ReSL: Enhancing Deep Clustering through Reset-based Self-Labeling

Anonymous authors
Paper under double-blind review

Abstract

The goal of clustering is to group similar data points together. Deep clustering enhances this process by using neural networks for inferring better data representations through a three-stage approach: pre-training for initial feature learning, deep clustering to structure the latent space, and self-labeling to iteratively refine both representations and cluster assignments. Ever since its inception, self-labeling has been a crucial element for reaching state-of-the-art performance in deep clustering. The samples for the self-labeling phase are obtained by setting a confidence threshold for the network's predictions and only using samples that exceed this threshold for further training. This often improves clustering performance but relies on training with noisy, self-constructed labels (pseudo-labels). As the model iteratively retrains on its own pseudo-labels, the certainty of its predictions tends to rise, increasing its confidence over time. The increasing confidence leads to a growing number of training samples also including more and more samples assigned to the wrong cluster, which can limit performance. Particularly, the model's initially learned biases are amplified by relying on easily learned but ultimately misleading patterns in pseudo-labels, hampering generalization.

In this paper, we propose ReSL, a framework that unites **Re**sets with **S**elf-**L**abeling. We demonstrate that employing weight-reset techniques during self-labeling increases clustering performance and improves generalization. Our findings address limitations of self-labeling and provide a foundation for future research in developing more robust approaches.

1 Introduction

Decades of research have been dedicated to the challenging task of clustering — partitioning data points into groups based on their similarity without utilizing any ground-truth annotations. Traditional clustering methods include k-means (MacQueen et al., 1967), Gaussian mixture models (Bishop & Nasrabadi, 2006), and spectral clustering (Von Luxburg, 2007). Despite their effectiveness, these methods face challenges when applied to high-dimensional data, due to the curse of dimensionality. In contrast, deep neural networks can learn feature representations directly from high-dimensional data by leveraging unsupervised representation learning techniques (Bengio et al., 2013; Abukmeil et al., 2021). Just like the clustering itself, these neural network representations can be trained without any annotations by solving so-called pretext tasks such as reconstruction or contrastive learning. Many well-established deep clustering (DC) algorithms rely on such tasks during a pre-training stage (Van Gansbeke et al., 2020; Zhong et al., 2021; Dang et al., 2021). In contrast, more modern DC algorithms (Qian et al., 2022; Qian, 2023) combine the representation-learning objective and clustering objective into a single end-to-end framework.

Whether a DC algorithm employs a multi-stage or end-to-end architecture, self-labeling has emerged as an indispensable tool to reach state-of-the-art performance in deep clustering (Van Gansbeke et al., 2020; Zhong et al., 2021; Qian, 2023; Miklautz et al., 2024). It fine-tunes the pre-trained deep clustering network by optimizing the cross-entropy loss on a subset of pseudo-labels generated by the model itself. This subset is defined by a confidence threshold, representing the minimum confidence the network must have in its assignment for a sample to be included in the training set. As training in the self-labeling phase progresses, the network becomes increasingly confident in its assignments, leading to the inclusion of more samples in each subsequent training iteration. This dynamic expansion of the training set, while beneficial in leveraging more data, introduces a critical challenge: **non-stationarity**.

The sequential arrival of new confident samples introduces non-stationarity in the input, while the subsequent optimization introduces non-stationarity in the targets. Inevitably, the training set will include samples assigned to the wrong cluster, leading to training on **noisy targets**. The inherent non-stationarity of self-labeling, coupled with the inherent risk of incorporating incorrect pseudo-labels, can negatively impact the training process. Figure 1 illustrates how the training set size grows over time due to this increasing confidence. Moreover, self-labeling fine-tunes the pre-trained network on a different objective, effectively **warm-starting** the network from the DC stage rather than initializing from scratch. The aforementioned warm-starting, noisy targets, and non-stationarity induce optimization issues in deep learning, which have been studied under the umbrella term "*plasticity loss*", referring to a loss of the network's ability to fit new targets (Klein et al., 2024; Ash & Adams, 2020; Lee et al., 2024).

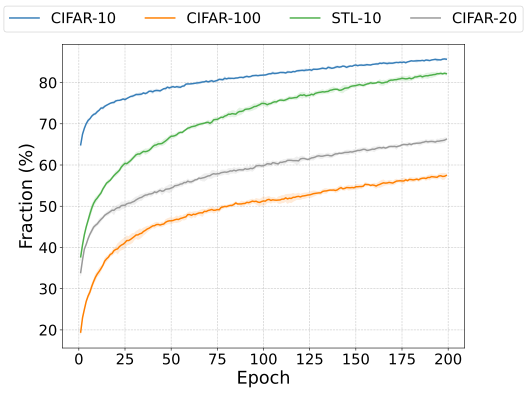


Figure 1: Self-labeling with SCAN gradually increases its probability estimates, leading to the inclusion of a growing proportion of samples to the confident training set in subsequent training rounds across different datasets. This expansion exacerbates non-stationarity and increases the risk of incorporating noisy pseudo-labels.

One effective way to reduce the negative impact of plasticity loss is by resetting the network's weights. This approach underlies several promising methods from the plasticity literature (Ash & Adams, 2020; Lee et al., 2024; Zaidi et al., 2023). Expanding on this concept, we propose a novel method, ReSL, that combines **Re**sets with **S**elf-**L**abeling. We conduct experiments using the DC method SCAN (Van Gansbeke et al., 2020), as it was the first to introduce self-labeling. Our quantitative experiments demonstrate improved clustering performance on STL-10 (Coates et al., 2011) and CIFAR-10/20/100 (Krizhevsky et al., 2009) datasets. To better understand the underlying mechanisms driving these improvements, we employ plasticity injection (Nikishin et al., 2024). Plasticity injection allows us to rule out trainability issues (recognized as one of the factors behind plasticity loss (Lee et al., 2024)), by introducing an output-invariant re-initialization scheme. Our investigation shows that ReSL stabilizes cluster label reassignments during training, leading to higher-quality pseudo-labels that generalize better to the clustering task. To summarize our contributions:

- We propose ReSL, an algorithm for self-labeling, and demonstrate that it consistently improves the clustering performance of the SCAN algorithm across multiple datasets.
- We propose a novel reset strategy that performs stronger resets at the beginning of training, where the effects of warm-starting are most pronounced.
- We investigate possible mechanisms behind performance improvements and demonstrate that less intense changes of pseudo-labels help decrease the compounding effect of noisy pseudo-labels.

2 Problem Setup

We are given an unlabeled dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where $\mathbf{x}_i \in \mathbb{R}^D$. Our goal is to partition $\mathcal{D}$ into C clusters without using any ground-truth labels. We assume access to a pretrained neural network $g_\phi \colon \mathbb{R}^D \to \mathbb{R}^d$ that provides a latent representation $\mathbf{z}_i = g_\phi(\mathbf{x}_i)$ for each $\mathbf{x}_i$. A clustering head $h_\theta \colon \mathbb{R}^d \to \Delta^{C-1}$ maps latent vectors $\mathbf{z}_i$ to probability distributions $\mathbf{q}_i = h_\theta(\mathbf{z}_i)$, where Δ^{C-1} denotes the $(C-1)$-simplex. The composite function $f_\eta = h_\theta \circ g_\phi$ assigns a probability distribution $\mathbf{q}_i$ over C clusters to each input $\mathbf{x}_i$.

Self-labeling. We iteratively refine the cluster assignments by creating pseudo-labels from the model's own predictions. Let $\tau \in [0, 1]$ be a confidence threshold. Define the set of confident samples as $S_\tau = \{i \mid \max_c(\mathbf{q}_i)_c \geq \tau\}$. For each $i \in S_\tau$, assign the pseudo-label $\tilde{y}_i = \arg\max_c(\mathbf{q}_i)_c$. and update the clustering function f_η by minimizing the cross-entropy loss on these pseudo-labels:

$$\mathcal{L}_{\text{SL}} \;=\; -\frac{1}{|S_\tau|}\sum_{i\,\in\,S_\tau}\sum_{c=1}^{C}\tilde{\mathbf{y}}_i(c)\,\log\Big[\big(f_\eta(\hat{\mathbf{x}}_i)\big)_c\Big],$$

where $\tilde{\mathbf{y}}_i$ is the one-hot encoding of $\tilde{y}_i$. Here, $\hat{\mathbf{x}}_i$ denotes an augmentation of the original input $\mathbf{x}_i$ (obtained through techniques such as cropping, rotation, or color jittering), which is employed to prevent overfitting. Our aim is to learn an accurate partition of $\mathcal{D}$ into C clusters, despite the risk of reinforcing initially biased pseudo-labels through this iterative self-labeling procedure.

3 Background and Related Work

3.1 Deep Clustering

Recent advancements in DC have bridged the gap between feature learning and clustering. The feature learning component can be realized with various architectures such as convolutional neural networks (CNNs), autoencoders (AEs), and contrastive learning frameworks such as SimCLR (Chen et al., 2020a) or MoCo (Chen et al., 2020b). Jointly optimizing representation learning and clustering objectives has enabled DC methods. Recent DC methods (Van Gansbeke et al., 2020; Qian, 2023; Zhong et al., 2021) have shown results close to supervised methods on widely used image benchmarks. These studies share a practice of fine-tuning clustering networks through self-labeling to enhance the quality of cluster assignments. Van Gansbeke et al. (2020) introduced the self-labeling procedure within their DC method, SCAN (Semantic Clustering by Adopting Nearest Neighbors). Further details on SCAN are provided in Appendix A.1.

3.2 Network Plasticity

To enable efficient learning, deep neural networks must possess plasticity — that is, the capacity to adapt and modify their weights during training. This concept is akin to neuroplasticity, which enables learning in the human brain. The term plasticity has gained broader attention in the fields of deep reinforcement learning (RL) (Klein et al., 2024) and continual learning (Elsayed & Mahmood, 2024). Once plasticity is lost, the ability to learn diminishes (Lyle et al., 2023). Weight-reset techniques have emerged as an effective strategy for mitigating the loss of plasticity (Lyle et al., 2023; Klein et al., 2024). Zaidi et al. (2023) show that when training on noisy labels, resetting results in a substantially improved generalization. To address suboptimal performance resulting from warm-starting and subsequent plasticity loss, Ash & Adams (2020) propose partial weight resets. Lee et al. (2024) then decomposed plasticity loss into trainability and generalizability. To minimize knowledge loss during resets, they developed a reset strategy with distillation, enabling rapid adaptation and gradual generalization. For a comprehensive overview of plasticity loss and mitigation strategies, we refer the reader to Klein et al. (2024).

4 Methodology

We propose ReSL, an algorithm for the self-labeling stage of deep clustering algorithms. ReSL is designed to mitigate the detrimental effects of non-stationarity and warm-starting discussed in Section 1 by incorporating a periodic weight-reset mechanism into the training pipeline. Algorithm 1 provides a generic implementation.

Algorithm 1 PyTorch-style pseudo-code of ReSL

```
# model: neural net
# reset_freq: interval for weight resets
# reset_strategy: a strategy for retrieving and resetting weights
# confidence_threshold: minimum confidence for including a sample in training
for epoch in range(max_epochs):
    pseudo_labels = obtain_pseudo_labels(model, dataloader, confidence_threshold)
    update_model(model, dataloader, pseudo_labels, optimizer, criterion)
    reset_strategy.update(model, epoch)
    if epoch % reset_freq == 0:
        reset_strategy.reset(model)
cluster_assignments = model(dataset)
```

At a high level, ReSL alternates between two key steps during training. In the first step, the network updates its parameters using pseudo-labels obtained from its current clustering assignments. A confidence threshold ensures that only samples with sufficiently reliable predictions are included in the training set. In the second step, after a fixed number of training epochs, the algorithm performs a weight reset. The specific mechanism for resetting the model's weights is abstracted into the reset strategy module, which permits various reset types. Depending on the chosen strategy, the reset may be implemented as a soft reset — resetting only a subset of the parameters — or as a hard reset, wherein all weights are changed. We further discuss two established reset strategies and propose a novel approach that gradually softens resets — addressing objective change from the DC stage to self-labeling by applying stronger resets early in training.

ReSL with Soft Resets ($\mathbf{ReSL_{SP}}$) The first reset strategy within our ReSL framework leverages soft resets by applying the Shrink and Perturb method (Ash & Adams, 2020) to the clustering head h_θ every R epoch. Specifically, we sample fresh parameters θ' from the original initializer (He et al., 2016) and perform a soft weight reset using a convex combination of the network's current weights and the freshly sampled weights $\theta \leftarrow \alpha\,\theta + (1-\alpha)\,\theta'$, where $\alpha \in (0,1)$ is a retention parameter that controls how much of the current state is preserved relative to the new initialization.

ReSL with Soft Resets ($\mathbf{ReSL_{SP*}}$) As part of ReSL, we propose a novel variant that "softens" the reset strength over time. Specifically, the retention parameter α starts at a lower value, enabling larger resets initially, and is then linearly increased to 1.0 by the final epoch using a softening factor $\frac{\text{epoch}}{E}$, meaning no resets at the end of training. At each reset interval R, these updated values are applied, ensuring that weight resets are stronger at the beginning of training, where warm-start initialization takes place.

ReSL with Hare & Tortoise Networks ($\mathbf{ReSL_{HT}}$) Inspired by the "hare and tortoise" approach of Lee et al. (2024), we maintain two networks with parameters η^{Hare} and η^{Tortoise}, the "hare" and the "tortoise" respectively. Both networks are initialized with the pretrained model resulting from the initial clustering stage. The hare rapidly adapts via SGD, whereas the tortoise's parameters are updated via distillation using an exponential moving average (EMA) of the hare's parameters: $\eta^{\text{Tortoise}} \leftarrow \mu\eta^{\text{Tortoise}} + (1-\mu)\eta^{\text{Hare}}$, where $\mu \in (0,1)$ is a momentum parameter. At each reset interval R, the hare network's parameters are reset to the current state of the tortoise network: $\eta^{\text{Hare}} \leftarrow \eta^{\text{Tortoise}}$. The tortoise network is used for subsequent data clustering.

5 Experiments

This section outlines our experimental methodology, beginning with a description of the datasets and setups employed in Section 5.1. We then present the results of ReSL applied with established weight-reset techniques in Section 5.2. Finally, we investigate the implications of our results in Section 5.3.

5.1 Experiment Setup

We evaluate clustering performance on CIFAR-10/20/100 (Krizhevsky et al., 2009) and STL-10 (Coates et al., 2011), following established benchmarks (Qian, 2023; Van Gansbeke et al., 2020; Zhong et al., 2021). Performance is measured using clustering accuracy (ACC) (Yang et al., 2010), Adjusted Rand Index (ARI) (Hubert & Arabie, 1985), and Normalized Mutual Information (NMI) (Kvalseth, 1987), averaged over five random seeds reported on the validation set, consistent with recent DC methods and surveys (Van Gansbeke et al., 2020; Zhong et al., 2021; Qian, 2023; Zhou et al., 2024; Lu et al., 2024; Huang et al., 2024). We use the original SCAN setup for our experiments. Self-labeling training starts with pretrained models trained using the SCAN codebase[1], with SCAN's hyperparameters (Appendix C) applied consistently throughout the training. For each dataset, we report the results corresponding to the configuration that achieved the best average accuracy over the five random seeds. The hyperparameter sensitivity of the ReSL's underlying reset strategies is analyzed in Appendix G.

[1]SCAN codebase `https://github.com/wvangansbeke/Unsupervised-Classification`.

5.2 Results

We summarize the clustering performance of our proposed ReSL variants against SCAN with standard self-labeling (SCAN+SL) across four benchmark datasets in Table 1.

Table 1: Clustering accuracy of ReSL variants against the standard self-labeling procedure. The best result in each column is in **bold**, and the second best is underlined.

Experiment	STL-10		CIFAR-10		CIFAR-20		CIFAR-100	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
ReSL$_{SP}$	76.75	77.12	<u>88.16</u>	**90.79**	48.60	49.24	31.56	32.19
ReSL$_{SP*}$	<u>76.91</u>	<u>77.21</u>	87.87	88.16	<u>48.64</u>	<u>49.47</u>	31.35	<u>33.02</u>
ReSL$_{HT}$	**77.80**	**78.11**	**88.21**	<u>88.85</u>	**49.13**	**50.68**	**34.89**	**35.6**
SCAN+SL	75.78	76.85	87.57	87.92	48.02	48.83	<u>31.68</u>	32.51

Both variants, ReSL$_{SP}$ and ReSL$_{SP*}$, show varying performance across datasets, with the best improvement observed on STL-10 and a marginal decrease in accuracy on CIFAR-100. While the shrink and perturb method risks losing valuable information through aggressive resets, the hare and tortoise approach preserves stability with an exponential moving average of network parameters, leading to its consistently superior performance. ReSL$_{HT}$ outperforms SCAN+SL across all datasets, with improvements of up to 3.21% for CIFAR-100. On CIFAR-10, it leads to an improvement of 0.64%, while on CIFAR-20 and STL-10, it yields an additional 1.11% and 2.02% improvement, respectively. Additionally, ReSL$_{HT}$ consistently achieves the highest scores in terms of ARI and NMI across all datasets (see Appendix E).

Table 2: ReSL$_{HT}$ reduces noisy labels in the set of **confident** samples S_τ. We evaluate the quality of S_τ by calculating the NMI between the confident pseudo-labels and the true classes (NMI Match). A higher NMI Match indicates less label noise within S_τ. Results are reported at epochs 100 and 200 of self-labeling (denoted as e=100 and e=200 in the table). The best result in each column is in **bold**.

Experiment	STL-10		CIFAR-10		CIFAR-20		CIFAR-100	
	e=100	e=200	e=100	e=200	e=100	e=200	e=100	e=200
ReSL$_{HT}$	**85.91**	**83.53**	**89.80**	**88.84**	**66.05**	**62.95**	**79.27**	**77.88**
SCAN+SL	80.34	76.71	89.69	88.49	64.99	62.14	72.18	68.28

ReSL$_{HT}$ further surpasses SCAN+SL in terms of the quality of pseudo-labels, as summarized in Table 2, where we report results at epochs 100 and 200 of self-labeling.

5.3 Detecting Self-Labeling Pitfalls

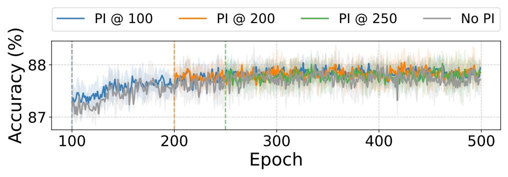


Figure 2: A comparison of CIFAR-10 average accuracy for SCAN+SL models trained with and without plasticity injection. Plasticity injection (PI) was applied at epochs 100, 200, and 250 to analyze its effect at various training stages.

Recent work by Lee et al. (2024) decouples plasticity loss into two distinct components: trainability — the network's ability to update its parameters — and generalizability — its capacity to perform well on unseen data. To diagnose whether the clustering head h_θ suffers from a loss of trainability, we use plasticity injection (Nikishin et al., 2024). Originally developed in deep RL community, plasticity injection evaluates the network's ability to update (i.e. trainability) without altering the total number of trainable parameters or the immediate predictions.

To restore the plasticity of the clustering head h_θ at a designated training step T, we construct three copies of: the *base* head h_θ (with the parameters learned up to step T), a *freshly initialized* adaptive head $h_{\theta'_1}$ (a random reinitialization), and a *frozen* static copy of the adaptive head $h_{\theta'_2}$ (identical to $h_{\theta'_1}$ at creation). After injection, we freeze the base head h_θ, allow the adaptive head $h_{\theta'_1}$ to continue training, and keep the static copy h'_{θ_2} unchanged. The combined output is computed as: $\mathbf{q}_i = h_\theta(\mathbf{z}_i) + h_{\theta'_1}(\mathbf{z}_i) - h_{\theta'_2}(\mathbf{z}_i)$, thereby preserving the original predictions at the moment of injection. If injection improves the performance, this suggests that the clustering head had indeed experienced trainability difficulties. Otherwise, other factors might be limiting further performance gains.

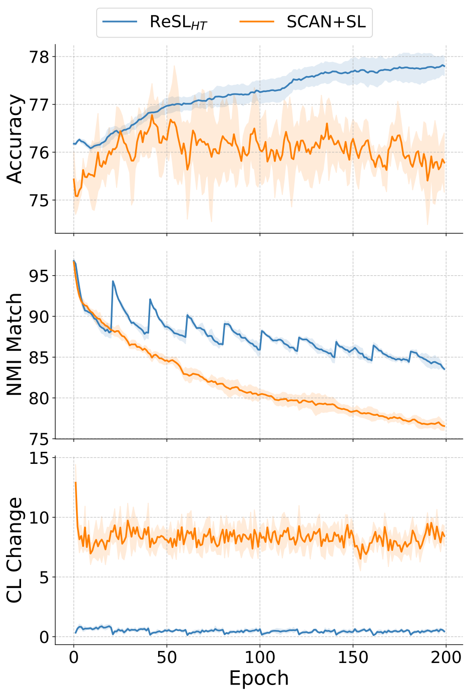


Figure 3: We measure the noise present in the confident set using NMI between pseudo-labels and true classes (NMI Match; see Appendix D). ReSL$_{HT}$ significantly slows the reinforcement of incorrect pseudo-labels during self-labeling on STL-10. Similar patterns are observed on the other datasets (Appendix 4).

Results on CIFAR-10 (Figure 2) and other datasets (Appendix H) show that all models, including the baseline without injection, achieved comparable accuracy, despite minor fluctuations immediately after the injection. Our results imply that while plasticity injection enhances trainability, it does little to counteract the decline in generalizability during self-labeling. This deterioration in generalizability is evident in Figure 3, where the decreasing NMI between confident pseudo-labels and the true classes (NMI Match; see Appendix D) reflects a degradation in pseudo-label quality, which in turn amplifies the initial biases, even as overall accuracy improves. We attribute the decreasing quality of confident pseudo-labels to the higher values of cluster-label reassignment frequencies (CL Change; see Appendix D). SCAN+SL undergoes frequent cluster-label reassignments (CL Change) while continuously adapting to an expanding set of noisy pseudo-labels, leading to overfitting on the pseudo-labeling task and ultimately harming its generalizability. In contrast, ReSL$_{HT}$ remains more committed to the clustering assignments (low CL Change) by periodically resetting to a slow-moving EMA, thereby limiting the compounding effect of noisy pseudo-labels.

6 Conclusion and Future Work

In this work, we investigate the self-labeling stage of deep clustering. We show that increasing number of pseudo-labels introduces non-stationarity and amplifies initial biases, ultimately limiting the model's ability to generalize. To address these issues, we introduce ReSL. Our experiments demonstrate that ReSL, particularly the hare and tortoise variant, outperforms the standard self-labeling procedure across multiple datasets. ReSL achieves this by stabilizing cluster cluster label reassignments, slowing pseudo-label quality degradation. Future work will extend our analysis to other deep clustering methods and reset strategies, and explore alternative reset schedules.

REPRODUCIBILITY

The complete codebase, including configuration files for reproducing all experiments reported in this work, is available at this link.

ETHICS STATEMENT

This work contributes foundational research to deep clustering, potentially advancing diverse scientific domains. While the released code could be misused, we believe its potential to facilitate research outweighs this possibility. In medical contexts, our could improve patient categorization for targeted treatments. However, its inherent limitations require careful implementation and human oversight to minimize potential errors and public concern. Similar caution is warranted in sensitive applications like finance.

REFERENCES

Mohanad Abukmeil, Stefano Ferrari, Angelo Genovese, Vincenzo Piuri, and Fabio Scotti. A survey of unsupervised generative models for exploratory data analysis and representation learning. *ACM Comput. Surv.*, 54(5), July 2021. ISSN 0360-0300. doi:10.1145/3450963.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. doi:10.1109/TPAMI.2013.50.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020b.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Zhiyuan Dang, Cheng Deng, Xu Yang, Kun Wei, and Heng Huang. Nearest neighbor matching for deep clustering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13693–13702, 2021.

Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27, 2014.

Mohamed Elsayed and A Rupam Mahmood. Addressing loss of plasticity and catastrophic forgetting in continual learning. *arXiv preprint arXiv:2404.00781*, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Huajuan Huang, Chen Wang, Xiuxi Wei, and Yongquan Zhou. Deep image clustering: A survey. *Neurocomputing*, 599:128101, 2024.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985. doi:10.1007/BF01908075.

Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9865–9874, 2019.

Timo Klein, Lukas Miklautz, Kevin Sidak, Claudia Plant, and Sebastian Tschiatschek. Plasticity loss in deep reinforcement learning: A survey. *arXiv preprint arXiv:2411.04832*, 2024.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Tarald O Kvalseth. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):517–519, 1987. doi:10.1109/TSMC.1987.4309069.

Hojoon Lee, Hyeonseo Cho, Hyunseung Kim, Donghu Kim, Dugki Min, Jaegul Choo, and Clare Lyle. Slow and steady wins the race: Maintaining plasticity with hare and tortoise networks. *arXiv preprint arXiv:2406.02596*, 2024.

Yiding Lu, Haobin Li, Yunfan Li, Yijie Lin, and Xi Peng. A survey on deep clustering: from the prior perspective. *Vicinagearth*, 1(1):4, 2024.

Clare Lyle, Zeyu Zheng, Evgenii Nikishin, Bernardo Avila Pires, Razvan Pascanu, and Will Dabney. Understanding plasticity in neural networks. In *International Conference on Machine Learning*, pp. 23190–23211. PMLR, 2023.

James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pp. 281–297. Oakland, CA, USA, 1967.

Lukas Miklautz, Timo Klein, Kevin Sidak, Collin Leiber, Thomas Lang, Andrii Shkabrii, Sebastian Tschiatschek, and Claudia Plant. Breaking the reclustering barrier in centroid-based deep clustering. *arXiv preprint arXiv:2411.02275*, 2024.

Evgenii Nikishin, Junhyuk Oh, Georg Ostrovski, Clare Lyle, Razvan Pascanu, Will Dabney, and André Barreto. Deep reinforcement learning with plasticity injection. *Advances in Neural Information Processing Systems*, 36, 2024.

Qi Qian. Stable cluster discrimination for deep clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16645–16654, 2023.

Qi Qian, Yuanhong Xu, Juhua Hu, Hao Li, and Rong Jin. Unsupervised visual representation learning by online constrained k-means. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16640–16649, 2022.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018. doi:10.48550/arXiv.1807.03748.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European conference on computer vision*, pp. 268–285. Springer, 2020.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *IEEE Transactions on Image Processing*, 19(10): 2761–2773, 2010. doi:10.1109/TIP.2010.2049235.

Sheheryar Zaidi, Tudor Berariu, Hyunjik Kim, Jorg Bornschein, Claudia Clopath, Yee Whye Teh, and Razvan Pascanu. When does re-initialization work? In *Proceedings on*, pp. 12–26. PMLR, 2023.

Huasong Zhong, Jianlong Wu, Chong Chen, Jianqiang Huang, Minghua Deng, Liqiang Nie, Zhouchen Lin, and Xian-Sheng Hua. Graph contrastive clustering. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9224–9233, 2021.

Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Zhao Li, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, and Martin Ester. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *ACM Computing Surveys*, 57(3):1–38, 2024.

Appendix

A Background

A.1 SCAN

Semantic clustering by adopting nearest neighbors (SCAN) (Van Gansbeke et al., 2020) is a deep clustering framework. It comprises three stages: representation learning, clustering, and self-labeling, the latter being a novel contribution to the deep clustering field.

In the first stage, a feature extractor function g_ϕ is trained on the dataset $\mathcal{D}$ using a self-supervised pretext task (e.g. SimCLR (Chen et al., 2020a)). The resulting feature embeddings are then used to identify K nearest neighbors for each sample $\mathbf{x}_i \in \mathcal{D}$, forming a set $\mathcal{N}_{\mathbf{x}_i}$ that is assumed to contain samples belonging to the same semantic cluster. The obtained semantically meaningful features are further used as a prior for clustering the images.

To encourage consistent clustering across neighbors, SCAN introduces a clustering function f_η which performs a soft assignment of samples to clusters $\mathcal{C} = \{1, \ldots, C\}$. The probability of assigning a sample $\mathbf{x}_i$ to cluster c is denoted as $f_\eta^c(\mathbf{x}_i) \in [0, 1]$. SCAN learns a clustering function by minimizing a proposed semantic clustering loss:

$$\mathcal{L}_{\text{SCAN}} = -\frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}_i \in \mathcal{D}} \sum_{k \in \mathcal{N}_{\mathbf{x}_i}} \log\langle f_\eta(\mathbf{x}_i), f_\eta(k)\rangle + \lambda \sum_{c \in \mathcal{C}} f_\eta^{c'} \log f_\eta^{c'},$$

where the term $f_\eta^{c'}$ represents the average cluster assignment across the dataset:

$$f_\eta^{c'} = \frac{1}{|\mathcal{D}|} \sum_{\mathbf{x}_i \in \mathcal{D}} f_\eta^c(\mathbf{x}_i).$$

The first term ensures consistent clustering for a sample and its neighbors by maximizing their similarity, while the second term spreads the predictions uniformly across the clusters C. This prevents the model from collapsing into trivial solutions where all samples are assigned to a single cluster.

The final stage of the SCAN algorithm refines the clustering assignments via self-labeling by using the assignments from the previous iteration as pseudo-labels.

A.2 SimCLR

Similar to previous contrastive learning algorithms (Dosovitskiy et al., 2014; Wu et al., 2018; Ji et al., 2019), SimCLR (Chen et al., 2020a) learns representations by maximizing agreement between differently augmented views of the same data sample via a contrastive loss. During SimCLR pretraining, from each sample in a batch of N samples, we derive two augmented versions of this sample, resulting in a batch size of $2N$. Given a positive pair, SimCLR treats the other $2(N-1)$ samples as negative samples. SimCLR utilizes the InfoNCE (van den Oord et al., 2018) loss by applying it in the latent space. For a given positive pair of samples (i, j), the contrastive loss with temperature parameter τ is defined as follows:

$$\mathcal{L}_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

where $\mathbb{1}_{[k \neq i]}$ is 1 if $k \neq i$, and $\text{sim}(\mathbf{z}_i, \mathbf{z}_j)$ denotes the cosine similarity between two embedded samples.

B Datasets

CIFAR-10/20/100 (Krizhevsky et al., 2009): CIFAR datasets include three variants: CIFAR-10, CIFAR-20, and CIFAR-100. CIFAR-10 consists of images with dimensions of 32×32×3 channels, categorized into 10 classes. CIFAR-100 expands this structure to 100 classes, grouped into 20 super-classes, forming the basis of CIFAR-20. In total, CIFAR dataset contains 50,000 training images and 10,000 validation images.

STL-10 (Coates et al., 2011): STL-10 dataset contains 10 classes of images, each of size 96 x 96 x 3 channels. It provides 500 training images per class, 800 validation images per class, and an additional 100,000 unlabeled samples for use during the training stage. Note that following the original implementation of SCAN (Van Gansbeke et al., 2020), we do not utilize these unlabeled samples.

C Hyperparameters

Table 3: SCAN's hyperparameters for self-labeling

Parameter	**CIFAR10**	**CIFAR-20 / 100**	**STL10**
General Training			
Confidence threshold	0.99	0.99	0.99
Criterion	Confidence cross entropy	Confidence cross entropy	Confidence cross entropy
Apply class balancing	True	True	True
Epochs	200	200	200
Batch size	1000	1000	1000
Model			
Backbone	resnet18	resnet18	resnet18
Number of heads	1	1	1
Augmentations (Train set)			
Augmentation strategy	SCAN	SCAN	SCAN
Crop size	32	32	96
Normalize mean	[0.4914, 0.4822, 0.4465]	[0.5071, 0.4867, 0.4408]	[0.485, 0.456, 0.406]
Normalize std	[0.2023, 0.1994, 0.2010]	[0.2675, 0.2565, 0.2761]	[0.229, 0.224, 0.225]
Cutout			
Num of holes	1	1	1
Length	16	16	32
Random	True	True	True
Transformations (Validation set)			
Crop size	32	32	96
Normalize mean	[0.4914, 0.4822, 0.4465]	[0.5071, 0.4867, 0.4408]	[0.485, 0.456, 0.406]
Normalize std	[0.2023, 0.1994, 0.2010]	[0.2675, 0.2565, 0.2761]	[0.229, 0.224, 0.225]
Optimizer			
Type	Adam	Adam	Adam
Learning rate	1e-4	1e-4	1e-4
Weight decay	1e-4	1e-4	1e-4

We use the same experimental setup as SCAN for each dataset, except that we train STL-10 for 200 epochs instead of 100 to ensure consistency with the CIFAR datasets. Unless stated otherwise in the corresponding experiment sections, the hyperparameters listed in Table 3 are consistently applied across all our self-labeling experiments and match those from the original SCAN codebase.

D METRICS

Clustering Accuracy (ACC) We measure the clustering accuracy by first allowing for an optimal matching (permutation) between predicted cluster labels and ground-truth labels. Concretely, given ground-truth labels $\{y_i\}_{i=1}^n$ and predicted labels $\{c_i\}_{i=1}^n$, we seek a one-to-one mapping g that maximizes the overall agreement. The Hungarian algorithm Kuhn (1955) can be used to find this mapping efficiently. Formally:

$$\mathrm{ACC}\big(\mathbf{y}, \mathbf{c}\big) = \max_{g} \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\{\, y_i = g\big(c_i\big) \big\}, \tag{1}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function. The maximization is over all possible bijections g from the set of predicted labels to the set of ground-truth labels.

Normalized Mutual Information (NMI) Normalized Mutual Information quantifies the similarity between clustering results and true class labels, while correcting for differences in their entropies:

$$\mathrm{NMI}\big(\mathbf{y}, \mathbf{c}\big) = \frac{I\big(\mathbf{y}; \mathbf{c}\big)}{\frac{1}{2}\big[\, H\big(\mathbf{y}\big) + H\big(\mathbf{c}\big)\big]}, \tag{2}$$

where $H(\cdot)$ denotes the entropy and $I(\cdot;\cdot)$ the mutual information. By normalizing with the average entropy of the label vectors, NMI is constrained to lie in $[0,1]$.

Adjusted Rand Index (ARI) The Rand Index (RI) measures the fraction of correctly paired samples among all possible pairs. Let TP and TN be the number of true-positive and true-negative pairs, respectively, among the $\binom{n}{2}$ possible sample pairs. Then the Rand Index is

$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{\binom{n}{2}}. \tag{3}$$

However, RI can be artificially inflated by chance alignments when the number of clusters is large. The *Adjusted Rand Index* (ARI) corrects for this effect by normalizing against the expected value of RI, yielding:

$$\mathrm{ARI}\big(\mathbf{y}, \mathbf{c}\big) = \frac{\mathrm{RI} - \mathbb{E}[\mathrm{RI}]}{\max(\mathrm{RI}) - \mathbb{E}[\mathrm{RI}]}, \tag{4}$$

where ARI ranges in $[-1, 1]$. A value of 1 indicates perfect agreement, 0 agreement expected by random chance, and -1 perfect disagreement.

Quality of Pseudo-Labels (NMI Match) Let $S_\tau = \{i \mid \max_c (q_i)_c \geq \tau\}$ denote the indices of confident samples from the unlabeled dataset $\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where each sample's latent representation is $\mathbf{z}_i = g_\phi(\mathbf{x}_i)$ and the clustering head produces probabilities $\mathbf{q}_i = h_\theta(\mathbf{z}_i)$. The pseudo-label for sample i is defined as

$$\tilde{y}_i = \arg\max_c (q_i)_c.$$

For evaluation (using ground-truth labels y_i), the quality of the confident pseudo-labels is measured by

$$\mathrm{NMI\_Match} = \mathrm{NMI}\Big(\{\tilde{y}_i\}_{i \in S_\tau}, \{y_i\}_{i \in S_\tau}\Big), \tag{5}$$

where $\mathrm{NMI}(\cdot,\cdot)$ denotes the normalized mutual information. A higher $\mathrm{NMI\_Match}$ indicates that the pseudo-labels closely reflect the true classes, implying lower label noise.

Cluster Label Reassignment (CL Change) Let V be a fixed validation set. For each $\mathbf{x}_i \in V$, denote the network's cluster assignment at epoch t by

$$\hat{y}_i^t = \arg\max_c \big(f_\eta(\mathbf{x}_i)\big)_c,$$

where $f_\eta(\mathbf{x}_i) = h_\theta(g_\phi(\mathbf{x}_i))$. Define

$$Q^t = \{\hat{y}_i^t \mid \mathbf{x}_i \in V\},$$

as the set of assignments at epoch t. The Cluster Label Reassignment metric is then defined as

$$\mathrm{CL\_Change} = (1 - NMI(Q^t, Q^{t-1})) \times 100 \tag{6}$$

This metric quantifies the percentage change in cluster assignments on the validation set between consecutive epochs. Lower values indicate more stable clustering evolution over time.

E ReSL All Metrics

Table 4: Clustering accuracy of ReSL variants against the standard self-labeling procedure. The best result in each column is in **bold**, and the second best is underlined.

Experiment	STL-10		CIFAR-10		CIFAR-20		CIFAR-100	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
ReSL$_{SP}$	76.75	77.12	88.16	**90.79**	48.60	49.24	31.56	32.19
ReSL$_{SP*}$	76.91	77.21	87.87	88.16	48.64	49.47	31.35	33.02
ReSL$_{HT}$	**77.80**	**78.11**	**88.21**	88.85	**49.13**	**50.68**	**34.89**	**35.6**
SCAN+SL	75.78	76.85	87.57	87.92	48.02	48.83	31.68	32.51

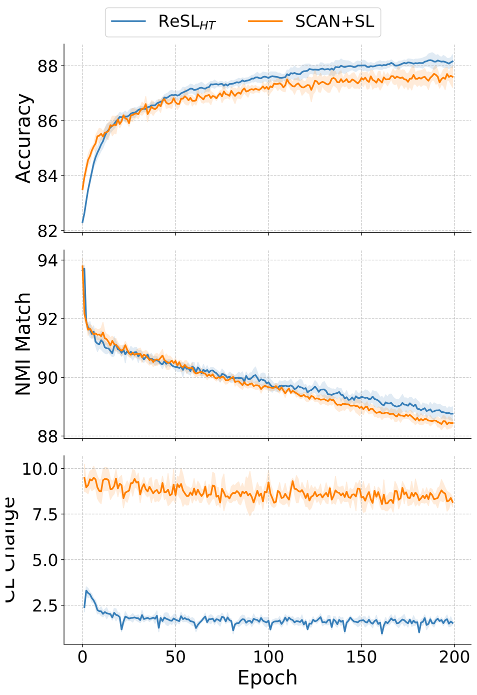
Table 5: ARI of ReSL variants against the standard self-labeling procedure. The best result in each column is in **bold**, and the second best is underlined.

Experiment	STL-10		CIFAR-10		CIFAR-20		CIFAR-100	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
ReSL$_{SP}$	60.81	61.86	77.31	**81.33**	32.76	33.12	21.88	22.44
ReSL$_{SP*}$	60.81	61.86	77.00	78.46	32.91	34.25	22.33	23.30
ReSL$_{HT}$	**62.23**	**62.47**	**76.82**	77.33	**33.72**	**35.29**	**24.17**	**24.59**
SCAN+SL	59.90	61.40	75.75	76.20	32.83	33.82	22.35	23.11

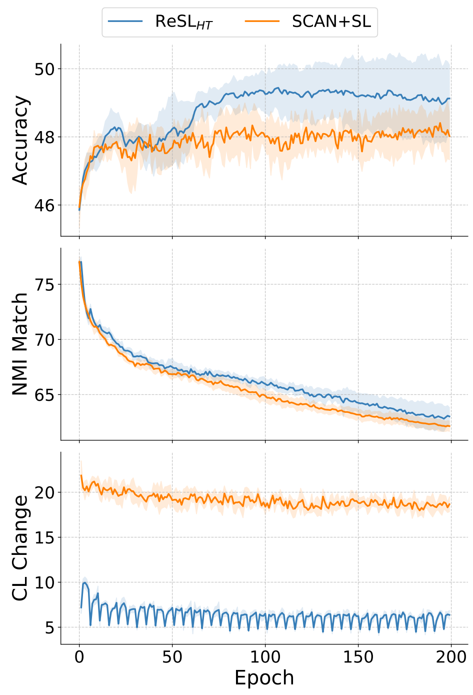
Table 6: NMI of ReSL variants against the standard self-labeling procedure. The best result in each column is in **bold**, and the second best is underlined.

Experiment	STL-10		CIFAR-10		CIFAR-20		CIFAR-100	
	Avg	Max	Avg	Max	Avg	Max	Avg	Max
ReSL$_{SP}$	67.01	67.83	79.28	**81.76**	48.03	48.42	53.72	54.06
ReSL$_{SP*}$	67.01	67.83	79.39	80.34	48.22	49.04	53.63	54.26
ReSL$_{HT}$	**68.00**	**68.18**	**79.71**	80.08	**49.13**	**50.20**	**55.75**	**55.84**
SCAN+SL	66.34	67.51	78.66	78.98	48.15	48.39	53.42	54.22

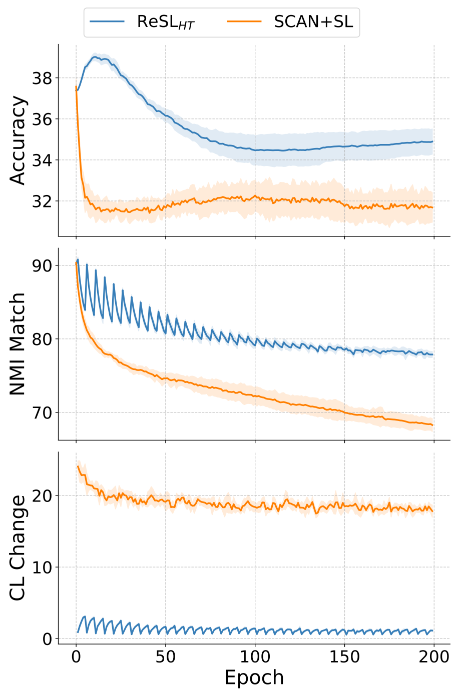
F ReSL HT Results



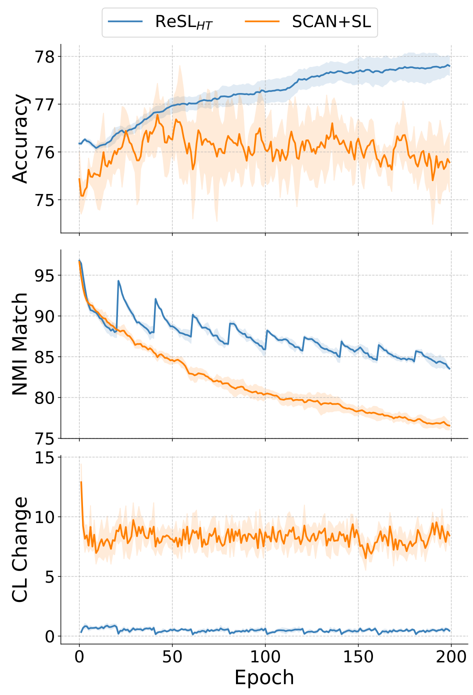
(a) CIFAR10



(b) CIFAR20



(c) CIFAR100

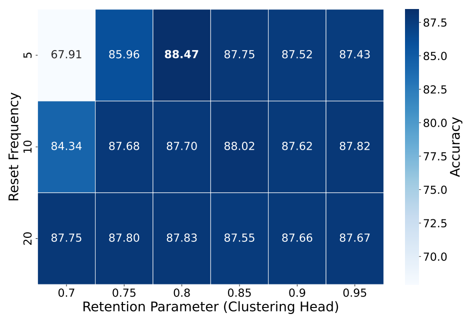


(d) STL10

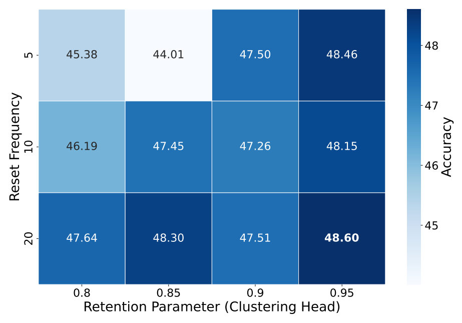
Figure 4: Comparison of ReSL$_{HT}$ and SCAN+SL analyzing accuracy, pseudo-label quality, and cluster label changes across multiple datasets.

G Hyperparameter sensitivity

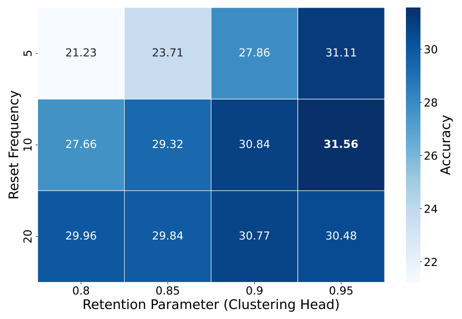
G.1 ReSL with Soft Resets



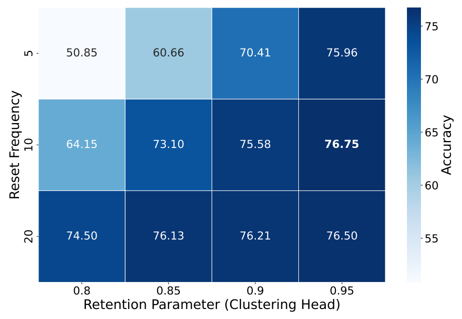
(a) CIFAR10



(b) CIFAR20



(c) CIFAR100

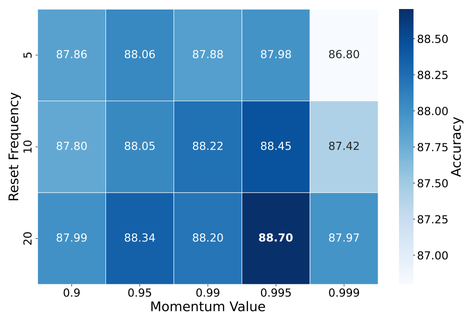


(d) STL10

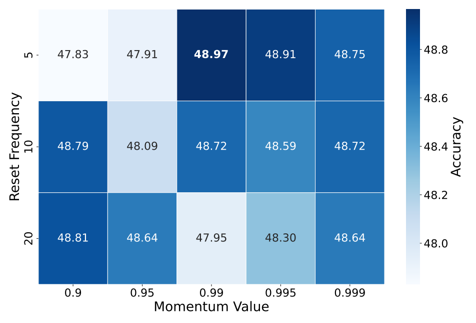
Figure 5: Heatmaps for ReSL$_{SP}$ showing accuracy variations with different reset frequencies and retention parameters across multiple datasets.

The performance of the ReSL$_{SP}$ reset strategy depends on two key hyperparameters: the retention parameter (α) and the reset frequency. The retention parameter controls the proportion of previous weights retained during resets, while the reset frequency determines how often resets occur. Accuracy generally improves as α increases, indicating that stronger retention mitigates the disruptive effects of resets. However, the optimal α varies based on dataset. CIFAR-10 achieves peak accuracy at $\alpha = 0.8$, while CIFAR-20 and CIFAR-100 perform best at $\alpha = 0.95$, emphasizing the importance of preserving learned representations in more complex datasets. STL-10 exhibits a gradual improvement with higher α, suggesting that excessive resets degrade feature stability.

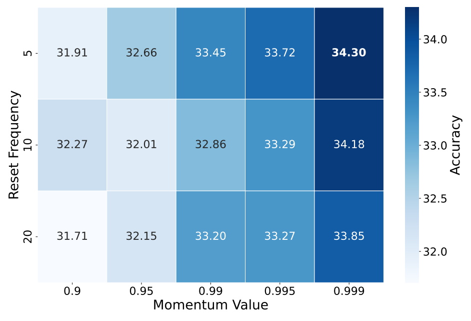
G.2 RESL WITH HARE & TORTOISE NETWORKS



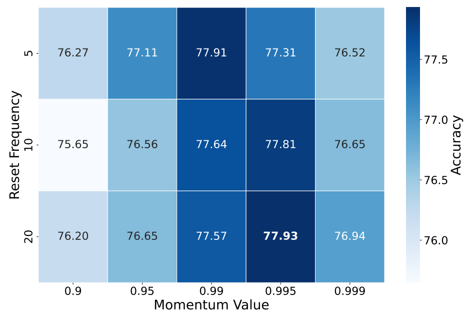
(a) CIFAR10



(b) CIFAR20



(c) CIFAR100

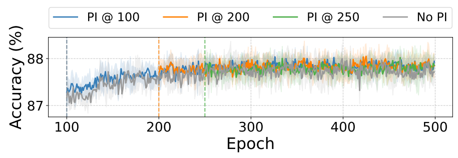


(d) STL10

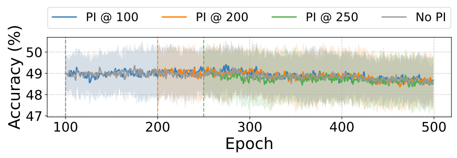
Figure 6: Heatmaps for ReSL$_{HT}$ analyzing the impact of momentum and reset frequency on accuracy across multiple datasets.

Figure 6 illustrates the accuracy variations for different configurations of ReSL$_{HT}$ across multiple datasets. A momentum value of 0.995 with a reset frequency of 20 consistently outperforms SCAN+SL across all datasets, demonstrating the effectiveness of this configuration in stabilizing updates and enhancing clustering performance. CIFAR-10, CIFAR-20, CIFAR-100, and STL-10 all exhibit improved accuracy under this setting.

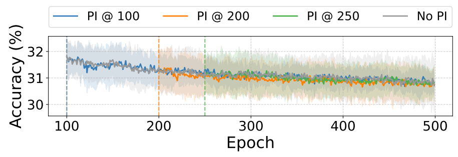
H PLASTICITY INJECTION



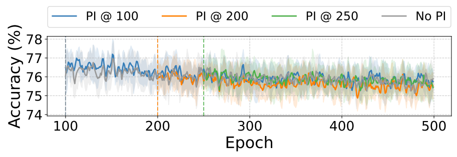
(a) CIFAR10



(b) CIFAR20



(c) CIFAR100



(d) STL10

Figure 7: Plasticity injection experiment across multiple datasets.

All models, including the baseline without injection, demonstrated similar accuracy levels, with only slight variations observed immediately after the injection, suggesting no loss of trainability.