

Exploring Beyond Logits: Hierarchical Dynamic Labeling Based on Embeddings for Semi-Supervised Classification

Yanbiao Ma, Licheng Jiao, Fang Liu, LingLing Li, Shuyuan Yang and Xu Liu

Key Laboratory of Intelligent Perception and Image Understanding of the Ministry of Education
International Research Center for Intelligent Perception and Computation, Xi'an 710071, China
School of Artificial Intelligence, Xidian University, Xi'an 710071, China
ybmamail@stu.xidian.edu.cn, lchjiao@mail.xidian.edu.cn

Abstract

In semi-supervised learning, methods that rely on confidence learning to generate pseudo-labels have been widely proposed. However, increasing research finds that when faced with noisy and biased data, the model's representation network is more reliable than the classification network. Additionally, label generation methods based on model predictions often show poor adaptability across different datasets, necessitating customization of the classification network. Therefore, we propose a Hierarchical Dynamic Labeling (HDL) algorithm that does not depend on model predictions and utilizes image embeddings to generate sample labels. We also introduce an adaptive method for selecting hyperparameters in HDL, enhancing its versatility. Moreover, HDL can be combined with general image encoders (e.g., CLIP) to serve as a fundamental data processing module. We extract embeddings from datasets with class-balanced and long-tailed distributions using pre-trained semi-supervised models. Subsequently, samples are re-labeled using HDL, and the re-labeled samples are used to further train the semi-supervised models. Experiments demonstrate improved model performance, validating the motivation that representation networks are more reliable than classifiers or predictors. Our approach has the potential to change the paradigm of pseudo-label generation in semi-supervised learning.

1 Introduction

With the advent of the era of large-scale models, researchers have an increasingly urgent demand for data [Liang *et al.*, 2022; Paullada *et al.*, 2021; Facchinetti *et al.*, 2022]. Manual annotation stands out as a reliable method to enhance label quality, but its cost becomes prohibitively expensive for large-scale datasets. Current automatic labeling methods mainly focus on utilizing Deep Neural Networks trained on labeled data to predict labels for unknown samples. Particularly in the field of semi-supervised learning, Confident Learning [Northcutt *et al.*, 2021] has become a ubiquitous strategy for obtaining pseudo-labels for samples [Lee and others, 2013a;

Wang *et al.*, 2022; Wang *et al.*, 2023]. Specifically, the approach involves training a model with labeled data while utilizing the model to generate pseudo-labels for unlabeled samples with high confidence. As the model's performance improves, more samples are selected based on high confidence, and semi-supervised learning benefits from this iterative optimization process. However, methods based on Confidence learning face two primary limitations:

- Generating prediction confidence requires training classification networks specifically for different datasets, making them less versatile.
- When labeled data is biased or contains noise (i.e., samples with incorrect labels), classifiers in deep neural networks tend to exhibit significant bias [Ma *et al.*, 2023c; Xu *et al.*, 2023] and excessive memorization of noisy samples [Zhu *et al.*, 2022; Pleiss *et al.*, 2020], leading to the mislabeling of unknown samples.

Motivation A promising annotation approach is one that does not rely on the model's predictions. Recent studies have found that in real-world scenarios, the representation network of a model appears to be more reliable than prediction networks or classifiers [Kang *et al.*, 2019; Ma *et al.*, 2023a]. In the realm of long-tailed image recognition, the motivation behind decoupled training [Zhou *et al.*, 2020] is rooted in the observation that biases in models stem from classifiers, whereas representation networks can obtain unbiased image embeddings. In the domain of noise detection, it has been observed that models tend to excessively memorize noisy samples, hindering accurate noise identification based on the model's prediction confidence [Zhu *et al.*, 2022]. The aforementioned studies suggest that, compared to methods relying on model predictions, developing embedding-based, data-centric approaches can avoid many shortcomings introduced by the learning process [Liu *et al.*, 2023]. Therefore, our focus is on developing an embedding-based image annotation method to enhance tasks like semi-supervised learning, leveraging unlabeled data to improve model performance. The main contributions of this paper are summarized as follows:

- We propose a Hierarchical Dynamic Labeling algorithm (HDL) that does not rely on model predictions (see Section 4.3). This method is highly flexible and can also be combined with CLIP to serve as a general-purpose data

preprocessing module (see Section 5.7).

- We propose a method for adaptively selecting the hyperparameter k for HDL, making HDL less reliant on empirical choices and thereby enhancing its universality.
- In both class-balanced (see Section 5.4) and class-imbalanced scenarios (see Section 5.5), our method significantly enhances the performance of semi-supervised models (Table 1 and 3), validating the notion that the representation network is more reliable than classifiers or predictors. Furthermore, our approach has the potential to shift the paradigm of generating pseudo-labels in semi-supervised learning. (see Section 5).

2 Related Works

Generating image pseudo-labels based on confidence learning The core idea of confidence learning [Northcutt et al., 2021] is to identify labeling errors in a dataset by the predictive confidence of the model. Specifically, when the prediction confidence of a sample is lower than a predetermined threshold, the sample is considered to be mislabeled with a high probability. In order to make the recognition results more reliable, it becomes a popular approach to use multiple models for joint prediction. The idea of confidence learning is widely used in the field of semi-supervised learning for the generation of image pseudo-labels [Lee and others, 2013b; Berthelot et al., 2019b; Berthelot et al., 2019a; Chen et al., 2023]. The basic paradigm is to train on labeled data first and then predict the pseudo-labels of unlabeled data, which in turn selects high-confidence pseudo-labels to train the model. For example, MixMatch [Berthelot et al., 2019b] augments unlabeled samples k times and then uses the model to make k predictions on the augmented samples and averages them to obtain pseudo-labels. In order to improve the reliability of model predictions, ReMixMatch [Berthelot et al., 2019a] adds a distribution alignment step to MixMatch with the aim of making the model predicted labels closer to the existing label distribution. FixMatch [Sohn et al., 2020] uses weak data augmentation to obtain pseudo-labels for the samples and is used to supervise the output of strong data augmentation. The recent Dash [Xu et al., 2021], FlexMatch [Zhang et al., 2021] and FreeMatch [Wang et al., 2022] enhance the quality of pseudo-labels by designing a better method of choosing the prediction confidence threshold. **Different from them**, we label unknown samples based on the representation of images and justify our motivation by improving existing methods.

3 Preliminary

3.1 Task and Symbol Definition

Given a labeled dataset $D := \{(x_n, y_n)\}_{n \in [N]}$, where $[N] := \{1, 2, \dots, N\}$ and $y_n \in \{1, 2, \dots, C\}$. Additionally, we have an unlabeled dataset $D' := \{x'_m\}_{m \in [M]}$, where y'_m represents the potential label corresponding to x'_m . There are two possible scenarios for the value of y'_m : (1) $\forall m \in [M], y'_m \in [C]$, and (2) $\exists m \in [M], y'_m \notin [C]$. We focus on closed-set image labeling, so the label spaces of datasets D and D' are consistent, i.e., $\forall m \in [M], y'_m \in \{1, 2, \dots, C\}$.

3.2 Label Clusterability

This work aims to achieve automatic image annotation without training, and the label clusterability [Zhu et al., 2022; Ohi et al., 2020] provides a basis for our research. Intuitively, clusterability means that two embeddings that are closer have a higher probability of belonging to the same true class. Label clusterability can be formally defined as follows.

Definition 1 ((k, δ_k) Label Clusterability). *Given a dataset D and an image encoder $f(\cdot)$, extract the embeddings $X = f(D)$ of all images in D . For $\forall x \in X$, if the probability that x and its k nearest neighbors belong to the same true class is at least $1 - \delta_k$, then the dataset D is said to satisfy (k, δ_k) label clusterability. When $\delta_k = 0$, the dataset D is said to satisfy k -NN label clusterability.*

δ_k represents the probability that the dataset D violates the clusterability, which is related to the size of k and the quality of the embedding. It is obvious that δ_k increases as k increases. This is because when k is larger, the k nearest neighbors of the central instance are more likely to contain embeddings of other categories. When the quality of the embedding is lower, it means that the image encoder has a worse ability to distinguish between categories, which may lead to confusion between embeddings of different categories, making δ_k larger. In the context of automatic image annotation in general scenarios, large-scale visual pre-training models with powerful visual representation capabilities, such as CLIP [Radford et al., 2021], have emerged as a compelling choice for extracting embeddings.

4 Automatic classification using embedding

In this section, we first introduce the simplest idea of utilizing label clusterability for annotation, and then progressively refine it to propose a more general and superior algorithm. For the sake of clarity in presentation, CLIP is utilized in this section to extract embeddings from the dataset; however, the experimental section will incorporate a variety of image encoders for a more extensive evaluation.

4.1 Intuition for image labeling relying on (k, δ_k) label clusterability

Given a set of labeled embeddings $D := \{(x_n, y_n)\}_{n \in [N]}$ and a set of unlabeled embeddings $D' := \{x'_m\}_{m \in [M]}$, all embeddings are extracted by the CLIP. Assuming that D satisfies the k_1 -NN label clusterability, this means that we can determine the label of the central instance by using the labels of its k_1 nearest neighbors. Specifically, for the embedding $x'_m, m \in [M]$ to be labeled, we select k_1 nearest neighbors $x'_{m_1}, x'_{m_2}, \dots, x'_{m_{k_1}}$ from D , and their corresponding labels are $y'_{m_1}, y'_{m_2}, \dots, y'_{m_{k_1}}$. We convert all labels into the form of one-hot encoding, and then the label of x'_m can be obtained by voting on its k_1 nearest neighbors, which is

$$y'_m = \operatorname{argmax}_{i \in [C]} \left(\frac{1}{k_1} \sum_{j=1}^{k_1} y'_{m_j} \right) [i], y'_m \in \mathbb{R}^C. \quad (1)$$

The above process seems to be a valuable method for image annotation using clusterability, but this method has high

requirements for clusterability and there is room for optimization. Let's analyze it below. Although the embeddings in D' have no label information, their label space is consistent with D . Therefore, after mixing the set D' of embeddings extracted by CLIP with D , all the embeddings still satisfy a certain degree of clustering. When selecting k_1 nearest neighbors for x'_m , the unlabeled embeddings are excluded, but these unlabeled embeddings may be closer to the central instance x'_m . Let the distance between the k_1 th nearest neighbor of x'_m and x'_m be d_{k_1} , and simultaneously select k_1 nearest neighbors for x'_m from D' and D . Let the distance between the k_1 th nearest neighbor and x'_m be d_{k_2} . Clearly, $d_{k_1} \geq d_{k_2}$. The above analysis shows that methods that do not use unlabeled embeddings for voting have higher requirements for label clusterability. In order to make the automatic labeling method based on embeddings more general and universal, we pursue the use of unlabeled embeddings to assist in labeling, thereby reducing the requirements of the labeling algorithm for label clusterability and improving the performance of the labeling algorithm.

4.2 Unlabeled embeddings can promote labeling

In the following, we refer to the method introduced in Section 4.1 as kNN-DV, and assume that all embedding sets satisfy the k -NN label clusterability. Introducing the unlabeled embedding set D' into the selection range of k -NN means that there may be more similar embeddings inside the hypersphere $B(x'_m, R)$ centered at x'_m and with radius R , compared to kNN-DV. In this case, if we keep the number of nearest neighbors consistent with the base algorithm, we can find enough similar embeddings within a hypersphere centered at x'_m with a volume smaller than $B(x'_m, R)$. The aforementioned concept allows for the reduction in the requirement of label clusterability for the embedding set while maintaining a constant number of nearest neighbors.

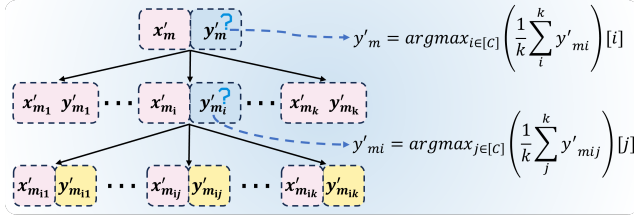


Figure 1: An example of a two-tier tree structure used for label searching. Blue boxes indicate the absence of a label, while yellow boxes represent uncertainty regarding the existence of a label.

However, we **face a problem** that must be solved: some of the k nearest neighbors of $x'_{m_1}, x'_{m_2}, \dots, x'_{m_k}$, may be unlabeled embeddings, i.e., there exists $j \in [k]$ such that $x'_{m_j} \in D'$, which makes it difficult to directly determine label of x'_m through voting. However, we found that the labels of each embedding can be gradually determined through depth-first search of the tree structure. As shown in Figure 1, assume that among k nearest neighbors of x'_m , $x'_{m_j} \in D'$, $j \in [k]$, then it is necessary to further find x'_{m_j} 's k nearest neighbors. If all of x'_{m_j} 's k -NN carry label information, or a certain por-

portion of nearest neighbors carry label information, then use these labels to vote for the category of x'_{m_j} , and then determine the category of x'_m . If most of x'_{m_j} 's k nearest neighbors do not have labels, continue depth-first search. Each step of the deep search satisfies the criterion of label clusterability with reduced requirements.

The intuitive understanding of using tree-based depth-first search for labeling is as follows: given the set D' of embeddings to be labeled, labeled the center instance with the largest number of labeled embeddings in its k -NN, then add the labeled embeddings to set D and use them to label other embeddings. That is, the elements of D are dynamically increased during the labeling process until all embeddings belonging to D' have been transferred to D . However, the disadvantage of depth-first search based on tree structure is also obvious, such as may fall into circular search, multiple branches repeat search and so on. Therefore, we propose a hierarchical dynamic labeling algorithm.

4.3 Hierarchical Dynamic Labeling (HDL)

Based on the analysis in section 4.2, introducing unlabeled embeddings into the automatic labeling process can reduce the algorithm's requirements for label clusterability. However, using tree-based search and voting has significant drawbacks, so we propose hierarchical dynamic labeling to achieve the same goal as tree-based automatic labeling.

Given a labeled embedding set $D := \{(x_n, y_n)\}_{n \in [N]}$ and an unlabeled embedding set $D' := \{x'_m\}_{m \in [M]}$, the **core idea** of hierarchical dynamic labeling with respect to D' involves two points: (1) prioritize labeling the center instance with the largest number of labeled embeddings among its k nearest neighbors; and (2) dynamically update D by transferring newly labeled embeddings to D in real-time.

Why is our approach called hierarchical? This comes from our first core idea. Specifically, for each embedding in D' , we search its k nearest neighbors from both D and D' , count the number of embeddings in the k nearest neighbors that belong to D , and represent it as $L_1, L_2, \dots, L_M$. First, we select all embeddings that satisfy $L_m = \max\{L_1, L_2, \dots, L_M\}$, $m \in [M]$ for labeling, and represent the selected embeddings as $x'_{max,1}, \dots, x'_{max,s}$, $s \leq M$. Furthermore, we need to determine the labeling order of the selected s embeddings. Obviously, during the entire labeling process, the labeling order of the embeddings is gradually determined by two levels. In the first level, s samples with priority labeling are determined, and in the second level, the labeling order of the s samples is further determined. Next, we will introduce how to determine the labeling order of the s samples in the second level.

Given s unlabeled embeddings $x'_{max,1}, \dots, x'_{max,s}$. Assume that the labeling of $x'_{max,i}$, $i \in [s]$ is first conducted, then $x'_{max,i}$ is transferred to D after the labeling is completed. Then we count the number $[l_{i,1}, \dots, l_{i,i-1}, l_{i,i+1}, \dots, l_{i,s}] \in \mathbb{R}^{s-1}$ in the k nearest neighbors of the remaining $s-1$ embeddings that belong to D . Perform the above operation for all s embeddings and place the statistical results for the i -th

Algorithm 1: Hierarchical Dynamic Labeling (HDL)

Input: k , labeled embedding set $D := \{(x_n, y_n)\}_{n \in [N]}$, unlabeled embedding set $D' := \{x'_m\}_{m \in [M]}$.

Output: D

```

1: repeat
2:   For each embedding in  $D'$ , we search its  $k$  nearest neighbors from both  $D$  and  $D'$ , count the number of embeddings in
   the  $k$  nearest neighbors that belong to  $D$ , and represent it as  $L_1, L_2, \dots, L_M$ .
3:    $First\_level\_data = []$  # First_level_data is used to store the embeddings of the first level.
4:   for  $m = 1$  to  $M$  do
5:     if  $L_m == np.max(L_1, L_2, \dots, L_M)$  then
6:        $First\_level\_data.append(x'_m)$  # Save the eligible embedding into First_level_data.
7:     end if
8:   end for
9:    $FLD\_number = len(First\_level\_data)$ ,  $order = torch.zeros(FLD\_number)$ 
10:  for  $i = 1$  to  $FLD\_number$  do
11:    # Find_kNN( $\cdot$ ) for finding  $k$  nearest neighbors.
12:     $First\_level\_data[i-1]_1, \dots, First\_level\_data[i-1]_k = Find\_kNN(First\_level\_data[i-1])$ 
13:    # Voting using labeled  $L_m == np.max(L_1, L_2, \dots, L_M)$  nearest neighbors.
14:     $y(First\_level\_data[i-1]) = Vote(First\_level\_data[i-1]_1, \dots, First\_level\_data[i-1]_k)$ 
15:     $D.append(First\_level\_data[i-1], y(First\_level\_data[i-1]))$ 
16:    Find the  $k$  nearest neighbors for each embedding in  $First\_level\_data$  except for  $First\_level\_data[i-1]$ , and count
    the number of nearest neighbors that belong to  $D$  for each embedding, denoted as  $l_m, m \in [FLD\_number], m \neq i-1$ .

17:    Remove  $(First\_level\_data[i-1], y(First\_level\_data[i-1]))$  from  $D$ .
18:     $order[i-1] = sum(l_m, m \in [FLD\_number], m \neq i-1)$ 
19:  end for
20:   $_, index = sort(order)$  # The labeling order of the  $FLD\_number$  embeddings is stored in index.
21:  for  $i = 1$  to  $FLD\_number$  do
22:    Label  $First\_level\_data[index[i-1]]$  and move into  $D$  after labeling.
23:    Remove  $First\_level\_data[index[i-1]]$  from  $D'$ .
24:  end for
25: until  $|D'| = 0$  #  $|D'|$  denotes the number of embeddings in  $D'$ 

```

embedding in the i -th row of matrix l . l is represented as

$$l = \begin{bmatrix} l_{1,2} & \cdots & l_{1,i} & l_{1,i+1} & \cdots & l_{1,s} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ l_{i,1} & \cdots & l_{i,i-1} & l_{i,i+1} & \cdots & l_{i,s} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ l_{s,1} & \cdots & l_{s,i-1} & l_{s,i} & \cdots & l_{s,s-1} \end{bmatrix} \in \mathbb{R}^{s \times (s-1)}.$$

Let $l[i]$ represent the i -th row of matrix l . We sum each row of the matrix to obtain $sum(L) = [sum(l[1]), \dots, sum(l[s])] = \sum_{j \neq 1}^s l_{1,j}, \dots, \sum_{j \neq s}^s l_{s,j} \in \mathbb{R}^s$. $sum(l[i])$ reflects the number of embeddings in the k nearest neighbors of the remaining $s-1$ embeddings that belong to D after the i -th embedding is labeled. Obviously, a larger $sum(l[i])$ is preferred, so we sort the elements of $sum(L)$ in descending order and prioritize labeling embeddings with larger element values. At this point, we have introduced the proposed hierarchical dynamic labeling algorithm in its entirety. The detailed calculation steps are shown in Algorithm 1.

The core parameter of the hierarchical dynamic Labeling (HDL) algorithm is the number of nearest neighbors k . How to adaptively choose k is the key to enhancing the generality of our method. In the next subsection, we will give an adaptive selection method for k in terms of the probability that clusterability holds and the probability that the k -nearest-

neighbor vote is successful.

4.4 Adaptive selection and analysis of k

When k is larger, the majority voting method is obviously more robust and reliable. However, it has to be considered that as k increases, the probability of satisfying the clusterability of the embedding decreases. Therefore, the selection of k value needs to balance the probability of satisfying the clusterability and the advantage brought by majority voting. Given a set of embeddings, assuming the probability of satisfying the clusterability is μ_k , then the probability of correctly voting [Zhu et al., 2022] by k nearest neighbors satisfies

$$P(k) \geq \mu_k \cdot I_{1-e}(k+1-k', k'+1), \quad (2)$$

where $k' = \lceil (k+1)/2 \rceil - 1$ and e represents the probability of incorrect labels in the embedding set. $I_{1-e}(k+1-k', k'+1)$ is a regularized incomplete beta function, calculated as $I_{1-e}(k+1-k', k'+1) = \frac{(k+1)!}{(k-k')!k'!} \int_0^{1-e} t^{k-k'} (1-t)^{k'} dt$. As k increases, μ_k usually decreases while $I_{1-e}(k+1-k', k'+1)$ increases. $P(k)$ can be used as a tool for selecting appropriate k values, where μ_k is affected by the quality of the embedding, and thus the μ_k corresponding to different k values need to be statistically obtained from the embedding set. Specifically, a certain proportion of center instances are randomly selected from the labeled embedding set, and their k

Listing 1: Adaptive selection of k

```

1  def auto_k(clip_feature_labeling, Label, p=0.1, k_upper_limit = 20):
2      PBTY = torch.zeros(k_upper_limit)    #PBTY is used to store the probability
        that clustrability holds.
3      I = torch.zeros(k_upper_limit)
4      for i in tqdm(range(1, k_upper_limit)):
5          features = clip_feature_labeling
6          m = int(features.shape[0]*p)
7          selected_features=features[np.random.randint(0, features.shape[0], (m,))]
8          norms = torch.norm(selected_features, dim=1)
9          selected_features = selected_features / norms.view(-1,1)
10         dist_matrix = 1 - torch.matmul(selected_features, features.T)
11         _, indices = dist_matrix.topk(i+1, dim=1, largest=False)
12         Y = Label[indices]
13         row_indices = list(range(Y.shape[0]))
14         result = list(map(lambda idx: check_row_equal(Y, idx), row_indices))
15         probability = sum(result) / Y.shape[0]
16         PBTY[i-1] = probability
17         k1 = math.ceil((i+1)/2)-1
18         I[i-1] = scipy.special.betainc(i+1-k1, k1+1, 0.85)
19         max_value, max_index = torch.max(PBTY*I, 0)
20         return max_index+1

```

nearest neighbors are found. The condition of whether the center instances and their k nearest neighbors have the same label is checked, and the proportion of center instances that satisfy this condition is counted to estimate μ_k . The steps for estimating μ_k and selecting k values are given in Listing 1.

5 Experiments

In this section, we introduce the experiments from the following three aspects: (1) Experimental objectives and the design of the experimental program. (2) The dataset used in the experiments and the related parameter settings. (3) Experimental results and analysis.

5.1 Experimental objectives and program design

Looking back at the fundamental motivation of this work again: **image embeddings are more reliable than model predictions**. Therefore, we propose the HDL algorithm based on image embeddings. The goal of this paper is to prove that HDL is superior to labeling methods based on confident learning. Semi-supervised learning with pseudo-labels provides us with a good experimental setting for this purpose. Specifically, the semi-supervised model trains the model while predicting pseudo-labels for unknown samples to achieve alternating optimization. **This leads to the question:** if HDL based on embeddings is more reliable than model predictions, then after semi-supervised learning convergence, using HDL to relabel unknown samples and continue training the semi-supervised model will further improve performance. If performance cannot be improved, it proves that our method does not bring any extra benefits. Based on this thinking, we extracted image embeddings from a converged semi-supervised model and conducted the above experiments in scenarios with class balance (see Section 5.4) and long-tailed distribution (see Section 5.5).

5.2 Datasets and Implementation Details

We evaluated the performance of HDL in improving semi-supervised methods on both class-balanced and class-imbalanced datasets. The balanced datasets included CIFAR-10, CIFAR-100, and STL-10, while the class-imbalanced dataset was CIFAR-10-LT with various imbalance factors.

Class-balanced datasets: CIFAR-10, CIFAR-100, STL-10. CIFAR-10 [Krizhevsky *et al.*, 2009] consists of 60,000 images across 10 classes, of which 50,000 images are used for training and 10,000 images are used for testing. We follow the common practice in semi-supervised learning to partition the data as follows: for each class, we randomly select 4, 25, and 400 samples as labeled data, and treat the remaining samples in the training set as unlabeled data. CIFAR-100 [Krizhevsky *et al.*, 2009] comprises 100 classes with 500 training samples and 100 testing samples per class. We randomly select 25 and 100 samples as labeled data for each class, and consider the remaining samples as unlabeled data. STL-10 [Coates *et al.*, 2011] includes 10 classes with 500 training samples, 800 testing samples, and an additional 10,000 unlabeled samples. For each class, we randomly select 100 samples as labeled data from the training set.

Class unbalanced dataset: CIFAR-10-LT (IF = 50, 100, 200). The degree of class imbalance is characterized by the imbalance factor (IF) [Cui *et al.*, 2019]. Assuming there are C classes, with sample sizes of $n_1, n_2, \dots, n_C$, where $n_1 \geq n_2 \geq \dots \geq n_C$, then IF is defined as $\frac{n_1}{n_C}$. CIFAR-10-LT [Ma *et al.*, 2023b] is a long-tail version of CIFAR-10, and we select CIFAR-10-LT with IFs of 50, 100, and 200 to verify HDL. Additionally, a labeling rate needs to be set. When the labeling rate is 10%, the most frequent class contains 500 labeled samples and 4500 unlabeled samples. In this work, the labeling rate is always set to 10% [Wang *et al.*, 2023].

Implementation Details. We used Wide ResNet-28-2

Table 1: Error rates (mean $\pm$ std %) on CIFAR-10, CIFAR-100, and stl-10 datasets. The best results are highlighted in **bold**, and the second-best results are underlined. Note that we evaluated our approach using 5 experiments with different random seeds.

Method	CIFAR-10			CIFAR-100		STL-10
	40 labels	250 labels	4000 labels	2500 labels	10000 labels	1000 labels
Π -Model	74.34 $\pm$ 1.76	54.26 $\pm$ 3.97	41.01 $\pm$ 0.38	57.25 $\pm$ 0.48	37.88 $\pm$ 0.11	32.78 $\pm$ 0.40
Pseudo-Labeling	74.61 $\pm$ 0.26	49.78 $\pm$ 0.43	16.09 $\pm$ 0.28	57.38 $\pm$ 0.46	36.21 $\pm$ 0.19	32.64 $\pm$ 0.71
Mean Teacher	70.09 $\pm$ 1.60	32.32 $\pm$ 2.30	9.19 $\pm$ 0.19	53.91 $\pm$ 0.57	35.83 $\pm$ 0.24	33.90 $\pm$ 1.37
UDA	29.05 $\pm$ 5.93	8.82 $\pm$ 1.08	4.88 $\pm$ 0.18	33.13 $\pm$ 0.22	24.50 $\pm$ 0.25	6.64 $\pm$ 0.17
FixMatch	13.81 $\pm$ 3.37	5.07 $\pm$ 0.65	4.26 $\pm$ 0.05	28.29 $\pm$ 0.11	22.60 $\pm$ 0.12	6.25 $\pm$ 0.33
Dash	8.93 $\pm$ 3.11	5.16 $\pm$ 0.23	4.36 $\pm$ 0.11	27.15 $\pm$ 0.22	21.88 $\pm$ 0.07	6.39 $\pm$ 0.56
MixMatch	47.54 $\pm$ 11.50	11.05 $\pm$ 0.86	6.42 $\pm$ 0.10	39.94 $\pm$ 0.37	28.31 $\pm$ 0.33	21.70 $\pm$ 0.68
+kNN-DV	47.18 $\pm$ 8.31	10.34 $\pm$ 1.31	5.91 $\pm$ 0.37	38.65 $\pm$ 0.35	27.73 $\pm$ 0.28	21.16 $\pm$ 0.53
+HDL	46.02$\pm$5.51	9.70$\pm$0.72	5.37$\pm$0.21	36.91$\pm$0.24	26.47$\pm$0.49	20.58$\pm$0.61
ReMixMatch	19.10 $\pm$ 9.64	5.44 $\pm$ 0.05	4.72 $\pm$ 0.13	27.43 $\pm$ 0.31	23.03 $\pm$ 0.56	6.74 $\pm$ 0.14
+kNN-DV	18.53 $\pm$ 5.66	4.87 $\pm$ 0.24	4.58 $\pm$ 0.09	26.85 $\pm$ 0.46	22.67 $\pm$ 0.33	6.49 $\pm$ 0.21
+HDL	17.92$\pm$3.93	4.32$\pm$0.17	4.02$\pm$0.15	26.13$\pm$0.57	22.15$\pm$0.15	6.02$\pm$0.08
FreeMatch	4.90 $\pm$ 0.04	4.88 $\pm$ 0.18	4.10 $\pm$ 0.02	26.47 $\pm$ 0.20	21.68 $\pm$ 0.03	5.63 $\pm$ 0.15
+kNN-DV	4.86 $\pm$ 0.13	4.29 $\pm$ 0.43	3.87 $\pm$ 0.11	25.96 $\pm$ 0.79	21.34 $\pm$ 0.52	5.41 $\pm$ 0.26
+HDL	4.78$\pm$0.08	3.91$\pm$0.35	3.65$\pm$0.07	25.17$\pm$0.45	20.85$\pm$0.58	5.13$\pm$0.18
DualMatch	5.75 $\pm$ 1.01	4.89 $\pm$ 0.52	3.88 $\pm$ 0.10	27.08 $\pm$ 0.23	20.78 $\pm$ 0.15	5.94 $\pm$ 0.08
+kNN-DV	5.48 $\pm$ 0.52	4.23 $\pm$ 0.29	3.75 $\pm$ 0.46	26.59 $\pm$ 0.62	20.45 $\pm$ 0.49	5.52 $\pm$ 0.16
+HDL	5.16$\pm$0.69	3.96$\pm$0.41	3.53$\pm$0.32	26.01$\pm$0.54	20.09$\pm$0.76	5.07$\pm$0.23

[Zagoruyko and Komodakis, 2016], Wide ResNet-28-2, Wide ResNet-28-8, and Wide ResNet-37-2 [He *et al.*, 2016] as the backbone networks on CIFAR-10, CIFAR-10-LT, CIFAR-100, and STL-10, respectively. For all semi-supervised learning methods, the training steps were set to 2^{20} on CIFAR-10 and stl-10, 2^{19} on CIFAR-100, and 2^{16} on CIFAR-10-LT [Wang *et al.*, 2023]. The remaining parameter settings are shown in Table 2. We have selected MixMatch, ReMixMatch, FreeMatch, and DualMatch as our baselines and employed HDL to enhance their performance. Specifically, after the four semi-supervised models have completed training, we utilize HDL to re-label the unlabeled data and continue training for an additional 2^5 steps.

Table 2: Details of the experimental setup.

Dataset	CIFAR-10	CIFAR-100	STL-10	CIFAR-10-LT
Model	WRN-28-2	WRN-28-8	WRN-37-2	WRN-28-2
Weight decay	0.0005	0.001	0.0005	0.0005
Batch size	64			
Learning rate	0.03			
SGD momentum	0.9			
EMA decay	0.999			
k	3	3	3	3

Compared Methods. Similar to HDL, we enhanced MixMatch [Berthelot *et al.*, 2019b], ReMixMatch [Berthelot *et al.*, 2019a], FreeMatch [Wang *et al.*, 2022], and DualMatch [Wang *et al.*, 2023] using kNN-DV (Section 4.2) and compared them with HDL. Additionally, we contrasted our method with common semi-supervised models, including Π -Model [Laine and Aila, 2016], Pseudo-Labeling [Lee and others, 2013a], Mean Teacher [Tarvainen and Valpola, 2017], UDA [Xie *et al.*, 2020], FixMatch [Sohn *et al.*, 2020], and

Dash [Xu *et al.*, 2021].

5.3 Setting of the hyperparameter k

The trained semi-supervised model to be improved is used to extract the image embeddings, and then the k is chosen based on the methodology presented in Section 4.4. We present the k values on each dataset in the last row of Table 2.

5.4 Results on Class-Balanced Datasets

Whether it is a consistency regularization model or a pseudo-labeling model, their common goal is to optimize the model, making its representations of similar samples more consistent and similar, thereby enhancing performance. Observing the results of kNN-DV and HDL in Table 1, it is evident that they significantly improve the performance of existing semi-supervised learning methods across various datasets. Experimental results indicate that image embeddings generated by adequately trained semi-supervised learning models are more reliable than confidence predictions. Therefore, we emphasize the importance of focusing on pseudo-label semi-supervised models based on image embeddings.

Specifically, on the CIFAR-10 with 40 labels, HDL improves the performance of MixMatch and ReMixMatch by **1.52%** and **1.18%**, respectively. On the CIFAR-10 with 250 labels, HDL yields performance gains of **1.35%**, **1.12%**, **0.97%**, and **0.93%** for MixMatch, ReMixMatch, FreeMatch, and DualMatch, respectively, with the enhanced FreeMatch achieving **state-of-the-art performance**. Furthermore, on the CIFAR-10 dataset with 4000 labels, HDL enhances the performance of MixMatch by **1.05%**, and DualMatch + HDL achieves **optimal performance**. For the CIFAR-100 and STL-10, HDL consistently enhances the performance of existing methods comprehensively. In contrast, the performance

of k-NN-DV is relatively mediocre. On the CIFAR-10 dataset with 4000 labels and the CIFAR-100 dataset with 10000 labels, k-NN-DV only brings about an average performance improvement of approximately 0.5% compared to existing methods. **It is important to note** that our approach does not require any changes to the existing method, only relabeling and continued training of the original network.

5.5 Results on Class-Imbalanced Datasets

The evaluation results on CIFAR-10-LT, which features multiple imbalance factors, are shown in Table 3. Compared to scenarios with class balance, HDL exhibits more pronounced performance on long-tailed datasets. This aligns with our expectations, as image embedding-based methods in long-tailed scenarios can reduce model bias. Specifically, at an Imbalance Factor (IF) of 50, HDL achieved performance gains of **1.7%**, **1.5%**, and **1.4%** for MixMatch, FixMatch, and DualMatch, respectively. With an IF of 100, HDL improved the performance of MixMatch by **1.5%**. However, the benefits provided by HDL diminished when the IF was increased to 200, likely due to a decrease in the number of tail-class samples participating in voting. Moreover, HDL’s performance comprehensively surpassed that of kNN-DV, further validating the soundness of our approach.

Table 3: Error rates (mean $\pm$ std %) on CIFAR-10-LT. The best results are highlighted in **bold**, and the second-best results are underlined. Note that we evaluated our approach using 5 experiments with different random seeds.

Method	IF=50	IF=100	IF=200
Pseudo-Labeling	47.5 $\pm$ 0.74	53.5 $\pm$ 1.29	58.0 $\pm$ 1.39
Mean Teacher	42.9 $\pm$ 3.00	51.9 $\pm$ 0.71	54.9 $\pm$ 1.28
MixMatch	30.9 $\pm$ 1.18	39.6 $\pm$ 2.24	45.5 $\pm$ 1.87
+kNN-DV	29.8 $\pm$ 1.27	38.8 $\pm$ 2.05	44.8 $\pm$ 1.72
+HDL	29.2$\pm$0.84	38.1$\pm$1.69	44.3$\pm$1.25
FixMatch	20.6 $\pm$ 0.65	33.7 $\pm$ 1.74	40.3 $\pm$ 0.74
+kNN-DV	19.7 $\pm$ 0.92	32.9 $\pm$ 1.31	39.9 $\pm$ 1.19
+HDL	19.1$\pm$1.13	32.1$\pm$1.52	39.6$\pm$0.82
DualMatch	19.0 $\pm$ 0.82	28.3 $\pm$ 1.38	37.3 $\pm$ 0.39
+kNN-DV	18.3 $\pm$ 1.08	27.5 $\pm$ 0.96	36.8 $\pm$ 0.68
+HDL	17.6$\pm$1.36	27.0$\pm$1.02	36.7$\pm$0.45

5.6 Additional analysis of kNN-DV and HDL

Our analysis in Section 4.2 suggests that kNN-DV, by not accounting for unlabeled samples, potentially raises the requirements for the clusterability of the image embedding sets to be valid. In this section, we set the hyperparameter k for both kNN-DV and HDL to 3 to quantify and analyze the aforementioned shortcomings of kNN-DV. In our investigation of the CIFAR-10, CIFAR-100, and STL-10 datasets, we found that the probability of clusterability for kNN-DV is consistently lower than that for HDL across all three datasets. To enhance clusterability, it becomes necessary to reduce the number of nearest neighbors. However, reducing the number of nearest neighbors can compromise the reliability of the algorithm. Therefore, it can be concluded that HDL theoretically surpasses kNN-DV in performance, a finding that is also corroborated by our experimental results.

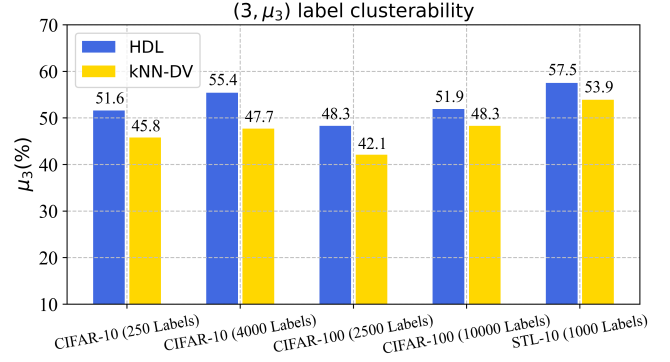


Figure 2: The probability μ_3 that clusterability holds on CIFAR-10, CIFAR-100 and STL-10 for $k = 3$.

5.7 HDL for data processing modules

When HDL is employed as a general data processing module, CLIP can be used to extract image embeddings. Given that CLIP is a universal image encoder, we hypothesize that the probability of the extracted embedding sets satisfying clusterability is roughly the same across different image datasets. We assessed the probability of label clusterability, μ_k , under different values of k on the mini-ImageNet, Clothing-1M, CIFAR-100, and Caltech 101 datasets. Subsequently, we calculated and plotted the mean and variance of μ_k in Figure 3. The experimental results reveal that the variance of μ_k is very small across all four datasets. This validates our viewpoint and implies that in practice, it is not necessary to statistically evaluate μ_k each time; instead, a reliable empirical value can be used as a substitute.

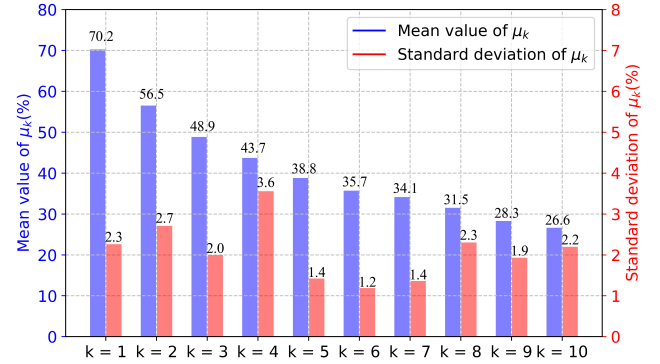


Figure 3: Under different values of k , the means and standard deviations of the probabilities that the embedding sets for the four image datasets satisfy clusterability. Please refer to the blue vertical axis for the magnitude of the mean and the red vertical axis for the magnitude of the standard deviation.

6 Conclusion

This work originates from the perspective that embeddings are more reliable than confidence in predictions. It introduces the Embedding-based Hierarchical Dynamic Labeling algorithm and significantly enhances existing semi-supervised models. This advancement holds the potential to foster improvements in the pseudo-label generation paradigm.

References

- [Berthelot *et al.*, 2019a] David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019.
- [Berthelot *et al.*, 2019b] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019.
- [Chen *et al.*, 2023] Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. Softmatch: Addressing the quantity-quality trade-off in semi-supervised learning. *arXiv preprint arXiv:2301.10921*, 2023.
- [Coates *et al.*, 2011] 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.
- [Cui *et al.*, 2019] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019.
- [Facchinetti *et al.*, 2022] Tullio Facchinetti, Guido Benetti, Davide Giuffrida, and Antonino Nocera. Slr-kit: A semi-supervised machine learning framework for systematic literature reviews. *Knowledge-Based Systems*, 251:109266, 2022.
- [He *et al.*, 2016] 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.
- [Kang *et al.*, 2019] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019.
- [Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.
- [Laine and Aila, 2016] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.
- [Lee and others, 2013a] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta, 2013.
- [Lee and others, 2013b] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896. Atlanta, 2013.
- [Liang *et al.*, 2022] Weixin Liang, Girmaw Abebe Tadesse, Daniel Ho, L Fei-Fei, Matei Zaharia, Ce Zhang, and James Zou. Advances, challenges and opportunities in creating data for trustworthy ai. *Nature Machine Intelligence*, 4(8):669–677, 2022.
- [Liu *et al.*, 2023] Zhi Liu, Weizheng Kong, Xian Peng, Zongkai Yang, Sannyuya Liu, Shiqi Liu, and Chaodong Wen. Dual-feature-embeddings-based semi-supervised learning for cognitive engagement classification in on-line course discussions. *Knowledge-Based Systems*, 259:110053, 2023.
- [Ma *et al.*, 2023a] Yanbiao Ma, Licheng Jiao, Fang Liu, Shuyuan Yang, Xu Liu, and Puhua Chen. Feature distribution representation learning based on knowledge transfer for long-tailed classification. *IEEE Transactions on Multimedia*, 2023.
- [Ma *et al.*, 2023b] Yanbiao Ma, Licheng Jiao, Fang Liu, Shuyuan Yang, Xu Liu, and Lingling Li. Curvature-balanced feature manifold learning for long-tailed classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15824–15835, 2023.
- [Ma *et al.*, 2023c] Yanbiao Ma, Licheng Jiao, Fang Liu, Shuyuan Yang, Xu Liu, and Lingling Li. Orthogonal uncertainty representation of data manifold for robust long-tailed learning. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 4848–4857, 2023.
- [Northcutt *et al.*, 2021] Curtis Northcutt, Lu Jiang, and Isaac Chuang. Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research*, 70:1373–1411, 2021.
- [Ohi *et al.*, 2020] Abu Quwsar Ohi, Muhammad F Mridha, Farisa Benta Safir, Md Abdul Hamid, and Muhammad Mostafa Monowar. Autoembedder: A semi-supervised dnn embedding system for clustering. *Knowledge-Based Systems*, 204:106190, 2020.
- [Paullada *et al.*, 2021] Amandalynne Paullada, Inioluwa Deborah Raji, Emily M Bender, Emily Denton, and Alex Hanna. Data and its (dis) contents: A survey of dataset development and use in machine learning research. *Patterns*, 2(11), 2021.
- [Pleiss *et al.*, 2020] Geoff Pleiss, Tianyi Zhang, Ethan Elenberg, and Kilian Q Weinberger. Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems*, 33:17044–17056, 2020.
- [Radford *et al.*, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

- [Sohn *et al.*, 2020] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.
- [Tarvainen and Valpola, 2017] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.
- [Wang *et al.*, 2022] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, et al. Freematch: Self-adaptive thresholding for semi-supervised learning. *arXiv preprint arXiv:2205.07246*, 2022.
- [Wang *et al.*, 2023] Cong Wang, Xiaofeng Cao, Lanzhe Guo, and Zenglin Shi. Dualmatch: Robust semi-supervised learning with dual-level interaction. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 102–119. Springer, 2023.
- [Xie *et al.*, 2020] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, 33:6256–6268, 2020.
- [Xu *et al.*, 2021] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yufeng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *International Conference on Machine Learning*, pages 11525–11536. PMLR, 2021.
- [Xu *et al.*, 2023] Hao Xu, Hui Xiao, Huazheng Hao, Li Dong, Xiaojie Qiu, and Chengbin Peng. Semi-supervised learning with pseudo-negative labels for image classification. *Knowledge-Based Systems*, 260:110166, 2023.
- [Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.
- [Zhang *et al.*, 2021] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34:18408–18419, 2021.
- [Zhou *et al.*, 2020] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9719–9728, 2020.
- [Zhu *et al.*, 2022] Zhaowei Zhu, Zihao Dong, and Yang Liu. Detecting corrupted labels without training a model to predict. In *International conference on machine learning*, pages 27412–27427. PMLR, 2022.