PivotAlign: Improve Semi-Supervised Learning by Learning Intra-Class Heterogeneity and Aligning with Pivots

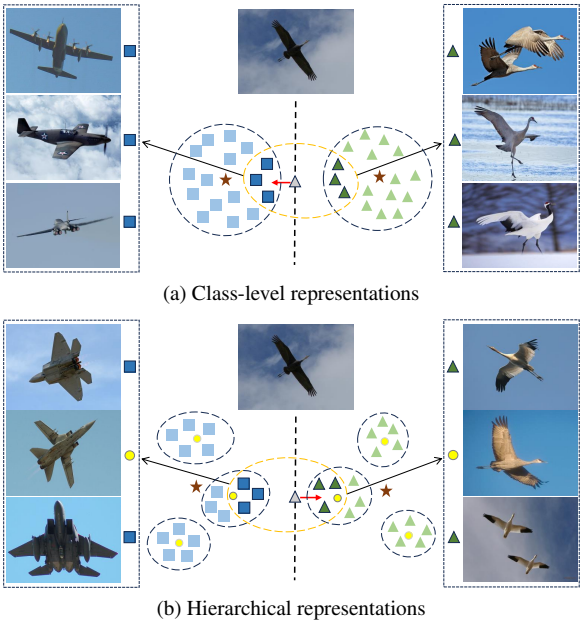
Lingjie Yi[1], Tao Sun[1], Yikai Zhang[2], Songzhu Zheng[2], Weimin Lyu[1], Haibin Ling[1], Chao Chen[1]
[1] Stony Brook University, [2] Morgan Stanley
{chris.yi, weimin.lyu, chao.chen.1}@stonybrook.edu,
{tao, hling}@cs.stonybrook.edu,
{yikai.zhang, songzhu.zheng}@morganstanley.com

Abstract

Self-supervised learning plays an important role in current state-of-the-art semi-supervised learning (SSL) methods. These methods learn inter-class heterogeneity among data and generate pseudo-labels based on class level representations. However, they often neglect intra-class heterogeneity, resulting in the under-exploitation of finer-grained semantic relationships within classes. To address this limitation, we introduce PivotAlign, a novel SSL approach that aims to 1) learn hierarchical representations to detect both inter-class and intra-class semantic relationships, and 2) refine pseudo-labels based on learned representations with a class-debiasing strategy. Specifically, we first learn a set of pivots as sub-prototypes of classes. We then train representations so that features align with the assigned pivot and are hierarchically grouped based on both inter-class and intra-class heterogeneity. This allows us to capture both inter-class and intra-class semantic relationships among data and leverage them to better assign and refine pseudo-labels. Additionally, since SSL methods are prone to bias toward classes that are easier to learn, we further re-balance class predictions to alleviate this class bias. We demonstrate the effectiveness of PivotAlign on various SSL benchmarks, where PivotAlign achieves state-of-the-art performances. The source code will be released upon publication of the work.

1. Introduction

Deep learning has achieved significant success in various fields by training on large-scale datasets with human annotations [23, 42]. Labeling these datasets, however, can be very expensive, especially when it requires expert knowledge or even infeasible due to privacy concerns. Recently, deep semi-supervised learning (SSL) [3, 4, 10, 24, 27, 30, 34, 37–39, 44, 46–48, 53] has emerged as an effective solution to reduce dependence on costly labeled datasets. SSL methods train models using a small amount of labeled data combined with a vast amount of unlabeled data.



(a) Class-level representations

(b) Hierarchical representations

Figure 1. A simple classification task. Colored squares and triangles are labeled data. Brown stars are class centers. Yellow dots are pivots. Black dashed lines are the decision boundaries. Distance indicates semantic closeness. (a) With class-level representations, an unlabeled flying bird (grey triangle) may be mislabeled as an airplane when compared to its nearby labeled birds and airplanes (in the orange dashed circle). (b) With hierarchical representations, where all flying birds are grouped together, finer comparisons to samples sharing intra-class semantic information are allowed, leading to more accurate label refinement (directions of refinement are shown as red arrows).

Self-supervised learning [8, 11, 13, 20, 40, 52] and pseudo-labeling [27] are two powerful tools for SSL. Many efforts have been made to incorporate self-supervised learning into SSL [17, 28, 33, 48, 58], aiming to learn more separated class-

level feature clusters to improve model predictions. Some works further design various methods to refine pseudo-labels by leveraging information from neighbouring (labeled [28, 33] or unlabeled [58]) samples in the feature space.

These self-supervised learning based SSL methods, while leveraging inter-class heterogeneity, neglect intra-class heterogeneity. They train unlabeled data with the unsupervised contrastive loss [8, 11] and retain their instance-specific information. Whereas labeled data are trained with the supervised contrastive loss [25] and their representations gradually shrink to class centers [18]. This shrinkage process forces labeled data to drop intra-class semantic information and only retain class-level semantic information. In SSL, models rely on semantic relationships between labeled and unlabeled data to assign and refine pseudo-labels for unlabeled data. Neglecting intra-class heterogeneity results in inferior semantic relationships, leading to suboptimal pseudo-label assignments and refinements.

In this paper, we introduce a novel self-supervised learning method, **PivotAlign**. It aims to 1) learn hierarchical representations that capture both inter-class and intra-class heterogeneity and 2) leverage these semantic relationships to refine pseudo-labels. To detect both inter-class and intra-class heterogeneity, we learn a set of pivots, each of which acts as a sub-prototype of the classes and carries a specific type of intra-class semantic information. We then assign each sample to the pivot that shares the most relevant semantic information. This is achieved through online pivot assignment and pivot learning (described in Sec. 4.1). The online nature of this method allows our model to work with mini-batch training and scale efficiently to large datasets (*e.g.*, ImageNet [16]), while offline methods (such as K-Means) based models [21, 49] struggle with batch-wise training and scalability. Importantly, to maintain intra-class heterogeneity, our method prevents the pivots from collapsing into their respective class centers throughout the training process.

To capture both inter-class and intra-class semantic relationships in feature representations, we expect samples assigned to the same pivot to align with it and group together, forming sub-groups. Additionally, sub-groups with similar inter-class semantic information should remain close to each other, forming super-groups that are positioned away from the decision boundaries. The entire feature space forms a hierarchical structure, maintaining intra-class diversity while ensuring inter-class separation. This unique type of representation distinguishes our model from other clustering, sub-clustering and prototype based models [17, 21, 33, 49]. We achieve this through hierarchical representation learning (described in Sec. 4.2).

With the learned hierarchical representations, our model refines the pseudo-label for each unlabeled sample by using labeled samples that share the most inter-class and intra-class similarities as references. Compared to previous methods [28, 33, 49, 58], hierarchical representations enable more accurate label refinement. In Fig. 1, we present an illustrative example showing how hierarchical representations help correct the pseudo-label of a flying bird, whereas relying solely on class-level representations fails. Furthermore, it has been empirically observed that pseudo-labels often bias toward classes that are easier to learn [30]. To mitigate this issue, we incorporate a debiasing strategy into the label refinement process (described in Sec. 4.3).

We conduct experiments on fives commonly-used datasets, and the results show that our model outperforms SOTA performance compared to popular SSL baselines.

Contributions of our work are summarized as the follows:

- We propose a novel SSL model that leverages both inter-class and intra-class heterogeneity for representation learning and label refinement.
- We introduce an online method to detect hierarchical semantic relationships that works with mini-batch training and scales efficiently to large datasets.
- We design a contrasting method to learn hierarchical representations and leverage them to refine pseudo-labels with a debiasing strategy to alleviate class bias.
- We demonstrate the effectiveness of PivotAlign across widely adopted SSL datasets. Our method outperforms SOTA baselines.

2. Related Works

2.1. Self-supervised Learning

Instance-based self-supervised learning focuses on pulling positive samples together and pushing negative samples away by optimizing the InfoNCE loss [45]. SimCLR [11] uses differently augmented views of the same input as positives and all other samples as negatives. SupCon [25] extends SimCLR [11] by using all instances from the same class as positives. These methods benefit from large batch sizes.

Cluster-based self-supervised learning comprises two steps: clustering and discrimination. The former generates cluster labels and the latter trains embeddings. DeepCluster [6] adopts K-means to generate cluster assignments. SeLa [2] transforms the balanced clustering task into an optimal transport problem. CoKe [36] modifies the clustering phase with an online constrained K-means. Similarly, SwAV [7] solves the clustering problem with optimal transport in an online manner. Following [2, 7], our method transforms hierarchical clustering into an optimal transport problem; but it updates pivots and optimizes representations (unlike other cluster-based methods) via instance-based contrastive learning.

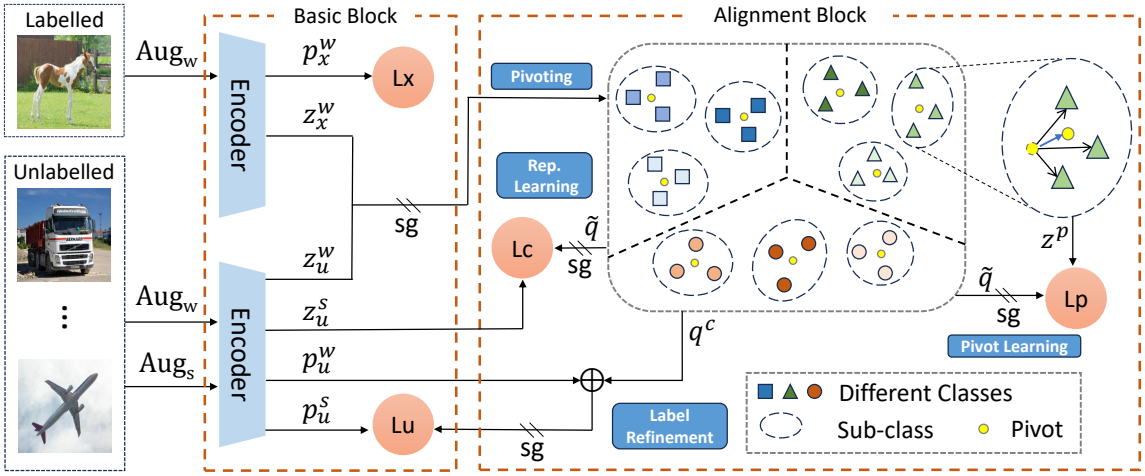


Figure 2. Framework of PivotAlign. Embeddings of weakly augmented labeled data z_x^w and confident unlabeled data z_u^w are used to calculate pivot assignments $\tilde{q}$ based on their affinities to pivots z^p. By taking pivot assignments as labels, hierarchical representations can be learned via the weighted contrastive loss L_c. With generated assignments $\tilde{q}$ and updating embeddings z^w, pivots z^p can be further updated via the pivot contrastive loss L_p. Each unlabeled sample, by leveraging intra-class semantic relations with nearby pivots, gets a cluster label p^c to refine its pseudo-label, which serve as the training target in unsupervised loss L_u. '$\mathtt{sg}$' denotes stop gradient.

2.2. Semi-supervised Learning

Pseudo labeling [27] uses model predictions to generate hard labels for unlabeled data to train against. This method is susceptible to confirmation bias [1] where a model is distorted by training on incorrect pseudo-labels. To overcome this problem, FixMatch [37] incorporates confidence thresholding such that pseudo-labels are only retained when their confidence surpasses a pre-defined threshold. This technique, while maintaining high pseudo-label quality, limits data utilization. Inspired by Dash [47], many studies [10, 44, 53] adjust the threshold dynamically to balance pseudo-label quality and quantity.

Self-supervised learning is also incorporated into the SSL framework. CoMatch [28] simultaneously optimizes representations and pseudo-labels. SimMatch [57, 58] extends consistency regularization in representation learning. Closely related to our work, ProtoCon [33] adopts online clustering proposed in [36] to find cluster labels and use them to refine pseudo-labels. However, unlike our method that learns hierarchical fine-grained representations, it optimizes representations via class-level prototypical loss. Suave [17] is a semi-supervised version of [7]. It improves performance by applying data augmentation techniques like MixUp [54], CutMix [50] and Multi-Crop [7].

Other semi-supervised tasks. In addition to classification task, clustering/sub-clustering/prototype based semi-supervised learning techniques are developed in other areas. [21] use sub-cluster to constrain segmentation process. [49] replies on prototype for class-level grouping. But none of these methods detect clusters or sub-clusters online, nor learn hierarchical fine-grained representations.

3. Preliminaries

Semi-Supervised Learning. Following the standard SSL framework, let $\mathcal{X} = \{(x_i, y_i) : i = 1, \dots, B\}$ be one batch of labeled data of size B from n different classes, where x_i is a labeled input and y_i is the ground truth label. Let $\mathcal{U} = \{u_i : i = 1, \dots, \mu B\}$ be one batch of unlabeled data of size μB where u_i is an unlabeled input and μ is the size ratio of $\mathcal{U}$ to $\mathcal{X}$. Following [37], weak $\mathrm{Aug}_w(\cdot)$ and strong $\mathrm{Aug}_s(\cdot)$ augmentations are applied to these inputs. Model predictions of augmented inputs are denoted as $p_i^{w/s} = p(y|\,\mathrm{Aug}_{w/s}(x_i))$. Labeled data are optimized using the supervised loss:

$$L_x = \frac{1}{B}\sum_{i=1}^{B} \mathrm{H}(y_i, p_i^w) \tag{1}$$

where $\mathrm{H}(y, p)$ denotes the cross-entropy. Unlabeled data are trained via the unsupervised loss:

$$L_u = \frac{1}{\mu B}\sum_{i=1}^{\mu B} \mathbb{1}(\max(q_i) \geq \tau)\mathrm{H}(q_i, p_i^s) \tag{2}$$

τ is a pre-defined threshold. Following [11, 28, 33], we denote $q_i = DA(p_i^w)$ as the soft label, which is the model prediction after distribution alignment [3], $y_i^u = \mathrm{argmax}(q_i)$ as the hard label and use the soft label instead of the hard one in Eq. (2).

Optimal Transport. For a typical OT problem, given two marginal distributions $r \in \mathbb{R}_+^N$ and $c \in \mathbb{R}_+^V$, a cost matrix M, assigning N r-distributed samples to c-distributed clusters involves finding a joint distribution Q which minimizes Wasserstein distance between Q and M with a regularization:

$$\min_{Q \in U(r,c)} \langle Q, M \rangle + \epsilon \mathrm{KL}(Q \| r c^\top) \tag{3}$$

where $\langle \cdot \rangle$ is the Frobenius dot-product between two matrices, KL is the Kullback-Leibler divergence and ϵ controls the smoothness of Q. The matrix Q to is an element of the transportation polytope:

$$U(r,c) := \left\{ Q \in \mathbb{R}_+^{N \times V} \mid Q\mathbf{1} = r, Q^\top \mathbf{1} = c \right\} \tag{4}$$

Here $\mathbf{1}$ denotes a vector of all ones with corresponding dimensions. r and c are the marginal distributions of data and clusters, respectively, defined in the uniform way:

$$r = \frac{1}{N} \cdot \mathbf{1}, \quad c = \frac{1}{V} \cdot \mathbf{1} \tag{5}$$

This problem can be solved with the Sinkhorn-Knopp algorithm [15] as:

$$Q = \mathrm{diag}(\alpha) \exp^{\frac{M}{\epsilon}} \mathrm{diag}(\beta) \tag{6}$$

where $\alpha \in \mathbb{R}^N$ and $\beta \in \mathbb{R}^V$ are scaling vectors.

4. PivotAlgin

The model framework and training pipeline are depicted in Fig. 2. Our model consists of: 1) a base encoder $f : \mathcal{X} \to h$ that extracts latent representations; 2) a prediction head $l : h \to p$ that produces model predictions p; 3) a projection head $g : h \to z$ that generates low-dimensional embeddings and we assume all zs being normalized z ($z^{w/s}$ denote the embedding of a weakly/strongly augmented input); 4) V learnable pivots $z^p_{1:V}$ that have the same dimension with $z^{w/s}$ and are randomly initialized.

4.1. Intra-class Heterogeneity Detection

Online Pivot Assignments. Now, suppose we have V pivots, evenly distributed across all K classes and thus each class has $\frac{V}{K}$ pivots. Denote y^p_v as the class label of v^{th} pivot. Then each pivot is a sub-prototype of class y^p_v. To capture intra-class semantic relationships, we need to know class labels of data. Therefore, we can only use labeled data and unlabeled data with high confidence from the confident set $\mathcal{C} = \{i : \max(q_i) \geq \tau\}$. For a sample $i \in \mathcal{C}$, define its affinity to k^{th} pivot as:

$$A^w_{iv} = (z^w_i \cdot z^p_v)\, \mathbb{1}(y^u_i = y^p_v) \tag{7}$$

Recall that y^u_i is the pseudo-label of sample i. $\mathbb{1}(y^u_i = y^p_v)$ ensures that data from one class can only be assigned to one of pivots from this class, maintaining correct intra-class semantic relationships. As SSL models are prone to bias toward easy classes [9], the class proportion within set $\mathcal{C}$ is also distorted. To maintain balanced assignments among different classes, we adjust the weight of each sample according to the class proportion:

$$r_{1i} = \frac{1}{K} \cdot \frac{1}{\sum_{j \in \mathcal{C}} \mathbb{1}(y^u_j = y_i)} \tag{8}$$

Thus for $i \in \mathcal{C}$, the pivot assignment $\tilde{q}_i$ can be solved by minimizing the objective below by substituting M with $-A^w$ and r with r_1 in Eq. (3):

$$\min_{\tilde{Q} \in U(r_1,c)} \langle \tilde{Q}, -A^w \rangle + \epsilon \mathrm{KL}\left(\tilde{Q} \| r_1 c^\top\right) \tag{9}$$

Note that keeping c remains uniform in Eq. (9) is crucial to prevent pivot assignments from degenerating. Also, in practice, the size of the confident set in one mini-batch might not be sufficient for good assignments among K pivots. We use the confident data from previous batches stored in a memory bank to enlarge the size of the confident set. In this case, the confident set is the union of the confident set from current batch $\mathcal{C}^b$ and the confident set from the memory bank $\mathcal{C}^m$, thus $\mathcal{C} = \mathcal{C}^b \cup \mathcal{C}^m$. The memory bank is updated in the same way as [22, 28].

For an unlabeled sample with low confidence, $i \notin \mathcal{C}$, it is not clear which pivot this sample should be assigned to exactly, we set its assignment in the following way:

$$A^w_{iv} = (z^w_i \cdot z^v_k)\, q_{iy^p_v}, \text{ where } \tilde{q}_{ik} = \frac{A^w_{iv}}{\sum_{j=1}^V A^w_{ij}} \tag{10}$$

Online Pivot Learning. With the pivot assignments, we now want to learn a set of good pivots to serve as sub-prototypes that carry intra-class semantic information. In our work, we use weakly augmented data to update pivots, ensuring a smooth change between two consecutive iterations. We keep $\tilde{q}_i$ and z_i frozen during the pivot learning stage. Recall that pivot assignments are identified using samples from the confident set $\mathcal{C}$, we use the same set of samples to update pivots. In each iteration, pivots are optimized via the following pivot contrastive loss:

$$L_p = \frac{1}{K} \sum_k \sum_{i \in \mathcal{C}} -\frac{\tilde{q}_{ik}}{\sum_{b \in \mathcal{C}} \tilde{q}_{bk}} \log \frac{\exp(z^w_i \cdot z^p_k / t)}{\sum_{i' \in \mathcal{C}} \exp(z^w_{i'} \cdot z^p_k / t)} \tag{11}$$

Intuitively, we treat each pivot as an anchor, confident data assigned to this pivot as positives, and all other confident data as negatives to contrast with. Here, we use the hard version of $\tilde{q}$ to ensure separations among pivots and prevent them from shrinking to class centers. We verify this choice in the Exp. 7 of ablation study (see Tab. 3). Note that no gradient is passed to z^w_i and $\tilde{q}_i$.

Table 1. Top-1 accuracy on CIFAR-10/100, STL-10 and SVHN with varying labeled set sizes over 3 different random splits.

Method	CIFAR-10			CIFAR-100			STL-10		SVHN
	40 labels	250 labels	4000 labels	400 labels	2500 labels	10000 labels	40 labels	1000 labels	1000 labels
UDA [46]	89.38 ± 3.75	94.84 ± 0.06	95.71 ± 0.07	53.61 ± 1.59	72.27 ± 0.21	77.51 ± 0.23	62.58 ± 8.44	93.36 ± 0.17	**98.11** ± 0.01
MixMatch [4]	63.81 ± 6.48	86.37 ± 0.59	93.34 ± 0.26	32.41 ± 0.66	60.24 ± 0.48	72.22 ± 0.29	45.07 ± 0.96	78.30 ± 0.68	96.31 ± 0.37
ReMixMatch [3]	90.12 ± 1.03	93.70 ± 0.05	95.16 ± 0.01	57.25 ± 1.05	73.97 ± 0.35	76.97 ± 0.56	67.88 ± 6.24	93.26 ± 0.14	94.84 ± 0.31
FixMatch [37]	92.53 ± 0.28	95.14 ± 0.05	95.79 ± 0.08	53.58 ± 0.82	71.97 ± 0.16	77.80 ± 0.12	64.03 ± 4.14	93.75 ± 0.33	98.04 ± 0.03
Dash [47]	91.07 ± 3.11	94.84 ± 0.23	95.64 ± 0.11	55.18 ± 0.96	72.85 ± 0.22	78.12 ± 0.07	65.48 ± 4.30	93.61 ± 0.56	98.03 ± 0.01
FlexMatch [53]	95.03 ± 0.06	95.02 ± 0.09	95.81 ± 0.01	60.06 ± 1.62	73.51 ± 0.20	78.10 ± 0.15	70.85 ± 4.16	94.23 ± 0.18	93.28 ± 0.30
FreeMatch [44]	95.10 ± 0.04	95.12 ± 0.18	95.90 ± 0.02	62.02 ± 0.42	73.53 ± 0.20	78.32 ± 0.03	84.44 ± 0.55	94.37 ± 0.15	98.04 ± 0.03
NP-Match [43]	95.09 ± 0.04	95.04 ± 0.06	95.89 ± 0.02	61.09 ± 0.99	73.97 ± 0.26	78.78 ± 0.13	85.80 ± 0.67	94.41 ± 0.24	–
RelationMatch [55]	93.13 ± 0.12	95.15 ± 0.04	95.78 ± 0.06	54.21 ± 0.59	72.10 ± 0.15	77.82 ± 0.13	66.58 ± 3.92	93.92 ± 0.29	–
CoMatch [28]	93.09 ± 1.39	95.09 ± 0.33	95.44 ± 0.20	58.11 ± 2.34	71.63 ± 0.35	79.14 ± 0.36	86.26 ± 4.20	94.29 ± 0.08	97.99 ± 0.04
SimMatch [58]	94.40 ± 1.37	95.16 ± 0.39	96.04 ± 0.01	62.19 ± 2.21	74.93 ± 0.32	79.42 ± 0.11	83.02 ± 4.34	94.26 ± 0.31	97.95 ± 0.05
PivotAlign (Ours)	**95.28** ± 0.24	**95.54** ± 0.02	**96.39** ± 0.09	**63.25** ± 0.65	**75.35** ± 0.36	**79.83** ± 0.20	**86.79** ± 3.10	**94.43** ± 0.20	98.05 ± 0.03

4.2. Hierarchical Representation Learning

In cluster-based self-supervised learning, given pivots and pivot assignments, input embeddings can be optimized via Eq. (11). As discussed in Sec. 2.1, instance-based self-supervised learning benefits from larger batch sizes or more samples to contrast with. If we view the discrimination step of cluster-based self-supervised learning as instance contrast with cluster centroids, the batch size is fixed as the number of cluster, or K in our case. This fixed and small batch size limits us from learning good hierarchical representations by training z_i via Eq. (11). We empirically prove this analysis in the Exp. 6 of ablation study (see Tab. 3).

To address this constraint, we adopt ideas from MoCo and SupCon. More specifically, for each input i in a mini-batch, we conduct supervised contrast with all data stored in the memory bank where pivot assignments are used as class labels. This allows us to learn desired hierarchical representations unlike all previous SSL methods. The representations are optimized via the following contrastive loss:

$$L_c = \frac{1}{\mu B} \sum_{i=1}^{\mu B} \sum_{j} -\frac{w_{ij}}{\sum_b w_{ib}} \log \frac{\exp(z_i^s \cdot z_j^m / t)}{\sum_{j'} \exp(z_i^s \cdot z_{j'}^m / t)} \tag{12}$$

where z^m denotes features stored in the memory bank. And contrastive weight is define as:

$$w_{ij} = \begin{cases} \tilde{q}_i \cdot \tilde{q}_j & \text{if } j \in \mathcal{C}^m \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

Note that as described in Sec. 4.1, pivot assignments $\tilde{q}_j$ of confident points in the memory bank are calculated together with $\tilde{q}_i$. Since no pivot assignments are generated for unconfident samples in the memory bank, they are always taken as negatives. Here, pivot assignments remains soft. This prevents sub-clusters with similar inter-class semantic information being pushed away and thus ensure the hierarchical structure to reflect both inter-class and intra-class semantic relationships.

4.3. Label Refinement

With ideal hierarchical representations, we are able to leverage intra-class semantic relationships to refine pseudo-labels. Intuitively, given an unlabeled sample i, we can guess its label by comparing it with pivots from nearby sub-clusters. To achieve this guessing process, we first calculate a cluster prediction based on its similarities to pivots:

$$p_i^a = \sum_k a_{ik} \cdot \text{Vec}(y_v^p), \text{ where } a_{ik} = \frac{\exp(z_i^w \cdot z_k^p / t)}{\sum_{k'} \exp(z_i^w \cdot z_{k'}^p / t)} \tag{14}$$

where $\text{Vec}(y_v^p)$ denotes the one-hot encoding of the class label of k^{th} pivots. p^a can be directly used for label refinement (see Exp. 4 in Tab. 3), however, similar to model predictions, p^a also bias toward easy classes. We introduce a debiasing strategy to alleviate this problem. Based on P^a whose i^{th} row is p_i^a, we conduct a balanced optimal transport to achieve the debiasing purpose. More specifically, we replace M with $-\log(P^a)$ in Eq. (3) and obtain the following objective function:

$$\min_{Q^c \in U(r,c)} \langle Q^c, -\log(P^a) \rangle + \epsilon \text{KL}(Q^c \| r c^\top) \tag{15}$$

We can obtain a debiased cluster label q_i^c for data i by solving Eq. (15). Similar to Sec. 4.1, we enlarge the sample size for OT by using features stored in the memory bank. Finally, we refine pseudo-labels with cluster labels:

$$\bar{q}_i = \alpha \cdot q_i + (1 - \alpha) \cdot q_i^c \tag{16}$$

And $\bar{q}$ is used to replace q in Eq. (2). The overall training objective for our model is :

$$L = L_x + \lambda_u L_u + \lambda_p L_p + \lambda_c L_c \tag{17}$$

Table 2. Top-1 Accuracy for ImageNet-1k with 1% and 10% labeled examples.

	Method	Pretraining Epoch	Semi-Supervised Epoch	Top-1 Label fraction 1 %	Top-1 Label fraction 10 %	Top-5 Label fraction 1 %	Top-5 Label fraction 10 %
Semi-Supervised	VAT+EntMin [19, 32]	-	-	-	68.8	-	88.5
	S4L-Rotation [5]	-	200	-	53.4	-	83.8
	UDA [46]	-	-	-	68.8	-	88.5
	FixMatch [37]	-	300	51.2	71.5	-	89.1
	CoMatch [28]	-	400	66.0	73.6	86.4	91.6
	SimMatch [58]	-	400	67.2	74.4	87.1	91.6
	ProtoCon [33]	-	300	65.5	73.1	-	-
	ProtoCon + EMAN	-	850	67.2	73.5	-	-
Self-Supervised	PIRL [31]	800	20	30.7	60.4	57.2	83.8
	PCL [29]	200	20	-	-	75.3	85.6
	SimCLR [11]	1000	60	48.3	65.6	75.5	87.8
	SimCLR V2 [12]	800	60	57.9	68.4	82.5	89.2
	BYOL [20]	1000	50	53.2	68.8	78.4	89.0
	SwAV [7]	800	20	53.9	70.2	78.5	89.9
	WCL [56]	800	60	65.0	72.0	86.3	91.2
	Suave [17]	800	100	66.2	75.0	-	-
	PivotAlign (Ours)	-	400	**69.6**	**75.5**	**88.6**	**92.0**

5. Experiments

5.1. CIFAR-10 & CIFAR-100 & STL-10 & SVHN

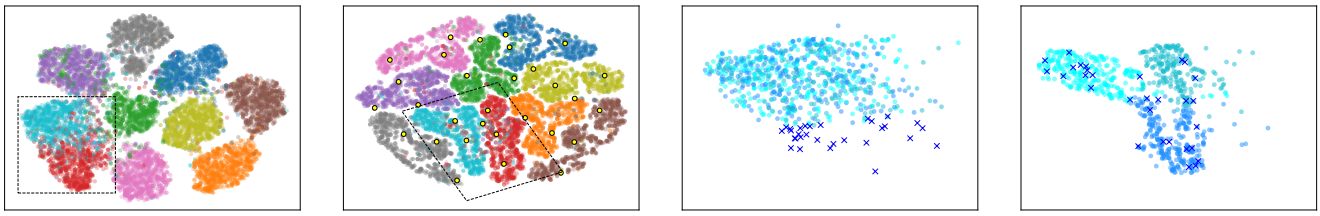
Datasets. We first evaluate PivotAlign on several small-scale SSL datasets. **CIFAR-10/100** [26] is a collection of 60k 32×32 color images in 10/100 classes. Each class has 5000/500 training samples and 1000/100 testing samples respectively. We conduct a class-balanced selection of 40, 250, and 4,000 samples from the training set as the labeled data and the remaining as the unlabeled data for CIFAR-10. For CIFAR-100, we select 400, 2500, and 10,000 samples in the same way as labeled data. **STL-10** [14] contains 5K labeled images, 100K unlabeled images, and 8K test images. We randomly select 40 and 1,000 labeled images as the labeled data and take all the remaining labeled and unlabeled images as the unlabeled data. **SVHN** [35] consists of 32×32 cropped digits from house numbers in Google Street View images. It contains about 73k training images, 26k testing images and 531k extra images. We combine the training and extra sets and randomly select 1,000 from them as the label data and use the rest as the unlabeled data.

Implementation Details. Our implementation follows [28, 58]. For the model backbone, we adopt WRN-28-2 [51] for CIFAR-10 and SVHN, WRN-28-8 for CIFAR-100, and WRN-37-2 for STL-10. The projection head is a 2-layer MLP that generates 128-dimensional embeddings. Our model is trained with a SGD optimizer with Nesterov momentum. The momentum is 0.9 and the weight decay is $1e^{-3}$ for all datasets except $5e^{-4}$ for CIFAR-100. The learning rate is 0.03 with a cosine decay schedule of $0.03 \cdot \cos\left(\frac{7\pi n}{16N}\right)$, where n is the current training step and $N = 2^{20}$ is the total number of iterations. For data augmentations, we use the same strategies as [37]. To ensure a fair comparison, we adopt hyperparameters from [28]. We set $B = 64$, $\mu = 7$, $\tau = 0.95$, $\alpha = 0.9$, $t = 0.1$, $\epsilon = 0.05$, $\lambda_u = 1$, $\lambda_c = 1$, $\lambda_p = 1$, and the size of the memory bank as 5120.

Results. Tab. 1 shows experiment results on CIFAR-10, CIFAR-100, STL-10 and SVHN. For baselines, we select classical models [3, 4, 37, 46], dynamic thresholding models [44, 47, 53], representation learning models [28, 58] and two recent works [43, 55]. We report the mean and standard deviation of top-1 testing accuracy over three runs. We can see that PivotAlign outperforms baseline models across almost all settings except SVHN with 1000 labeled data, where PivotAlign is the second best and very close to the best model. When compared with the representation earning baseline CoMatch [28], PivotAlign boosts performance on CIFAR-10 with 40 labels by 2.19 %, CIFAR-100 with 400 labels by 5.14 % and obtains non-trivial improvements under all other settings.

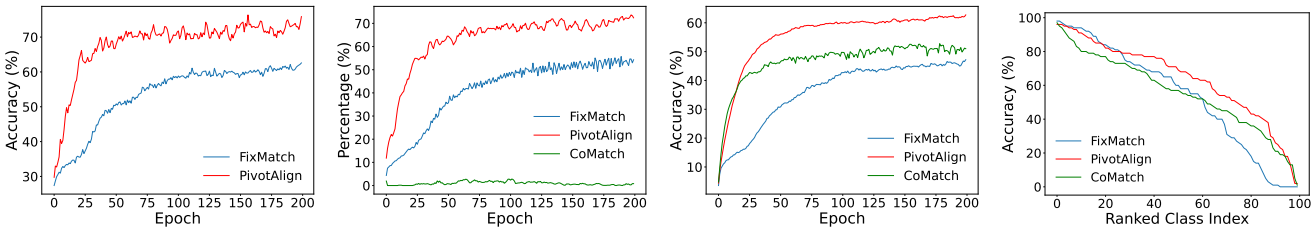
5.2. ImageNet-1k

Datasets. We also evaluate PivotAlign on a large-scale dataset. ImageNet-1k [16] consists of approximately 1.2M high-resolution training images and 50k validation images from 1000 distinct classes. Following the same protocol as [28, 58], we use 1% or 10% of samples, which is 13 or 128 samples per class from the training set as labeled data and take the rest as unlabeled data. For a fair comparison, instead of sampling randomly, we use a predefined labeled and unlabeled splits that have been widely adopted in previous works [7, 11, 12, 56].



(a) FixMatch (b) PivotAlign (c) FixMatch class 5 (d) PivotAlign class 5

Figure 3. t-SNE visualization of CIFAR-10 training set. (a) and (b) show representations learned by FixMatch [37] and PivotAlign. Each color denotes one of the 10 distinct classes. Yellow dots denote learned pivots. Dashed boxes denote two classes (class 3 and class 5) that are not separated well in FixMatch. (c) and (d) are zoom-in-views of class 5, and data are colored by three sub-clusters (from light blue to dark blue) identified by PivotAlign. It can be seen that samples with similar intra-class semantic information are dispersed randomly within the class-level cluster in (c) but are grouped together in (d). '$\times$' denotes samples mislabeled by FixMatch but corrected by PivotAlign.



(a) (b) (c) (d)

Figure 4. Results of CIFAR-100 with 400 labeled data. (a) Pseudo label accuracy. (CoMatch [28] is omitted due to insufficient confident samples.) (b) Proportion of confident samples. (c) Overall testing accuracy. (d) Class-wise accuracies ranked from highest to lowest.

Implementation Details. Our implementation follows [7, 28, 58]. We adopt ResNet50 [23] as the model backbone. The projection head is a 2-layer MLP that generates 128-dimensional embeddings. The model is optimized with a SGD optimizer with Nesterov momentum. The momentum is 0.9 and the weight decay is $1e^{-4}$. For the learning rate, the initial rate is 0.03 and it will be decayed to 0 with the cosine scheduler. For data augmentations, we adopt the weak augmentation strategy from [37] and the strong augmentation strategy from [7]. Instead of using six augmented views in the regular multi-crop strategy, we use one large plus four small strongly augmented views together with the one large weakly augmented views. To ensure a fair comparison, we adopt hyperparameters from [28]. We set $B = 160$, $\mu = 2$, $\tau = 0.7$, $\alpha = 0.9$, $t = 0.1$, $\epsilon = 0.05$ and the size of the memory bank as 30K for both settings. And we set $\lambda_u = 10$, $\lambda_c = 10$, $\lambda_p = 10$ for 1% setting and $\lambda_u = 5$, $\lambda_c = 5$, $\lambda_p = 5$ for 10% setting Also, for large datasets like ImageNet-1k, where there is higher diversity within each sub-cluster, adding more 'reference points' can better capture the data diversity. To achieve it, we maintain one more memory bank to save all labeled data and use pivots and labeled data together as 'reference points'.

Results. Tab. 2 shows experiment results on ImageNet-1k. Our baselines include two types of models. The first type is semi-supervised learning models [5, 19, 27, 28, 32, 37, 46, 58]. The second type is self-supervised pre-training models with supervised fine-tuning [7, 11, 12, 20, 29, 31, 56]. We show that PivotAlign achieves state-of-the-art performance in both settings where 1% and 10% labels are available. When compared with two closely related cluster-based self-supervised learning works ProtoCon [33] and Suave [17], PivotAlign improves top-1 accuracy under 1% and 10% settings by 2.4%, 2.0%, 3.4% and 0.5% respectively.

5.3. Analysis

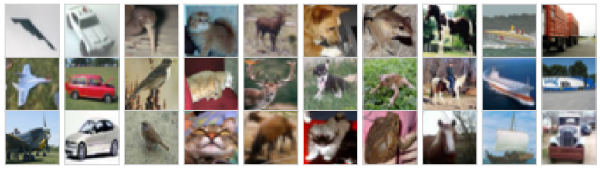


Figure 5. Images of 30 pivots of CIFAR-10 training set identified by PivotAlign.

Hierarchical Fine-grained Representations. In Fig. 3, we visualize the learned representations of CIFAR-10 training images with 40 labeled data using t-SNE [41] (more details are in the appendix). It's clearly shown in Fig. 3b that our model learns hierarchical fine-grained representations. Fig. 5 depicts images of pivots (yellow dots in Fig. 3b). We can see that pivots from the same class serve as sub-prototypes of that class (*e.g.*, pivots in the airplane class are bottom view, top view, and view from the ground respectively). Fig. 3d

Exp.	1	2	3	4	5	6	7
L_p w/ soft $\tilde{q}$							✓
L_p w/ hard $\tilde{q}$		✓	✓	✓	✓	✓	
L_c		✓	✓	✓	✓		✓
label prop			✓				
label refine				✓	✓	✓	✓
debiasing					✓	✓	✓
Accuracy	92.53	94.02	94.52	94.77	**95.28**	93.56	94.08
Δ		+1.49	+1.99	+2.24	**+2.75**	+1.03	+1.55

Table 3. Ablating model components.

shows that our model clusters samples with similar intra-class information together while in FixMatch [37] these samples spread arbitrarily within the class-level cluster. Sub-clusters with similar inter-class semantic information also stay close to each other, whereas they could be split apart in other cluster-based methods. Fig. 3d further shows samples mislabeled by FixMatch but corrected by PivotAlign (the grey triangle in Fig. 1), which indicates that learned hierarchical fine-grained representations improve pseudo-labels.

Pseudo-label quality. Fig. 4 depicts the results of models trained on CIFAR-100 with 400 labeled data. Fig. 4a demonstrates that the pseudo-labels of confident samples generated by PivotAlign are more accurate. Fig. 4b shows PivotAlign generates a higher number of confident pseudo-labels while CoMatch [28] sometimes over-smooths pseudo-labels, limiting their confidence. Fig. 4c validates that the higher-quality pseudo-labels generated by PivotAlign lead to better test performance.

Class-wise Performance. Fig. 4d shows the ranked testing accuracy. It demonstrates 'class bias' [30], where a SSL model (*e.g.* [37]) learns 'easy' classes well but fails to learn some 'difficult' classes, with testing accuracy even dropping to zero. Compared to FixMatch [37], CoMatch [28] achieves higher overall accuracy and higher accuracy for 'difficult' classes, but at the cost of 'easy' classes. In contrast, PivotAlign boosts performance in almost every class and alleviates class bias.

5.4. Ablation Study

We evaluate the design of PivotAlign with ablation study on CIFAR-10 with 40 labeled data. Similar to Tab. 1, we report average of top-1 testing accuracy over 3 runs with random split of labeled data.

Effectiveness of Design. We first evaluate the effectiveness of each block of our model and results are displayed in Tab. 3. Exp. 1 provides the baseline performance from [37]. Exp. 1 vs. 2 show that hierarchical fine-grained representations (learned via L_p and L_c) alone bring a 1.49% performance improvement, which is already better than the representation learning baseline [28]. Building upon this, we investigate

α	0.7	0.8	0.9	1.0
Accuracy	92.10	92.89	95.28	94.02
K	10	30	50	70
Accuracy	94.23	95.28	93.75	93.39

Table 4. Analysis of refinement weight α and number of pivots K.

the contribution of label refinement strategies. Exp. 3 adopts the label propagation technique used in [28], which brings a 0.50% extra performance gain (vs. Exp. 2). When conducting label refinement by comparing to pivots (using p^a), Exp. 4 gains a 0.75% extra improvement (vs. Exp. 2). When equipped with the debiasing strategy (using q^c), Exp. 5 obtains a 1.26% additional performance gain in total (vs. Exp. 2). As discussed in Sec. 4.2, we introduce L_c (Eq. (12)) for representation learning. To validate this design, we unfreeze z_i in L_p in Eq. (11), drop L_c, and train z_i via L_p. Results (Exp. 5 vs. 6) show our design boost performance by 1.72% through learning more compact representations. As mentioned in Sec. 4.1, we use hard $\tilde{q}$ instaed of soft one in Eq. (11). Results (Exp. 5 vs. 7) show using hard $\tilde{q}$ is a better choice.

Refinement Weight. α, as indicated in Eq. (17), controls the weight for leveraging cluster labels learned from the feature space to refine pseudo-labels. Tab. 4 shows the performance with varying values of α in $\{0.7, 0.8, 0.9, 1.0\}$. Results indicate that $\alpha = 0.9$ is the optimal value.

Number of Pivots. This number determines how many pivots we need to learn across the entire dataset. Tab. 4 shows the performance with varying values of K in $\{10, 30, 50, 70, 100\}$, which correspond $\{1, 3, 5, 7, 10\}$ pivots of each class. Results show that 3 pivots per class achieve the best performance. This explains why we choose 300 pivots and 3000 pivots in CIFAR-100 and ImageNet-1k experiments.

6. Conclusion

In this paper, we introduce a novel SSL approach that detects inter-class and intra-class semantic relationships in data. Our method groups samples hierarchically by both inter-class and intra-class heterogeneity in the feature space. This allows us to use samples with the most relevant semantic information for each each unlabeled sample to better refine pseudo-labels. Extensive experiments on various SSL benchmarks demonstrate the effectiveness of our model.

Acknowledgements

This research was partially supported by the National Science Foundation (NSF) grant CCF-2144901 and the Stony Brook Trustees Faculty Award. Lingjie is also supported in part by Bloomberg Data Science Fellowship.

References

[1] Eric Arazo, DIego Ortego, Paul Albert, Noel E. O'Connor, and Kevin McGuinness. Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning. In *IJCNN*, 2020. 3

[2] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. In *ICLR*, 2020. 2

[3] 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. In *ICLR*, 2020. 1, 3, 5, 6

[4] David Berthelot, Nicholas Carlini, Ian Goodfellow, Avital Oliver, Nicolas Papernot, and Colin Raffel. MixMatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019. 1, 5, 6

[5] Lucas Beyer, Xiaohua Zhai, Avital Oliver, and Alexander Kolesnikov. S4L: Self-supervised semi-supervised learning. In *ICCV*, 2019. 6, 7

[6] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, 2018. 2

[7] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, 2020. 2, 3, 6, 7

[8] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 1, 2

[9] Baixu Chen, Junguang Jiang, Ximei Wang, Pengfei Wan, Jianmin Wang, and Mingsheng Long. Debiased Self-Training for Semi-Supervised Learning. In *NeurIPS*, 2022. 4

[10] 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. In *ICLR*, 2023. 1, 3

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 1, 2, 3, 6, 7

[12] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020. 6, 7

[13] Xinlei Chen and Kaiming He. Exploring simple Siamese representation learning. In *CVPR*, 2021. 1

[14] Adam Coates, Honglak Lee, and Andrew Y. Ng. An analysis of single-layer networks in unsupervised feature learning. In *JMLR*, 2011. 6

[15] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *NeurIPS*, 2013. 4

[16] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 6

[17] Enrico Fini, Pietro Astolfi, Karteek Alahari, Xavier Alameda-Pineda, Julien Mairal, Moin Nabi, and Elisa Ricci. Semi-supervised learning made simple with self-supervised clustering. In *CVPR*, 2023. 1, 2, 3, 6, 7

[18] Florian Graf, Christoph Hofer, Marc Niethammer, and Roland Kwitt. Dissecting supervised contrastive learning. In *ICML*, 2021. 2

[19] Yves Grandvalet and Yoshua Bengio. entropy minimization: Semi-supervised Learning by Entropy Minimization. In *NeurIPS*, 2004. 6, 7

[20] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent a new approach to self-supervised learning. In *NeurIPS*, 2020. 1, 6, 7

[21] Dayan Guan, Jiaxing Huang, Aoran Xiao, and Shijian Lu. Unbiased subclass regularization for semi-supervised semantic segmentation. In *CVPR*, 2022. 2, 3

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum Contrast for Unsupervised Visual Representation Learning. In *CVPR*, 2020. 4

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1, 7

[24] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. In *CVPR*, 2019. 1

[25] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *NeurIPS*, 2020. 2

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[27] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML Workshop: Challenges in Representation Learning*, 2013. 1, 3, 7

[28] Junnan Li, Caiming Xiong, and Steven C.H. Hoi. CoMatch: Semi-supervised Learning with Contrastive Graph Regularization. In *ICCV*, 2021. 1, 2, 3, 4, 5, 6, 7, 8

[29] Junnan Li, Pan Zhou, Caiming Xiong, and Steven C. H. Hoi. Prototypical Contrastive Learning of Unsupervised Representations. In *ICLR*, 2020. 6, 7

[30] Suichan Li, Bin Liu, Dongdong Chen, Qi Chu, Lu Yuan, and Nenghai Yu. Density-aware graph for deep semi-supervised visual recognition. In *CVPR*, 2020. 1, 2, 8

[31] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020. 6, 7

[32] Takeru Miyato, Shin Ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning. *TPAMI*, 2019. 6, 7

[33] Islam Nassar, Munawar Hayat, Ehsan Abbasnejad, Hamid Rezatofighi, and Gholamreza Haffari. ProtoCon: Pseudo-label Refinement via Online Clustering and Prototypical Consistency for Efficient Semi-supervised Learning. In *CVPR*, 2023. 1, 2, 3, 6, 7

[34] Islam Nassar, Samitha Herath, Ehsan Abbasnejad, Wray Buntine, and Gholamreza Haffari. All Labels Are Not Created Equal: Enhancing Semi-supervision via Label Grouping and Co-training. In *CVPR*, 2021. 1

[35] Yuval Netzer. Reading Digits in Natural Images with Unsupervised Feature Learning. In *NeurIPS*, 2011. 6

[36] Qi Qian, Yuanhong Xu, Juhua Hu, Hao Li, and Rong Jin. Unsupervised Visual Representation Learning by Online Constrained K-Means. In *CVPR*, 2022. 2, 3

[37] Kihyuk Sohn, David Berthelot, Chun Liang Li, Zizhao Zhang, Nicholas Carlini, Ekin D. Cubuk, Alex Kurakin, Han Zhang, and Colin Raffel. FixMatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*, 2020. 1, 3, 5, 6, 7, 8

[38] Zhiquan Tan, Kaipeng Zheng, and Weiran Huang. Otmatch: Improving semi-supervised learning with optimal transport. *ICML*, 2024. 1

[39] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *NeurIPS*, 2017. 1

[40] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding. *arXiv preprint arXiv:1807.03748*, 2018. 1

[41] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *JMLR*, 2008. 7

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1

[43] Jianfeng Wang, Thomas Lukasiewicz, Daniela Massiceti, Xiaolin Hu, Vladimir Pavlovic, and Alexandros Neophytou. Npmatch: When neural processes meet semi-supervised learning. In *ICML*, 2022. 5, 6

[44] Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. FreeMatch: Self-adaptive Thresholding for Semi-supervised Learning. In *ICLR*, 2023. 1, 3, 5, 6

[45] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised Feature Learning via Non-parametric Instance Discrimination. In *CVPR*, 2018. 2

[46] Qizhe Xie, Zihang Dai, Eduard Hovy, Minh Thang Luong, and Quoc V. Le. Unsupervised data augmentation for consistency training. In *NeurIPS*, 2020. 1, 5, 6, 7

[47] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-Supervised Learning with Dynamic Thresholding. In *ICML*, 2021. 1, 3, 5, 6

[48] Fan Yang, Kai Wu, Shuyi Zhang, Guannan Jiang, Yong Liu, Feng Zheng, Wei Zhang, Chengjie Wang, and Long Zeng. Class-Aware Contrastive Semi-Supervised Learning. In *CVPR*, 2022. 1

[49] Weiyi Yang, Richong Zhang, Junfan Chen, Lihong Wang, and Jaein Kim. Prototype-guided pseudo labeling for semi-supervised text classification. In *ACL*, 2023. 2, 3

[50] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *CVPR*, 2019. 3

[51] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *BMVC*, 2016. 6

[52] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stephane Deny. Barlow Twins: Self-Supervised Learning via Redundancy Reduction. In *ICML*, 2021. 1

[53] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. FlexMatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling. In *NeurIPS*, 2021. 1, 3, 5, 6

[54] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. MixUp: Beyond empirical risk minimization. In *ICLR*, 2018. 3

[55] Yifan Zhang, Jingqin Yang, Zhiquan Tan, and Yang Yuan. Relationmatch: Matching in-batch relationships for semi-supervised learning. *arXiv*, 2023. 5, 6

[56] Mingkai Zheng, Fei Wang, Shan You, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Weakly Supervised Contrastive Learning. In *ICCV*, 2021. 6, 7

[57] Mingkai Zheng, Shan You, Lang Huang, Chen Luo, Fei Wang, Chen Qian, and Chang Xu. SimMatchV2: Semi-Supervised Learning with Graph Consistency. In *ICCV*, 2023. 3

[58] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian, and Chang Xu. SimMatch: Semi-supervised Learning with Similarity Matching. In *CVPR*, 2022. 1, 2, 3, 5, 6, 7