universität wien

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Investigating Plasticity Loss During the Self-Labeling Stage in Deep Clustering

verfasst von | submitted by

Andrii Shkabrii BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 066 921
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Informatik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Inform.Univ. Dr. Claudia Plant
Mitbetreut von | Co-Supervisor:	Dipl.-Ing. Dr.techn. Lukas Miklautz BSc

Acknowledgements

This work was made possible thanks to the continuous support and guidance of my mentors, Timo Klein and Lukas Miklautz. I would like to extend my heartfelt gratitude to Lukas for years of collaboration and encouragement. Special thanks to Timo for always finding the right words of support and emphasizing the importance of maintaining a work-life balance. I am also deeply grateful to Sebastian Tschiatschek for introducing me to this team and for his guidance during my initial steps in my academic journey. Additionally, I thank Claudia Plant, who found a place for me in the team, supported me during my master's program, and offered valuable advice for my future academic career.

I am especially grateful to my family for their unwavering love, encouragement, and support throughout this journey.

Finally, I would like to acknowledge the financial support of the Republic of Austria, Bundesministerium für Bildung, Wissenschaft und Forschung (BMBWF), and the OeAD-GmbH — Agency for Education and Internationalisation, through the Ernst Mach Scholarship (EM UKR/Batch I).

Abstract

Clustering algorithms organize data points into meaningful groups based on their similarity. Deep clustering builds on this foundation by leveraging deep neural networks to learn richer and more representative data embeddings to improve the quality of cluster assignments. To further improve cluster performance practitioners often fine-tune networks through self-labeling, a separate training stage after deep clustering. However, the self-labeling procedure relies on training with noisy, self-constructed labels from nearest neighbors. To mitigate the influence of noise, the authors propose a confidence threshold. This introduces another challenge during training — the increasing size of the training data during the self-labeling stage. In practice, this scenario may encounter generalization difficulties.

In this thesis, we draw inspiration from recent works on neural network plasticity and apply weight-reset techniques during the self-labeling stage. We demonstrate that employing weight-reset techniques during the self-labeling process in the SCAN algorithm increases performance and improves generalizability. Our findings address the limitations of self-labeling and highlight the importance of reliable, algorithm-agnostic techniques and provide a foundation for future research in developing more robust self-labeling approaches.

Kurzfassung

Beim Clustering geht es darum, Datenpunkte auf der Grundlage von Ähnlichkeiten in sinnvolle Gruppen zu unterteilen. Deep Clustering baut auf dieser Grundlage auf und nutzt tiefe neuronale Netze, um umfassendere und repräsentativere Dateneinbettungen zu erlernen. Um die Qualität der Cluster-Zuordnungen zu verbessern, werden Netzwerke häufig durch Self-Labeling feinabgestimmt. Das Self-Labeling-Verfahren basiert jedoch auf dem Training mit verrauschten, selbst erstellten Labels anhand der nächsten Nachbarn. Um den Einfluss des Rauschens zu reduzieren, schlagen die Autoren eine Konfidenzschwelle vor. Dies stellt eine weitere Herausforderung während des Trainings dar — die zunehmende Größe der Trainingsdaten während der Self-Labeling-Phase. In der Praxis kann dieses Szenario zu Generalisierungsproblemen führen.

In dieser Arbeit lassen wir uns von neueren Forschungen zur Plastizität von neuronalen Netzwerken inspirieren und wenden Techniken zum Zurücksetzen von Gewichten während des Self-Labeling-Prozesses an. Wir zeigen, dass der SCAN-Algorithmus durch den Einsatz von Gewichtsrücksetzungstechniken während des Self-Labelings seine Leistung steigert und seine Generalisierungsfähigkeit verbessert. Unsere Ergebnisse verdeutlichen die Grenzen des Self-Labelings, unterstreichen die Bedeutung zuverlässiger, algorithmusagnostischer Techniken und bieten eine Grundlage für die Entwicklung robusterer Self-Labeling-Methoden.

Contents

List of Tables

List of Figures

List of Algorithms

Listings

1. Introduction

Decades of active 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 [M+67], Gaussian mixture models [BN06], and spectral clustering [VL07]. These methods effectively uncover data patterns and have been widely applied in many domains. Yet, when applied to high-dimensional data, these methods face challenges due to the curse of dimensionality and the inefficiency of similarity metrics. An intuitive solution is manual feature engineering and dimensionality reduction. However, these approaches have limited representational capability and can be cumbersome to implement. In contrast, neural networks learn feature representations directly from high-dimensional data by leveraging deep learning and representation learning techniques [BCV13, AFG+21]. The network can be trained without any annotations by solving secondary tasks, known as pretext tasks, such as compressing and then reconstructing the input, predicting noise, colorizing the input, and many more. Several end-to-end learning deep clustering (DC) approaches were proposed [VGVG+20, Qia23, ZWC+21], which simultaneously learn feature representations and cluster assignments.

These end-to-end approaches can further benefit from algorithm-agnostic techniques designed to enhance clustering performance. One such technique is self-labeling [VGVG+20], which significantly increases the performance of the state-of-the-art deep clustering algorithms. Self-labeling fine-tunes the network by training on a subset of pseudo-labels obtained from previous clustering assignments. This subset is defined by a confidence threshold, which represents the minimum confidence the network must have in its assignment for a sample to be included in the training set. Throughout this process, the network becomes increasingly confident in its assignments, gradually adding new confident samples to the training set, as illustrated in Figure 1.1.

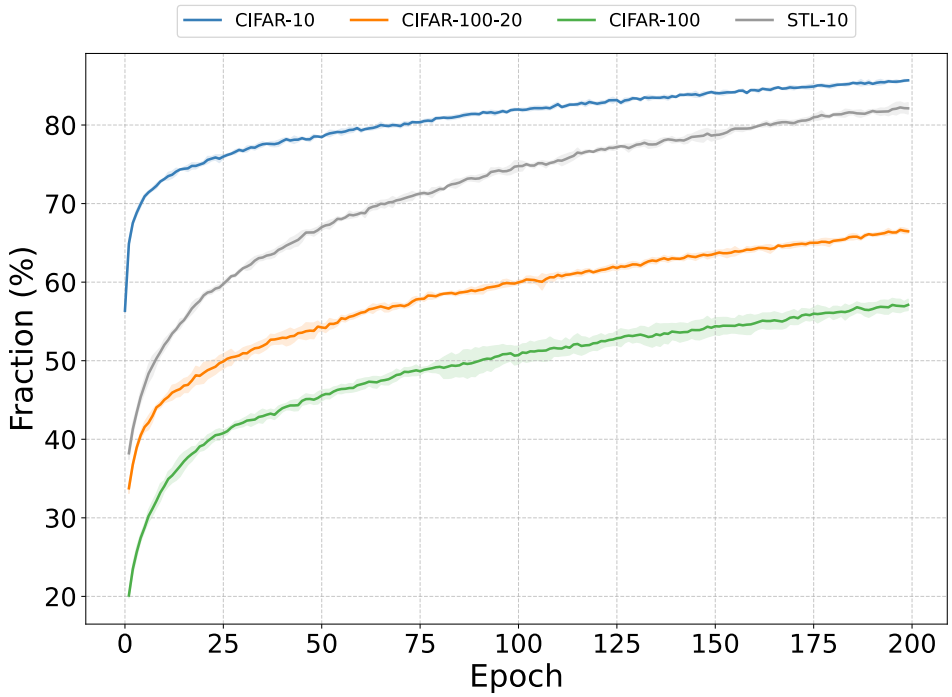


Figure 1.1.: Fraction of confident samples in batch during self-labeling on various datasets

The sequential arrival of new confident samples introduces non-stationarity in the input, while the subsequent optimization introduces non-stationarity in the targets, both as a consequence of self-labeling. Inevitably, the training set will include samples assigned to the wrong cluster, leading to training on noisy targets. A similar challenge has been studied in the plasticity literature, which refers to the network's ability to fit new targets. The non-stationarity challenge is considered among the key driving factors of plasticity loss [KMS+24, AA20, LCK+24].

This thesis investigates the following hypothesis: Does plasticity loss occur during the self-labeling stage of deep clustering? We conduct experiments using the deep clustering method SCAN [VGVG+20] (the first method to introduce self-labeling) and use an established plasticity loss diagnostic tool — plasticity injection [NOO+24]. We show that using plasticity injection is not enough to escape a performance plateau. We further extend SCAN with well-established weight-reset techniques. Resetting the weights of the network is one of the key mechanisms behind promising approaches [LCK+24, AA20] from the plasticity literature to reduce the negative impact of non-stationarity and noisy targets on the network's performance. Our quantitative experiments using soft resets [AA20] and Hare & Tortoise networks [LCK+24] improve clustering performance on STL-10 [CNL11] and CIFAR-10/20/100 [KH+09] datasets.

In summary, the main contributions of this thesis are as follows.

- Using plasticity injection, we empirically demonstrate that trainability might not be an issue during the self-labeling stage.
- We experiment with two approaches: periodically applying soft resets and the Hare and Tortoise approach.
- We propose a novel scheduling strategy for weight resets, which is more reliable than regular soft resets when applied during the self-labeling stage.

The rest of this thesis is organized as follows: Chapter 2 introduces the problem setup. In Chapter 3, we provide the background and review related work on deep clustering and the loss of plasticity. Chapter 4 details the datasets, evaluation metrics, and methods used to detect and mitigate plasticity loss, including weight resets and scheduling. The experiments are presented in Chapter 5. Chapter 6 discusses the implications of our findings and provides an outlook on future work. Finally, Chapter 7 concludes the thesis.

2. Problem Setup

Consider a dataset $\mathcal{D} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ comprising n instances, where each $\mathbf{x}_i \in \mathbb{R}^D$ represents a sample in a D-dimensional feature space. Let $\mathcal{C} = \{1, 2, ..., C\}$, our objective is to partition this dataset into C distinct clusters. To achieve this, each data point $\mathbf{x}_i$ is mapped to a lower-dimensional latent representation $\mathbf{z}_i \in \mathbb{R}^d$ (with $d \ll D$) using a feature extractor network g_ϕ. The network g_ϕ, parameterized by learnable weights ϕ, is trained self-supervised on a pretext task.

A clustering head h_θ, parameterized by learnable weights θ, is then applied to map the latent representation $\mathbf{z}_i$ to a cluster assignment vector $\mathbf{q}_i = h_\theta(\mathbf{z}_i)$. Here, $\mathbf{q}_i \in \mathbb{R}^C$ is a probability distribution over the C clusters, where each element of $\mathbf{q}_i$ corresponds to the probability of $\mathbf{x}_i$ belonging to a specific cluster. The final cluster assignment for $\mathbf{x}_i$ is determined by selecting the cluster with the highest probability: $c_i = \arg\max_c(\mathbf{q}_i)_c$.

The clustering function $f_\eta = h_\theta \circ g_\phi$ is composed of the feature extractor network g_ϕ and the clustering head h_θ, which together map input data points to their corresponding cluster assignments.

3. Background and Related Work

3.1. Deep Clustering

Recent advancements in deep clustering 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 [CKNH20] or MoCo [CFGH20]. Jointly optimizing representation learning and clustering objectives has enabled deep clustering methods. Recent deep clustering methods [VGVG$^+$20, Qia23, ZWC$^+$21] have shown results close to supervised methods on widely used image dataset benchmarks. These studies share a practice of fine-tuning clustering networks through self-labeling to enhance the quality of cluster assignments. In the following subsections, we focus on the SCAN [VGVG$^+$20] method, the pioneering approach to self-labeling in deep clustering. We also revisit the SimCLR framework, the backbone SCAN used for the datasets we are considering.

3.1.1. SimCLR

Similar to previous contrastive learning algorithms [DSRB14, WXYL18, JHV19], SimCLR 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 [vdOLV18] 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.

3.1.2. SCAN

Semantic clustering by adopting nearest neighbors (SCAN) 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). 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 by using the assignments from the previous iteration as pseudo-labels. We provide more details about this technique in Section 3.1.3.

3.1.3. Self-labeling

Self-labeling works by iteratively refining cluster assignments. The network's own predictions are used to generate pseudo-labels, which are then used as targets for further training. The authors of self-labeling propose the following setting.

Given that $f_\eta = h_\theta \circ g_\phi$ is our clustering function, composed of the feature extractor g_ϕ and the trainable clustering head h_θ. Denote by

$$\mathbf{q}_i = f_\eta(\mathbf{x}_i) \;=\; h_\theta\big(g_\phi(\mathbf{x}_i)\big) \;\in\; \mathbb{R}^C$$

the cluster assignment probabilities for sample $\mathbf{x}_i$. Let τ be a confidence threshold, and define the set of *confident* samples as

$$S_\tau \;=\; \Big\{ i \;\Big|\; \max_c \mathbf{q}_i(c) \;\geq\; \tau \Big\}.$$

For each $i \in S_\tau$, we form a pseudo-label $\tilde{y}_i = \arg\max_c \mathbf{q}_i(c)$. Let $\hat{\mathbf{x}}_i = t(\mathbf{x}_i)$ represent an augmentation of $\mathbf{x}_i$, under an augmentation function $t(\cdot)$.

The training loss is taken to be the standard cross-entropy between the pseudo-labels $\tilde{y}_i$ and the predicted distributions $f_\eta(\hat{\mathbf{x}}_i)$. In one-hot notation, let $\tilde{\mathbf{y}}_i \in \{0,1\}^C$ be the one-hot vector encoding of $\tilde{y}_i$. Then the self-labeling loss $\mathcal{L}_{\text{SL}}$ can be written as

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

By training on strongly-augmented inputs $\hat{\mathbf{x}}_i$, the clustering head becomes more robust to perturbations, reducing overfitting. Algorithm 1 summarizes the steps.

Algorithm 1 Self-labeling

Input: Dataset $\mathcal{D}$, Clusters $\mathcal{C}$, Neural Net $f_\eta = h_\theta \circ g_\phi$, Maximum Epochs E
Output: $f_\eta(\mathcal{D})$

1: **for** epoch $\leftarrow 1$ **to** E **do**
2: $\quad Y \leftarrow (f_\eta(\mathcal{D}) >$ threshold$)$
3: $\quad$ Update h_θ with cross-entropy loss $\mathcal{L}_{\text{SL}}(\theta)$
4: **end for**
5: **return** $f_\eta(\mathcal{D})$ ▷ $\mathcal{D}$ is divided into $\mathcal{C}$ clusters

The clustering head itself can be implemented as a simple linear layer mapping features to cluster assignments, or as a small multilayer perceptron that transforms features into cluster assignment probabilities.

The simplicity and effectiveness of the approach have made self-labeling a popular technique for improving deep clustering performance.

3.2. Loss of 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. Once plasticity is lost, the ability to learn diminishes [LZN+23]. The term plasticity has gained broader attention in the fields of deep reinforcement learning (RL) [KMS+24] and continual learning [EM24]. Various methods have been proposed to address plasticity loss. In the following subsections, we review the related literature on detecting loss of trainability and exploring weight manipulation strategies to mitigate the loss of plasticity. Our analysis is restricted to methods that are directly applicable to self-labeling.

3.2.1. Plasticity Injection

Plasticity injection [NOO+24] is an established method in deep RL for detecting plasticity loss. It serves as a diagnostic tool because of its ability to leave predictions unaffected at

the point of intervention and preserve the number of trainable parameters throughout the course of training. The method is based on the idea that the loss of plasticity is concentrated in the head of a neural network, where task-specific outputs are generated.

The intervention is applied as follows. At a given step T, the final layer is reinitialized with fresh parameters. Let θ denote the current network parameters, θ'_1 a set of freshly initialized parameters, and θ'_2 a frozen copy of θ'_1. If h_θ denotes the function representing the network's head, parameterized by θ, then the output function $h_\theta(x)$ is defined as:

$$h_\theta(x) = h_\theta(x) + h_{\theta'_1}(x) - h_{\theta'_2}(x)$$

Consequently, the trainability of the network is improved by freshly initialized parameters θ'_1. Hence, if injection leads to improved performance, it suggests that the network was losing its plasticity. However, if the network's performance remains unchanged, it indicates that the issue might not stem from network's trainability or that the plasticity injection is not well-suited to the current task. In such cases, further diagnostics may be necessary to address other factors that could restrict network learning capabilities.

3.2.2. Soft Resets

Soft resets, known as Shrink and Perturb [AA20], allow the neural network to be efficiently updated without sacrificing generalization performance. The authors study the scenario where learning starts with a pre-trained model, instead of having to train a freshly initialized model. Their results show that warm-starting does not damage training accuracy, but hurts generalization in neural networks. Instead of retraining from scratch, the authors propose performing a reset using a linear combination of the current parameters θ and random parameters ϕ from the initial parameter distribution:

$$\theta \leftarrow \alpha\theta + (1-\alpha)\theta', \theta' \sim initializer$$

The retention parameter α is in the range $(0, 1)$ and allows control over how much of the current network parameters should be retained. The special case of $\alpha = 1$ results in no change in the network's parameters, while $\alpha = 0$ leads to a full reset, where the current parameters are replaced with freshly initialized parameters from the initial parameter distribution.

3.2.3. Hare and Tortoise Networks

Inspired by the dynamics observed in neural processes linking the hippocampus and neocortex, the Hare and Tortoise networks [LCK+24] work analogously by balancing rapid adaptation with stable learning. The authors propose decoupling plasticity into trainability (the network's ability to train) and generalizability (the network's ability to generalize). The learning is done as follows.

Given an input x and its output y, the hare network's parameters, θ_h, are updated using:

$$\theta_h \leftarrow \theta_h - \alpha \nabla_{\theta h} \mathcal{L}(h(x; \theta_h), y)$$

where α denotes the learning rate, and L is the loss function.

The parameters of the tortoise network, θ_t are updated using an exponential moving average based on the hare network's parameters:

$$\theta_t \leftarrow \mu \theta_t + (1 - \mu)\theta_h$$

where μ denotes the momentum.

Throughout the training, the parameters of hare network are periodically reset to the tortoise network's parameters:

$$\theta_h \leftarrow \theta_t$$

The update rate is defined by the parameter μ. Larger values of μ enable the tortoise network to progressively absorb information from the hare network.

3.3. Resets in Deep Clustering

In BRB [MKS+24], the authors introduce the concept of a "reclustering barrier", which arises when clustering algorithms fail to improve late in training despite frequent reclustering. They show that combining reclustering with soft weight resets enables centroid-based deep clustering methods to escape the performance plateaus. Although the deep clustering experiments in the BRB study did not examine the effects of plasticity loss, weight resets were found to be effective in preventing early over-commitment to suboptimal clustering assignments.

4. Methodology

This section summarizes the methodology behind the experiments. We justify our choice of benchmark datasets in Section 4.1 and introduce the evaluation metrics in Section 4.2. In Section 4.3, we outline the experimental setup, including the chosen hyperparameters, architecture, and design decisions relevant to all experiments. In the remaining sections, we describe the necessary prerequisites and the modifications made to the self-labeling stage in each experiment.

4.1. Datasets

Similar to recent deep clustering methods [Qia23, VGVG$^+$20, ZWC$^+$21], we evaluate the clustering performance on CIFAR-10, CIFAR-20, CIFAR-100 [KH$^+$09], and STL-10 [CNL11], which are established image dataset benchmarks.

- **CIFAR-10/20/100** [KH$^+$09]: 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. Visualizations of sample images are provided in Figures A.1 and A.2.

- **STL-10** [CNL11]: 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 [VGVG$^+$20], we do not utilize these unlabeled samples. See Figure A.3 for a visualization.

4.2. Evaluation Metrics

Similar to the benchmarking pipelines of recently proposed deep clustering algorithms [VGVG$^+$20, ZWC$^+$21, Qia23], and following recommendations from deep clustering surveys [ZXZ$^+$24, LLL$^+$24, HWWZ24], we evaluate performance using *clustering accuracy (ACC)* [YXN$^+$10], *Adjusted Rand Index (ARI)* [HA85], and *Normalized Mutual Information (NMI)* [Kva87]. We conduct experiments across five random seeds and report averages with standard deviations for all metrics. All metrics are reported on the validation dataset.

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 [Kuh55] can be used to find this mapping efficiently. Formally:

$$\mathrm{ACC}\big(\mathbf{y}, \mathbf{c}\big) = \max_g \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}\big\{ y_i = g\big(c_i\big) \big\}, \tag{4.1}$$

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:

$$\mathrm{NMI}\big(\mathbf{y}, \mathbf{c}\big) = \frac{I\big(\mathbf{y}; \mathbf{c}\big)}{\frac{1}{2}\big[\, H\big(\mathbf{y}\big) + H\big(\mathbf{c}\big)\big]}, \tag{4.2}$$

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

$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{\binom{n}{2}}. \tag{4.3}$$

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:

$$\mathrm{ARI}\big(\mathbf{y}, \mathbf{c}\big) = \frac{\mathrm{RI} - \mathbb{E}[\mathrm{RI}]}{\max(\mathrm{RI}) - \mathbb{E}[\mathrm{RI}]}, \tag{4.4}$$

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

4.3. Experimental Setup

To maintain focus on exploring plasticity loss, we reuse the same experimental setup as SCAN for each dataset. The relevant parameters for self-labeling are summarized in Table 4.1.

Table 4.1.: Self-labeling Config

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

Self-labeling training begins with pretrained models obtained after the pretext and clustering stages of SCAN, using the original SCAN code available at `https://github.com/wvangansbeke/Unsupervised-Classification`. Unless stated otherwise in the corresponding experiment sections, the hyperparameters listed in Table 4.1 are consistently applied across all our self-labeling experiments and correspond to those from the original SCAN codebase.

The model's architecture follows the setup of SCAN [VGVG+20]. We use a Resnet-18 [HZRS16] encoder consisting of four residual blocks and a clustering head consisting of a single linear layer. Following BRB [MKS+24], we consider only the last Resnet-18 layer and the clustering head for resets.

4.4. Soft Resets During Self-Labeling

We implement soft resets by selectively resetting only specific layers of the network. The selective resetting allows the model to preserve the learned representations in the earlier layers of the network, while allowing the final layers to adapt periodically through the reset mechanism. Algorithm 2 summarizes our implementation:

Algorithm 2 Self-labeling with Soft Resets

Input: Dataset $\mathcal{D}$, Clusters $\mathcal{C}$, Neural Net $f_\eta = h_\theta \circ g_\phi$, Maximum Epochs E, Reset Frequency R, Layers to Reset $\mathcal{S}$, Retention Parameters $\{\alpha_l\}_{l \in \mathcal{S}}$
Output: $f_\eta(\mathcal{D})$

for epoch $\leftarrow 1$ **to** E **do**
  $Y \leftarrow (f_\eta(\mathcal{D}) >$ threshold)
  Update h_θ with cross-entropy loss $\mathcal{L}_{\text{SL}}(\theta)$
  if epoch mod R $= 0$ **then**
    for each layer l in $\mathcal{S}$ **do**
      Apply soft reset to parameters ϕ_l with retention parameter α_l ▷ As defined in Section 3.2.2
    end for
  end if
end for
return $f_\eta(\mathcal{D})$ ▷ $\mathcal{D}$ is divided into $\mathcal{C}$ clusters

Here, ϕ_l represents the parameters of layer l. The retention parameter $\alpha_l \in (0, 1)$ controls the extent of the reset, with $\alpha_l = 1$ retaining the current parameters entirely, and $\alpha_l = 0$ performing a full reset to the freshly sampled parameters.

4.5. Soft Resets with Scheduling During Self-Labeling

We focus on the scheduling logic for the retention parameters $\{\alpha_l\}_{l \in \mathcal{S}}$, which determine the proportion of the current model parameters retained versus reset to the fresh weights. Algorithm 3 outlines the implementation

Algorithm 3 Self-labeling with Soft Resets and Softening Retention Parameters

Input: Dataset $\mathcal{D}$, Clusters $\mathcal{C}$, Neural Net $f_\eta = h_\theta \circ g_\phi$, Maximum Epochs E, Reset Frequency R, Layers to Reset $\mathcal{S}$, Initial Retention Parameters $\{\alpha_l^0\}_{l\in\mathcal{S}}$
Output: $f_\eta(\mathcal{D})$

Initialize $\{\alpha_l^0\}_{l\in\mathcal{S}}$ as initial retention parameters
for epoch $\leftarrow 1$ **to** E **do**
    $Y \leftarrow (f_\eta(\mathcal{D}) >$ threshold$)$
    Update h_θ with cross-entropy loss $\mathcal{L}_{\text{SL}}(\theta)$

    if epoch mod R $= 0$ **then**
        for each layer l in $\mathcal{S}$ **do**
            Apply soft reset to parameters ϕ_l with retention parameter α_l ▷ As defined in Section 3.2.2
        end for
    end if

    $t \leftarrow$ epoch$/E$ ▷ Compute soften factor $t \in [0, 1]$
    for each layer l in $\mathcal{S}$ **do**
        $\alpha_l \leftarrow \alpha_l^0 + (1.0 - \alpha_l^0) \cdot t$ ▷ Linearly increase retention parameter
    end for
end for
return $f_\eta(\mathcal{D})$ ▷ $\mathcal{D}$ is divided into $\mathcal{C}$ clusters

At the start of training, the retention parameters $\{\alpha_l^0\}_{l\in\mathcal{S}}$ are initialized with low values, allowing for a more substantial reset. As training progresses, the retention parameters are gradually increased in a linear fashion, governed by the soften factor $t = \text{epoch}/E$. This factor linearly interpolates the retention parameters from their initial values to a maximum value of 1.0 by the final epoch. Soft resets are applied at intervals defined by the reset frequency R, with the retention parameters updated after each reset.

This scheduling approach ensures that the network undergoes larger resets during the initial stages of training, potentially encouraging exploration and reducing overconfidence. As training progresses and the model stabilizes, the resets become less aggressive, preserving the learned representations while still injecting minimal perturbations.

4.6. Hare and Tortoise Networks with Self-Labeling

To extend the self-labeling procedure with Hare and Tortoise networks, we initialize both networks with the weights obtained from the best clustering model based on the lowest training loss. The tortoise network is used for clustering the data and gradually integrates knowledge by applying an exponential moving average (EMA) to the hare's weights. Algorithm 4 outlines our implementation.

Algorithm 4 Self-labeling with Hare and Tortoise Networks

Input: Dataset $\mathcal{D}$, Clusters $\mathcal{C}$, Hare network $h_{\theta_h} = h_\theta \circ g_\phi$, Tortoise network t_{θ_t}, Maximum Epochs E, Reset Frequency R, Momentum μ
Output: $h_{\theta_h}(\mathcal{D})$

for epoch $\leftarrow 1$ **to** E **do**
  $Y \leftarrow (h_{\theta_h}(\mathcal{D}) > \text{threshold})$
  Update h_θ with cross-entropy loss $\mathcal{L}_{\text{SL}}(\theta)$
  Update t_{θ_t} using $\theta_t \leftarrow \mu\theta_t + (1-\mu)\theta_h$
  if epoch mod R $= 0$ **then**
    Update h_{θ_h} using $\theta_h \leftarrow \theta_t$
  end if
end for
return $h_{\theta_t}(\mathcal{D})$ ▷ $\mathcal{D}$ is divided into $\mathcal{C}$ clusters

Similarly to the methodology of self-labeling with soft resets, we introduce two hyperparameters: reset frequency, which determines how often the hare network is updated with the tortoise network, and momentum value, which determines how much of the tortoise weights are preserved.

By employing this dual-network approach, we aim to enhance the model's ability to adapt quickly to new data through the hare network while simultaneously ensuring that the learned representations are stable and generalizable via the tortoise network.

4.7. Confidence Threshold Scheduling Methodology

We extend self-labeling with soft resets and self-labeling with Hare and Tortoise networks by enabling confidence threshold scheduling. Instead of using a fixed value, we define starting and target values for the confidence. The resulting Algorithm 5 and Algorithm 6 allow scheduling confidence in a decreasing manner (from higher to lower values) and in an increasing direction (from lower to higher values).

Algorithm 5 Self-labeling with Soft Resets and Threshold Scheduling

Input: Dataset $\mathcal{D}$, Clusters $\mathcal{C}$, Neural Net $f_\eta = h_\theta \circ g_\phi$, Maximum Epochs E, Reset Frequency R, Layers to Reset $\mathcal{S}$, Retention Parameters $\{\alpha_l\}_{l\in\mathcal{S}}$, Start Threshold t_{start}, Target Threshold t_{target}

Output: $f_\eta(\mathcal{D})$

Initialize: $\theta \leftarrow \theta_0$, $t \leftarrow t_{\text{start}}$ ▷ Set current threshold to start threshold
Compute: $\Delta \leftarrow \frac{(t_{\text{target}} - t_{\text{start}})}{(R-1)}$ ▷ Threshold increment per reset cycle
for epoch $\leftarrow 1$ **to** E **do**
    $Y \leftarrow (f_\eta(\mathcal{D}) > t)$ ▷ Select confident samples via current threshold t
    Update h_θ using cross-entropy loss $\mathcal{L}_{\text{SL}}(\theta)$
    if epoch mod R $= 0$ **then**
        for each layer l in $\mathcal{S}$ **do**
            Apply soft reset to parameters ϕ_l with retention parameter α_l ▷ As defined in Section 3.2.2
        end for
        Reset threshold: $t \leftarrow t_{\text{start}}$
    else
        Update threshold (by direction):
        if $\Delta > 0$ **then**
            $t \leftarrow \min\big(t_{\text{target}},\ t + \Delta\big)$
        else
            $t \leftarrow \max\big(t_{\text{target}},\ t + \Delta\big)$
        end if
    end if
end for
return $f_\eta(\mathcal{D})$ ▷ $\mathcal{D}$ is divided into $\mathcal{C}$ clusters

In this extension of self-labeling with soft resets, we introduce a confidence threshold schedule defined by t_{start} and t_{target}. The threshold t is initialized at t_{start} and updated each epoch by $\Delta = \frac{(t_{\text{target}} - t_{\text{start}})}{(R-1)}$. After every R epochs, we reset t to t_{start} and partially reset the parameters ϕ_l of selected layers $\mathcal{S}$ using the retention parameters $\{\alpha_l\}_{l\in\mathcal{S}}$. This dynamic threshold influences how samples are masked (via $Y = (f_\eta(\mathcal{D}) > t)$), while soft resets mitigate overfitting by periodically "forgetting" parts of the network state.

Similarly, we extend self-labeling with Hare and Tortoise networks to enable scheduling. Algorithm 6 summarizes our approach.

Algorithm 6 Self-labeling with Hare and Tortoise Networks and Threshold Scheduling

Input: Dataset $\mathcal{D}$, Clusters $\mathcal{C}$, Hare network $h_{\theta_h} = h_\theta \circ g_\phi$, Tortoise network t_{θ_t}, Maximum Epochs E, Reset Frequency R, Momentum μ, Start Threshold t_{start}, Target Threshold t_{target}

Output: $h_{\theta_h}(\mathcal{D})$

Initialize: $t \leftarrow t_{\text{start}}$
Compute: $\Delta \leftarrow \frac{(t_{\text{target}} - t_{\text{start}})}{(R-1)}$
for epoch $\leftarrow 1$ **to** E **do**
    $Y \leftarrow (h_{\theta_h}(\mathcal{D}) > t)$
    Update h_{θ_h} with cross-entropy loss $\mathcal{L}_{\text{SL}}(\theta_h)$
    Update t_{θ_t} using $\theta_t \leftarrow \mu\,\theta_t + (1-\mu)\,\theta_h$
    if epoch mod R $= 0$ **then**
        Update h_{θ_h} using $\theta_h \leftarrow \theta_t$
        Reset threshold: $t \leftarrow t_{\text{start}}$
    else
        Update threshold (by direction):
        if $\Delta > 0$ **then**
            $t \leftarrow \min\big(t_{\text{target}},\ t + \Delta\big)$
        else
            $t \leftarrow \max\big(t_{\text{target}},\ t + \Delta\big)$
        end if
    end if
end for
return $h_{\theta_t}(\mathcal{D})$ ▷ $\mathcal{D}$ is divided into $\mathcal{C}$ clusters

For the Hare and Tortoise approach, we similarly schedule t from t_{start} to t_{target}, but maintain a tortoise network t_{θ_t} that is updated by momentum from the hare network h_{θ_h} each epoch. Every R epochs, the hare network is replaced by the tortoise network parameters ($\theta_h \leftarrow \theta_t$) and t is reset to t_{start}.

5. Experiments

In the following section we summarize the conducted experiments. We start by presenting the baselines in Section 5.1, followed by an experiment involving ground-truth labels in Section 5.2. The results of the plasticity injection experiment are summarized in Section 5.3. Sections 5.4.1 and 5.4.2 present the outcomes of extending the algorithm with soft resets and Hare Tortoise networks, respectively. We experiment with scheduling the confidence threshold in Section 5.5.1 and illustrate the effects of longer training in Section 5.6.1. Finally, Section 5.6 concludes with Table 5.6, summarizing the best experimental results.

5.1. Baselines

To establish a benchmark, we first replicate the results reported in the original SCAN study [VGVG$^+$20]. Table 5.1 summarizes the average clustering accuracy over five runs and its standard deviation across various datasets before and after the self-labeling stage, along with the corresponding percentage improvement.

Table 5.1.: Baseline averages for ACC, NMI, and ARI over five runs, including standard deviations, on various datasets. Results are presented before and after self-labeling, along with the percentage change.

Dataset	**ACC**			**NMI**			**ARI**		
	Before	After	Change (%)	Before	After	Change (%)	Before	After	Change (%)
CIFAR-10	81.97	87.57 ± 0.19	6.84%	71.06	79.03 ± 0.39	11.21%	66.38	76.19 ± 0.58	14.77%
CIFAR-20	44.98	48.46 ± 0.94	7.73%	45.19	47.94 ± 0.40	6.08%	29.01	32.67 ± 0.57	12.61%
CIFAR-100	37.39	31.72 ± 1.36	-15.15%	55.40	53.44 ± 0.45	-3.56%	23.78	22.12 ± 0.52	-6.98%
STL-10	76.14	76.45 ± 0.97	0.41%	65.51	66.14 ± 0.63	0.96%	58.29	59.48 ± 0.59	2.04%

As in the original work, applying self-labeling generally improves performance. However, the degree of improvement varies across datasets. For instance, while CIFAR-10 and CIFAR-20 show accuracy gains of 6.84% and 7.73%, respectively, STL-10 shows only a marginal increase. In contrast, CIFAR-100 experiences a significant decline in performance after self-labeling, suggesting that the effectiveness of self-labeling depends on the dataset and may not always lead to better results.

One possible hypothesis for the decreasing performance during self-labeling is the poor initial clustering performance on CIFAR-100, which could result in a training set containing a higher proportion of incorrect pseudo-labels. To investigate this, we introduce an

additional metric that represents the fraction of pseudo-labels in a batch that align with the ground-truth labels. Figure 5.1 presents the results across all benchmarking datasets.

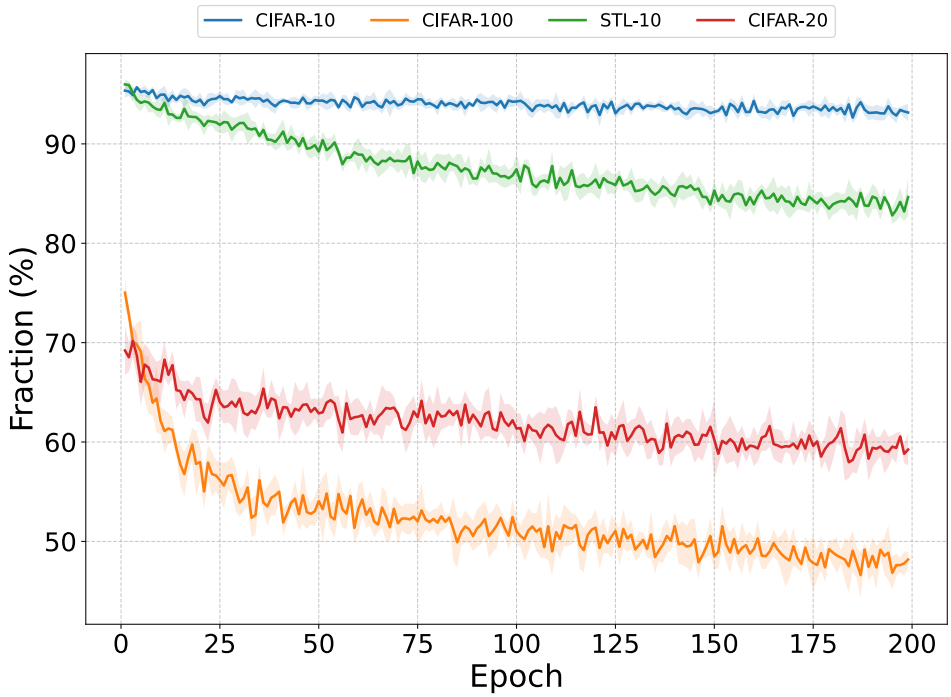


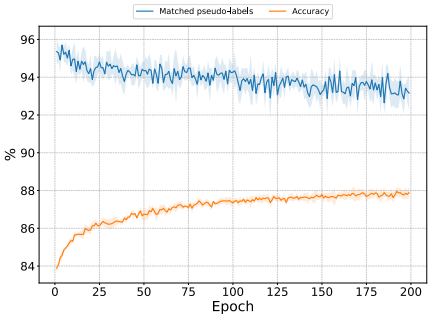
Figure 5.1.: Comparison of Correct Pseudo-Labels in a Batch Across CIFAR-10, CIFAR-20, CIFAR-100, and STL-10 During Self-Labeling

As Figure 5.1 shows, CIFAR-100 indeed contains a higher proportion of incorrect pseudo-labels. CIFAR-10 and STL-10 datasets exhibit a fraction of correctly labeled samples at 95.36% and 95.99%, respectively, whereas CIFAR-100 begins training with 75.03% correct pseudo-labels. Interestingly, CIFAR-20 dataset starts with an even lower initial fraction of 69.21% correct pseudo-labels, yet its performance improves during self-labeling by 7.73%. In contrast, CIFAR-100 experiences a performance drop of 15.15%.

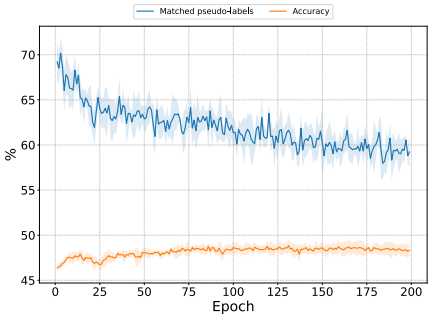
While CIFAR-20 begins with the lowest proportion of correct pseudo-labels, it shows greater improvement through self-labeling compared to other datasets in terms of ACC. In contrast, CIFAR-100 starts with a similarly low fraction but exhibits a trend of performance decline. This divergence between CIFAR-20 and CIFAR-100 indicates that self-labeling is influenced by the specific characteristics of the datasets, and the proportion of correct pseudo-labels alone fails to explain self-labeling performance outcomes.

To further illustrate the relationship between accuracy and the fraction of correct pseudo-labels in a batch, we provide Figure 5.2, which showcases these metrics in separate

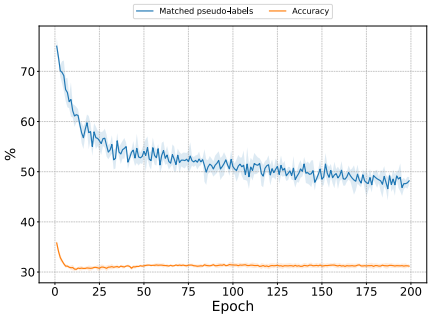
subfigures throughout the self-labeling process for all four benchmarking datasets.



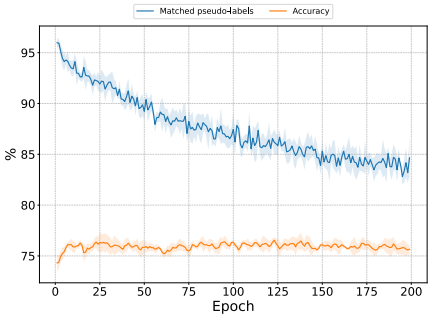
(a) CIFAR10



(b) CIFAR20



(c) CIFAR100



(d) STL10

Figure 5.2.: The relationship between accuracy and the fraction of correct pseudo-labels in a batch per dataset.

Figure 5.2 suggests that the models may be overly relying on pseudo-labels generated during training, which fail to generalize to new data. This is further evidenced by the declining fraction of correct pseudo-labels while accuracy stagnates or improves minimally.

5.2. Self-labeling with Ground Truth

To disentangle the effect of noisy targets from the effectiveness of self-labeling, we perform experiments utilizing ground-truth labels during the self-labeling stage. This allows us to establish target accuracy and investigate whether the model used is capable of achieving higher accuracy. We conduct three experiments:

- **Initialize randomly**: Random initialization, then supervised learning on the full training set using ground truth labels. Training is performed for a total of 100 epochs on STL-10 and 200 epochs on CIFAR datasets.

- **Initialize with SCAN, start supervised**: Initialize with SCAN, then train on the full training set using ground truth labels. Training is performed for a total of 100 epochs on STL-10 and 200 epochs on CIFAR datasets.

- **Initialize with SCAN, start supervised after unsupervised**: Initialize with SCAN, then train using the pseudo-labels for 50 epochs on STL-10 and 100 epochs on CIFAR datasets. Subsequently, training continues on the full training set using ground truth labels for a total of 100 epochs on STL-10 and 200 epochs on CIFAR datasets.

We evaluated the performance of each experimental setup. The results are summarized in Tables 5.2.

Table 5.2.: Self-Labeling with Ground-Truth on CIFAR-10, STL-10, CIFAR-20, and CIFAR-100.

Experiment	CIFAR-10 Acc (±Std)	STL-10 Acc (±Std)	CIFAR-20 Acc (±Std)	CIFAR-100 Acc (±Std)
Unsupervised SCAN	87.57 ± 0.19	76.45 ± 0.97	48.46 ± 0.94	31.72 ± 1.36
Supervised, Random Init	89.60 ± 0.23	66.58 ± 1.20	74.61 ± 0.30	66.44 ± 0.32
Supervised, SCAN Init	94.21 ± 0.07	**90.38 ± 0.21**	82.67 ± 0.14	73.35 ± 0.33
Unsupervised + Supervised, SCAN Init	**94.28 ± 0.07**	89.95 ± 0.20	**83.44 ± 0.17**	**73.84 ± 0.42**

The results indicate that when using ground-truth labels, the model's performance improves drastically. Notably, initializing with SCAN yields significantly higher accuracy compared to random initialization. Moreover, on STL-10 dataset, networks with random initialization cannot achieve the performance levels of those trained with SCAN in unsupervised settings.

However, these results represent an ideal learning scenario and cannot be directly applied to unsupervised clustering tasks due to their inherent nature. We interpret these results as an upper bound on possible self-labeling gains and summarize the maximum possible gains in Table 5.3.

Table 5.3.: Unsupervised (SCAN) vs. Unsupervised + Supervised (SCAN Init), and their change in accuracy.

Dataset	Baseline After Self-labeling	Unsupervised + Supervised SCAN Init	Change (%)
CIFAR-10	87.57 ± 0.19	94.28 ± 0.07	6.71%
STL-10	76.45 ± 0.97	89.95 ± 0.20	13.50%
CIFAR-20	48.46 ± 0.94	83.44 ± 0.17	34.98%
CIFAR-100	31.72 ± 1.36	73.84 ± 0.42	42.12%

5.3. Detecting Plasticity Loss

To determine whether plasticity loss can be detected during self-labeling by apply plasticity injection [NOO+24], we train all of our models for a total of 500 epochs. Longer training allows us to observe long-term learning behavior and ensures that the baseline performance plateaus. This results in four experimental setups:

- **Baseline Experiment**: The model is trained continuously for 500 epochs.
- **Injection Experiments**: Three separate models undergo plasticity injection at epochs 100, 200, and 250, respectively. At each injection epoch t, the model's head parameters are updated using the following rule:

  $$h_{\theta_{t+1}}(x) = h_{\theta_t}(x) + h'_{\theta_1}(x) - h'_{\theta_2}(x)$$

  Here, $h_{\theta_t}(x)$ represents the clustering head's output at epoch t. The parameters θ'_1 are freshly initialized, and θ'_2 are frozen copies of θ'_1.

This parameter update ensures that the network's output $h_\theta(x)$ remains unchanged at epoch $t+1$, as the intervention does not directly modify the output. Instead, it introduces new parameters θ'_1 to enhance the network's trainability while preserving the same total number of trainable parameters.

The injection points at epochs 100, 200, and 250 were chosen to assess the impact of plasticity injection at different training stages. After applying the injection, each model continued training until epoch 500. Figures 5.3 and 5.4 summarize the runs by visualizing the average accuracy on CIFAR-10 and STL-10, respectively, with standard deviation bands.

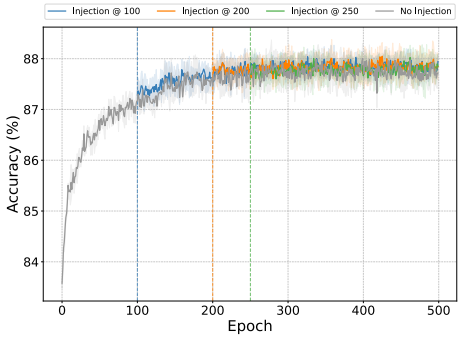


Figure 5.3.: Plasticity injection on CIFAR10

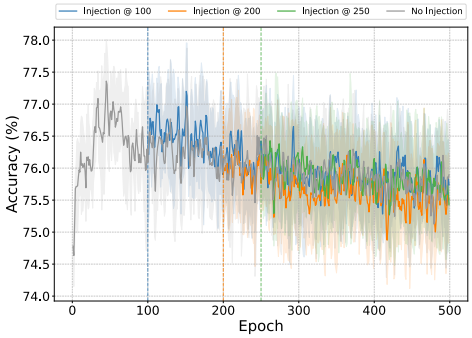


Figure 5.4.: Plasticity injection on STL10

We observe that plasticity injection did not significantly increase model accuracy on CIFAR-10 and STL-10 datasets, as shown by the results on CIFAR-10 in Figure 5.3 and on STL-10 in Figure 5.4. All models, including the baseline without an injection, converge to comparable values, despite slight accuracy variations right after the injection.

5.4. Weight Resets

Further, we conduct quantitative experiments using soft resets [AA20] and Hare and Tortoise networks [LCK+24]. Soft resets are periodically applied using a linear combination of the current weights and random weights drawn from the initial weight distribution. Meanwhile, Hare and Tortoise networks utilize two networks, where the hare is updated with SGD and distilled [Hin15] into the tortoise, which is then used to periodically reset the hare. In the following subsections, we provide pseudocode for the implementations and report the results.

5.4.1. Soft Resets

We evaluate Algorithm 2 by conducting an extensive hyperparameter sweep on CIFAR-10 dataset. Following the approach of BRB [MKS+24], we experiment with resetting the parameters of the last layer of the ResNet-18 model and of the clustering head. The hyperparameter search space is detailed in Table 5.4.

Table 5.4.: Hyperparameter Search Space for Soft Reset

Parameter	**Values**
Reset Frequency (R)	5, 10, 20
Retention Parameter (α) for Resnet-18 Last Layer	0.7, 0.75, ..., 0.95, 1
Retention Parameter (α) for Clustering Head	0.7, 0.75, ..., 0.95

We explored 126 hyperparameter combinations, each evaluated with five different seeds, resulting in a total of 630 runs on CIFAR-10 dataset. The box plot in Figure 5.5 illustrates the accuracy distribution for various soft reset retention parameters when applied to the last Resnet18 layer.

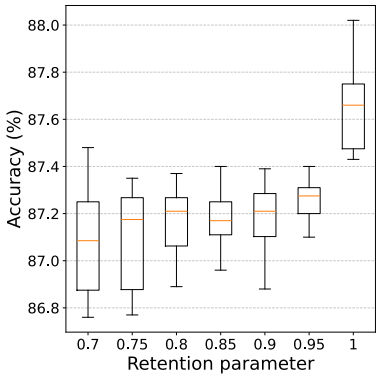


Figure 5.5.: Accuracy distributions under varying last Resnet-18 layer reset strengths on CIFAR-10

As the reset strength increases, we observe a clear trend of decreasing accuracy and an overall downward shift in the performance distribution. In contrast, when the layer is not reset, the model maintains a higher accuracy. This finding strongly suggests that resetting last Resnet18 layer during self-labeling negatively impacts the model's clustering ability, thereby degrading overall clustering performance.

Similarly, we present Figure 5.6, which depicts the accuracy distribution on CIFAR-10 when resets with varying reset strengths are applied.

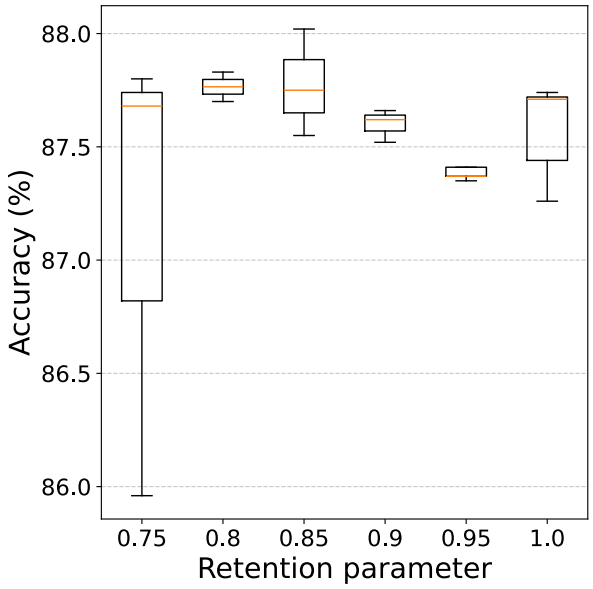
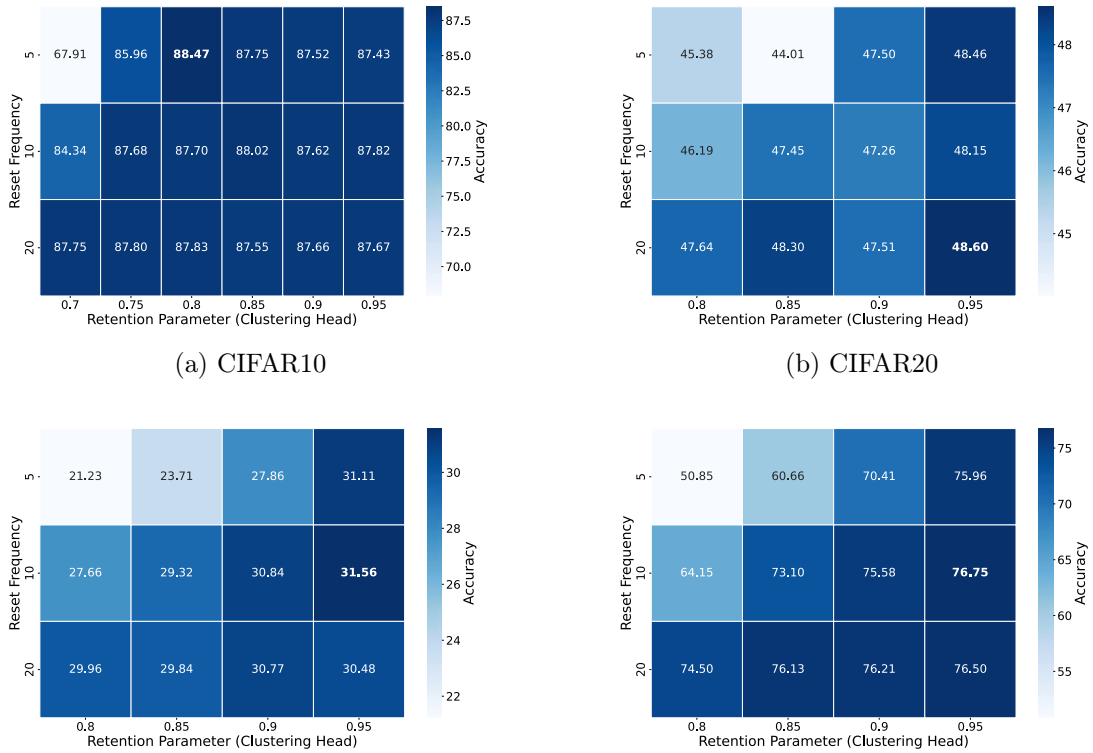


Figure 5.6.: Accuracy distributions under varying reset strengths of clustering head on CIFAR-10

Resetting only the clustering head results in the best performance, observed around a retention parameter of 0.8. This finding is further supported by the heatmap visualization of self-labeling with soft resets on CIFAR-10 5.7a.

The following experiments are thus conducted without resetting the last Resnet18 layer, but instead concentrate on resetting only the clustering head during self-labeling. The heatmap in Figure 5.7 summarizes the average accuracy for different combinations of reset frequency and clustering head retention parameter on all benchmark datasets, focusing only on runs where the last Resnet18 layer was not reset.



(a) CIFAR10

(b) CIFAR20

(c) CIFAR100

(d) STL10

Figure 5.7.: Soft Resets Accuracy Heatmap for Various Datasets

Accuracy generally improves with less aggressive resets, having retention parameter between 0.9 and 0.95 consistently perform well with less frequent resets of every 10 or 20 epochs.

5.4.2. Hare and Tortoise Networks

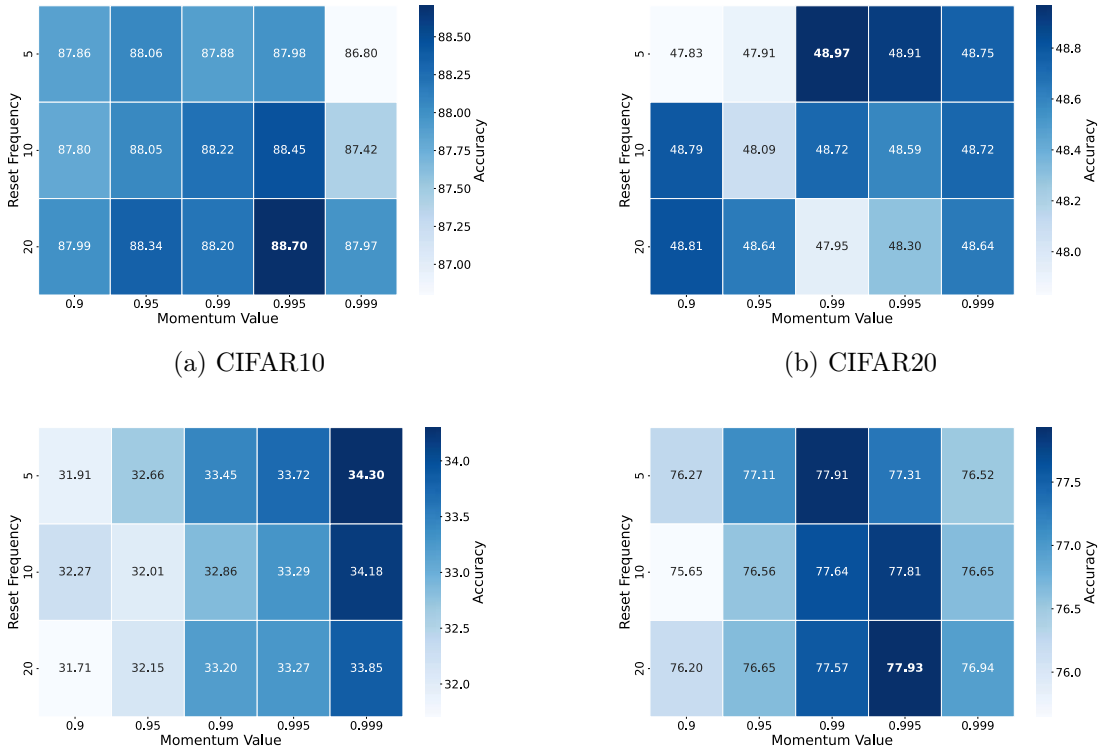
Following the original Hare and Tortoise paper [LCK+24], we evaluate Algorithm 4 with the recommended parameters for momentum values. We use the same reset frequency intervals as in the soft reset experiments.

Table 5.5.: Hyperparameter Search Space for Hare and Tortoise Networks

Parameter	**Values**
Reset frequency (R)	5, 10, 20
Momentum value (μ)	0.9, 0.95, 0.99, 0.995, 0.999

We conduct a grid search using the parameters listed in Table 5.5. For each pair of

reset frequency R and momentum value μ, the experiment is repeated using five different seeds. We provide a summary of the accuracy distribution in Figure 5.8



(a) CIFAR10

(b) CIFAR20

(c) CIFAR100

(d) STL10

Figure 5.8.: Hare and Tortoise Heatmap for Various Datasets

The results reveal that performance is highly sensitive to the choice of reset frequency and momentum values, with noticeable differences between datasets. The optimal configurations are dataset-specific and non-trivial to identify.

5.5. Weight Resets with Scheduling

In this section, we experiment with two scheduling techniques. To test whether we can utilize more data during the self-labeling stage, we investigate whether resets enable the use of less confident samples by scheduling the confidence threshold between resets. Additionally, we experiment with scheduling the strength of weight resets throughout training, applying stronger resets at the beginning when the task shift occurs and gradually softening them toward the end of training.

5.5.1. Scheduling Confidence Threshold

We conduct experiments using two opposing threshold schedules. In the first schedule, we begin with a strict confidence level of 0.99 immediately after a reset, then linearly lower it to 0.9 as the model stabilizes, just before the subsequent reset. In the second schedule, we start with a more permissive threshold of 0.9 right after a reset and later tighten it to 0.99 just before performing the next reset. We replicate the experiment using the soft reset approach, as well as the Hare and Tortoise networks. See Algorithm 5 and Algorithm 6 for details. For both reset techniques, we select the best-performing hyperparameter combination and evaluate whether they demonstrate robustness while trained on a larger set of samples.

- **Baseline with 0.99 Threshold**: Self-labeling training is performed once with soft resets and once with the Hare and Tortoise approach, using a constant confidence threshold set to 0.99.
- **Baseline with 0.9 Threshold**: Similar to the previous case, however the confidence threshold is lowered to 0.9 to include more samples during training.
- **Schedule threshold from 0.99 to 0.9**: The confidence threshold is initially set to 0.99 immediately after a reset and then decreased as the model stabilizes. This process is repeated at each reset.
- **Schedule threshold from 0.9 to 0.99**: The opposite scheduling of the previous approach is applied. Here, training begins with a threshold of 0.9 immediately after a reset, which is then gradually increased to 0.99 just before the next reset. This process is repeated at each reset.

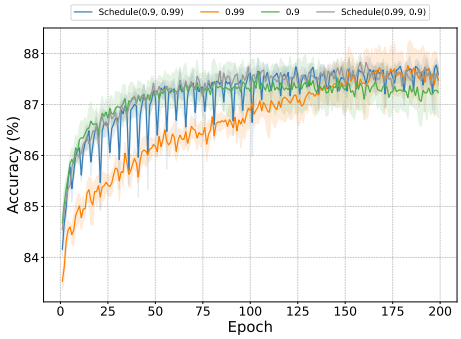


Figure 5.9.: SP CIFAR-10 Scheduling Confidence

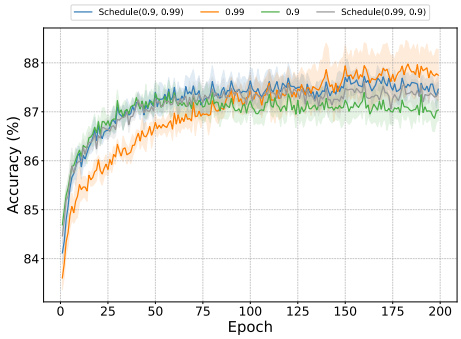


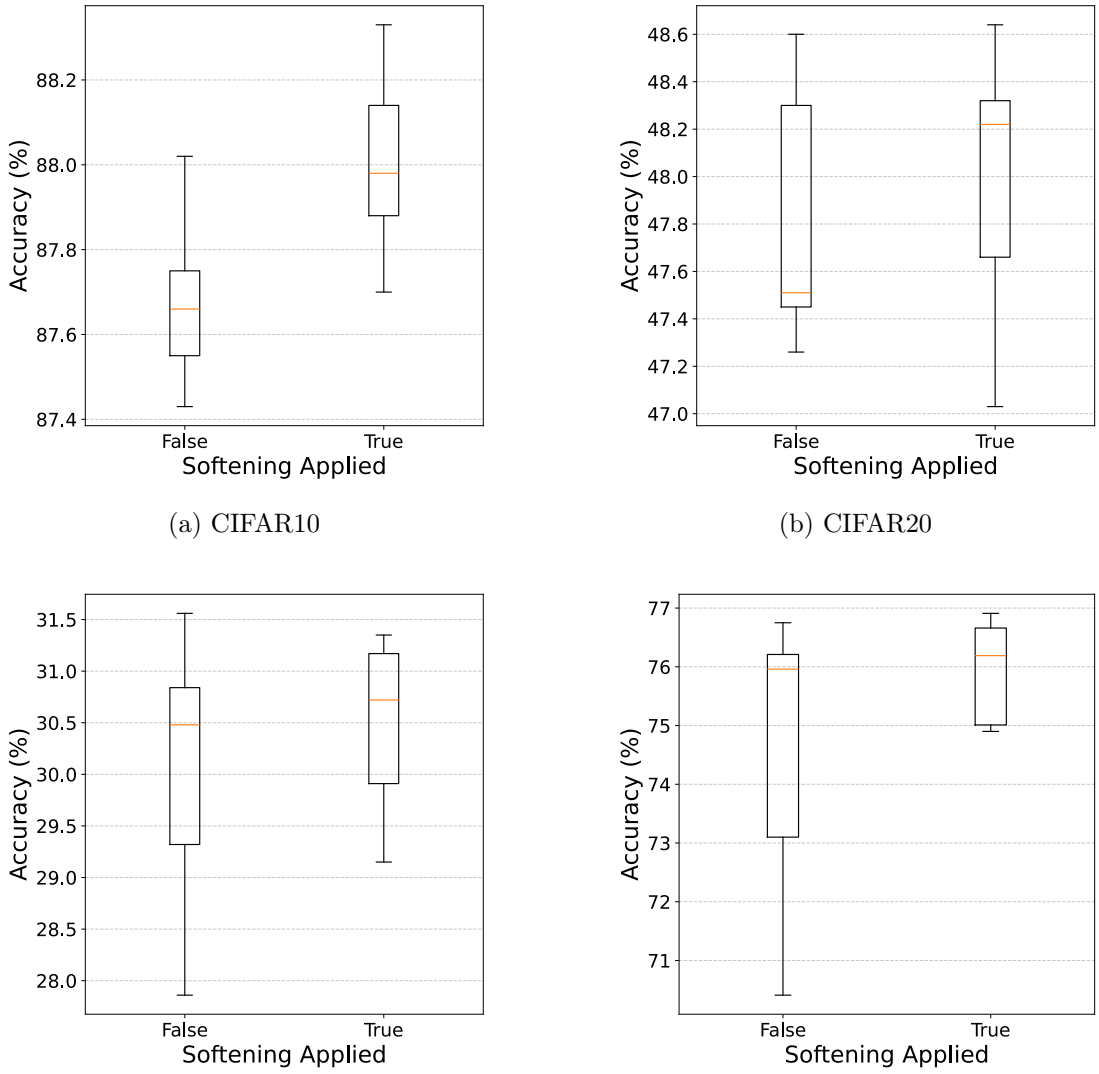
Figure 5.10.: HT CIFAR-10 Scheduling Confidence

Self-labeling achieves optimal performance when a strict constant confidence threshold of 0.99 is applied. This could happen, because the sample size gradually increases over the course of training. In settings with less strict thresholds the network initially shows

rapid gains and then gradually converges. These early rapid gains lead to a spike in the number of confident samples, causing the network to become overconfident.

5.5.2. Softening Reset Strength

We experiment with Algorithm 3, which schedules the intensity of weight resets during training. The process begins with stronger resets and gradually reduces their intensity as training progresses.



(a) CIFAR10

(b) CIFAR20

(c) CIFAR100

(d) STL10

Figure 5.11.: Softening the reset strength of the clustering head over the course of training on various datasets

Figure 5.11 depicts box-and-whisker plots comparing final average accuracies over five runs across four different datasets. Each subfigure contrasts runs where no scheduling was applied (*False* on the horizontal axis) against runs where the reset schedule gradually shifts from a higher reset strength to a lower one (*True*). Across all four datasets, the median accuracy is consistently higher when softening is applied, while the spread (variance) of accuracies is generally narrower, indicating more stable convergence. In certain cases, the difference in performance becomes especially pronounced, as seen in datasets like CIFAR10 5.11a and CIFAR20 5.11b.

5.6. Experiments Summary

We summarize the experiments in a table with three reset methods and a baseline, resulting in four experiment groups. SP(R, α) denotes self-labeling with soft resets (Section 5.4.1) every R epochs and retention parameter α. SP+(R, α) is its extension, featuring a linear increase of the retention parameter (Algorithm 3) throughout training. HT(R, μ) represents self-labeling with Hare and Tortoise networks (Section 5.4.2), where the hare network is reset to the tortoise network every R epochs with momentum value μ. The baseline corresponds to self-labeling without resets (Algorithm 1).

For each experiment group, we select the best setting for each of the four benchmark datasets, resulting in up to four settings per group. These best-performing settings are highlighted in the corresponding heatmaps for soft resets and Hare and Tortoise extensions in Figure 5.7 and Figure 5.8, respectively.

Table 5.6.: Summary Table (cells in blue are strictly greater than the baseline). 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
SP $(R = 5, \alpha = 0.8)$	50.85	56.38	88.47	**90.79**	45.38	47.29	21.23	22.83
SP $(R = 10, \alpha = 0.95)$	76.75	77.12	87.82	88.82	48.15	49.13	31.56	32.19
SP $(R = 20, \alpha = 0.95)$	76.50	77.29	87.67	87.94	48.60	49.24	30.48	31.88
SP+ $(R = 5, \alpha = 0.85)$	71.76	74.05	88.33	89.15	47.12	48.56	27.76	28.58
SP+ $(R = 10, \alpha = 0.95)$	76.91	77.21	87.95	88.87	48.64	49.47	31.17	31.53
SP+ $(R = 20, \alpha = 0.95)$	76.66	77.21	87.98	88.97	48.54	50.05	31.35	33.02
HT $(R = 5, \mu = 0.99)$	77.91	78.14	87.88	88.38	**48.97**	**50.80**	33.45	38.87
HT $(R = 5, \mu = 0.999)$	76.52	76.58	86.80	86.97	48.75	49.19	**34.30**	**39.09**
HT $(R = 20, \mu = 0.995)$	**77.93**	**78.34**	**88.704**	89.20	48.30	50.28	33.27	38.91
Baseline	76.45	77.80	87.57	87.74	48.46	49.76	31.72	33.33

Overall, the results presented in Table 5.6 demonstrate improvements over the baseline in

multiple settings, as highlighted by the blue cells. Similar to the comparison heatmaps of soft resets in Figure 5.7 and the Hare and Tortoise approach in Figure 5.8, we observe that dataset-specific tuning is necessary. Notably, the setting SP($R = 5, \alpha = 0.8$) yields a significant improvement over the baseline on CIFAR-10, however, stronger resets result in performance degradation on all other datasets. Hare and Tortoise approach, on the other hand, allows to significantly decrease the performance degradation on CIFAR-100. Additionally, the setting HT($R = 5, \alpha = 0.99$) consistently outperforms the baselines on all four datasets.

5.6.1. Longer Training

We further investigate the behavior of the self-labeling method in a scenario involving a longer training period. The same experimental groups from Section 5.6 are considered, and the results are summarized in Table 5.7. For each dataset, we report the accuracy of a model trained for 200 epochs, followed by the accuracy of a model trained for 500 epochs. To emphasize dynamics, we include a color-coded column that indicates a change in accuracy. Blue cells represent increasing values, while orange cells indicate decreasing values.

Table 5.7.: Mean accuracy at epochs 199 and 499 and their percentage change.

Experiment	STL-10			CIFAR-10			CIFAR-20			CIFAR-100		
	199	499	Change %	199	499	Change %	199	499	Change %	199	499	Change %
SP ($R = 5, \alpha = 0.8$)	50.85	33.21	-34.63%	88.47	88.67	0.23%	45.38	45.36	-0.04%	21.23	18.30	-13.78%
SP ($R = 10, \alpha = 0.95$)	76.75	75.70	-1.37%	87.82	88.05	0.26%	48.15	47.42	-1.52%	31.56	30.51	-3.33%
SP ($R = 20, \alpha = 0.95$)	76.50	75.12	-1.80%	87.67	87.76	0.10%	48.60	48.07	-1.09%	30.48	30.37	-0.36%
SP+ ($R = 5, \alpha = 0.85$)	71.76	54.88	-23.53%	88.33	88.30	-0.03%	47.12	44.70	-5.14%	27.76	26.18	-5.69%
SP+ ($R = 10, \alpha = 0.95$)	76.91	75.04	-2.43%	87.95	87.96	0.01%	48.64	47.13	-3.10%	31.17	30.47	-2.24%
SP+ ($R = 20, \alpha = 0.95$)	76.66	75.94	-0.94%	87.98	88.07	0.10%	48.54	47.48	-2.18%	31.35	30.82	-1.69%
HT ($R = 5, \mu = 0.99$)	77.91	77.53	-0.37%	87.88	88.32	0.44%	48.97	48.18	-0.79%	33.45	32.79	-0.66%
HT ($R = 5, \mu = 0.999$)	76.52	77.05	0.53%	86.80	87.93	1.13%	48.75	49.13	0.37%	34.30	34.71	0.41%
HT ($R = 20, \mu = 0.995$)	77.93	77.28	-0.65%	88.70	88.29	-0.42%	48.30	47.71	-0.59%	33.27	32.74	-0.54%
Baseline	76.45	75.30	-1.50%	87.57	88.10	0.61%	48.46	47.25	-2.50%	31.72	30.47	-3.94%

Although experiments on CIFAR-10 showcase minor benefits from longer training, we generally observe a trend where models degrade when trained for more epochs. We observe, that strong and frequent resets (e.g., SP($R = 5, \alpha = 0.8$) and SP+($R = 5, \alpha = 0.85$)) lead to significant performance declines. This happens due to the intensive weight manipulation involved. In contrast, Hare and Tortoise networks remain more stable and, even may enable longer training as in case of HT($R = 5, \mu = 0.999$).

6. Discussion

This chapter delves into the complexities and opportunities of self-labeling in deep clustering. It begins by addressing the challenges posed by noisy targets and the resulting self-reinforcing cycles that degrade model performance. Strategies like the Hare and Tortoise approach and soft resets are analyzed to mitigate these effects, while limitations such as dataset sensitivity and performance degradation are critically examined. The chapter concludes with a forward-looking discussion on enhancing self-labeling, offering a promising outlook for future research.

6.1. Noise-Reinforcing Cycle

Despite its effectiveness, self-labeling comes with an inherent challenge of noisy targets. As a consequence, this noise can propagate through subsequent optimization steps, ultimately degrading the quality of the learned assignments. As shown in Figure 5.1, our baseline experiments on CIFAR-100 demonstrate that the model may end up reinforcing its own initial biases or misclassifications, leading to a self-reinforcing cycle that degrades performance or locks the model into suboptimal local minima. Figure 5.6.1 depicts the issue in more detail. When training continues from epoch 200 to 500, we observe a significant degradation in performance across all benchmark datasets except for CIFAR-10.

6.2. Hare and Tortoise For a Better Generalization

We demonstrate that the Hare and Tortoise approach significantly reduces degradation in clustering accuracy on CIFAR-100 dataset during self-labeling. Moreover, for other datasets with more stable training dynamics, the Hare and Tortoise approach achieves improvements over baseline performance. It is primarily attributed to the tortoise network, which is updated gradually using an exponential moving average and generalizes more effectively.

6.3. On Plasticity Loss

Barely affected by plasticity injection, accuracy suggests that either trainability is not an issue during self-labeling or that plasticity injection is not the right tool to detect the loss of plasticity in this setting (See Figures 5.3 and 5.4). The work on maintaining plasticity with Hare and Tortoise networks [LCK+24] concludes that modern networks tend to retain trainability but lose generalizability after warm-starting. In the ablation

study, the authors of BRB [MKS+24], summarize that deep clustering algorithms do not suffer from plasticity loss, rather, the issue lies in premature convergence to sub-optimal local minima. We further support the claim that trainability is not an issue by presenting experiments in Figure 5.2. We show that introducing noise-free targets allows the network to continue training and achieve substantially higher clustering accuracy.

6.4. Scheduling Soft Resets

Poorer initial clustering performance makes soft resets increasingly sensitive to the selection of reset frequency and retention parameters. This behavior is expected, as resetting the network in these scenarios increases the risk of losing valuable information. However, implementing linear scheduling of reset strength allows to perform stronger resets at the beginning of training and further soften the interventions. In Figure 5.11, we show, that in summary across all datasets it is beneficial for the clustering performance. The results suggest that periodically reducing the strength of resets allows the model to preserve the initial performance of the clustering head while still benefiting from some degree of weights manipulation. Strong early resets help counteract task shift, but overly strong resets in later epochs may discard valuable information.

6.5. Performance Degradation

The authors of SCAN [VGVG+20] suggest training using self-labeling until the increase in the number of confident samples plateaus. However, as illustrated in Figure 1.1, the fraction of confident samples continues to grow throughout the training process. Although this approach may serve as a stopping criterion, the relationship between this growth and plateauing or potential performance degradation has not yet been sufficiently explored. We observe a tendency for more incorrect pseudo-labels to be included over time 5.1, which diminishes the effectiveness of self-labeling. An important direction for future research is to guarantee that self-labeling does not degrade initial performance.

6.6. Dataset Sensitivity

A significant limitation of self-labeling lies in its high sensitivity to the characteristics of the dataset it is applied to. While self-labeling may demonstrate strong performance on specific datasets (e.g., CIFAR-10 in Table 5.1), this success is not consistently replicated across all datasets. In some cases, its performance degrades significantly (See CIFAR-100 results in Table 5.1), yielding misleading results. Such inconsistency poses challenges for generalization and limits the effectiveness of self-labeling. In the unsupervised domain, where access to the ground truth is unavailable, further research is needed to enhance the generalizability of self-labeling.

6.7. Promising Outlook for Self-Labeling

Recent deep clustering methods [Qia23, ZWC+21] have adopted a self-labeling stage as an effective, algorithm-agnostic way to improve clustering performance. We believe that the effectiveness of this technique can be further enhanced by addressing its limitations. One possible area of research is to explore different sampling techniques. Confidence thresholding limits the number of utilized samples at the beginning of training and leads to generalizability issues. Future research directions may include linearly increasing the number of most confident samples, thereby avoiding the need for thresholding, or developing adaptive sampling strategies that adjust the selection criteria based on model performance throughout the training process. Additionally, integrating uncertainty estimation could enhance the reliability of self-labeling by better identifying ambiguous samples. Exploring these avenues could lead to a more generalizable approach to self-labeling, ultimately advancing the field.

7. Conclusion

In this thesis, we explored the impact of established techniques from the plasticity literature on deep clustering during the self-labeling stage. To detect plasticity loss during training, we employed plasticity injection. However, it did not reveal any trainability issues during self-labeling. This result aligns well with the findings of the authors of the Hare and Tortoise networks paper [LCK+24], who concluded that networks tend to retain trainability but lose generalizability after warm starting.

Although we could not confirm the presence of plasticity loss during the self-labeling stage, our extensive experiments demonstrate that multiple configurations of both weight manipulation techniques — soft resets and Hare and Tortoise networks — outperform the baseline when applied during self-labeling. Most results are highly sensitive to reset frequency and weight retention parameters. However, experiments with Hare and Tortoise networks show improved generalizability during self-labeling.

We introduced a scheduling mechanism for the retention parameter in soft resets. We observed that it is more reliable for the self-labeling stage than using a constant reset strength. Linear softening of reset strength prevents the discarding of valuable information while still helping to counteract initial overconfidence.

Experimenting with a smaller confidence threshold during resets revealed that a strict threshold is necessary to maintain gradual learning and to avoid initial suboptimal spikes followed by a plateau in clustering accuracy.

Finally, this master's thesis proposes an outlook for future work on self-labeling. The effectiveness of self-labeling motivates the exploration of more reliable, algorithm-agnostic techniques for fine-tuning clustering performance using pseudo-labels. In the unsupervised domain, guarantees against performance degradation are highly relevant, and future work may include addressing this challenge.

Bibliography

[AA20] Jordan Ash and Ryan P Adams. On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894, 2020.

[AFG+21] 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.

[BCV13] 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.

[BN06] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[CFGH20] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[CKNH20] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[CNL11] 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*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[DSRB14] 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.

[EM24] Mohamed Elsayed and A Rupam Mahmood. Addressing loss of plasticity and catastrophic forgetting in continual learning. *arXiv preprint arXiv:2404.00781*, 2024.

[HA85] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.

[Hin15] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[HWWZ24] Huajuan Huang, Chen Wang, Xiuxi Wei, and Yongquan Zhou. Deep image clustering: A survey. *Neurocomputing*, 599:128101, 2024.

[HZRS16] 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*, pages 770–778, 2016.

[JHV19] 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*, pages 9865–9874, 2019.

[KH+09] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[KMS+24] 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.

[Kuh55] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[Kva87] Tarald O Kvalseth. Entropy and correlation: Some comments. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(3):517–519, 1987.

[LCK+24] 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.

[LLL+24] 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.

[LZN+23] 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*, pages 23190–23211. PMLR, 2023.

[M+67] 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, pages 281–297. Oakland, CA, USA, 1967.

[MKS+24] 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.

[NOO+24] 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.

[Qia23] Qi Qian. Stable cluster discrimination for deep clustering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16645–16654, 2023.

[vdOLV18] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[VGVG+20] 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*, pages 268–285. Springer, 2020.

[VL07] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

[WXYL18] 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*, pages 3733–3742, 2018.

[YXN+10] 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.

[ZWC+21] 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*, pages 9224–9233, 2021.

[ZXZ+24] 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.

A. Appendix

A.1. Data

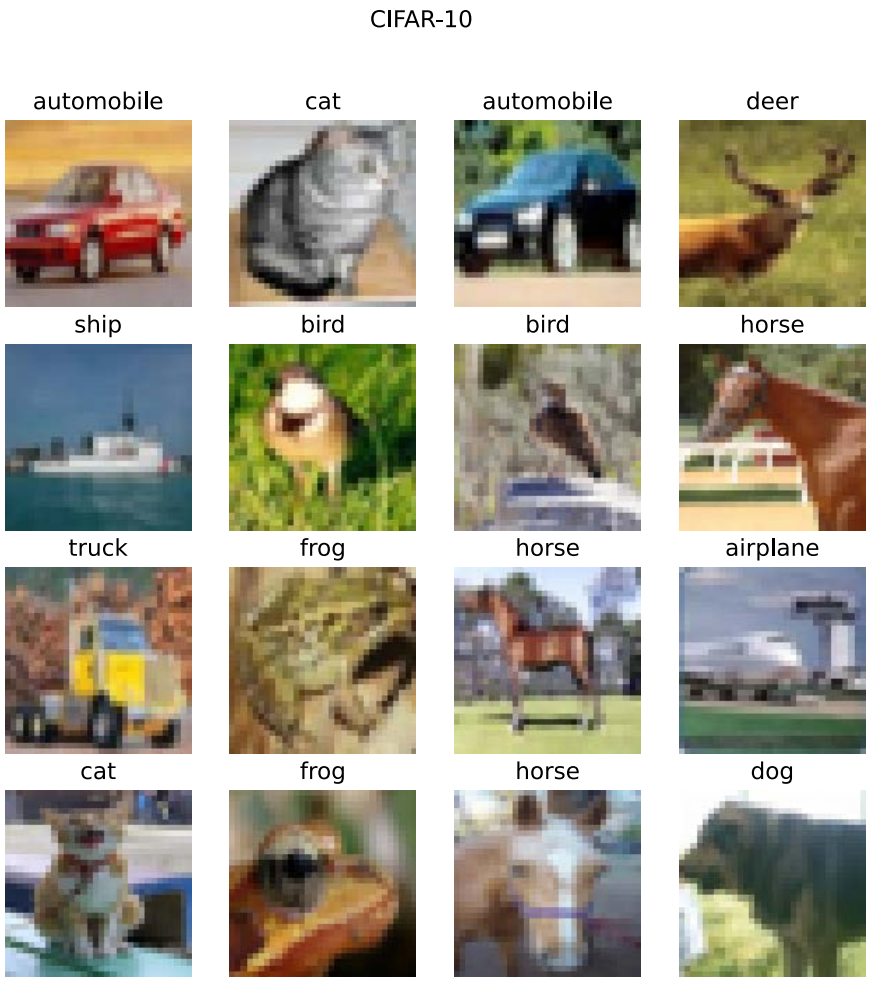


Figure A.1.: CIFAR-10 Samples

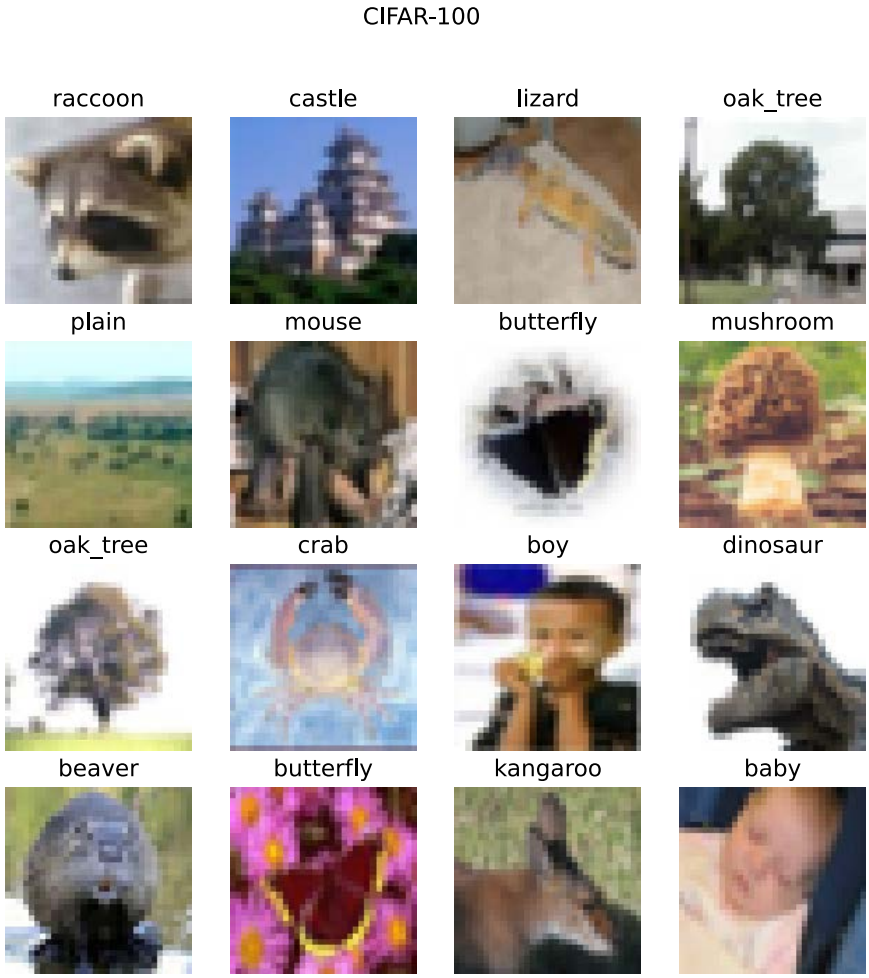


Figure A.2.: CIFAR-100 Samples

Figure A.3.: STL-10 Samples