Contrastive Credibility Propagation for Reliable Semi-supervised Learning

Brody Kutt, Pralay Ramteke, Xavier Mignot, Pamela Toman, Nandini Ramanan, Sujit Rokka Chhetri, Shan Huang, Min Du, William Hewlett

Palo Alto Networks, AI Research

{bkutt, pramteke, xmignot, ptoman, nramanan, schhetri, shuang1, midu, whewlett}@paloaltonetworks.com

Abstract

Producing labels for unlabeled data is error-prone, making semi-supervised learning (SSL) troublesome. Often, little is known about when and why an algorithm fails to outperform a supervised baseline. Using benchmark datasets, we craft five common real-world SSL data scenarios: few-label, open-set, noisy-label, and class distribution imbalance/misalignment in the labeled and unlabeled sets. We propose a novel algorithm called Contrastive Credibility Propagation (CCP) for deep SSL via iterative transductive pseudo-label refinement. CCP unifies semi-supervised learning and noisy label learning for the goal of reliably outperforming a supervised baseline in any data scenario. Compared to prior methods which focus on a subset of scenarios, CCP uniquely outperforms the supervised baseline in all scenarios, supporting practitioners when the qualities of labeled or unlabeled data are unknown.

1 Introduction

A fundamental goal of semi-supervised learning (SSL) is to ensure the use of unlabeled data results in a classifier that outperforms a baseline trained only on labeled data (supervised baseline). However, this is often not the case (Oliver et al. 2018). The problem is often overlooked as SSL algorithms are frequently evaluated only on clean and balanced datasets where the sole experimental variable is the number of given labels. Worse, in the pursuit of maximizing label efficiency, many modern SSL algorithms such as (Berthelot et al. 2019; Sohn et al. 2020; Zheng et al. 2022; Li, Xiong, and Hoi 2021) and others rely on a mechanism that directly encourages the marginal distribution of label predictions to be close to the marginal distribution of ground truth labels (known as distribution alignment). This assumption is rarely true in practice. We identify five key dataset quality variables (data variables) that can strongly impact SSL algorithm performance and are common in real-world datasets.

1. **Few-label**: Varying the number of labeled samples per class typically while the amount of unlabeled data grows or is held constant.

2. **Open-set**: Including and varying the ratio of out-of-distribution (OOD) samples, *i.e.* samples which belong to no class, in the unlabeled data.

3. **Noisy-label**: Varying the percent of given labels that are incorrect.

4. **Class distribution imbalance/misalignment**: Varying the disparity of class frequency distributions between labeled and unlabeled data. We explore increasingly imbalanced distributions in the labeled and unlabeled data separately (while keeping the other uniform to ensure misalignment).

Thoroughly evaluating the practicality of an SSL algorithm requires analyzing its response to these variables at differing severity. Identifying specific quality issues in a dataset can be challenging for practitioners. Often, real-world SSL workflows encounter scarcity and noise in labeled data, which is also sampled from a distribution distinct from the unlabeled data, derived from the often unknown target distribution. Moreover, the unlabeled data distribution may be inaccessible due to privacy concerns or real-time data collection in fully autonomous systems. Consequently, such systems necessitate external components like Out-of-Distribution (OOD) detectors to prevent failures, albeit at the cost of increased complexity. Instead of maximizing the robustness to any one data variable, *we strive to build an SSL algorithm that is robust to all data variables, i.e. can match or outperform a supervised baseline.* To address this challenge, we first hypothesize that sensitivity to pseudo-label errors is the root cause of all failures. This rationale is based on the simple fact that a hypothetical SSL algorithm consisting of a pseudo-labeler with a rejection option and means to build a classifier could always match or outperform its supervised baseline if the pseudo-labeler made no mistakes. Such a pseudo-labeler is unrealistic, of course. Instead, we build into our solution means to work *around* those inevitable errors.

Our contributions can be summarized as follows. To the best of our knowledge, our work is the first to 1) define an SSL algorithm that unifies a pseudo-labeling strategy and an approach to overcome label noise 2) use credibility vectors to properly represent uncertainty during pseudo-label generation 3) propose a generalized contrastive loss for non-discrete positive pairs 4) demonstrate a reliable performance boost over a supervised baseline across five real-world data scenarios at differing levels of severity. The rest of this work is organized as follows. We overview related works in Sec. 2. We introduce CCP in Sec. 3. In Sec. 4, we detail our experimental results before concluding in Sec. 5.

2 Related Work

SSL has a rich history in AI research (Yang et al. 2021; Chapelle, Schölkopf, and Zien 2006). We focus on two dominant approaches. These are pseudo-labeling and consistency training. CCP draws inspiration from both approaches.

Pseudo-labeling These methods typically constitute transductive learning (Shi et al. 2018; Iscen et al. 2019). Here, the objective is to generate proxy labels for unlabeled instances to enhance the learning of an inductive model. Seminal work in (Lee et al. 2013) simply uses the classifier in training to produce pseudo-labels inductively which are trained upon directly. This is problematic in that it actively promotes confirmation bias *i.e.* the model will learn to confirm its predictions. This was later extended to include a measure of confidence in (Shi et al. 2018). Closely related is the concept of self-training which iteratively integrates into training the most confident of these pseudo-labeled samples and repeats (Dong and Schäfer 2011; Sahito, Frank, and Pfahringer 2021; Xie et al. 2020b). These techniques can become unstable when pseudo-label error accumulates across iterations. More recent work has addressed accumulating errors by using independent models to utilize pseudo-labels (Chen et al. 2022). LPA (Zhu and Ghahramani 2002) is a popular graph-based technique for generating pseudo-labels transductively but has many failure cases (Dong et al. 2020). Much work tends to use LPA as a transductive inference mechanism for pseudo-labeling. Such inferred pseudo-labels are highly noisy and thus problematic for SSL. The CCLP algorithm (Kamnitsas et al. 2018) tries to circumvent this by instead using LPA-derived pseudo-labels only for graph-based regularization of the encoder. Like CCP, (Wang and Wu 2022) proposed repeatedly re-predicting pseudo-labels during an optimization framework akin to self-training. Despite the popularity of pseudo-labeling-based SSL approaches, they tend to break down when faced with a large amount of pseudo-label errors.

Consistency Training Analogous to perturbation-based SSL and contrastive learning, these techniques train a network to produce a single, distinct output for a sample under different augmentations (transformations). Work in (Xie et al. 2020a) minimizes a consistency loss for unlabeled data and a standard classification loss for labeled data simultaneously which could introduce irreconcilable gradients. Other work (Chen et al. 2020a,b), shown superior to (Xie et al. 2020a) in SSL, employ an unsupervised consistency-like loss in a pretraining stage before training solely with given labels. Self-supervised consistency loss is attractive in that it eliminates supervision errors but lacks power in that class structure from unlabeled data is never leveraged directly. Instead, CCP introduces a *softly* supervised consistency (contrastive) loss with true labels and pseudo-labels.

3 Contrastive Credibility Propagation (CCP)

Here we present our method for transductive pseudo-label refinement and classifier training called Contrastive Credibility Propagation (CCP). We establish notation, provide a high-level overview, introduce credibility vectors, formalize our loss functions, detail the CCP iteration, introduce our sub-sampling procedure, and then explain how to build a classifier on CCP-derived pseudo-labels.

3.1 Notation

Consider a classification task among a set of K classes $c = [0, 1, \ldots, K-1]$. Consider a partially labeled dataset X. Denote all indices to labeled (unlabeled) samples as L (U). Denote $I = L \cup U$. Each labeled sample $l \in L$ has a known one-hot class vector $y_l \in [0, 1]^K$ with a 1 (0) in the on (off) position. We maintain a real-valued credibility vector q_i of length K for each sample i: $q_i \in [-1, 1]^K$ (see Sec. 3.3). In training, denote the indices of a random batch of size n as $\mathcal{B} \subseteq I$. We decompose $\mathcal{B}$ into the indices of labeled, $\mathcal{B}_l$, and unlabeled, $\mathcal{B}_u$, samples. We define a set of data transformations, $\mathcal{T}$. We draw two transformations, $t_1, t_2 \in \mathcal{T}$, randomly for every batch. A transformed pair will share one q_i. A batch of transformed data and credibility vector tuples will be of size $2n$ denoted $\{(x_i, q_i), (x_{i+n}, q_i)\}_{i \in \mathcal{B}}$ where i and $i + n$ denote the indices of a transformed pair.

3.2 Overview

CCP aims to equip an SSL training process to be robust to errors in the pseudo-labels assigned to unlabeled data. CCP thus consists of iterative pseudo-label refinement *before* building a classifier. When combined with our credibility representation, this refinement helps to nudge pseudo-labels in the direction of the true class and nullify the data for which true labels are unknowable or ambiguous. To do this, we merge a process of transductively propagating pseudo-labels (in the form of credibility vectors) and an outer loop built for overcoming instance-dependent noisy labels based on the SEAL algorithm (Chen et al. 2021). To train a transductive pseudo-label generator, we define a generalization of unsupervised (SimCLR (Chen et al. 2020a)) and supervised (SupCon (Khosla et al. 2020)) contrastive loss that can make use of non-discrete, *i.e.* uncertain, positive pairs. We also define a pseudo-label subsampling strategy that limits the divergence of pseudo-label class distributions before and after subsampling which we hypothesize to be the primary cause of instability of similar subsampling procedures. We show it can lead to higher pseudo-label accuracy and faster convergence, especially with increasingly unclean data. Further details are provided in Fig. 1.

The network architecture consists of an encoder f_b, that computes a vector encoding $f_b(x) = b$. One linear projection head f_z computes an encoding used for contrastive learning, $f_z(b) = z$. A separate linear projection head, f_g, computes $f_g(b) = g \in \mathbb{R}^K$ for a classification loss. Attaching f_g and f_z to f_b is motivated by (Chen et al. 2020a,b; Khosla et al. 2020). These architectural components are illustrated in Fig. 2. Between iterations and before classifier training, we reset the variables in f_b, f_z back to a prior state (either a random initialization or pretrained using $\mathcal{L}_{\text{SSC}}$'s unsupervised counterpart, SimCLR).

Motivated by (Chen et al. 2020a; Cer et al. 2018), the similarity function we use is angular similarity defined by $\phi(z_i, z_j) = 1 - \arccos\left(\frac{z_i \cdot z_j}{\|z_i\| \|z_j\|}\right)/\pi$.
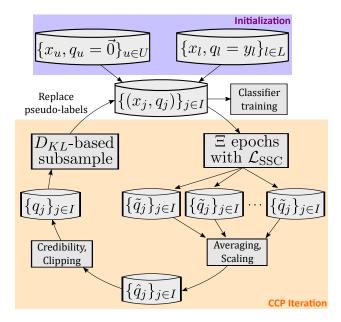
Figure 1: The credibility vectors for all samples, $\{q_j\}_{j \in I}$, are initialized with the given labels, label guesses, or $\vec{0}$ if unknown. For trusted labeled data, the associated q_l may be permanently clamped to that value. A single CCP iteration first transductively generates new $\{\tilde{q}_j\}_{j \in I}$ for each of Ξ epochs of training via Algorithm 2 while minimizing $\mathcal{L}_{SSC}$. All $\{\tilde{q}_j\}_{j \in I}$ are averaged and scaled such that the maximum of the strongest credibility vector is 1 to form $\{\hat{q}_j\}_{j \in I}$. We then perform a final credibility adjustment and clip all values outside $[0, 1]$ to form $\{q_j\}_{j \in I}$. Once pseudo-labels have been refined through CCP iterations, they are then used to build an inductive classifier while minimizing both $\mathcal{L}_{SSC}$ and $\mathcal{L}_{CLS}$. The (inherently error-prone) results of Algorithm 2 are never used to directly supervise error-sensitive contrastive and classifier losses.

3.3 Credibility Adjustments

The CCP algorithm uses credibility vectors, q_i, in place of one-hot or softmax label vectors. Credibility vectors are more expressive. A credibility score of 1 means maximum confidence in class membership, 0 means uncertainty, and -1 means maximum confidence in class non-membership. *Each credibility score given a sample and class is the similarity between that sample and class minus the highest similarity to all other classes.* Credibility adjustments are formalized in Algorithm 1, Lines 16 to 18 and Algorithm 2, Lines 9, 10 and 12. The core idea of credibility, similar to (Dong et al. 2020), is to extend similarity measurements to capture ambiguity brought forth by competing similarities *i.e.* conditional similarity. Unlike (Dong et al. 2020), we represent credibility in our label vectors for use throughout the algorithm. For trusted labeled data, credibility vectors, $\{q_l\}_{l \in L}$, are clamped with 1 (-1) in the on (off) position. Before applying credibility vectors to $\mathcal{L}_{SSC}$, $\mathcal{L}_{CLS}$, and Algorithm 2, we clip to a $[0, 1]$ range. A clipped credibility vector thus consists of 0 everywhere except the strongest value which is scaled down by
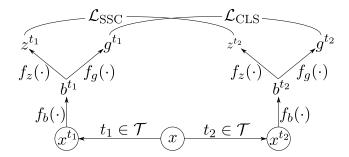

Figure 2: CCP uses an encoder $f_b(\cdot)$ and two projection heads: $f_z(f_b(\cdot))$ for contrastive loss and $f_g(f_b(\cdot))$ for classification loss.

the second strongest value. However, negative values are still useful when averaging across epochs in Line 14. The single non-zero value reflects the strength (confidence) of the label prediction. At initialization, unlabeled data receive $q_u = \vec{0}$ (maximum uncertainty). Throughout CCP, the influence of x_i scales with the magnitude of the non-zero value in q_i ($q_i = \vec{0}$ will ensure x_i exhibits no effect anywhere). Because CCP is designed to work with noisy (pseudo-)labels, if uncertain label guesses exist, these can be used for initialization instead. Fig. 3 provides a concrete example of a credibility vector calculation and its effect on a cross-entropy (Xent) gradient.

3.4 Soft Loss Functions

$\mathcal{L}_{SSC}$ and $\mathcal{L}_{CLS}$ are contrastive and classification loss functions, respectively. Both are supervised by $\{q_i\}_{i \in I}$ provided by the CCP algorithm. Each sample in a transformed pair has identical credibility vectors and is treated independently. Thus, to simplify the notation, we formalize $\mathcal{L}_{SSC}$ and $\mathcal{L}_{CLS}$ for a batch of data and (clipped) credibility vector tuples, $\{(x_i, q_i)\}_{i \in \mathcal{B}}$.

We define an $n \times n$ pairwise matching matrix, M, where $m_{i,j} = q_i \cdot q_j$ for $i, j \in \mathcal{B}$. Each row of M contains the weights for a weighted arithmetic mean of normalized contrastive losses for that sample to all other samples based on the evidence of a positive pair relationship. We scale pairwise similarities by temperature τ and take the exponential as in (Chen et al. 2020a,b; Khosla et al. 2020) to form an $n \times n$ matrix A defined by $a_{i,j} = \exp(\phi(z_i, z_j)/\tau)$ for $i, j \in \mathcal{B}$. We construct a strength vector, ω, defined by $\omega_i = \max(q_i)$ for $i \in \mathcal{B}$. Each ω_i serves to scale the magnitude of contrastive loss on sample i and the corresponding entries in the normalizing factor for each $a_{i,j}$. We ignore comparisons of the same sample with the following modifications $M = M \odot (1 - \mathbb{I})$, $A = A \odot (1 - \mathbb{I})$ where $\odot$ is element-wise multiplication and $\mathbb{I}$ is the identity matrix. We can now define $\mathcal{L}_{SSC}$,

$$\mathcal{L}_{SSC} = -\frac{1}{n} \sum_{i \in \mathcal{B}} \frac{\omega_i}{\sum m_{i,\cdot}} \sum_{j \in \mathcal{B}} m_{i,j} \log \left(\frac{a_{i,j}}{a_{i,\cdot} \cdot \omega} \right) \quad (1)$$

SimCLR and SupCon loss are special cases of $\mathcal{L}_{SSC}$. In $\mathcal{L}_{SSC}$, positive pairs are not specified discretely. Every pair of samples has a score in M between 0 and 1 which corresponds
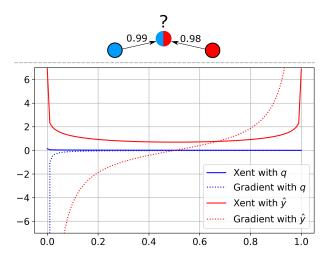
Figure 3: *Top:* An unlabeled sample features high similarity scores of 0.99 and 0.98 for both classes. A conventional softmax label is computed as $\hat{y} = [0.502, 0.498]$. A credibility vector, q, is computed as $0.99 - 0.98 = 0.01$ and $0.98 - 0.99 = -0.01$ (clipped to 0 before it's used in Xent). Thus, $q = [0.01, 0.0]$. *Bottom:* Consider x to be the softmax output of a binary classifier for the blue class. Using $\hat{y}$ to compute Xent induces strong gradients at either pole despite the true class being nearly ambiguous. Using q will ensure the gradient for this sample is near 0 everywhere except $x = 0$. If $q = \vec{0}$, i.e. the sample is equally similar to both classes, the gradient is 0 everywhere.

to how much evidence there is that they constitute a positive pair. If all data is labeled and all $\omega_i = 1$, $\mathcal{L}_{\text{SSC}}$ becomes SupCon loss. If all data is unlabeled, only transformed pairs have a maximum positive pair relationship, and all $\omega_i = 1$, $\mathcal{L}_{\text{SSC}}$ becomes SimCLR loss. Circumventing the discrete selection of positive pairs is simpler and better reflects the true state of pseudo-labels compared to employing a tunable decision threshold in (Zhang et al. 2022) and others.

When training a classifier, we use an Xent loss similar to SEAL loss except that we feed in q_i instead of softmax label vectors. We use the output from f_g instead of f_z. We send each g_i through a softmax, $\sigma(g_i)$, and compute,

$$\mathcal{L}_{\text{CLS}} = -\frac{1}{n} \sum_{i \in \mathcal{B}} \sum_{k \in c} q_{i,k} \log(\sigma(g_i)_k) \qquad (2)$$

One can interpret $\mathcal{L}_{\text{CLS}}$ as a credibility-weighted version of categorical Xent. Recall each q_i will contain only one non-zero entry. Assume this non-zero entry is index k. The value of $\mathcal{L}_{\text{CLS}}$ for sample i is $\omega_i \log(\sigma(g_i)_k)$.

3.5 The CCP Iteration

We formalize the CCP iteration in Algorithm 1, the propagation mechanism in Algorithm 2, and our subsampling procedure in Algorithm 3. Each assumes $\{q_l\}_{l \in L}$ are trusted labels. If this is not true, *e.g.* in the noisy-label scenario, all operations applied to unlabeled data are applied to all data. During

a single CCP iteration, we continuously predict new q_i's for samples while minimizing $\mathcal{L}_{\text{SSC}}$. We store all credibility vectors and then average them together at the end of the iteration. The SEAL algorithm (Chen et al. 2021), bearing similarity to CCP's outermost iteration, is designed to correct the label of mislabeled samples in supervised problems. It was found that, during training, network predictions frequently oscillate between incorrect and correct labels in the presence of label noise. This suggests averaging the predictions made across epochs to correct label noise. Similarly, propagated pseudo-labels are subjected to the randomness of the instantaneous network state *and* batch selection. In the presence of pseudo-label error, and, more generally, uncertainty arising from within the model or data, we observe the same oscillatory behavior reported in (Chen et al. 2021) of predicted pseudo-labels (shown in Appendix D). This grants us a similar theoretical motivation for averaging across epochs in CCP. Assume there is a latent, preferable pseudo-label for sample x_i denoted $q_i^* \in \mathbb{R}^K$. Denote the pseudo-label obtained at the end of the m-th CCP iteration as $q_i^{[\cdot, m]}$. Based on the oscillatory behavior of pseudo-labels, we can approximate the pseudo-label at the ξ-th epoch of the m-th iteration for $m \geq 1$ as,

$$q_i^{[\xi, m]} \approx \alpha_i^{[\xi, m]} \beta_i^{[\xi, m]} + (1 - \alpha_i^{[\xi, m]}) q_i^{[\cdot, m-1]} \qquad (3)$$

Where $\xi \in \{1, 2, \ldots, \Xi\}$, $q_i^{[\cdot, 0]}$ are zero vectors, $\alpha_i^{[\xi, m]} \in [0, 1]$ are coefficients dependent on instances, the network, and $q_i^{[\cdot, m-1]}$ with $\alpha_i^{[\xi, 1]} = 1 \quad \forall \xi$, and $\beta_i^{[\xi, m]} \in \mathbb{R}^K$ are i.i.d. random vectors with $\mathbb{E}[\beta_i^{[\xi, m]}] = q_i^*$. Consider a uniformly chosen random epoch and iteration denoted ξ' and m', respectively. We can see CCP is expected to improve the quality of pseudo-labels iteratively and new pseudo-labels at the end of an iteration have lower variance due to averaging.

$$\|\mathbb{E}[q_i^{[\cdot, m'+1]}] - q_i^*\| \leq \|q_i^{[\cdot, m']} - q_i^*\| \qquad (4)$$

$$\text{var}(q_{i,k}^{[\cdot, m']}) \leq \text{var}(q_{i,k}^{[\xi', m']}) \quad \forall k \in c \qquad (5)$$

In Algorithm 1, Lines 14 to 19, we adjust labels in several ways to make them more suitable as labels. In Line 15, we multiply by a scaling factor $1/\gamma$ after the average. We use $\gamma = \max_{u \in U, c \in K} \hat{q}_{u,c}$. This ensures the strongest pseudo-label will have a strength of 1. Before we clip all values outside the range of $[0, 1]$ in Line 19, we compute a final credibility adjustment. The scaled, credibility adjusted, and clipped vectors are denoted $\{q_u\}_{u \in U}$. In Line 7, we find more stable performance when using *balanced* batches during CCP iterations. We guarantee there are ≥ 1 (pseudo-)labeled samples per class in every batch via oversampling. Pseudo-labels obtained from repeated uses of a sample are simply gathered together in the average. This ensures 1) the correct class can be propagated and 2) credibility can be properly measured. Lastly, we found it beneficial to use a small learning rate in the first iteration. Especially for few-label scenarios, the network quickly overfits to the small labeled set, and pseudo-label quality suffers. Specifically, we reduced the original learning rate of 0.06 to 0.0006 in the first iteration for all experiments.

Algorithm 1: An iteration of the CCP algorithm.

1: **Given** Ξ, $\mathcal{T}$, $\{q_l\}_{l \in L}$, X, p_{last}, $d_{\max}$
2: **if** $\{q_u\}_{u \in U}$ are not available **then**
3:      Initialize $q_u = \vec{0}$ for $u \in U$
4: **end if**
5: Reset f_b, f_z to a random or pretrained state
6: **for** $\xi \in \{1, 2, \ldots, \Xi\}$ **do**
7:      **for** balanced batches $\{(x_i, q_i)\}_{i \in \mathcal{B}_l \cup \mathcal{B}_u}$ **do**
8:         Form $\{(x_i, q_i), (x_{i+n}, q_i)\}_{i \in \mathcal{B}_l \cup \mathcal{B}_u}$ with randomly drawn $t_1, t_2 \in \mathcal{T}$
9:         Compute $z_i = f_z(f_b(x_i))$, $z_{i+n} = f_z(f_b(x_{i+n}))$ for $i \in \mathcal{B}_l \cup \mathcal{B}_u$
10:         Compute and store $\{\tilde{q}_j\}_{j \in \mathcal{B}_u}$ using Algorithm 2
11:         Train f_b, f_z using the gradient of $\mathcal{L}_{\text{SSC}}$ computed with $\{(z_i, q_i), (z_{i+n}, q_i)\}_{i \in \mathcal{B}_l \cup \mathcal{B}_u}$
12:      **end for**
13: **end for**
14: Compute $\{\hat{q}_u = \text{average of all stored } \tilde{q}_u\}_{u \in U}$
15: Scale all $\hat{q}_u$ via multiplying by $\frac{1}{\gamma}$
16: **for** $k \in c$ **do**   ▷ Credibility adjustment
17:      $q_{u,k} = \hat{q}_{u,k} - \max_{k' \in c \setminus k}(\hat{q}_{u,k'})$ for $u \in U$
18: **end for**
19: Clip all values in $\{q_u\}_{u \in U}$ to lie in $[0, 1]$
20: Compute p, $\{\hat{\omega}_u\}_{u \in U}$ using Algorithm 3
21: Reset bottom $p\%$ of $\{q_u\}_{u \in U}$ ordered by $\{\hat{\omega}_u\}_{u \in U}$
22: Update $p_{\text{last}} = p$, $d_{\max} = \frac{d_{\max}}{2}$
23: **Return** f_b, p_{last}, $d_{\max}$, $\{q_u\}_{u \in U}$ (used in next iteration)

Algorithm 2: Compute propagated credibility vectors.

1: **Given** $\{(z_i, q_i), (z_{i+n}, q_i)\}_{i \in \mathcal{B}_l \cup \mathcal{B}_u}$
2: Define expanded indices $\mathcal{B}' = \{0, 1, \ldots, 2n - 1\}$
3: Define $q_{i+n} = q_i$ for $i \in \mathcal{B}$
4: **for** $j \in \mathcal{B}_u$ **do**
5:      **for** $k \in c$ **do**   ▷ Compute class similarities
6:         $\psi_{j,k} = \sum_i^{\mathcal{B}' \setminus j} \phi(z_j, z_i) q_{i,k} / \sum_i^{\mathcal{B}' \setminus j} q_{i,k}$
7:         $\psi_{j+n,k} = \sum_i^{\mathcal{B}' \setminus j+n} \phi(z_{j+n}, z_i) q_{i,k} / \sum_i^{\mathcal{B}' \setminus j+n} q_{i,k}$
8:      **end for**
9:      **for** $k \in c$ **do**   ▷ Credibility adjustment
10:         $\tilde{q}_{j,k} = \psi_{j,k} - \max_{k' \in c \setminus k}(\psi_{j,k'})$
11:         $\tilde{q}_{j+n,k} = \psi_{j+n,k} - \max_{k' \in c \setminus k}(\psi_{j+n,k'})$
12:      **end for**
13:      Store $\tilde{q}_j = \frac{\tilde{q}_j + \tilde{q}_{j+n}}{2}$ ▷ Average across t_1, t_2
14: **end for**
15: **Return** $\{\tilde{q}_j\}_{j \in \mathcal{B}_u}$

Algorithm 3: Compute a subsample percentage.

1: **Given** $\{\hat{q}_u\}_{u \in U}$, $\{q_u\}_{u \in U}$, p_{last}, $d_{\max}$
2: Compute $\{\hat{\omega}_u = \max(\hat{q}_u)\}_{u \in U}$
3: $Q = \sum_u^U q_u / \sum_k^c \sum_u^U q_{u,k}$ ▷ Anchor distribution
4: **for** $p_i \in \{0\%, 1\%, \ldots, p_{\text{last}} - 1\%\}$ **do**
5:      Reset bottom $p_i\%$ of $\{q_u\}_{u \in U}$ ordered by $\{\hat{\omega}_u\}_{u \in U}$
6:      $P = \sum_u^U q_u / \sum_k^c \sum_u^U q_{u,k}$ ▷ Candidate distribution
7:      $d_i = D_{\text{KL}}(P \parallel Q) = \sum_{k \in c} P_k \log_2 (P_k / Q_k)$
8: **end for**
9: $p = \max(\{p_i \text{ for all } i \text{ such that } d_i < d_{\max}\})$
10: **Return** p, $\{\hat{\omega}_u\}_{u \in U}$

D_{KL}-**Based Subsampling** We use Algorithm 3 to decide which pseudo-labels to reset to $\vec{0}$ between CCP iterations. This is inspired by self-training (Amini et al. 2022) which solves the similar but inverted task of iteratively *adding* data to the labeled set. These techniques typically use the max score of a learned classifier as an indicator of confidence. Similarly, our confidence indicator for a credibility vector is the maximum of its *unclipped* values. We compute a percentage $p \in \{0\%, 1\%, \ldots, 99\%\}$ that represents what percent of least confident pseudo-labels will be reset. Aggressive p values can cause instability or slow convergence when too many correct pseudo-labels are reset. We hypothesize the instability is similar in cause to the instability of self-training mechanisms (Chen et al. 2022). CCP is shown highly stable without any subsampling (Sec. 4.2). Accordingly, our approach is to balance resetting weak pseudo-labels with limiting the divergence of the predicted class distribution of the selected unlabeled data from the predicted class distribution of all unlabeled data. Concretely, in Line 3 we compute the anchor distribution, Q, by summing the weights for every class and dividing by the total mass. We search over candidate p's with one minus the p used in the previous iteration, $p_{\text{last}} - 1\%$, as the maximum to ensure we don't increase p between iterations. At Line 6, we compute the new distribution, P, obtained after resetting the candidate percentage of vectors. At Line 7, we compute the Kullback–Leibler (KL) divergence (Csiszar 1975) between these distributions. At Line 9, we choose p by selecting the maximum candidate that obeys a strict limit on the divergence, $d_{\max}$. We divide $d_{\max}$ by 2 for the next iteration to support convergence. Our method is unsupervised, free of imposing assumptions, and normalized to the dataset size. This suggests, in theory, that a single schedule for $d_{\max}$ should generalize well across datasets. Indeed, we find an initial value of $d_{\max} = 0.01$ to work *generally* well across all of our experiments.

3.6 Building a Classifier

After CCP iterations have concluded, we apply the soft labels, $\{q_j\}_{j \in I}$, to build a classifier consisting of f_b, f_g. We use Stochastic Gradient Descent with randomly selected minibatches (regardless of whether the data was originally labeled). We find that minimizing both Eq. (1) (using f_z) and Eq. (2) together consistently outperforms Eq. (2) alone. In our experiments, resetting f_b, f_z (to a random or pretrained state) after the final iteration of CCP is marginally more performant than reusing the state after the last iteration – this prior state will have been learned without the latest pseudo-label adjustments. However, the latter converges much faster. We find that using Algorithm 3 to subsample the final $\{q_j\}_{j \in I}$ greatly increases training set accuracy but has a minimal and inconsistent effect on the test set. This is because cropping the "hard" samples with weak pseudo-labels at this stage makes the training set easier but reduces generalization.

4   Experimental Results[1]

We leverage CIFAR-10 and CIFAR-100 for our data variable sensitivity experiments (Krizhevsky and Hinton 2009). We use these datasets as starting points to explore the five data variables we introduce in Sec. 1. For comparability among the data variables, we define a base case that represents a minimal severity of each data variable. We study each data variable independently by increasing the severity from the base case at three levels. For the base cases, we take the first 40% of classes and define them as in-distribution (ID) e.g. for CIFAR-10, classes $[0, 3]$ are ID while $[4, 9]$ are out-of-distribution (OOD). For CIFAR-10, the base case is defined as 400 labeled and 4600 unlabeled samples per ID class. For CIFAR-100, we have 100 labeled and 400 unlabeled samples per ID class. We then independently introduce the following perturbations:

1. **Few-label**: We move labeled data to the unlabeled set such that the number of labeled samples per ID class reduces to 25, 4, and 2.

2. **Open-set**: We move all data from OOD classes into the unlabeled set incrementally *e.g.* on CIFAR-10 the unlabeled set contains data from classes $[0, 5]$, $[0, 7]$, and $[0, 9]$.

3. **Noisy-label**: We randomly select 20%, 40%, and 60% of labels and randomly perturb them to an incorrect ID class.

4. **Class distribution imbalance/misalignment**: We perturb labeled and unlabeled sets separately. We take the last 50% of ID classes, *e.g.* for CIFAR-100 the ID classes we perturb are $[20, 39]$, and reduce their quantity (discarding samples). When perturbing unlabeled sets, we reduce to 20%, 10%, and 0% of the original quantity. For labeled sets, we reduce such that 25, 4, and 2 samples remain.

We perform our data variable sensitivity analysis on CCP, CoMatch (Li, Xiong, and Hoi 2021), OpenMatch (Saito, Kim, and Saenko 2021), ACR (Wei and Gan 2023), and FixMatch (Sohn et al. 2020) with and without distribution alignment. CoMatch and FixMatch were developed to optimize performance in the few-label scenario, whereas OpenMatch and ACR were designed to address open-set and misalignment in the unlabeled set, respectively. We used the original author's implementations for all algorithms. For all algorithms and datasets, we use the standard WRN-28-2 and WRN-28-8 (Zagoruyko and Komodakis 2016) as the backbone network architecture for CIFAR-10 and CIFAR-100, respectively. We closely match all training hyperparameters and settings of (Sohn et al. 2020; Li, Xiong, and Hoi 2021). For algorithm-specific hyperparameters, we use the values originally recommended in the work for the corresponding dataset. CCP-specific hyperparameter choices are detailed in Tab. 2. Some minor differences exist across all algorithm implementations *e.g.* the parameter choices for each $t \in \mathcal{T}$, the deep learning software package used, and regularization used. We report the performance of a fully supervised control for each algorithm implementation to help understand the effect.

---

[1]Code available at: https://github.com/PaloAltoNetworks/CCP_CIFAR and https://github.com/PaloAltoNetworks/ccp-as-pytorch

Computational Expense   CCP iterations can potentially be computationally taxing at large values of Ξ. In the worst case, one must fully train a new network for every CCP iteration. Additionally, if Ξ is too large, incorrect pseudo-labels are memorized and $\hat{q}$ becomes biased towards the error. We found beneficial a course of pretraining using $\mathcal{L}_{SSC}$'s unsupervised counterpart, SimCLR, to determine the initialization of f_b, f_z. This pretrained state significantly reduced the inherent randomness of early pseudo-label predictions. Also, due to faster convergence, using this pretrained state allowed us to use aggressive early stopping after the first CCP iteration which has a fixed number of epochs and a low learning rate (50 and 0.0006, respectively). Specifically, we maintain a batch-wise exponential moving average (EMA) of $\mathcal{L}_{SSC}$ with decay 0.99 during each CCP iteration and halt quickly after the EMA stops dropping. This resulted in the equivalent of ~ 15 and ~ 50 epochs per iteration with 24 and 12 CCP iterations for CIFAR-10 and CIFAR-100, respectively. However, shown in Sec. 4.2, CCP often converges after only a few iterations. The only deviation from this is in the open-set experiments, in which we report the performance after a single CCP iteration. Further iterations increased pseudo-label confidence in both ID and OOD samples and thus did not provide value. Using Algorithm 3 with a more aggressive d_{max} and schedule mitigated this problem however we found satisfying results simply with a single CCP iteration which was more consistent with the other experiments.

4.1   Data Variable Sensitivity Analysis

We report the test accuracy of each algorithm on each data variable experiment at all levels of severity in Tab. 1. We find relatively consistent performance across the control experiments. Performance on the base case reflects an algorithm's inherent label efficiency (which impacts performance in all scenarios) and helps to contextualize all results in this table. We also normalized results to the base case to help visualize an algorithm's sensitivity to each data variable in isolation. These plots can be found in Appendix B. Although CCP doesn't achieve superiority in every scenario, particularly concerning label efficiency, it demonstrates remarkable consistency. For instance, on CIFAR-10, its accuracy never drops below 90% for any scenario representing only a 5% drop from the base case. Moreover, CCP uniquely outperforms a supervised baseline in every experiment. This supports the core reliability thesis. Where CCP does not achieve superiority, label efficiency seems to be a large contributing factor. Other algorithms fail, sometimes catastrophically below CCP or even the supervised baseline, in the presence of certain data variations. This is most prevalent when perturbing the labeled distribution and the relatively poorly explored noisy-label scenario where CCP often outperforms by a large margin. Intuitively, DA seems to help FixMatch marginally in the few-label scenario but hurts when perturbing the unlabeled data distribution. Note that all algorithms perform relatively well on open-set experiments. All algorithms tested besides CCP build off FixMatch in some way. This is common practice in the literature. We hypothesize that, for these algorithms, FixMatch's pseudo-label threshold parameter effectively removes unconfident pseudo-labeled OOD samples
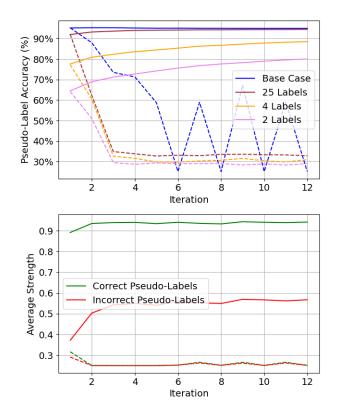
Figure 4: Performance of CCP iterations on the base case and few-label experiments of CIFAR-10 with and without credibility. Solid lines indicate the use of credibility adjustments. Dashed lines indicate a softmax function. *Top:* Pseudo-label accuracy. *Bottom:* The average strength of correct and incorrect pseudo-labels in the base case.
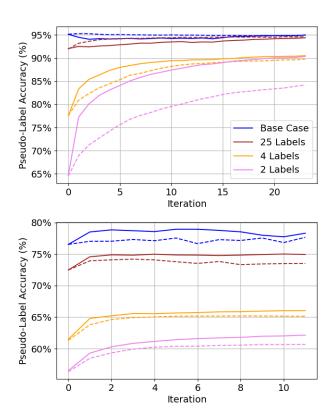
Figure 5: Pseudo-label accuracy during CCP iterations in the base case and few-label experiments of CIFAR-10 (top) and CIFAR-100 (bottom). Solid lines depict the usage of D_{KL}-based subsampling with the parameters and schedule presented in Algorithm 1 with an initial $d_{max} = 0.01$. Dashed lines indicate no subsampling *i.e.* $d_{max} = 0.0$.

from consideration. For CCP, credibility adjustments shrink the pseudo-labels of OOD samples to near zero automatically.

4.2 Ablation Analysis

We investigate the criticality of credibility adjustments to successful CCP iterations as well as the effectiveness of subsampling via Algorithm 3. We focus on the base case and few-label experiments, however, repeating these experiments with the other data variables provided a similar result.

In Fig. 4, we study the effect of credibility adjustments by measuring the difference in CCP iteration performance when traditional softmax functions replace them. We also omit scaling, clipping, and subsampling when using softmax such that it resembles the SEAL algorithm. We often find quick and catastrophic degradation to maximum entropy pseudo-labels when using softmax. When considering the average strength (maximum value) of pseudo-labels, it is clear that credibility strongly differentiates correct and incorrect pseudo-labels, making it a better fit as a measure of confidence. Recall in Fig. 3 that, when faced with uncertainty, an Xent gradient with a softmax label pushes the network to produce a high entropy pseudo-label. That mirrors what we see here.

In Fig. 5, we see D_{KL}-based subsampling provides substan-

tial benefit to the pseudo-label accuracy during CCP iterations at high data variable severity. At worst, it appears to provide no benefit as in the CIFAR-10 base case experiment. Note we used the same initial $d_{max} = 0.01$ and schedule presented in Algorithm 1 for all experiments in this work. If instability is observed, which can occur if d_{max} is too large, tuning is necessary. Additional subsampling ablation analysis with a similar result can be found in Tab. 6.

5 Conclusion

We have presented an algorithm that combines a soft contrastive approach to pseudo-labeling with an outer iteration designed for learning under instance-dependent label noise. The result is a highly reliable and effective SSL algorithm that does not perform worse than a supervised baseline across five common real-world SSL scenarios. Future work may include augmenting Algorithm 2 to include successful components from prior work such as consistency training between weak/strong augmentation (Sohn et al. 2020) and instance/semantic similarity (Zheng et al. 2022) to combine CCP's reliability with the label efficiency of other work.

CIFAR-10 / CIFAR-100	Control Base Case	Few-label	Open-set	Noisy-label	Class Imbalance/Misalignment U	L
Supervised Baseline	97.00% 89.18%	63.33% 54.93% 49.98%	89.18% 89.18% 89.18%	78.83% 73.08% 58.60%	89.40% 89.40% 89.40%	75.90% 63.98% 58.08%
	84.10% 67.30%	43.85% 20.03% 15.38%	67.30% 67.30% 67.30%	55.45% 47.35% 35.70%	67.30% 67.30% 67.30%	54.03% 38.95% 36.13%
CoMatch	97.00% 97.74%	96.97% 97.45% 97.17%	97.82% 97.69% 97.79%	91.82% **68.69%** **50.05%**	94.07% 91.54% **87.77%**	96.11% 89.73% 90.90%
	85.54% 85.02%	82.12% 66.02% 61.51%	84.90% 84.72% 85.24%	78.72% 67.16% 47.94%	80.23% 78.52% 77.75%	81.00% 61.41% 48.04%
ACR	97.38% 97.20%	95.45% 67.66.% 60.66%	96.80% 96.42% 96.78%	82.90% **62.73%** **39.75%**	93.90% 94.45% **88.55%**	96.05% 94.85% 87.75%
	85.70% 82.80%	74.97% 42.83% 22.07%	82.78% 82.88% 82.33%	72.70% 56.33% **33.38%**	78.67% 76.95% 75.80%	77.70% 64.30% 52.65%
OpenMatch	96.80% 96.65%	92.20% 68.18% **43.08%**	93.58% 91.93% 91.00%	91.28% 85.93% 76.75%	94.90% 93.93% 91.48%	76.43% **50.55%** **49.80%**
	85.20% 83.68%	77.30% 40.35% 27.88%	83.13% 82.63% 82.33%	77.35% 67.10% 53.55%	80.80% 79.18% 76.05%	79.55% 54.95% 49.15%
FixMatch w/o DA	97.20% 97.97%	97.37% 96.35% 76.26%	97.59% 97.52% 97.67%	91.57% **66.44%** **44.62%**	95.76% 93.82% 90.67%	96.11% 68.27% **54.40%**
	85.44% 85.66%	80.80% 56.82% 42.39%	85.09% 85.02% 85.24%	75.42% 62.43% 41.84%	80.73% 78.57% 76.41%	80.01% 60.32% 53.00%
FixMatch w/ DA	96.80% 98.12%	97.12% 97.12% 96.80%	97.52% 97.62% 97.30%	91.12% **65.97%** **46.97%**	93.58% 91.57% **88.44%**	97.40% 92.65% 80.37%
	85.59% 85.29%	81.40% 58.09% 43.23%	84.97% 84.42% 84.75%	75.97% 61.81% 40.95%	79.32% 77.21% 75.15%	81.75% 69.27% 63.79%
CCP (Ours)	96.85% 95.03%	94.50% 90.23% 90.28%	94.83% 94.50% 94.40%	94.33% 94.40% 94.28%	93.60% 92.90% 90.45%	94.45% 90.28% 90.25%
	84.10% 78.55%	74.05% 65.38% 61.38%	78.40% 77.68% 77.65%	74.60% 74.48% 71.08%	76.25% 74.63% 73.70%	75.68% 69.60% 65.88%

Table 1: Test set accuracy of each algorithm in each data variable scenario on CIFAR-10 (upper) and CIFAR-100 (lower). In the upper/lower portions of cells, the performance on each severity level is presented in descending order. U (L) refers to the perturbation of the unlabeled (labeled) distribution. Bold indicates under-performing the supervised baseline in that scenario.

Acknowledgements

References

Amini, M.-R.; Feofanov, V.; Pauletto, L.; Devijver, E.; and Maximov, Y. 2022. Self-Training: A Survey.

Berthelot, D.; Carlini, N.; Cubuk, E. D.; Kurakin, A.; Sohn, K.; Zhang, H.; and Raffel, C. 2019. ReMixMatch: Semi-Supervised Learning with Distribution Alignment and Augmentation Anchoring. *CoRR*, abs/1911.09785.

Bhattacharjee, A.; Karami, M.; and Liu, H. 2022. Text Transformations in Contrastive Self-Supervised Learning: A Review.

Cer, D.; Yang, Y.; Kong, S.-y.; Hua, N.; Limtiaco, N.; John, R. S.; Constant, N.; Guajardo-Cespedes, M.; Yuan, S.; Tar, C.; Sung, Y.-H.; Strope, B.; and Kurzweil, R. 2018. Universal Sentence Encoder.

Chang, M.-W.; Ratinov, L.; Roth, D.; and Srikumar, V. 2008. Importance of Semantic Representation: Dataless Classification. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, 830–835. AAAI Press. ISBN 9781577353683.

Chapelle, O.; Schölkopf, B.; and Zien, A., eds. 2006. *Semi-Supervised Learning*. The MIT Press. ISBN 9780262033589.

Chen, B.; Jiang, J.; Wang, X.; Wan, P.; Wang, J.; and Long, M. 2022. Debiased Self-Training for Semi-Supervised Learning.

Chen, P.; Ye, J.; Chen, G.; Zhao, J.; and Heng, P.-A. 2021. Beyond Class-Conditional Assumption: A Primary Attempt to Combat Instance-Dependent Label Noise. In *AAAI*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020a. A Simple Framework for Contrastive Learning of Visual Representations. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 1597–1607. PMLR.

Chen, T.; Kornblith, S.; Swersky, K.; Norouzi, M.; and Hinton, G. 2020b. Big Self-Supervised Models are Strong Semi-Supervised Learners.

Csiszar, I. 1975. *I*-Divergence Geometry of Probability Distributions and Minimization Problems. *The Annals of Probability*, 3(1): 146 – 158.

Dong, C.; and Schäfer, U. 2011. Ensemble-style Self-training on Citation Classification. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, 623–631. Chiang Mai, Thailand: Asian Federation of Natural Language Processing.

Dong, Y.; Chen, W.; Zhao, H.; Ma, X.; Gao, T.; and Li, X. 2020. Label propagation algorithm based on Roll-back detection and credibility assessment. *Mathematical Biosciences and Engineering*, 17(3): 2432–2450.

Dwork, C. 2011. The Promise of Differential Privacy: A Tutorial on Algorithmic Techniques. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, 1–2.

Heinzerling, B.; and Strube, M. 2018. BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages. In chair), N. C. C.; Choukri, K.; Cieri, C.; Declerck, T.; Goggi, S.; Hasida, K.; Isahara, H.; Maegaard, B.; Mariani, J.; Mazo, H.; Moreno, A.; Odijk, J.; Piperidis, S.; and Tokunaga, T., eds., *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs).

Iscen, A.; Tolias, G.; Avrithis, Y.; and Chum, O. 2019. Label propagation for deep semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5070–5079.

Johnson, R.; and Zhang, T. 2015. Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 919–927. Curran Associates, Inc.

Kamnitsas, K.; Castro, D.; Folgoc, L. L.; Walker, I.; Tanno, R.; Rueckert, D.; Glocker, B.; Criminisi, A.; and Nori, A. 2018. Semi-Supervised Learning via Compact Latent Space Clustering. In Dy, J.; and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 2459–2468. PMLR.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised Contrastive Learning.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1746–1751.

Kingma, D. P.; and Ba, J. 2015. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980.

Krizhevsky, A.; and Hinton, G. 2009. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*.

Lee, D.-H.; et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 896.

Lehmann, J.; Isele, R.; Jakob, M.; Jentzsch, A.; Kontokostas, D.; Mendes, P.; Hellmann, S.; Morsey, M.; Van Kleef, P.; Auer, S.; and Bizer, C. 2014. DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web Journal*, 6.

Li, J.; Xiong, C.; and Hoi, S. C. H. 2021. CoMatch: Semi-supervised Learning with Contrastive Graph Regularization. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9455–9464.

Oliver, A.; Odena, A.; Raffel, C. A.; Cubuk, E. D.; and Goodfellow, I. 2018. Realistic Evaluation of Deep Semi-Supervised Learning Algorithms. In Bengio, S.; Wallach, H.;

Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Pang, B.; and Lee, L. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, 115–124. Ann Arbor, Michigan: Association for Computational Linguistics.

Sahito, A.; Frank, E.; and Pfahringer, B. 2021. Better Self-training for Image Classification through Self-supervision.

Saito, K.; Kim, D.; and Saenko, K. 2021. OpenMatch: Open-Set Semi-supervised Learning with Open-set Consistency Regularization. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 25956–25967. Curran Associates, Inc.

Shi, W.; Gong, Y.; Ding, C.; Tao, Z. M.; and Zheng, N. 2018. Transductive semi-supervised deep learning using min-max features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 299–315.

Sohn, K.; Berthelot, D.; Carlini, N.; Zhang, Z.; Zhang, H.; Raffel, C. A.; Cubuk, E. D.; Kurakin, A.; and Li, C.-L. 2020. FixMatch: Simplifying Semi-Supervised Learning with Consistency and Confidence. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 596–608. Curran Associates, Inc.

Wang, G.-H.; and Wu, J. 2022. R2-D2: Repetitive Reprediction Deep Decipher for Semi-Supervised Deep Learning. *arXiv preprint arXiv:2202.08955*.

Wei, T.; and Gan, K. 2023. Towards Realistic Long-Tailed Semi-Supervised Learning: Consistency Is All You Need. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 3469–3478.

Xie, Q.; Dai, Z.; Hovy, E.; Luong, M.-T.; and Le, Q. V. 2020a. Unsupervised Data Augmentation for Consistency Training. arXiv:1904.12848.

Xie, Q.; Luong, M.-T.; Hovy, E.; and Le, Q. V. 2020b. Self-Training With Noisy Student Improves ImageNet Classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10684–10695.

Yang, X.; Song, Z.; King, I.; and Xu, Z. 2021. A Survey on Deep Semi-supervised Learning. *CoRR*, abs/2103.00550.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. *CoRR*, abs/1605.07146.

Zhang, X.; Zhao, J. J.; and LeCun, Y. 2015. Character-level Convolutional Networks for Text Classification. *CoRR*, abs/1509.01626.

Zhang, Y.; Zhang, X.; Qiu, R. C.; Li, J.; Xu, H.; and Tian, Q. 2022. Semi-supervised Contrastive Learning with Similarity Co-calibration. *ArXiv*, abs/2105.07387.

Zheng, M.; You, S.; Huang, L.; Wang, F.; Qian, C.; and Xu, C. 2022. SimMatch: Semi-supervised Learning with Similarity Matching. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 14451–14461. Los Alamitos, CA, USA: IEEE Computer Society.

Zhu, X.; and Ghahramani, Z. 2002. Learning from Labeled and Unlabeled Data with Label Propagation.

Hyperparameter	Value
Batch Size (Warmup/CCP Iterations)	512
Warmup Epochs	512
Batch Size (Classifier Training)	64
Classifier Training Epochs	512
Learning Rate (First CCP Iteration)	0.0006
Learning Rate	0.06
Similarity Metric	Angular
f_z output dimension	128
Activation	ReLU
Temperature τ	0.01
EMA loss decay	0.99
EMA model decay	0.999
λ for L2 regularization	0.0005
Optimizer	SGD with Nesterov Momentum $= 0.9$
Learning Rate Schedule	Cosine Decay $:= 0.06 \cos \frac{7\pi s}{S}$ where $s =$ step number, $S =$ total number of steps
Initial $d_{\max}$	0.01

Table 2: CCP hyperparameter choices.

A CIFAR Training Details

Tab. 2 details some of the CCP-specific and common hyperparameter choices used in all experiments.

A.1 Image Transformations

Each image transformation used in this work features one or more parameters that control the magnitude of its effect. Along with the transformation type, these parameters are randomly varied during training within predefined bounds. All transformations are applied sequentially per image with a probability of occurring. Random crop, horizontal flip, color jitter, and grayscale transformation occurs with probability 100%, 50%, 80%, 20%, respectively. A description of each $t \in \mathcal{T}$ can be found below.

- **Random crop**: Take a random crop of the image containing, at a minimum, 10% of the original image. Given that the aspect ratio (width over height) of CIFAR-10 images is 1.0, The aspect ratio of the crop must fall within $[0.75, 1.25]$. The image crops are resized back to 32×32 using bicubic interpolation.

- **Horizontal flip**: Flip images left-to-right.

- **Color Jitter**: Randomly distort the brightness, contrast, saturation, and hue of an image in a randomly chosen order. The maximum delta for brightness, contrast, and saturation jitter is set to 0.72 and 0.18 for hue jitter.

- **Grayscale**: Transform the colors of the entire image to grayscale.

B Normalized Data Variable Sensitivity Analysis

In Fig. 6, we report plots of the performance of each algorithm in our data variable sensitivity analysis that has been normalized with respect to the performance on the base case. This helps control for the effect of inherent label efficiency on the base case such that each data variable can be better analyzed independently.

C Additional Text Classification Experiments

As explained in Sec. 1, our CIFAR experiments focus on testing the reliability of CCP to outperform a supervised baseline in five common SSL data scenarios. We use CIFAR-10 and CIFAR-100 as starting points for those experiments. To ensure CCP works well with a completely different data environment and encoder, we additionally used four well-known text classification datasets: AG News (Zhang, Zhao, and LeCun 2015), DBpedia (Zhang, Zhao, and LeCun 2015; Lehmann et al. 2014), Rotten Tomatoes (Pang and Lee 2005), and Yahoo! Answers (Chang et al. 2008; Zhang, Zhao, and LeCun 2015) while investigating the few-label scenario. Some summary information can be found about these datasets in Tab. 3.

For text data, we compare CCP to a supervised Xent control with access to all labels, a supervised Xent baseline, and three algorithms that bear important similarities to CCP at varying levels of label availability in Tab. 5. SimCLR (Chen et al. 2020a) and SupCon (Khosla et al. 2020) are included as they are specific cases of $\mathcal{L}_{\text{SSC}}$. Further, the Compact Clustering via Label Propagation (CCLP) regularizer (Kamnitsas et al. 2018) can also be seen as a semi-supervised counterpart of SupCon. Since SimCLR and SupCon require a training session with clean labels only, they should be vulnerable to the few-label scenario. SimCLR and SupCon use the same f_z and f_g that we use with CCP. For CCLP, we attach f_g to f_b and apply CCLP to the hidden layer of f_g for a fair comparison. CCLP calls for linear decision boundaries formed after the hidden space where CCLP

Dataset	No. Classes	Balanced	No. Train Samples	Classification Task
AG News	4	Yes	120,000	Topic
DBpedia	14	Yes	560,000	Topic
Rotten Tomatoes	2	No	271,772	Sentiment
Yahoo! Answers	10	Yes	700,000	Topic

Table 3: Description of each text benchmark dataset.

Hyperparameter	AG News	DBpedia	Rotten Tomatoes	Yahoo! Answers
Input Size	512	512	256	1024
Dropout Rate	0.0	0.0	0.8	0.8
Batch Size	256	256	256	256
Optimizer	Adam	Adam	Adam	Adam
Activation	GELU	GELU	GELU	GELU
τ	0.1	0.1	0.1	0.1
Ξ	40	40	40	40
Weight Decay	0.0	0.0	0.001	0.001
Learning Rate	0.00004	0.00004	0.00004	0.00004
First Sequence of Parallel Convolutional Layers	$32@5\times100 \rightarrow 16@3\times1$	$32@5\times100 \rightarrow 16@3\times1$	$32@5\times100 \rightarrow 16@3\times1$	$64@5\times100 \rightarrow 32@3\times1$
Second Sequence of Parallel Convolutional Layers	$32@9\times100 \rightarrow 16@7\times1$	$32@9\times100 \rightarrow 16@7\times1$	$32@9\times100 \rightarrow 16@7\times1$	$64@9\times100 \rightarrow 32@7\times1$
Third Sequence of Parallel Convolutional Layers	-	-	-	$64@13\times100 \rightarrow 32@9\times1$
Global Max Pooling	True	True	True	True

Table 4: Hyperparameter settings of f_b for each text dataset. "$32@5\times100$" refers to a convolutional layer with 32 filters each of size 5×100 (Kingma and Ba 2015; Hendrycks and Gimpel 2016).

is applied via a traditional classification loss applied immediately after a final fully connected layer. We also search over the CCLP-specific hyperparameters such as the number of steps and CCLP weight and report the best results. All networks are trained until convergence according to their originally prescribed procedure. Each CCP experiment consists of 8 CCP iterations with a fixed $\Xi = 40$. We use the same subsampling procedure as our CIFAR experiments (initial $d_{\max} = 0.01$, divided by 2 after each iteration) between CCP iterations.

We use a common encoder, f_b for each of the text datasets for all algorithms. The solution is similar to other work that uses convolutional networks for text classification (Zhang, Zhao, and LeCun 2015; Kim 2014; Johnson and Zhang 2015). Briefly, text data is tokenized and each token index is used to look up a corresponding embedded vector (EV). The sequence of EVs is then processed by a convolutional neural network. Unlike our CIFAR experiments, we found no substantial benefit to using self-supervised pretraining to initialize f_b and f_z. We use the BPEmb byte-pair encoder (Heinzerling and Strube 2018) to transform a text document into a sequence of token indices cropped to a certain size. Each token index is used to look up an embedding vector (EV) whose initial values are set to the pretrained BPEmb values. The sequence of EVs is read by parallel convolutional layers whose output then feeds into additional convolutional layers depth-wise. The final activation maps undergo global max pooling to obtain a single floating point value per filter. These maximum activations are concatenated together to form b_i. f_z and f_g are designed as a 2-layer MLP. For f_z, the size of the hidden (output) layer is 64 (32). For f_g the hidden layer is of size 64 and the output layer size is the number of classes. Small adjustments are made to f_b to ensure the control and baseline reached the desired performance on each text dataset. An illustration of f_b used for text datasets can be found in Fig. 7. More information about the hyperparameters of f_b for each dataset can be found in Tab. 4.

C.1 Text Transformations

We design a set of text transformations, $\mathcal{T}$, inspired by the simplest and computationally cheapest set of transformations found in (Bhattacharjee, Karami, and Liu 2022). All transformations are implemented as tensor operations in the computational graph. All transformations act on a sample after it has been converted to a sequence of EVs and allow a gradient to pass through them. These consist of applying Laplacian noise consistent with Differential Privacy (Dwork 2011), applying Gaussian noise, randomly hiding and scrambling the order of EVs, and randomly swapping paragraphs and EVs. The use of transformations alone garners large accuracy increases for every algorithm particularly when label information is scarce. To eliminate this effect on our text experiments, every algorithm tested use the same $\mathcal{T}$ (and augmented batches). Below are more details on each $\mathcal{T}$ used on text:

% of Labels	Algorithm	DLP	AG News	DBpedia	Rotten Tomatoes	Yahoo! Answers
100%	Control	99.95%	91.95%	98.79%	82.01%	73.17%
1%	Baseline	99.23%	85.51%	94.25%	66.39%	63.24%
	CCLP	99.03%	79.54%	97.57%	64.48%	51.88%
	SimCLR	99.46%	86.89%	95.91%	68.10%	57.72%
	SupCon	99.40%	85.92%	96.12%	66.97%	60.19%
	CCP (ours)	**99.70%**	**88.89%**	98.24%	**71.14%**	**68.26%**
0.1%	Baseline	97.17%	78.37%	86.47%	59.89%	53.44%
	CCLP	95.56%	50.66%	67.24%	57.02%	37.80%
	SimCLR	97.50%	80.57%	87.75%	59.42%	37.69%
	SupCon	97.56%	74.84%	84.97%	57.96%	36.38%
	CCP (ours)	**98.88%**	**87.04%**	**96.20%**	**60.85%**	**63.23%**
0.05%	Baseline	94.71%	71.91%	72.71%	57.15%	48.86%
	CCLP	91.71%	50.17%	60.13%	55.53%	33.62%
	SimCLR	95.32%	77.09%	83.08%	56.07%	38.79%
	SupCon	94.96%	68.46%	80.70%	**57.42%**	28.25%
	CCP (ours)	**96.03%**	**85.14%**	**93.84%**	57.35%	**59.58%**

Table 5: The best test set accuracy across all text datasets. Bold indicates the best performance found across all algorithms.

- **Differential Privacy**: Laplacian noise is applied to all EVs in a fashion that is common in the practice of Differential Privacy (Dwork 2011). We randomly vary the strength of the noise, ϵ, between 10 and 100.
- **Gaussian Noise**: Gaussian noise is applied to all EVs. We randomly vary μ and $sigma$ between $[-0.5, 0.5]$ and $[0.01, 0.05]$, respectively.
- **Vector Hide**: We randomly replace EVs with the learned padding vector used to pad short inputs. The amount of EVs replaced is randomly varied from 10% to 25% of the total length.
- **Paragraph Swap**: We choose a random index along the length of a sequence of EVs and swap the content above and below that index.
- **Random Vector Swap**: We randomly replace EVs with randomly chosen EVs from the full vocabulary. The amount of EVs replaced is randomly varied from 10% to 25% of the total length.
- **Scramble**: We randomly select indices across the length of a sequence of EVs and randomly scramble their order. The amount of EVs we choose to scramble randomly varies from 10% to 25% of the total length.

C.2   Few-Label Performance

Mirroring our CIFAR experiments, note that only CCP outperforms or matches the baseline in every experiment in Tab. 5. SimCLR often outperforms the baseline, but CCP outperforms SimCLR in every experiment by the widest margin when label availability is lowest. CCLP underperforms the baseline in every experiment except on DBpedia with 1% label availability *i.e.* when the baseline is highly accurate and the classes are balanced. When CCLP succeeds, it outperforms the pretraining methods likely because it makes direct use of pseudo-labels. CCLP and CCP, like many other SSL algorithms, rely on network fitness for quality pseudo-labels. Accordingly, the margin of success for CCP shrinks as the baseline becomes less accurate *e.g.* on Rotten Tomatoes with 0.05% label availability where the baseline is $< 8\%$ better than random guessing.

C.3   CCP Iteration Performance

In Tab. 6 we display the accuracy of pseudo-labels produced by CCP at each iteration with and without subsampling at all levels of label availability for every text dataset. We can see most runs have converged after seven iterations although minor improvements are still occurring in some runs. Using subsampling can occasionally decrease performance across iterations *e.g.* on the Rotten Tomatoes dataset at 0.1% label availability. After seven iterations, the accuracy is higher than when not using subsampling, but further iterations appear to be decreasing accuracy. This necessitates the careful tuning of a subsampling schedule. Also, occasionally, runs without subsampling appear to return a higher accuracy at termination by a small margin despite a slower start. Again, this is explainable by an accumulation of errors brought forth by subsampling. A smaller initial $d_{\max}$ would alleviate this situation. We use a fixed $\Xi = 40$ in each CCP iteration for every text dataset. This means all eight CCP iterations consisted of only 320 epochs.

Dataset	% of Labels	Subsampling	Iteration						
			1	2	3	4	5	6	7
AG News	1%	N	83.51%	86.68%	87.30%	87.60%	87.79%	87.95%	87.89%
		Y	83.51%	86.50%	86.54%	87.23%	87.67%	88.00%	88.13%
	0.1%	N	75.29%	81.62%	83.27%	84.03%	84.48%	84.67%	84.84%
		Y	75.29%	82.17%	83.37%	84.18%	84.74%	85.15%	85.49%
	0.05%	N	67.14%	76.71%	79.66%	80.78%	81.70%	82.27%	82.72%
		Y	67.14%	77.56%	80.36%	81.56%	82.00%	82.63%	82.98%
DBpedia	1%	N	96.24%	97.23%	97.55%	97.69%	97.71%	97.78%	97.81%
		Y	96.24%	96.80%	97.06%	97.44%	97.58%	97.66%	97.85%
	0.1%	N	89.88%	92.84%	93.79%	94.23%	94.58%	94.81%	94.87%
		Y	89.88%	93.45%	94.11%	94.42%	94.84%	95.01%	95.25%
	0.05%	N	82.75%	87.43%	89.08%	89.99%	90.57%	90.86%	91.09%
		Y	82.75%	88.17%	90.04%	91.06%	91.50%	91.96%	92.26%
Rotten Tomatoes	1%	N	64.29%	67.84%	68.89%	69.42%	69.55%	69.57%	69.70%
		Y	64.29%	68.94%	69.92%	70.46%	70.60%	70.66%	70.61%
	0.1%	N	57.04%	57.75%	58.17%	58.32%	58.38%	58.44%	58.50%
		Y	57.04%	58.30%	59.11%	59.34%	59.28%	59.22%	59.14%
	0.05%	N	54.75%	55.28%	55.65%	55.79%	55.89%	55.93%	55.96%
		Y	54.75%	55.53%	56.31%	56.41%	56.38%	56.42%	56.41%
Yahoo! Answers	1%	N	63.06%	65.70%	66.46%	66.88%	66.90%	67.13%	67.22%
		Y	63.06%	66.20%	66.43%	66.87%	67.06%	67.30%	67.31%
	0.1%	N	52.82%	58.23%	59.76%	60.53%	60.99%	61.37%	61.61%
		Y	52.82%	58.57%	59.83%	60.32%	60.60%	60.76%	60.92%
	0.05%	N	45.25%	52.94%	55.28%	56.45%	57.19%	57.70%	57.99%
		Y	45.25%	53.50%	55.89%	56.98%	57.56%	57.89%	58.13%

Table 6: The accuracy of pseudo-labels after seven iterations of CCP. All runs with subsampling uses an initial $d_{\max} = 0.01$.

D Pseudo-Label Oscillation

Similar to (Chen et al. 2021), we observe oscillations of the predicted pseudo-label across epochs in the presence of a pseudo-label error. The credibility vector value for the correct class begins high (or at least reflects uncertainty *i.e.* near zero) but the network eventually memorizes the incorrect pseudo-label as training continues. When there is no pseudo-label error, the magnitude of oscillation is significantly less. Oscillations are also observed during the first iteration when no prior pseudo-label exists. This is attributable to the effects of random batch selection and network state. Randomly chosen examples of these behaviors are illustrated in Fig. 8. This is empirical evidence for our theoretical justification of CCP iterations in Sec. 3.5.
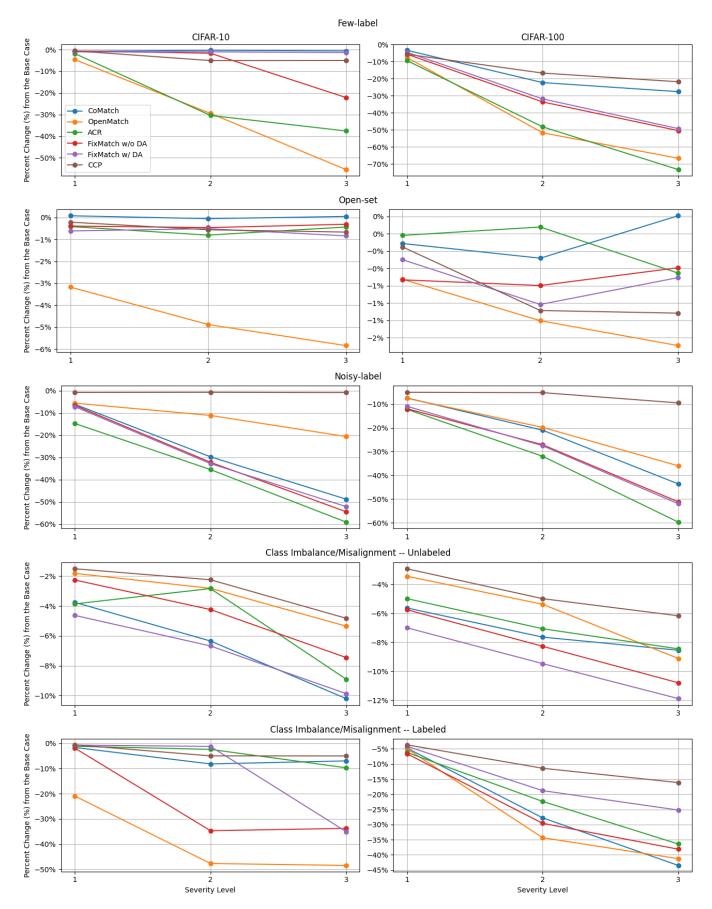
Figure 6: Normalizing the performance of each algorithm in our data variable sensitivity analysis with respect to the corresponding performance on the base case.
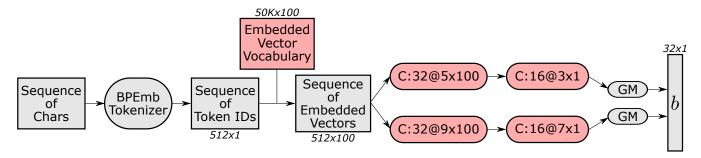
Figure 7: An illustration of an example encoder used in these text classification experiments, f_b. "C:32@5×100" refers to a convolutional layer with 32 filters each of size 5×100. "GM" refers to global max pooling over the feature activation maps. Red indicates learnable variables.
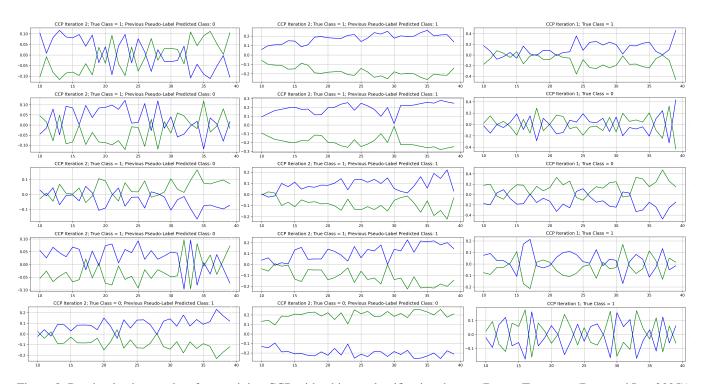

Figure 8: Randomly chosen plots from training CCP with a binary classification dataset (Rotten Tomatoes (Pang and Lee 2005)) with 1% label availability depicting pseudo-label oscillation over epochs. On the x-axis is the epoch number. On the y-axis is the non-scaled credibility value for class 1 in blue and class 0 in green. The leftmost column shows oscillation in the presence of a pseudo-label error. The middle column displays significantly degraded oscillation in the presence of a correct pseudo-label. The rightmost column depicts oscillations which can also be observed in the first CCP iteration.