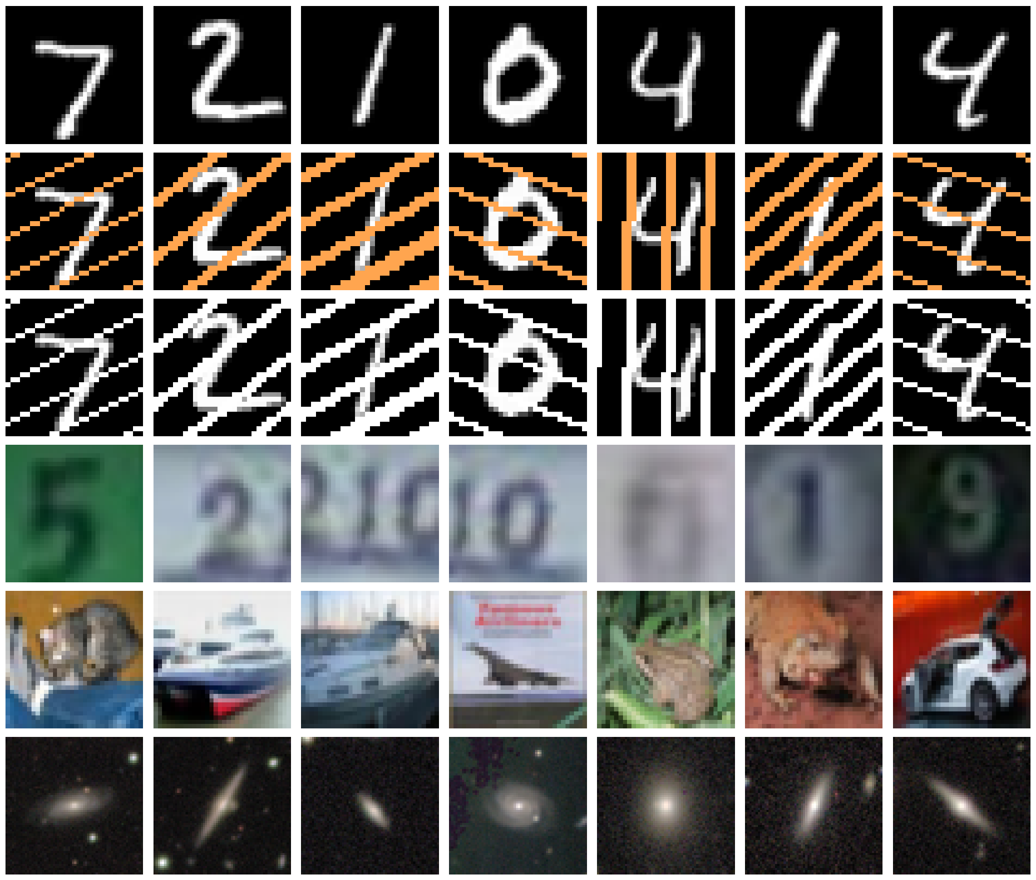

Figure 2: Samples per row from each dataset. From top to bottom: MNIST, MNIST with yellow stripes, MNIST with white stripes, SVHN, CIFAR-10, GalaxyMNIST.

SVHN (Street View House Numbers): The SVHN dataset [8] consists of 73,257 training and 26,032 testing images of real-world house numbers captured from Google Street View. Each 32x32 RGB image contains one or more digits, but the task is to classify the digit in the center, adding complexity compared to simpler datasets like MNIST.

CIFAR-10: The CIFAR-10 dataset [4] contains 60,000 32x32 color images, with 50,000 for training and 10,000 for testing, across 10 different classes. Each class represents a distinct object category such as airplanes, cars, or animals.

Galaxy MNIST: GalaxyMNIST [16] contains 10,000 galaxy images (64x64), categorized into four morphological classes: smooth and round, smooth and cigar-shaped, edge-on disk, and unbarred spiral. Derived from Galaxy Zoo DECaLS, it offers a balanced 80/20 train/test split. We use GalaxyMNIST due to its complexity and domain shift, as its astronomical images differ significantly from natural images, allowing us to assess model adaptation in specialized domains.

These datasets introduce varying levels of complexity and domain shifts compared to the large-scale natural image datasets typically used to pre-train foundation models. While MNIST and CIFAR-10 are relatively simple, the MNIST variations introduce perturbations that challenge the vision foundation models' generalization capabilities. Furthermore, datasets like SVHN and GalaxyMNIST present distinct challenges that highlight the issue of distribution shift. SVHN contains real-world digit images, often including surrounding digits and complex backgrounds, complicating the model's ability to isolate relevant information. GalaxyMNIST, meanwhile, introduces astronomical patterns, which considerably differ from the natural image distributions on which foundation models were trained. This divergence leads to suboptimal performance, making targeted fine-tuning essential, especially when labeled data is limited and unlabeled data is insufficient to train a new foundation model from scratch.

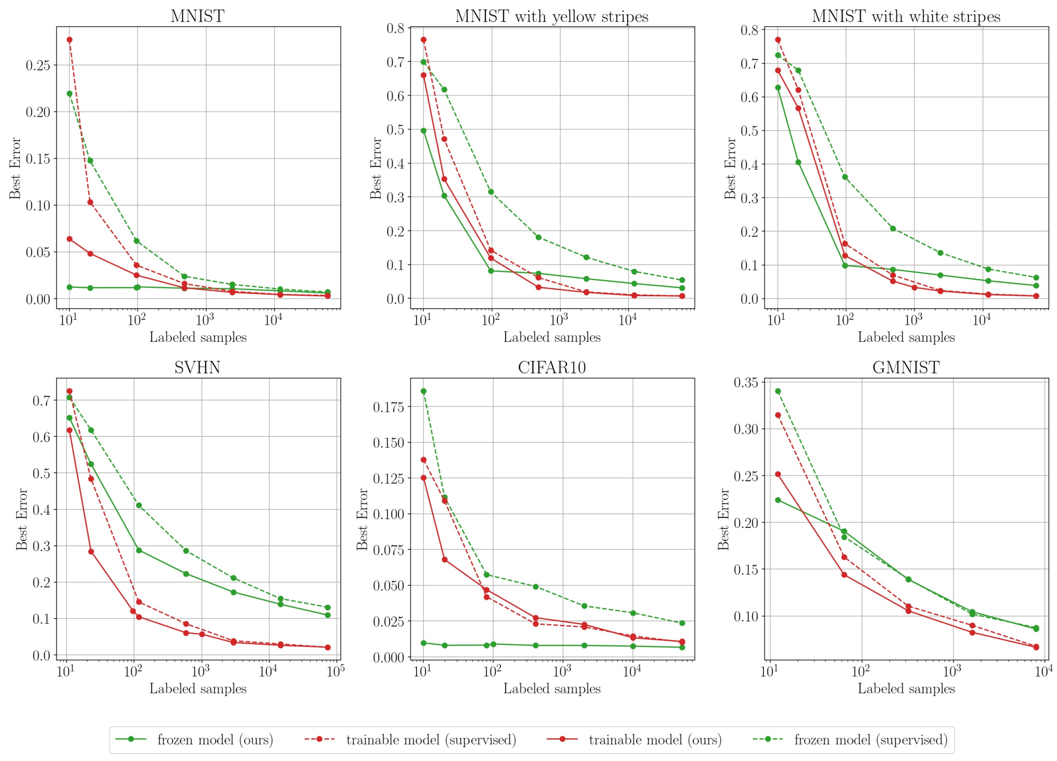

Figure 3: RADIOv2 model results. The y-axis represents the best error rate (1 - accuracy), and the x-axis represents the number of labeled samples. For each dataset, the classifier is trained with both supervised learning and our proposed method for frozen and trainable backbone.

3 Results and Discussion

We evaluated the performance of our proposed semi-supervised fine-tuning method across six datasets: MNIST, MNIST with yellow stripes, MNIST with white stripes, SVHN, CIFAR-10, and GalaxyMNIST. Experiments were conducted with both **trainable** and **frozen** backbones using three models: RADIOv2.5 Base [11], DINOv2 Small [9], and CLIP Base [3, 10, 1]. We focused on comparing our **semi-supervised method** to the **purely supervised learning** baseline, as it provides a well-established point of reference. Methods like LoRA[2] were left out of the scope of this work to avoid introducing additional parameters that would require extensive tuning, allowing us to prioritize a clearer evaluation of our semi-supervised fine-tuning approach. Results for each backbone are shown in Figures 3, 4, and 5.

3.1 Performance Across Datasets

Overall, our semi-supervised method consistently improves the performance across all datasets compared to the purely supervised approach. This is particularly evident in the low-labeled data regime (10-100 samples), where our method leverages the combination of labeled and unlabeled data to enhance the learning process. For simpler datasets like MNIST and CIFAR-10, frozen backbones benefit significantly from our method, while for more complex datasets like SVHN and GalaxyMNIST, the trainable models take advantage of the extra flexibility provided by fine-tuning.

MNIST and CIFAR-10: Frozen Models Excel In the MNIST and CIFAR-10 datasets, frozen backbones consistently outperform trainable models under our semi-supervised method, especially in the low-labeled data regime. The simplicity of these datasets, where the content-label relationship

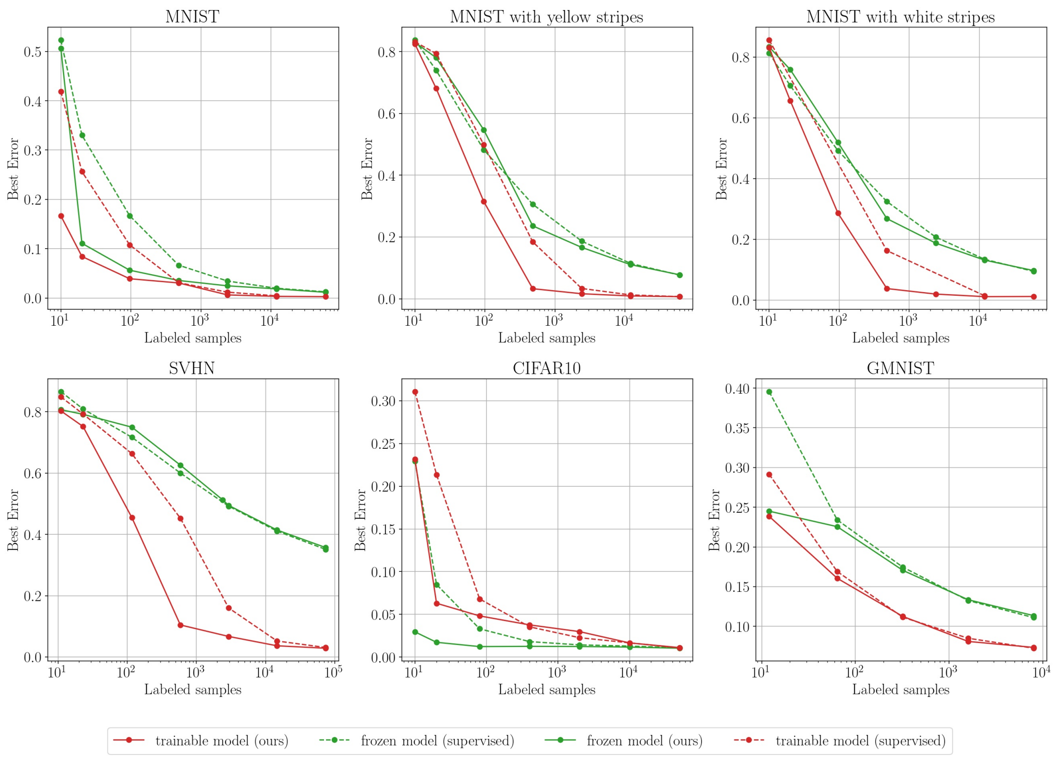

Figure 4: DINOv2 model results. The y-axis represents the best error rate (1 - accuracy), and the x-axis represents the number of labeled samples. For each dataset, the classifier is trained with both supervised learning and our proposed method for frozen and trainable backbone.

is straightforward, allows the frozen models to maintain strong initial representations, and further fine-tuning often leads to overfitting or suboptimal adjustments.

RADIOv2 consistently achieves the best performance across all models, particularly in the frozen setup. As the number of labeled samples increases, trainable models start to close the gap, but frozen RADIOv2 remains competitive. CLIP shows similar patterns, although the performance improvement is less significant compared to RADIOv2; DINOv2, being smaller than RADIOv2 and CLIP, has its frozen backbone underperforming compared to the trainable one under our method. This suggests that RADIOv2's pre-trained representations are better suited to these datasets.

MNIST with Yellow and White Stripes: Trainable Overtakes for RADIOv2 For MNIST with yellow and white stripes, we observe a different dynamic. In the low-data regime, frozen models under our method still perform well, particularly for RADIOv2. However, as more labeled data becomes available, the trainable backbone trained with semi-supervision starts to outperform the frozen one for RADIOv2. This indicates that the added complexity introduced by the stripes (which obscure part of the digits) requires the backbone to adapt its features to fully capture the content-label relationship.

For CLIP and DINOv2, trainable backbones perform better consistently, even in the low-data regime, suggesting that these models require more flexibility to handle the added complexity. Again, RADIOv2 shows the strongest overall performance, indicating that its initial representations are more resilient to perturbations like the stripes.

SVHN and GalaxyMNIST: Trainable Models Are Essential In more complex datasets like SVHN and GalaxyMNIST, the trainable models clearly outperform the frozen models across all backbones. These datasets exhibit larger intra-class variability and a significant distribution shift from the pre-trained representations. In these cases, fine-tuning the backbone is necessary to adapt the model's features to the specific task, especially as more labeled samples become available.

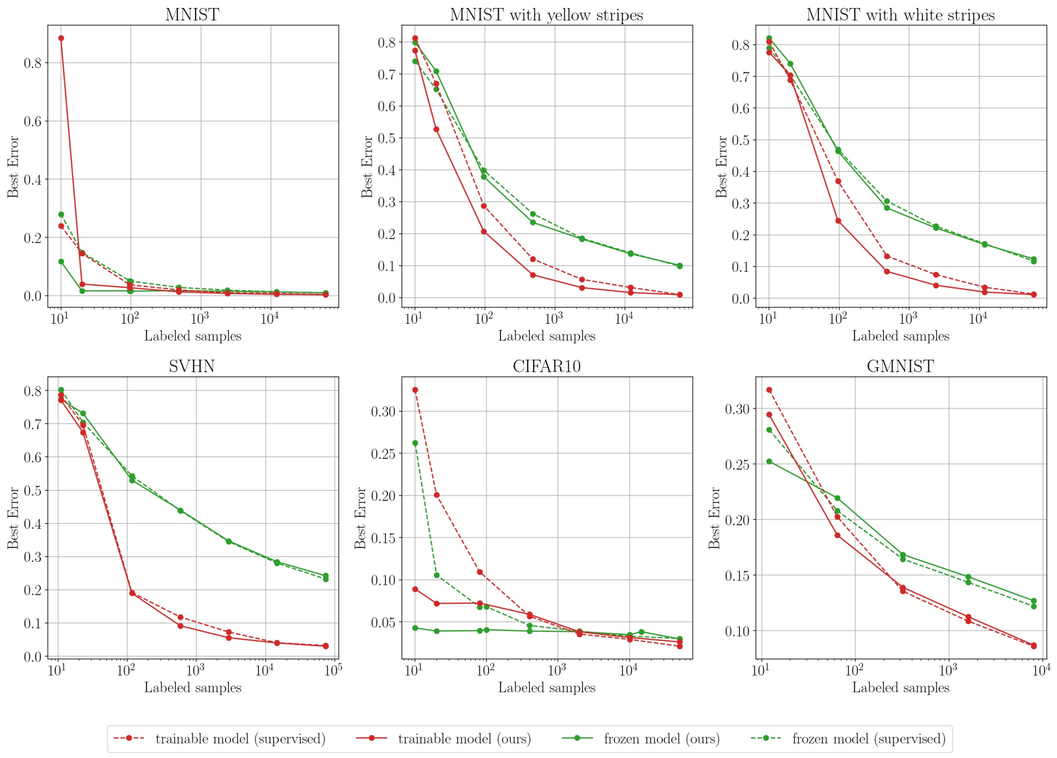

Figure 5: CLIP model results. The y-axis represents the best error rate (1 - accuracy), and the x-axis represents the number of labeled samples. For each dataset, the classifier is trained with both supervised learning and our proposed method for frozen and trainable backbone.

Our semi-supervised method shows improvements in the low-labeled data regime for both frozen and trainable models, though the trainable backbones benefit much more. RADIOv2 continues to show the best performance overall, followed by CLIP and then DINOv2. Our method's improvements on CLIP are rather limited, which means that the model does not benefit much from the extra unlabeled samples. In contrast, RADIOv2 demonstrates greater capacity to leverage unlabeled data, making it the most adaptable model across datasets.

3.2 Discussion: Frozen vs Trainable with Our Method

Our semi-supervised method reveals interesting dynamics between frozen and trainable models. In simpler datasets like MNIST and CIFAR-10, frozen models benefit more from our method, likely because the backbone representations are already well-suited to the task, and fine-tuning may introduce unnecessary changes. This is particularly true for RADIOv2, which consistently outperforms the other models in frozen setups. However, as dataset complexity increases (e.g., MNIST with stripes, SVHN, GalaxyMNIST), fine-tuning backbones become necessary. The trainable models can adapt to more complex content-label relationships and handle the increased intra-class variability, especially in datasets with significant distribution shifts like GalaxyMNIST.

Overall, our method improves the scores compared to the purely supervised approach for most of the studied datasets with both frozen and trainable backbones.

4 Conclusion

This paper emphasizes the critical role of fine-tuning in adapting foundation models to specific downstream tasks, especially in scientific domains with limited labeled data. By leveraging content-style decomposition within an information-theoretic framework, we can effectively tailor the latent representations of foundation models, ensuring their suitability for specific applications. Our findings underscore the importance of aligning model training with the ultimate objectives of downstream tasks to achieve optimal performance.

Our experiments demonstrate that the proposed semi-supervised approach improves performance across both frozen and trainable backbones. The method consistently delivers better results than purely supervised baselines in the majority of the cases. Our method offers a step toward mitigating distribution shift, particularly in the early stages of fine-tuning. However, further research is required to fully understand how different backbone architectures respond to domain shifts and how unlabeled data can be leveraged more effectively in various scientific tasks.

Future work will focus on extending this framework to image reconstruction tasks, expanding the scope of downstream applications and studied foundation models, and exploring alternative fine-tuning strategies to address distribution mismatch effectively.

References

[1] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829, 2023.

[2] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[3] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. If you use this software, please cite it as below.

[4] Alex Krizhevsky, Vinod Nair, Geoffrey Hinton, et al. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5):2, 2014.

[5] Erica Lastufka, Omkar Bait, Olga Taran, Mariia Drozdova, Vitaliy Kinakh, Davide Piras, Marc Audard, Miroslava Dessauges-Zavadsky, Taras Holotyak, Daniel Schaerer, and Svyatoslav Voloshynovskiy. Self-supervised learning on meerkat wide-field continuum images, 2024.

[6] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

[7] Siqi Lu, Junlin Guo, James R Zimmer-Dauphinee, Jordan M Nieusma, Xiao Wang, Parker VanValkenburgh, Steven A Wernke, and Yuankai Huo. Ai foundation models in remote sensing: A survey. *arXiv preprint arXiv:2408.03464*, 2024.

[8] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 4. Granada, 2011.

[9] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[11] Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. Am-radio: Agglomerative vision foundation model reduce all domains into one. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12490–12500, June 2024.

[12] Samuel Stevens, Jiaman Wu, Matthew J Thompson, Elizabeth G Campolongo, Chan Hee Song, David Edward Carlyn, Li Dong, Wasila M Dahdul, Charles Stewart, Tanya Berger-Wolf, et al. Bioclip: A vision foundation model for the tree of life. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19412–19424, 2024.

[13] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *arXiv preprint physics/0004057*, 2000.

[14] P Umesh. Image processing in python. *CSI Communications*, 23, 2012.

[15] Slava Voloshynovskiy, Olga Taran, Mouad Kondah, Taras Holotyak, and Danilo Rezende. Variational information bottleneck for semi-supervised classification. *Entropy*, 22(9):943, 2020.

[16] Mike Walmsley, Chris Lintott, Tobias Géron, Sandor Kruk, Coleman Krawczyk, Kyle W Willett, Steven Bamford, Lee S Kelvin, Lucy Fortson, Yarin Gal, et al. Galaxy zoo decals: Detailed visual morphology measurements from volunteers and deep learning for 314 000 galaxies. *Monthly Notices of the Royal Astronomical Society*, 509(3):3966–3988, 2022.

[17] Shaoting Zhang and Dimitris Metaxas. On the challenges and perspectives of foundation models for medical image analysis. *Medical Image Analysis*, page 102996, 2023.

A Information-Theoretic Details

Content Decoder Loss

For the content decoder, the Kullback-Leibler (KL) divergence measures the difference between the true distribution $p_{c_{a_x}}(\mathbf{c}_{a_x})$ of the content label and the model's predicted distribution $p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x})$. The KL divergence is defined as:

$$\mathcal{D}_{c_{a_x}} := \mathbb{D}_{KL}(p_{c_{a_x}}(\mathbf{c}_{a_x}) || p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x})) = \mathbb{E}_{p_{c_{a_x}}(\mathbf{c}_{a_x})}\left[\log \frac{p_{c_{a_x}}(\mathbf{c}_{a_x})}{p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x})}\right]. \tag{1}$$

This term ensures that the model's content prediction aligns with the true content distribution.

Additionally, we include the cross-entropy loss for the supervised content prediction:

$$\mathcal{L}_{\text{CE}} = -\mathbb{E}_{p(\mathbf{c}_{a_x}, \mathbf{x})}\left[\log p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x})\right]. \tag{2}$$

Style Decoder Loss

For the style decoder, the KL divergence measures the difference between the true prior distribution $p_{s_{a_x}}(\mathbf{s}_{a_x})$ and the model's predicted distribution $p_{\theta_{s_{a_x}}}(\mathbf{s}_{a_x})$. The KL divergence is defined as:

$$\mathcal{D}_{s_{a_x}} := \mathbb{D}_{KL}(p_{s_{a_x}}(\mathbf{s}_{a_x}) || p_{\theta_{s_{a_x}}}(\mathbf{s}_{a_x})) = \mathbb{E}_{p_{s_{a_x}}(\mathbf{s}_{a_x})}\left[\log \frac{p_{s_{a_x}}(\mathbf{s}_{a_x})}{p_{\theta_{s_{a_x}}}(\mathbf{s}_{a_x})}\right]. \tag{3}$$

`[CLS]` Token Decoder Loss

For the `[CLS]` token, the KL divergence is computed between the true prior distribution $p_{y_x}(\mathbf{y}_x)$ and the model's predicted distribution $p_{\theta_{y_x}}(\mathbf{y}_x)$:

$$\mathcal{D}_{y_x} := \mathbb{D}_{KL}(p_{y_x}(\mathbf{y}_x) || p_{\theta_{y_x}}(\mathbf{y}_x)) = \mathbb{E}_{p_{y_x}(\mathbf{y}_x)}\left[\log \frac{p_{y_x}(\mathbf{y}_x)}{p_{\theta_{y_x}}(\mathbf{y}_x)}\right]. \tag{4}$$

Additionally, we include the cosine similarity loss for the reconstruction of the `[CLS]` token:

$$\mathcal{L}_{y_x}(\tilde{\mathbf{y}}_x, \hat{\mathbf{y}}_x) = 1 - \cos(\tilde{\mathbf{y}}_x, \hat{\mathbf{y}}_x). \tag{5}$$

This loss can be considered as a conditional cross-entropy for the reconstrcution.

Total Loss Function

The total loss combines the content, style, and `[CLS]` token losses, along with their respective KL divergence regularizers and the cross-entropy loss for supervised content prediction:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda_c \mathcal{D}_{c_{a_x}} + \lambda_s \mathcal{D}_{s_{a_x}} + \lambda_y \mathcal{D}_{y_x} + \lambda_{y\hat{y}} \mathcal{L}_{y_x}. \tag{6}$$

In our experiments $\lambda_c = \lambda_s = \lambda_y = \lambda_{y\hat{y}} = 1$ following ablation studies conducted in [15].

B Model Descriptions

This appendix provides a detailed description of the models used in our experiments. The architecture includes an encoder content and style outputs ($p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x}|\mathbf{y}_x)$ and $p_{\theta_{s_{a_x}}}(\mathbf{s}_{a_x}|\mathbf{y}_x)$), a decoder for reconstruction ($p_{\theta_{y_x}}(\mathbf{y}_x|\mathbf{c}_{a_x}, \mathbf{s}_{a_x})$), and three discriminators: one for content ($\mathcal{D}_{c_{a_x}}$), one for style ($\mathcal{D}_{s_{a_x}}$), and one for the `CLS` token ($\mathcal{D}_{y_x}$). Each component plays a crucial role in ensuring that the model performs well under semi-supervised learning.

Encoder

The encoder is responsible for processing the input features through a shared MLP structure that maps them into meaningful representations. The shared MLP is shown in Table 1.

Content and Style Heads: After passing through the shared MLP, the features are split into two separate heads: for content $\mathbf{c}_{a_x}$ and one for style $\mathbf{s}_{a_x}$.

Table 1: Shared Encoder Structure

Size	Layer
`CLS` Token size	Input
8000	Linear
8000	Leaky ReLU (slope = 0.01)
8000	Dropout (probability = 0.3)

Content Head

The content encoder $p_{\theta_{c_{a_x}}}(\mathbf{c}_{a_x}|\mathbf{y}_x)$ outputs the content representation. The structure of the head is shown in Table 2.

Table 2: Content Head Structure

Size	Layer
8000	Linear
1024	Leaky ReLU (slope = 0.01)
Number of classes	Output (Linear)

Style Head

The style encoder $p_{\theta_{s_{a_x}}}(\mathbf{s}_{a_x}|\mathbf{y}_x)$ generates the style representation. Its head structure is shown in Table 3.

Table 3: Style Head Structure

Size	Layer
8000	Linear
100	Output

Decoder

The decoder, denoted by $p_{\theta_{y_x}}(\mathbf{y}_x|\mathbf{c}_{a_x}, \mathbf{s}_{a_x})$, reconstructs the `CLS` token by stacking the content and style. The decoder structure is shown in Table 4.

Table 4: Decoder Structure

Size	Layer
Content size + Style size	Input
2560	Linear
2560	Leaky ReLU (slope = 0.01)
2560	Dropout (probability = 0.3)
2560	Linear
2560	Leaky ReLU (slope = 0.01)
2560	Dropout (probability = 0.3)
`CLS` Token size	Output (Linear)

Discriminators

Our method uses binary cross entropy for discriminator losses. The model has three discriminators : one for content $\mathbf{c}_{a_x}$, one for style $\mathbf{s}_{a_x}$, and one for the `CLS` token $\mathbf{y}_x$. The structure of each discriminator is outlined below.

Content Discriminator

The content discriminator $\mathcal{D}_{\mathbf{c}_{a_x}}$ ensures that the content vector has one-hot representation for all unpaired inputs. The structure of this discriminator is shown in Table 5.

Table 5: Content Discriminator Structure

Size	**Layer**
Number of classes	Input
500	Linear
500	Linear
1	Sigmoid

.

Style Discriminator

The style discriminator $\mathcal{D}_{\mathbf{s}_{a_x}}$ ensures that the style representation is Gaussian. The structure of this discriminator is shown in Table 6.

Table 6: Style Discriminator Structure

Size	**Layer**
100	Input
50	Linear
500	Linear
1	Sigmoid

Class Token Discriminator

The `CLS` token discriminator $\mathcal{D}_{\mathbf{y}_{a_x}}$ evaluates the final `CLS` representation, ensuring that given the combination of one-hot content and Gaussian style the decoder can generate the `CLS` token. The structure of this discriminator is shown in Table 7.

Table 7: Class Token Discriminator Structure

Size	**Layer**
`CLS` Token size	Input
128	Linear
128	Leaky ReLU (slope = 0.02)
64	Linear
64	Leaky ReLU (slope = 0.02)
32	Linear
32	Leaky ReLU (slope = 0.02)
1	Sigmoid

Training Setup

The training process involves optimizing the network using the AdamW optimizer and a learning rate of 5×10^{-5}. A warmup schedule of 0 for components and 0.1 for transformers is applied. Batch sizes are set to 512 for frozen models (unsupervised losses) and 32 for trainable models.

The following losses are used:

- **Adversarial losses:** Applied for all discriminators.
- **Supervised loss:** Cross-entropy loss applied to the content head.
- **Reconstruction loss:** Cosine similarity loss for the `CLS` token reconstruction.

C t-SNE in the classifier features for RADIOv2

Here, we visualize classifier features before the last linear layer, where the features have a dimensionality of 1024, for both frozen and trainable RADIOv2 models. We first apply PCA to reduce the features to 5 components, followed by t-SNE for visualization. The images are from the test set, and the colors correspond to the classes. The classifier is fully trained. The upper row shows our model, and the lower row shows the supervised method. The images are ordered by an increasing number of labeled data available for supervised updates.

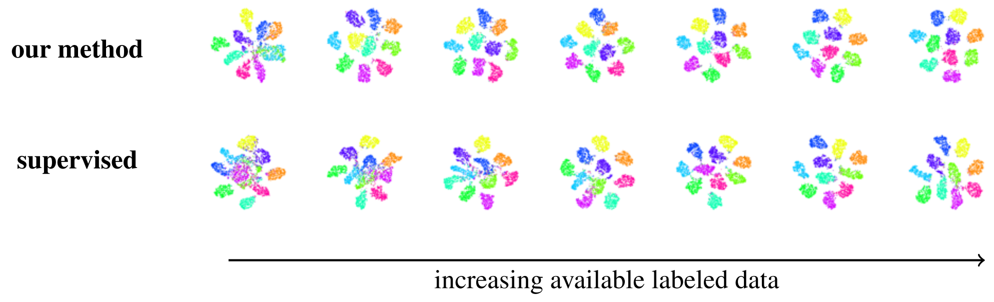

Figure 6: Latent space (dimensionality of 1024) of the classifier for the MNIST dataset with a **frozen** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples increases: 10, 19, 96, 480, 2,400, 12,000, 60,000.

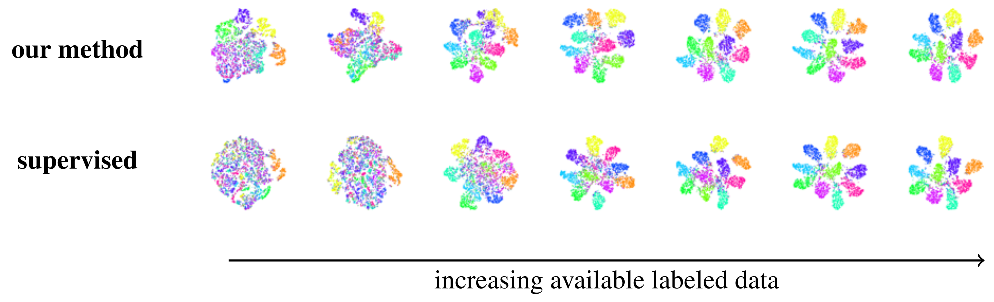

Figure 7: Latent space (dimensionality of 1024) of the classifier for the MNIST dataset with yellow stripes with a **frozen** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 10, 19, 96, 480, 2,400, 12,000, 60,000.

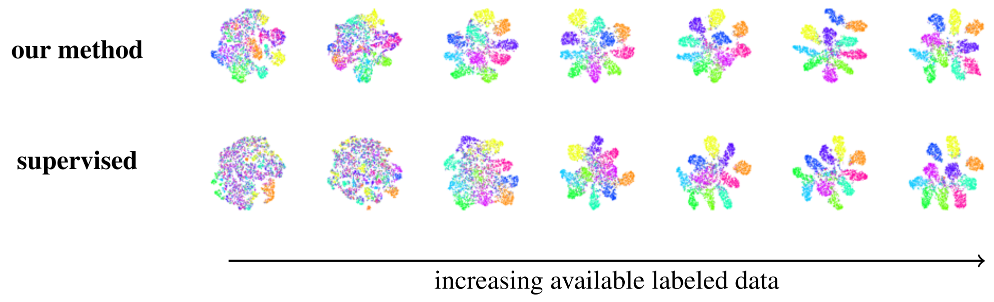

Figure 8: Latent space (dimensionality of 1024) of the classifier for the MNIST dataset with white stripes with a **frozen** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 10, 19, 96, 480, 2,400, 12,000, 60,000.

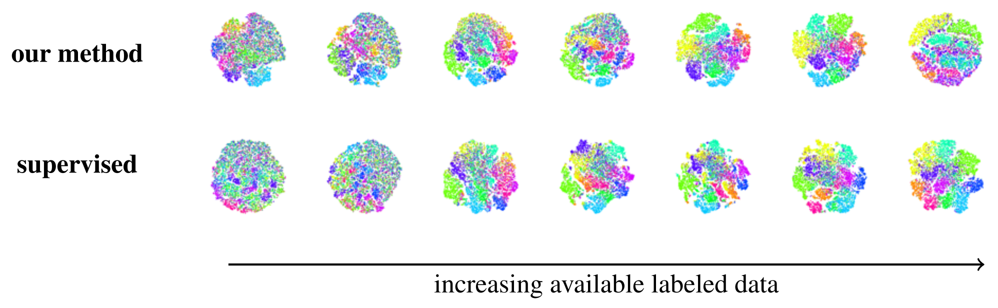

Figure 9: Latent space (dimensionality of 1024) of the classifier for the SVHN dataset with a **frozen** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 10, 23, 117, 586, 2,930, 14,651, 73,257.

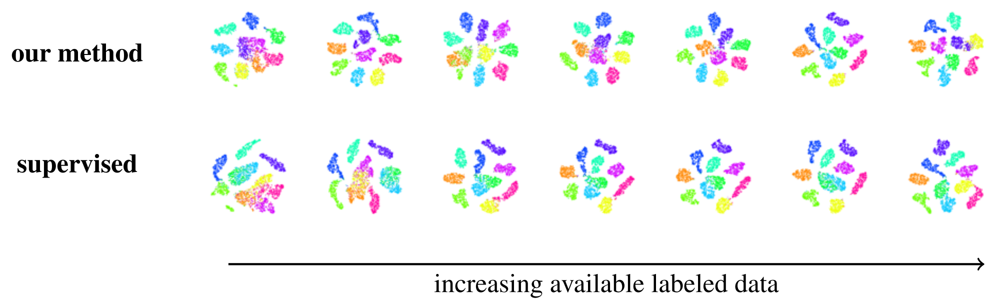

Figure 10: Latent space (dimensionality of 1024) of the classifier for the CIFAR-10 dataset with a **frozen** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 10, 16, 80, 400, 2,000, 10,000, 50,000.

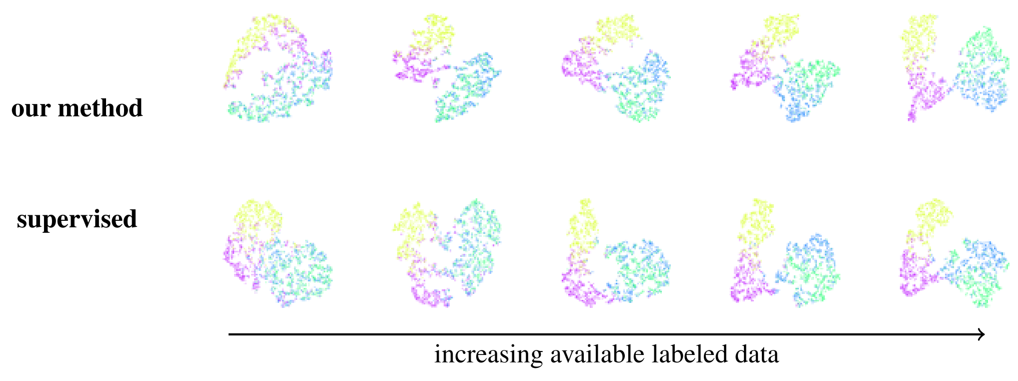

Figure 11: Latent space (dimensionality of 1024) of the classifier for the GalaxyMNIST dataset with a **frozen** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 4, 12, 64, 320, 1,600

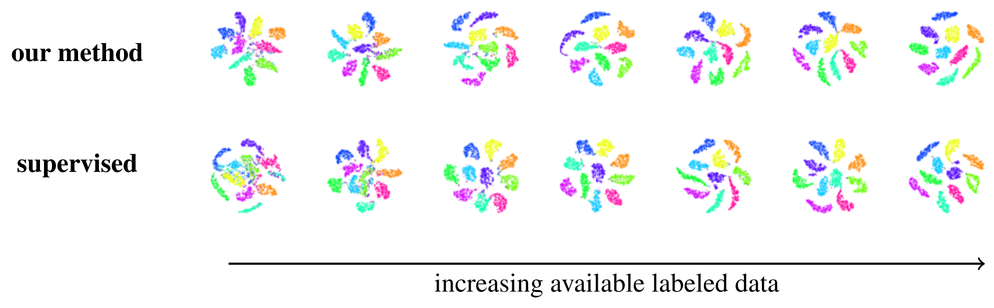

Figure 12: Latent space (dimensionality of 1024) of the classifier for the MNIST dataset with a **trainable** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples increases: 10, 19, 96, 480, 2,400, 12,000, 60,000.

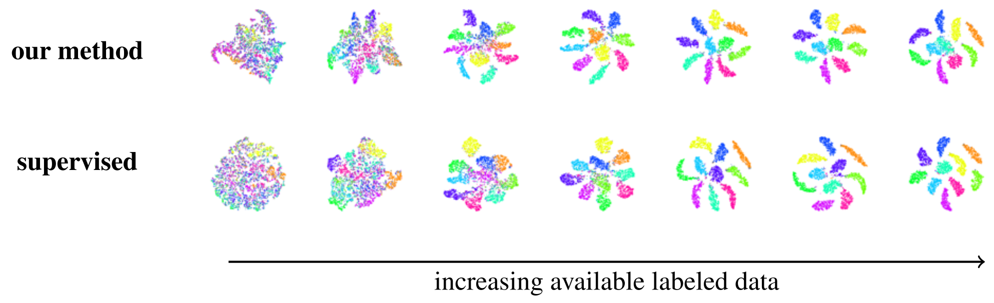

Figure 13: Latent space (dimensionality of 1024) of the classifier for the MNIST dataset with yellow stripes with a **trainable** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 10, 19, 96, 480, 2,400, 12,000, 60,000.

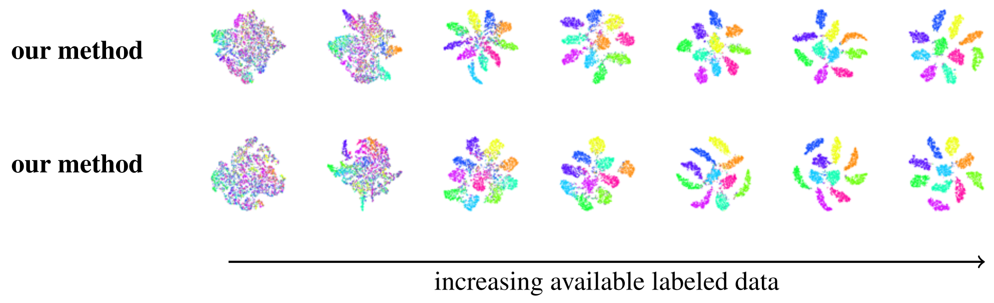

Figure 14: Latent space (dimensionality of 1024) of the classifier for the MNIST dataset with white stripes with a **trainable** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 10, 19, 96, 480, 2,400, 12,000, 60,000.

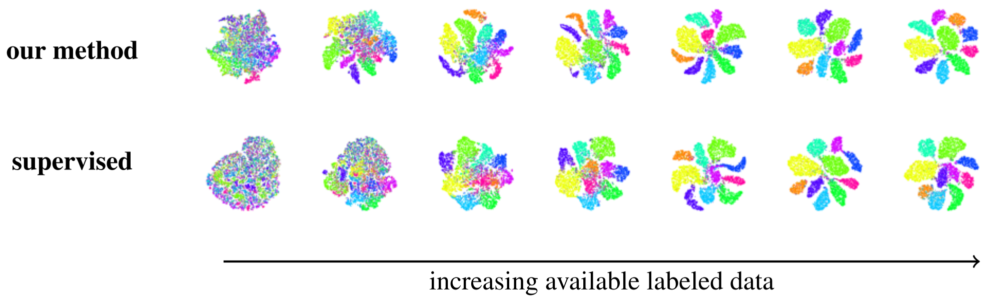

Figure 15: Latent space (dimensionality of 1024) of the classifier for the SVHN dataset with a **trainable** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 10, 23, 117, 586, 2,930, 14,651, 73,257.

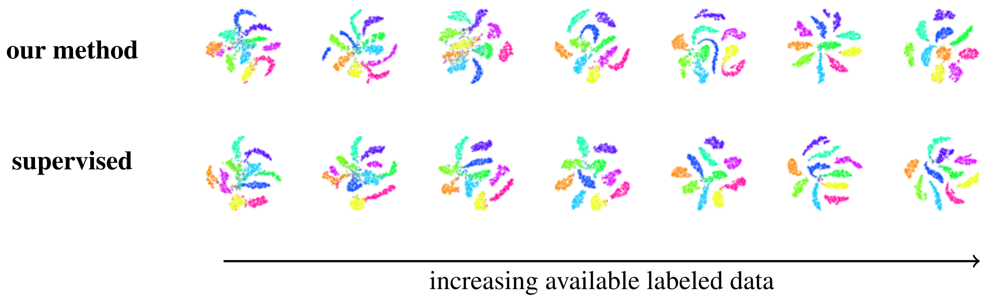

Figure 16: Latent space (dimensionality of 1024) of the classifier for the CIFAR-10 dataset with a **trainable** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 10, 16, 80, 400, 2,000, 10,000, 50,000.

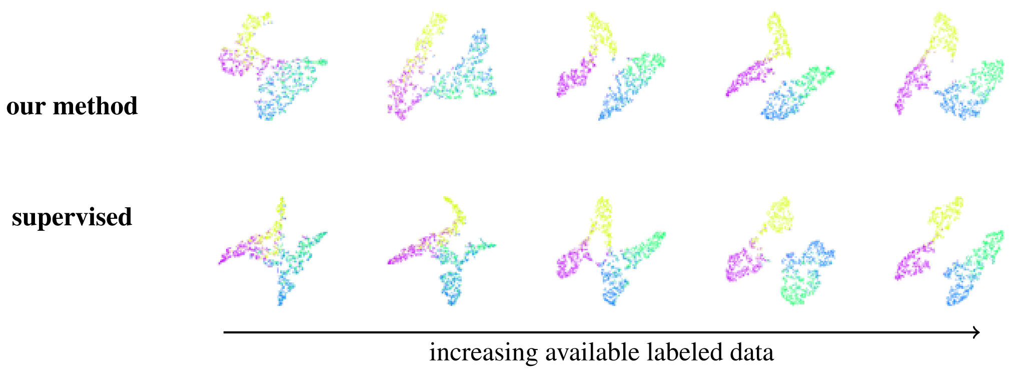

Figure 17: Latent space (dimensionality of 1024) of the classifier for the GalaxyMNIST dataset with a **trainable** backbone. The first row shows our method, and the second row shows the supervised method. From left to right, the total number of labeled samples used for training increases: 4, 12, 64, 320, 1,600