Positive Unlabeled Contrastive Learning

Anish Acharya[*] UT Austin
Sujay Sanghavi UT Austin & Amazon
Li Jing Meta
Bhargav Bhushanam Meta
Dhruv Choudhary Meta
Michael Rabbat McGill University & Meta
Inderjit Dhillon UT Austin & Google

Abstract

Self-supervised pretraining on unlabeled data followed by supervised finetuning on labeled data is a popular paradigm for learning from limited labeled examples. In this paper, we investigate and extend this paradigm to the classical positive unlabeled (PU) setting - the weakly supervised task of learning a binary classifier only using a few labeled positive examples and a set of unlabeled samples. We propose a novel PU learning objective positive unlabeled Noise Contrastive Estimation (puNCE) that leverages the available explicit (from labeled samples) and implicit (from unlabeled samples) supervision to learn useful representations from positive unlabeled input data. The underlying idea is to assign each training sample an individual weight; labeled positives are given unit weight; unlabeled samples are duplicated, one copy is labeled positive and the other as negative with weights π and $(1-\pi)$ where π denotes the class prior. Extensive experiments across vision and natural language tasks reveal that puNCE consistently improves over existing unsupervised and supervised contrastive baselines under limited supervision.

1 Introduction

Learning from limited amount of labeled data is a long-standing challenge in modern machine learning. Owing to its recent widespread success in both computer vision and natural language processing tasks (Chen et al., 2020c; Grill et al., 2020; Radford et al., 2021; Gao et al., 2021; Dai and Le, 2015; Radford et al., 2018) *self supervised pretraining followed by supervised finetuning* has become the de-facto approach to learn from limited supervision.

These approaches typically leverage unlabeled data in a task agnostic manner during pretraining and use the available label information during finetuning (Hinton et al., 2006; Bengio et al., 2006; Mikolov et al., 2013; Kiros et al., 2015; Devlin et al., 2018; Zbontar et al., 2021; Assran et al., 2020). In this context, contrastive learning has emerged as one of the most promising approach to learn useful representations from unlabeled data by encouraging them to be invariant to distortions.

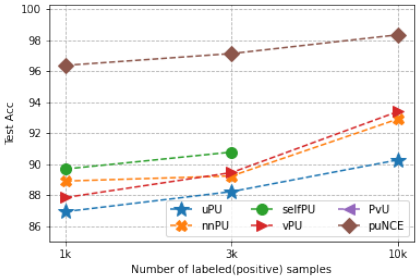


Figure 1: **Comparison of PU Learning algorithms**: All-Conv trained on PU CIFAR (vehicle / animal) for varying number of labeled positive samples n_P = 1k(2%), 3k(6%), 10k(20%). The proposed contrastive loss **puNCE** significantly improves over current PU baselines.

This paper investigates and extends contrastive representation learning to the Positive Unlabeled (PU) Learning (Denis, 1998) setting - the *weakly supervised* task of learning a binary classifier in

[*]work done while interning at Meta.

Preprint. Under review.

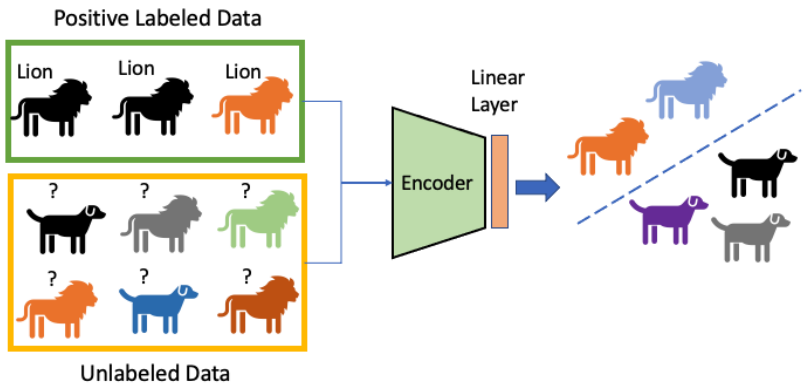


Figure 2: **Learning from Positive Unlabeled data.** No negative examples are labeled i.e. a classifier needs to be trained from an incomplete set of positive and a set of unlabeled samples such that it can discriminate between samples from positive and negative classes.

absence of any explicitly labeled negative examples i.e. *using an incomplete set of positive and a set of unlabeled samples. PU learning also implies learning under distribution shift since it involves generalization to unseen negative class* (Garg et al., 2021). This setting arises naturally in several real world applications where obtaining negative samples is either too resource intensive or improbable. For example, consider *personalized recommendation systems* (Naumov et al., 2019) where the training data is typically extracted from user feedback. Since explicit user feedback (e.g. user ratings) is hard to obtain, most practical recommendation systems rely on implicit user feedback (e.g. browsing history) (Kelly and Teevan, 2003) which usually indicates user's positive preference (e.g. if a user browses a product frequently or watched a movie then the user-item pair is labeled positive) (Chen et al., 2021). This setup has also been motivated by applications like gene and protein identification (Elkan and Noto, 2008), anomaly detection (Blanchard et al., 2010) and matrix completion (Hsieh et al., 2015).

While self supervised contrastive losses e.g. InfoNCE (Gutmann and Hyvärinen, 2010) have gained remarkable success in unsupervised representation learning; recent works (He et al., 2020; Kolesnikov et al., 2019) have pointed out that purely self-supervised approaches often produce inferior visual representation compared to fully supervised approaches. To address this issue, researchers (Khosla et al., 2020; Assran et al., 2020; Graf et al., 2021) proposed supervised variants of contrastive loss (SCL) that, by leveraging available supervision, is able to obtain equally (or even more) discriminative representations as fully supervised cross entropy objective. Motivated by these observations, *our goal is to design a contrastive loss leveraging the available weak supervision in an efficient manner to learn rich representation $g_B(\mathbf{x})$ from Positive Unlabeled dataset, which is able to discriminate between positive and negative classes*.

To this end, we propose **puNCE** (positive unlabeled Noise Contrastive Estimation) - *a novel contrastive training objective that extends the standard self supervised infoNCE (Gutmann and Hyvärinen, 2010; Chen et al., 2020b) loss to the positive unlabeled setting; by incorporating available biased supervision*. Unlike recent supervised variants of infoNCE (Khosla et al., 2020; Assran et al., 2020; Zhong et al., 2021) which can only leverage explicit (strong) supervision (e.g in form of labeled data), puNCE is also able to leverage implicit (weak) supervision from the unlabeled data. The main idea is to use the fact that unlabeled data is distributed as a mixture of positive and negative marginals where the mixture proportion is given by class prior π (Elkan and Noto, 2008; Niu et al., 2016; Du Plessis et al., 2014; Elkan, 2001). puNCE treats each unlabeled sample as a positive example and a negative example with appropriate probabilities and further uses the available explicit label information (i.e. set of positive labeled samples) to pull together multiple positive samples in the embedding space (Khosla et al., 2020). Our experiments across PU and binary semi-supervised settings suggest that *in settings with limited supervision puNCE can be particularly effective and often produce stronger representations than both infoNCE and its supervised variants* (Khosla et al., 2020; Assran et al., 2020). Our experiments on standard PU learning benchmarks reveal that even with small amount of weak supervision, puNCE can dramatically improve the quality of the resulting embedding compared to unsupervised contrastive training using infoNCE. For example, on PU CIFAR10 with

ResNet-18 puNCE achieves over 3% improvement compared to infoNCE while using 20% labeled data, and over 2% improvement when using only 5% labeled data. We further show that the resulting PU Learning approach i.e. *contrastive training the encoder using puNCE, followed by freezing the encoder and training a linear layer on top using standard cost-sensitive PU loss* (Elkan and Noto, 2008; Du Plessis et al., 2014; Kiryo et al., 2017) results in significant improvement in generalization performance compared to current state-of-the-art (Figure 1, Table 2). For example, on PU CIFAR-10 benchmark with only 1k positive labeled samples (i.e. rest 49k training samples unlabeled), our contrastive approach improves over existing PU learning approaches by 8.9% with a ResNet-18.

Our contributions can be summarized as follows:

- We investigate the limitation of general self-supervised pretraining and finetuning approach on the weakly supervised PU learning task. We observe that self supervised contrastive losses are unable to leverage available supervision and produce inferior representations compared to fully supervised approaches. The gains from existing supervised contrastive losses are also limited when amount of explicitly labeled data is small (Table 1).
- We propose a novel contrastive training objective - puNCE (positive unlabeled Noise Contrastive Estimation), that extends contrastive loss to the positive unlabeled setting by incorporating the available biased supervision via appropriately re-weighting the samples.
- We perform experiments on three settings with limited supervision - (a) Positive Unlabeled Classification, (b) Binary semi-supervised (PNU) classification and (c) Contrastive few shot finetuning of pretrained language model. Across all the setting, puNCE consistently outperforms supervised and unsupervised contrastive losses and is especially powerful when available supervision is scarce. On PU classification task, contrastive pretraining with puNCE results in large improvement over current state-of-the-art PU learning algorithms (Table 2).

2 Problem Setup.

2.1 Positive Unlabeled Learning.

Let $\mathrm{x} \in \mathbb{R}^d$, $d \in \mathbb{N}$ and $\mathrm{y} \in \{\pm 1\}$ be the input and output random variables respectively and $p(\mathrm{x}, \mathrm{y})$ be the true underlying joint density of (x, y). PU dataset $\mathcal{X}_{PU}$ is composed of two independent sets of data, sampled uniformly at random from $p(\mathrm{x}, \mathrm{y})$: an incomplete set $\mathcal{X}_P$ of n_p data-points with positive class and a set $\mathcal{X}_U$ of n_u unlabeled samples:

$$\mathcal{X}_P = \{\mathbf{x}_i^P\}_{i=1}^{n_p} \sim p(\mathrm{x}|\mathrm{y}=1);\ \mathcal{X}_U = \{\mathbf{x}_i^U\}_{i=1}^{n_u} \sim p(\mathrm{x});\ \mathcal{X}_{PU} = \mathcal{X}_P \cup \mathcal{X}_U \tag{1}$$

Since information about y is unavailable for all samples, it is convenient to define an indicator random variable s such that: **s = 1 if x is labeled and 0 otherwise**. Now viewing $\mathbf{x}, y, s$ as random variables allows us to assume that there is some true underlying distribution $p(\mathbf{x}, y, s)$ over triplets $(\mathbf{x}, y, s)$ however, for each sample only $(\mathbf{x}, s)$ is recorded. The definition of PU dataset (1) immediately implies the following two results:

$$P(y=+1|s=1) = 1\ ,\ p(s=1|y=-1) = 0 \tag{2}$$

Assumption 1 (Known Class Prior). *Throughout the paper, we assume that the class prior $\pi = p(y=+1)$ is either known or can be efficiently estimated from $\mathcal{X}_{PU}$ via mixture proportion estimation algorithm (Christoffel et al., 2016; Ramaswamy et al., 2016; Ivanov, 2020; Garg et al., 2021).*

Note that, this is a standard assumption made in PU Learning literature and is at the heart of most classical cost sensitive PU Learning algorithms (Elkan and Noto, 2008; Kiryo et al., 2017; Du Plessis et al., 2014; Chen et al., 2020a; Niu et al., 2016).

The PU Learning task is thus to train a classifier $f : \mathbb{R}^d \to \mathbb{R}$ using only PU observations $(\mathbf{x}, s)$ and knowledge of π, such that $f(\mathbf{x})$ is a close approximation of $p(y=+1|\mathbf{x})$. Where, f is parameterized in terms of a linear layer $\mathbf{v} \in \mathbb{R}^k$ and feature extractor $g_B(\mathbf{x}) \in \mathbb{R}^{k \times d}$ i.e.

$$f_{\mathbf{v},B} = \mathbf{v}^T g_B(\mathbf{x}) : \mathbb{R}^d \to \mathbb{R}\ ,\ g_B(\mathbf{x}) \in \mathbb{R}^{k \times d}\ ,\ \mathbf{v} \in \mathbb{R}^k \tag{3}$$

2.2 Contrastive Representation Learning.

The main idea of contrastive learning (Sohn, 2016; Wu et al., 2018) is to contrast semantically similar and dissimilar samples - encouraging the representations of similar pairs to be close and that of the dissimilar pairs to be more orthogonal.

In particular, at iteration t, given a randomly sampled batch $\mathcal{D}_t = \{\mathbf{x}_i \sim p(\mathbf{x})\}_{i=1}^{b}$ of b data points, the corresponding *multi-viewed batch* consists of 2b data points $\tilde{\mathcal{D}}_t = \{\mathbf{x}_i\}_{i=1}^{2b}$ where $\mathbf{x}_{2i-1}$ and $\mathbf{x}_{2i}$ are two views of $\mathbf{x}_i \in \mathcal{D}_t$ obtained via stochastic augmentations (Chen et al., 2020d; Tian et al., 2020). Within the multi-viewed batch, consider an arbitrary augmented data point indexed by $i \in \mathbb{I} \equiv \{1, \ldots, 2b\}$ and let $a(i)$ be the index of the corresponding augmented sample originating from the same source sample. Denote the embedding of sample $\mathbf{x}_i$ as $\mathbf{z}_i = g_{B_t}(\mathbf{x}_i)$, then the *self supervised contrastive loss* **infoNCE** (Gutmann and Hyvärinen, 2010; Oord et al., 2018) is given as:

$$\mathcal{L}_{infoNCE} = \sum_{i\in\mathbb{I}} \ell^{(i)}_{InfoNCE} = \sum_{i\in\mathbb{I}} \left[-\log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_{a(i)}/\tau\right)}{\sum_{k\in\mathbb{I}\setminus\{i\}} \exp\left(\mathbf{z}_i \cdot \mathbf{z}_k/\tau\right)} \right] \tag{4}$$

In this context $\mathbf{x}_i$ is commonly referred to as the **anchor**, $\mathbf{x}_{a(i)}$ is called **positive** and the other $2(b-1)$ samples are considered **negatives**. Simply put, InfoNCE loss pulls the anchor and the positive in the embedding space while pushing away anchor from negatives.

Projection Layer. Rather than directly using the output of the encoder $g_B(\mathbf{x})$ to contrast samples, it is common practice (Chen et al., 2020b; Zbontar et al., 2021; Grill et al., 2020; Khosla et al., 2020; Assran et al., 2020) to feed the representation through a projection network $h_{\theta_{proj}}$ to obtain a lower dimensional representation before comparing the representations i.e. instead of using $\mathbf{z}_i = g_{B_t}(\mathbf{x}_i)$, in practice we consider $\mathbf{z}_i = h_{\theta_{proj}}(g_{B_t}(\mathbf{x}_i))$. *The projector is only used during optimization and discarded during downstream transfer task.*

Contrastive learning with supervision. Supervised Contrastive Learning (SCL) (Khosla et al., 2020; Zhong et al., 2021; Graf et al., 2021; Assran et al., 2020) is a supervised variant of infoNCE that considers multiple positive pairs from other samples belonging to the same class as anchor in addition to the augmented view. in fully supervised setting, SCL can be computed as:

$$\mathcal{L}_{SCL} = \sum_{i\in\mathbb{I}} \ell^{(i)}_{SCL} = -\sum_{i\in\mathbb{I}} \frac{1}{|\mathbb{Q}(i)|} \sum_{j\in\mathbb{Q}(i)} \log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_j/\tau\right)}{\sum_{k\in\mathbb{I}\setminus\{i\}} \exp\left(\mathbf{z}_i \cdot \mathbf{z}_k/\tau\right)} \tag{5}$$

where, $\mathbb{Q}(i) = \{t \in \{\mathbb{I}\setminus\{i\}\} : y_t = y_i\}$ denote the set of all samples in the current batch that belong to the same class as the anchor.

3 Positive Unlabeled NCE

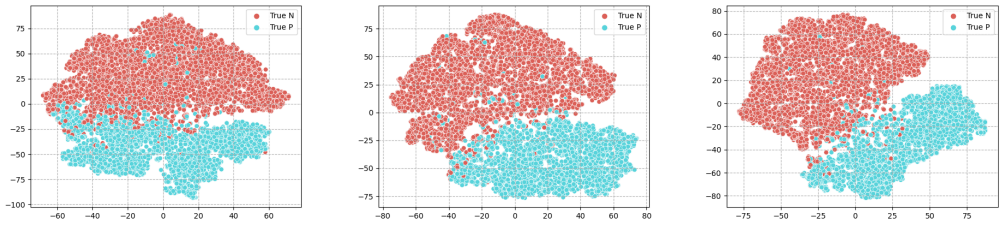
We propose **puNCE** (positive unlabeled InfoNCE) that extends the self supervised InfoNCE loss to the PU learning setting. *The main idea of puNCE is to consider an unlabeled sample as positive with probability π and negative with probability $(1-\pi)$.* This follows from Bayes rule since we can write the marginal density of the unlabeled data as:

$$p(\mathbf{x}) = \pi p(\mathbf{x}|y=1) + (1-\pi)p(\mathbf{x}|y=-1) \tag{6}$$

In particular, consider a **labeled anchor** $\mathbf{x}_i \in \mathcal{X}_P$, the puNCE risk corresponding to $\mathbf{x}_i$ is computed by pulling together normalized embeddings of all the available labeled samples in multi-viewed batch $\tilde{\mathcal{D}}_t$ as opposed to InfoNCE which only considers $\mathbf{x}_{a(i)}$ as positive. This holds, since all labeled samples are guaranteed to come from the same latent class as $\mathbf{x}_i$ (2) . Given multi-view batch $\tilde{\mathcal{D}}_t$, let $\mathbb{P} \equiv \{k \in \mathbb{I} : s_k = 1\}$ denote the set of indices of all **labeled samples** and $\mathbb{U} \equiv \{k \in \mathbb{I} : s_k = 0\}$ indexes the **unlabeled samples** i.e. $\mathbb{U} \equiv \mathbb{I} \setminus \mathbb{P}$. Then, puNCE empirical risk ℓ_P for the labeled samples is given as:

$$\ell_P = \sum_{i\in\mathbb{P}} \ell^{(i)}_{puNCE} = -\sum_{i\in\mathbb{P}} \frac{1}{|\mathbb{P}|-1} \sum_{j\in\mathbb{P}\setminus\{i\}} \log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_j/\tau\right)}{\sum_{k\in\mathbb{I}\setminus\{i\}} \exp\left(\mathbf{z}_i \cdot \mathbf{z}_k/\tau\right)} \tag{7}$$

On the other hand, **unlabeled anchor** $\mathbf{x}_i \in \mathcal{X}_U$, is treated as positive example with probability $\pi = p(y=1|s=0)$ and as negative example with probability $(1-\pi)$. When considered positive



(a) infoNCE (b) puNCE ($n_p = 1k$) (c) puNCE ($n_P = 3k$)

Figure 3: **Performance with increased Label Information** t-SNE visualization of learned representations on CIFAR10. Classes are indicated by colors. Clearly the modified PUNCE objective leads to better class separation than InfoNCE. Further, we also note that the quality of the representation improves with increasing available positive labeled samples as expected intuitively.

Dataset	$g_B(\cdot)$	n_P	infoNCE	DCL	SCL	puNCE
PU MNIST (odd/even)	MLP	1k	94.15±0.15	94.32±0.42	94.24±0.09	**94.70**±0.19
		3k	94.84±0.25	95.09±0.36	95.82±0.18	**96.01**±0.21
		10k	95.15±0.05	95.45±0.08	**98.29**±0.08	**98.27**±0.11
PU CIFAR (animal / vehicle)	ResNet18	1k	96.33±0.14	96.21±0.11	97.42±0.07	**97.59**±0.17
		3k	96.51±0.06	96.49±0.03	97.66±0.04	**97.87**±0.04
		10k	96.58±0.04	96.50±0.02	97.70±0.07	**97.95**±0.02

Table 1: Linear Evaluation of different contrastive losses - infoNCE (Chen et al., 2020b), DCL (Chuang et al., 2020), SCL (Khosla et al., 2020) under Positive Unlabeled setting. puNCE is particularly effective when available supervision is limited. As we increase supervision, SCL starts to improve and become comparable. This interplay follows from the fact that they become equivalent in the fully supervised setting.

($y = +1$), all the labeled samples along with unlabeled augmentation $\mathbf{x}_{a(i)}$ are used as positive pairs for $\mathbf{x}_i$. Whereas, when it is considered negative ($y = -1$), since there are no available labeled negative samples puNCE treats only augmentation $\mathbf{x}_{a(i)}$ as positive pair. The empirical risk on unlabeled samples ℓ_U is thus computed as:

$$\ell_U = \sum_{i\in\mathbb{U}} \ell^{(i)}_{puNCE} = -\sum_{i\in\mathbb{U}} \left[\frac{\pi}{|\mathbb{P}|+1} \sum_{j\in\{\mathbb{P},a(i)\}} \log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_j/\tau\right)}{\sum_{k\in\mathbb{I}\setminus\{i\}} \exp\left(\mathbf{z}^t_i \cdot \mathbf{z}_k/\tau\right)} + (1-\pi) \log \frac{\exp\left(\mathbf{z}_i \cdot \mathbf{z}_{a(i)}/\tau\right)}{\sum_{k\in\mathbb{I}\setminus\{i\}} \exp\left(\mathbf{z}_i \cdot \mathbf{z}_k/\tau\right)} \right] \tag{8}$$

The first term of (8) denotes the loss incurred by the positive contribution of the unlabeled samples and the second term corresponds to the negative counterpart. Combining (7), (8) the puNCE empirical risk can be computed as:

$$\mathcal{L}_{puNCE} = \frac{1}{|\tilde{D}_t|} \sum_{i\in\mathbb{I}} \ell^{(i)}_{puNCE} = \frac{1}{2b} \left[\sum_{i\in\mathbb{P}} \ell^{(i)}_{puNCE} + \sum_{i\in\mathbb{U}} \ell^{(i)}_{puNCE} \right] = \frac{1}{2b} \left[\ell_P + \ell_U\right] \tag{9}$$

This simply says that, in essence *puNCE assigns each training example an individual weight. In particular, all the labeled samples are given unit weight and the unlabeled samples are duplicated; one copy is labeled positive with weight* π *and the other copy is labeled negative with weight* $(1-\pi)$.

Extension to binary semi-supervised learning. PU Learning is closely related to semi-supervised learning. In semi-supervised learning labeled samples from both the classes along with unlabeled samples are available i.e. it contains three independent sets of samples: positive labeled, negative labeled and unlabeled samples (referred to as positive negative unlabeled (PNU) learning). A natural question to ask is if the idea of using implicit weak supervision from the unlabeled samples can also help in the general binary semi-supervised learning setting i.e. when the training data is limited but unbiased. Let $\mathbb{P}$, $\mathbb{N}$ and $\mathbb{U}$ denote the set of labeled positive, labeled negative and unlabeled samples

respectively and $\mathbb{Q}(i) = \{t \in \{\mathbb{I} \setminus \{i\}\} : y_t = y_i\}$ denote the set of all samples in the current batch that belong to the same class as the anchor and $\mathbb{P}(i) = \{\mathbb{P}, a(i)\}$, $\mathbb{N}(i) = \{\mathbb{N}, a(i)\}$, $\mathbb{I}(i) = \mathbb{I} \setminus i$. Then the puNCE loss in the PNU setting takes the following form:

$$\mathcal{L}_{puNCE}^{PNU} = -\frac{1}{2b}\left[\sum_{i\in\mathbb{P}\cup\mathbb{N}}\frac{1}{|\mathbb{Q}(i)|}\sum_{\mathbb{Q}(i)}\log\frac{\exp\left(\mathbf{z}_i\cdot\mathbf{z}_j/\tau\right)}{\sum_{k\in\mathbb{I}\setminus\{i\}}\exp\left(\mathbf{z}_i\cdot\mathbf{z}_k/\tau\right)}\right.$$
$$\left.+\sum_{i\in\mathbb{U}}\left(\frac{\pi}{|\mathbb{P}(i)|}\sum_{j\in\mathbb{P}(i)}\log\frac{\exp\left(\mathbf{z}_i\cdot\mathbf{z}_j/\tau\right)}{\sum_{k\in\mathbb{I}(i)}\exp\left(\mathbf{z}_i^t\cdot\mathbf{z}_k/\tau\right)}+\frac{1-\pi}{|\mathbb{N}(i)|}\sum_{j\in\mathbb{N}(i)}\log\frac{\exp\left(\mathbf{z}_i\cdot\mathbf{z}_j/\tau\right)}{\sum_{k\in\mathbb{I}(i)}\exp\left(\mathbf{z}_i^t\cdot\mathbf{z}_k/\tau\right)}\right)\right] \tag{10}$$

Supervised and Unsupervised Setting. It is straightforward to see that in the fully supervised setting puNCE reduces to SCL and in the unsupervised setting it is equivalent to infoNCE.

4 Related Work

PU Learning. Owing to its importance in several real world problems (e.g. recommendation), developing specialized learning algorithms for PU setting have received renewed impetus in the machine learning community. Most of the recent research in this area can be broadly categorized into two major class of algorithms - based on how they handle unlabeled samples. At a high level, the first class of algorithms (Liu et al., 2002, 2003; Bekker and Davis, 2020; Luo et al., 2021; Chen et al., 2020d) tries to identify potential (with high confidence) negative, positive or both samples from the unlabeled pool of data; then perform traditional supervised learning together with the available labeled P data. The second set of algorithms (Elkan and Noto, 2008; Du Plessis et al., 2014; Kiryo et al., 2017) instead rely on finding appropriate weights for the positive and unlabeled samples and then performing standard cost sensitive training (Elkan, 2001). Our approach is closely related to the second set of methods as we treat each unlabeled sample as a convex combination of positive and negative samples and weight them accordingly.

Contrastive Loss. Self-supervised learning has demonstrated superior performances over supervised methods on various benchmarks. Joint-embedding methods (Chen et al., 2020b; Grill et al., 2020; Zbontar et al., 2021; Caron et al., 2021) are one the most promising approach for self-supervised representation learning where the embeddings are trained to be invariant to distortions. To prevent trivial solutions, a popular method is to apply pulsive force between embeddings from different images, known as contrastive learning. Contrastive loss is shown to be useful in various domains, including natural language processing (Gao et al., 2021), multimodal learning (Radford et al., 2021). Contrastive loss can also benefit supervised learning (Khosla et al., 2020).

5 Experiments

In this section we state our experimental setup, present our empirical findings and discuss insights about how puNCE benefits learning from weak supervision. We compare puNCE with other popular contrastive losses including losses that can leverage available supervision. We also perform several ablations on puNCE and downstream PU classification. To ensure reproducibility, all the experiments are run with deterministic cuDNN back-end and repeated 5 times with different random seeds and the confidence intervals are noted.

Datasets. We evaluate puNCE on several common PU Learning image and text classification benchmarks consistent with recent literature (Garg et al., 2021; Chen et al., 2020d; Kiryo et al., 2017). In particular, for *vision benchmarks* we consider the following binary versions of MNIST and CIFAR10. (i) **CIFAR10 (Vehicle / Animal)**: Animal images (bird, cat, deer, dog, frog, horse) are treated as negative and Vehicle images (airplane, automobile, ship, truck) are treated as positive. (ii) **MNIST (Odd / Even)**: Separate the odd digits (1, 3, 5, 7, 9) from the even digits (0, 2, 4, 6, 8). We perform *text classification*, on **SST2** sentiment dataset (Socher et al., 2013) where the goal is to classify positive and negative reviews. We simulate **PU settings** from the binary data as follows: from the true positive samples in the training data we sample n_P samples uniformly at random and label them positive while we mark the remaining training samples as unlabeled. For example, PU

Dataset	Encoder	Algorithm	n_P = 1k	n_P = 3k	n_P = 10k
PU CIFAR (vehicle/animal)	ResNet 18	PN (Elkan and Noto, 2008)	68.19±1.38	68.12±0.17	79.64±0.28
		PvU (Elkan and Noto, 2008)	88.39±0.15	90.16±0.19	93.54±0.21
		uPU(Du Plessis et al., 2014)	87.94±0.54	89.53±0.12	92.37±0.35
		nnPU (Kiryo et al., 2017)	88.69±0.08	89.46±0.22	91.58±0.21
		vPU(Chen et al., 2020a)	75.71±0.31	83.64±0.28	89.65±0.19
		puNCE(This work)	**97.59**±0.17	**97.87**±0.04	**97.95**±0.02
	All Conv	PvU (Elkan and Noto, 2008)	59.86±0.15	61.20±0.24	75.25±0.79
		uPU (Du Plessis et al., 2014)	86.93±0.28	88.22±0.83	90.26±1.60
		nnPU (Kiryo et al., 2017)	88.90±0.30	89.23±0.33	92.90±0.58
		SelfPU (Chen et al., 2020d)	89.68±0.22	90.77±0.21	-
		vPU (Chen et al., 2020a)	87.84±0.47	89.45±0.39	93.38±0.02
		puNCE (This work)	**96.37**±0.14	**97.13**±0.10	**98.34**±0.09
PU MNIST (Odd/even)	MLP	PN (Elkan and Noto, 2008)	65.32±0.90	94.16±0.26	97.02±0.19
		PvU(Elkan and Noto, 2008)	91.10±0.91	93.71±0.91	95.31±0.48
		uPU(Du Plessis et al., 2014)	91.14±0.87	93.84±0.16	96.43±0.20
		nnPU(Kiryo et al., 2017)	91.83±0.79	95.53±0.29	97.18±0.12
		puNCE	**94.70**±0.19	**96.01**±0.21	**98.27**±0.11

Table 2: PU Computer Vision Benchmarks. We compare our contrastive PU Learning approach i.e. training encoder g_B using puNCE, subsequently freezing the encoder and only training a linear layer on top. For all these experiments linear probing is done using nnPU loss.

MNIST(Odd / Even) training dataset with $n_P = 1k$ consists of $1k$ labeled samples (positive) and $59k$ unlabeled ($29k$ true positive and $30k$ true negatives) samples. On the other hand to simulate **binary semi-supervised PNU setting**, we instead randomly sample n_l indices and keep the corresponding true label (labeled samples can contain both true positives and negatives), while all the other samples are marked as unlabeled ($s = 0$).

Learner Architecture. Recall that, our PU learneris parameterized in terms of an encoder $g_B(\cdot)$ with parameters B and a linear layer with parameters $\mathbf{v}$. On PU-CIFAR(Animal/Vehicle) we perform experiments with ResNet18 (He et al., 2016) and All convolution net (Springenberg et al., 2014) encoders. For PU-MNIST(Odd/Even) we train a MLP encoder with ReLU activation. For SST2 dataset we use a pretrained RoBERTa encoder (Liu et al., 2019).

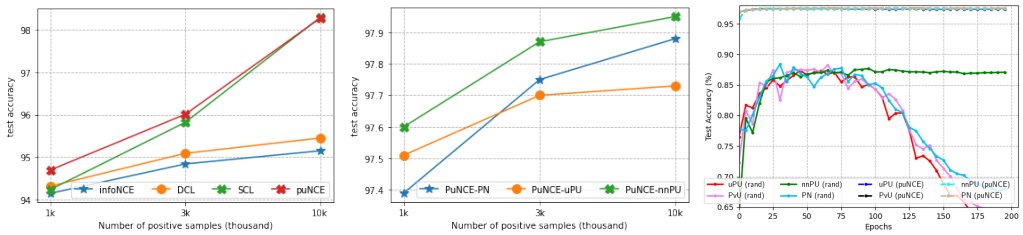
Contrastive Loss Baselines. We compare puNCE with several popular contrastive losses including unsupervised variants - InfoNCE (Chen et al., 2020b), Debiased Contrastive Learning (DCL) (Chuang et al., 2020) as well as variants that can leverage explicit supervision - Supervised Contrastive Learning (abbreviated SupCon) (Khosla et al., 2020; Assran et al., 2020). Following the setup of prior work on learning with limited supervision (Chen et al., 2020c; Assran et al., 2020) we adopt SCL to the PU setting in the following manner: At iteration t, given an augmented batch $\tilde{\mathcal{D}}_t$, on the labeled samples the contrastive loss is computed using SupCon loss while on the unlabeled samples in the augmented batch we compute the loss using standard infoNCE objective.

$$\mathcal{L}_{SCL}^{PU} = -\frac{1}{2b}\left[\sum_{i\in\mathbb{P}}\frac{1}{|\mathbb{P}|-1}\sum_{j\in\mathbb{P}\setminus\{i\}}\log\frac{\exp\left(\mathbf{z}_i\cdot\mathbf{z}_j/\tau\right)}{\sum_{k\in\mathbb{I}(i)}\exp\left(\mathbf{z}_i\cdot\mathbf{z}_k/\tau\right)}+\sum_{i\in\mathbb{U}}\ell_{infoNCE}^{(i)}\right] \tag{11}$$

PU Learning Baselines. We compare the downstream PU classification performance of puNCE against many related PU Learning algorithms at different levels of supervision ($n_P = 1k, 3k, 10k$). Our baselines include: PN (Naumov et al., 2019; Elkan and Noto, 2008) - considering all the unlabeled samples as negative and performing binary classification using standard loss (e.g. CE), cost sensitive approaches like PvU (Elkan, 2001), uPU (Du Plessis et al., 2014), nnPU (Kiryo et al., 2017), and other approaches like vPU (Chen et al., 2020a),SelfPU (Chen et al., 2020d).

5.1 Contrastive Positive Unlabeled Learning

In this section we evaluate the performance of the proposed puNCE loss on downstream PU classification problem. As discussed before, our contrastive PU learning approach involves training



(a) Contrastive pretraining (b) Linear Probing (c) Training Stability

Figure 4: (ResNet18 - PU CIFAR10) (4a) Comparing different contrastive learning approaches on downstream PU classification task across different amount of supervision ($n_P = 1k, 3k, 10k$). (4b) Linear probing on top of puNCE features using standard supervised cross entropy (PN), nnPU (Kiryo et al., 2017), uPU (Du Plessis et al., 2014). (4c)Test accuracy during training with puNCE initialization and random initialization of linear layer.

encoder $g_B(\cdot)$ using contrastive loss, followed by training a linear layer on top using standard positive unlabeled loss (e.g. nnPU (Kiryo et al., 2017)). For non contrastive classic PU baselines both the encoder (B) and the linear layer ($\mathbf{v}$) are jointly trained using the PU loss. Contrastive training is done using LARS optimizer (You et al., 2019), temperature set to 0.5. We used batch size 2048 for CIFAR10 experiments and 1024 for MNIST experiments. We used an initial learning rate of 0.01 with cosine annealing learning rate for 300 epochs on PU CIFAR10 and 200 epochs for PU MNIST.

Fine Tune vs Linear Probe.

After initializing the learner with the trained encoder parameters, we explore two popular transfer methods to transfer to the downstream Positive Unlabeled task - fine tuning (updating both $\mathbf{v}$ and B), linear probing (updating $\mathbf{v}$ while freezing the lower layers) using the PU training dataset.

Table 3: Comparing different transfer methods on training ResNet18 on PU CIFAR 10 - Linear probing performs consistently better than fine-tuning.

n_P	LP	FT
1k	**97.59**±0.17	92.96±0.19 (↓ 4.36)
3k	**97.87**±0.04	95.83±0.15 (↓ 2.04)
10k	**97.95**±0.02	97.41±0.09 (↓ 0.54)

In standard ID setting i.e. when finetuning data and test data come from the same distribution, it is well known that finetuning results in better ID generalization performance than linear probing (He et al., 2020; Zhai et al., 2019). However, in the PU setting we observe that finetuning consistently under-performs linear probing. In fact, as the bias increases (i.e. for lower n_P), the gap becomes more significant (see Table 3). One possible way to explain this phenomenon is through feature distortion theorem (Kumar et al., 2022). Suppose, there exists an optimal B_* and v_* and suppose our representation quality is good such that our encoder parameters B_0 satisfies $\min_U \|B_0 - UB_*\| \leq \epsilon$ over all rotation matrices $U \in \mathbb{R}^{k\times k}$ and v is initialized randomly to v_0. Then we know that $v_0 v_0^T - B_0 B_0^T = v_{ft} v_{ft}^T - B_{ft} B_{ft}^T$ where v_{ft}, B_{ft} denote the parameters post finetuning (Du et al., 2020). This suggest that since in PU setting the training data is biased, finetuning on it can result in significant feature distortion resulting in large error on data with large distribution shift. This is consistent with our observation, since as we reduce number of labeled samples i.e. we increase the bias and for small n_P, the distribution shift is large resulting in larger drop in performance.

Fine tune loss. We experiment with different cost sensitive PU losses to during linear probing. We notice that they perform similarly with nnPU performing marginally better (Figure 4b).

Stability of PU Learning. The classical cost-sensitive PU learning algorithms are notoriously unstable and tend to overfit the data. These methods need careful hyper parameter tuning and early stopping. Interestingly, we observe that when the model is initialized with parameters obtained from puNCE pretraining, the PU training becomes much more stable (Figure 4c) suggesting that puNCE pretraining is also inherently robust.

Effect of Number of Labeled Samples. To understand the benefit of increased supervision, we perform several experiments across different values of n_P. Our experiments (see Table 1 and Figure 4a) reveal that supervised approaches(SCL, puNCE) are consistently able to learn more powerful representation when supervision is available. The downstream performance of these approaches

Dataset	$g_B(\cdot)$	Labeled	infoNCE	SCL	puNCE
PNU CIFAR 10 (vehicle/animal)	All Conv	1%	83.96±2.45	86.70±0.93	**88.09±1.88**
		5%	93.92±0.43	**96.16±0.06**	**96.14±0.34**
		10%	94.98±0.34	96.39±0.02	**96.57±0.11**

Table 4: Linear Evaluation of different contrastive losses under Semi-Supervised (PNU) setting - with 1%, 5% and 10% labeled training data. puNCE proves to be superior than infoNCE (Chen et al., 2020b) and semi-supervised SCL (Assran et al., 2020) especially in low supervision regime.

Dataset	$g_B(\cdot)$	Labeled	CE	CE + SCL	CE + puNCE
PNU SST2	RoBERTa	20	85.92±2.11	88.18±3.30	**91.15±1.75**
		100	91.10±1.36	92.83±1.23	**93.76±0.17**
		1000	94.03±0.64	94.11±0.53	**94.58±0.79**

Table 5: Few shot learning test results for N=10,100,1000. puNCE achieves significant gains over previous SCL based approach (Gunel et al., 2020) and standard CE baseline.

improve with increased supervision. *The results also suggest that puNCE can be particularly powerful when available supervision is small i.e. for smaller values of n_P and results in significant gains over both unsupervised and supervised contrastive approaches further emphasizing the importance of leveraging implicit supervision from unlabeled samples*.

5.2 Contrastive Positive Negative Unlabeled learning

We further evaluate puNCE in the binary semi-supervised setting. Training data contains samples from both the classes and a set of unlabeled samples. In particular, we perform experiments when only 1%, 5% and 10% of the data is available (Figure 4). It is important to note that, unlike PU Learning settings, here we perform downstream tuning only over the labeled data. We see similar trends as in PU Learning experiments - puNCE improves over infoNCE by 4.13% and over SCL by 1.39% when using only 1% data on PU CIFAR10. With only 10% data puNCE is able to achieve similar accuracy as a fully supervised baseline (with cross entropy) improving over semi-supervised adaptation of SCL (Assran et al., 2020) by 0.18%.

5.3 PNU few shot learning.

Additionally, we also apply puNCE for pretrained language model finetuning (FT). In the few-shot learning setting, the goal is to FT a pretrained model on a downstream task using only a few labeled samples. FT with cross-entropy(CE) loss with only a few examples is highly unstable and can give very different results across different runs (Dodge et al., 2020; Zhang and Sabuncu, 2018). To mitigate this, recent works (Gunel et al., 2020; Chen et al., 2022) use contrastive loss in conjunction with CE i.e. $\mathcal{L} = \lambda\mathcal{L}_{CE} + (1-\lambda)\mathcal{L}_{CL}$. We closely follow the same setup, however, in addition to N labeled samples (N=10, 100, 1000), we also assume access to 500 unlabeled samples during finetuning. Note that for unlabeled samples, the CE term has no contribution. In this setting, we adopt a pretrained RoBERTa model for downstream SST2 binary sentiment classification and observe that puNCE results in 2.97% (N=10), 0.93%(N=100) and 0.47% improvement over the previous state-of-the-art that uses SCL.

6 Conclusions

In this work, we investigated the limitation of a general self-supervised pretraining and finetuning approach on weakly-supervised tasks. We proposed a novel contrastive training objective puNCE (positive unlabeled Noise Contrastive Estimation), that extends contrastive loss to the positive unlabeled setting by incorporating available biased supervision. Our method achieved state-of-the-art performances on several PU learning and semi-supervised settings across vision and natural language processing and is particularly powerful when available supervision is limited.

References

Assran, M., Ballas, N., Castrejon, L., and Rabbat, M. (2020). Supervision accelerates pre-training in contrastive semi-supervised learning of visual representations. *arXiv preprint arXiv:2006.10803*.

Bekker, J. and Davis, J. (2020). Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19.

Blanchard, G., Lee, G., and Scott, C. (2010). Semi-supervised novelty detection. *The Journal of Machine Learning Research*, 11:2973–3009.

Caron, M., Touvron, H., Misra, I., J'egou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9630–9640.

Chen, H., Liu, F., Wang, Y., Zhao, L., and Wu, H. (2020a). A variational approach for learning from positive and unlabeled data. *Advances in Neural Information Processing Systems*, 33:14844–14854.

Chen, J.-L., Cai, J.-J., Jiang, Y., and Huang, S.-J. (2021). Pu active learning for recommender systems. *Neural Processing Letters*, 53(5):3639–3652.

Chen, Q., Zhang, R., Zheng, Y., and Mao, Y. (2022). Dual contrastive learning: Text classification via label-aware data augmentation. *arXiv preprint arXiv:2201.08702*.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020b). A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. E. (2020c). Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255.

Chen, X., Chen, W., Chen, T., Yuan, Y., Gong, C., Chen, K., and Wang, Z. (2020d). Self-pu: Self boosted and calibrated positive-unlabeled training. In *International Conference on Machine Learning*, pages 1510–1519. PMLR.

Christoffel, M., Niu, G., and Sugiyama, M. (2016). Class-prior estimation for learning from positive and unlabeled data. In *Asian Conference on Machine Learning*, pages 221–236. PMLR.

Chuang, C.-Y., Robinson, J., Yen-Chen, L., Torralba, A., and Jegelka, S. (2020). Debiased contrastive learning. *arXiv preprint arXiv:2007.00224*.

Dai, A. M. and Le, Q. V. (2015). Semi-supervised sequence learning. *Advances in neural information processing systems*, 28.

Denis, F. (1998). Pac learning from positive statistical queries. In *International Conference on Algorithmic Learning Theory*, pages 112–126. Springer.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dodge, J., Ilharco, G., Schwartz, R., Farhadi, A., Hajishirzi, H., and Smith, N. (2020). Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

Du, S. S., Hu, W., Kakade, S. M., Lee, J. D., and Lei, Q. (2020). Few-shot learning via learning the representation, provably. *arXiv preprint arXiv:2002.09434*.

Du Plessis, M. C., Niu, G., and Sugiyama, M. (2014). Analysis of learning from positive and unlabeled data. *Advances in neural information processing systems*, 27:703–711.

Elkan, C. (2001). The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd.

Elkan, C. and Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220.

Gao, T., Yao, X., and Chen, D. (2021). Simcse: Simple contrastive learning of sentence embeddings. *ArXiv*, abs/2104.08821.

Garg, S., Wu, Y., Smola, A. J., Balakrishnan, S., and Lipton, Z. (2021). Mixture proportion estimation and pu learning: A modern approach. *Advances in Neural Information Processing Systems*, 34.

Graf, F., Hofer, C., Niethammer, M., and Kwitt, R. (2021). Dissecting supervised constrastive learning. In *International Conference on Machine Learning*, pages 3821–3830. PMLR.

Grill, J.-B., Strub, F., Altch'e, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning. *ArXiv*, abs/2006.07733.

Gunel, B., Du, J., Conneau, A., and Stoyanov, V. (2020). Supervised contrastive learning for pre-trained language model fine-tuning. *arXiv preprint arXiv:2011.01403*.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hsieh, C.-J., Natarajan, N., and Dhillon, I. (2015). Pu learning for matrix completion. In *International conference on machine learning*, pages 2445–2453. PMLR.

Ivanov, D. (2020). Dedpul: Difference-of-estimated-densities-based positive-unlabeled learning. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 782–790. IEEE.

Kelly, D. and Teevan, J. (2003). Implicit feedback for inferring user preference: a bibliography. In *Acm Sigir Forum*, volume 37, pages 18–28. ACM New York, NY, USA.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. (2020). Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. *Advances in neural information processing systems*, 28.

Kiryo, R., Niu, G., Du Plessis, M. C., and Sugiyama, M. (2017). Positive-unlabeled learning with non-negative risk estimator. *Advances in neural information processing systems*, 30.

Kolesnikov, A., Zhai, X., and Beyer, L. (2019). Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1920–1929.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., and Liang, P. (2022). Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*.

Liu, B., Dai, Y., Li, X., Lee, W. S., and Yu, P. S. (2003). Building text classifiers using positive and unlabeled examples. In *Third IEEE International Conference on Data Mining*, pages 179–186. IEEE.

Liu, B., Lee, W. S., Yu, P. S., and Li, X. (2002). Partially supervised classification of text documents. In *ICML*, volume 2, pages 387–394. Sydney, NSW.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Luo, C., Zhao, P., Chen, C., Qiao, B., Du, C., Zhang, H., Wu, W., Cai, S., He, B., Rajmohan, S., et al. (2021). Pulns: Positive-unlabeled learning with effective negative sample selector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8784–8792.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Naumov, M., Mudigere, D., Shi, H.-J. M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C.-J., Azzolini, A. G., et al. (2019). Deep learning recommendation model for personalization and recommendation systems. *arXiv preprint arXiv:1906.00091*.

Niu, G., du Plessis, M. C., Sakai, T., Ma, Y., and Sugiyama, M. (2016). Theoretical comparisons of positive-unlabeled learning against positive-negative learning. *Advances in neural information processing systems*, 29:1199–1207.

Oord, A. v. d., Li, Y., and Vinyals, O. (2018). Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision. In *ICML*.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Ramaswamy, H., Scott, C., and Tewari, A. (2016). Mixture proportion estimation via kernel embeddings of distributions. In *International conference on machine learning*, pages 2052–2060. PMLR.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Sohn, K. (2016). Improved deep metric learning with multi-class n-pair loss objective. In *Advances in neural information processing systems*, pages 1857–1865.

Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*.

Tian, Y., Krishnan, D., and Isola, P. (2020). Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 776–794. Springer.

Wu, Z., Efros, A. A., and Yu, S. X. (2018). Improving generalization via scalable neighborhood component analysis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 685–701.

You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., and Hsieh, C.-J. (2019). Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*.

Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. (2021). Barlow twins: Self-supervised learning via redundancy reduction. In *ICML*.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., et al. (2019). A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*.

Zhang, Z. and Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, 31.

Zhong, Z., Fini, E., Roy, S., Luo, Z., Ricci, E., and Sebe, N. (2021). Neighborhood contrastive learning for novel class discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10867–10875.

A Appendix

A.1 Additional Experimental Details

A.1.1 Computing exact π

For our experiments we use the optimal π i.e. probability of positive sample in unlabeled data exactly as follows: suppose n_P positive samples are labeled and let P^* and N^* denote true number of positive and negatives. then we can readily use $\pi^* = \frac{P^* - n_P}{P^* + N^* - n_P}$.

A.1.2 Mixture Proportion Estimation

In real settings when we do not have knowledge about the dataset exact π cannot be computed and needs to be estimated - referred to as the Mixture proportion estimation (MPE) problem. Formally speaking, Mixture proportion estimation (MPE) refers to the task of estimating the weight of a component distribution in a mixture, given samples from the mixture (unlabeled data) and component (positive labeled data). We refer the reader to (Ramaswamy et al., 2016; Garg et al., 2021; Ivanov, 2020) for a detailed discussion on MPE algorithms.

A.1.3 MNIST Multi Layer Perceptron

```
MLPContrastive(
  (backbone): Sequential(
    (0): Linear(in_features=784, out_features=5000, bias=True)
    (1): BatchNorm1d(5000, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=5000, out_features=5000, bias=True)
    (4): BatchNorm1d(5000, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=5000, out_features=50, bias=True)
    (7): BatchNorm1d(50, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
    (8): ReLU(inplace=True)
  )
  (projector): Sequential(
    (0): Linear(in_features=50, out_features=300, bias=True)
    (1): BatchNorm1d(300, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=300, out_features=50, bias=True)
  )
  (online_head): Linear(in_features=50, out_features=1, bias=True)
)
------------------
Num of Params = 29231151
```

A.1.4 All Convolution N/w

```
AllConv(
  (backbone_cnn): Sequential(
    (0): Conv2d(3, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1,
    1))
    (1): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(96, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1,
     1))
    (4): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True,
    track_running_stats=True)
    (5): ReLU(inplace=True)
```

```
    (6): Conv2d(96, 96, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (7): BatchNorm2d(96, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): Conv2d(96, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (14): ReLU(inplace=True)
    (15): Conv2d(192, 192, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (16): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (17): ReLU(inplace=True)
    (18): Conv2d(192, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (19): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (20): ReLU(inplace=True)
    (21): Conv2d(192, 192, kernel_size=(1, 1), stride=(1, 1))
    (22): BatchNorm2d(192, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (23): ReLU(inplace=True)
    (24): Conv2d(192, 10, kernel_size=(1, 1), stride=(1, 1))
    (25): BatchNorm2d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (26): ReLU(inplace=True)
  )
  (backbone_lin): Sequential(
    (0): Linear(in_features=640, out_features=1000, bias=True)
    (1): ReLU(inplace=True)
    (2): Linear(in_features=1000, out_features=1000, bias=True)
    (3): ReLU(inplace=True)
  )
  (projector): Sequential(
    (0): Linear(in_features=1000, out_features=1000, bias=True)
    (1): BatchNorm1d(1000, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=1000, out_features=1000, bias=True)
  )
  (online_head): Linear(in_features=1000, out_features=1, bias=True)
)
------------------
Num of Params = 5019255
```

A.1.5 ResNet18

```
ResNet18Contrastive(
  (backbone): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Sequential(
      (0): BasicBlock(
```

```
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (4): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (5): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
```

```
          (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (6): Sequential(
      (0): BasicBlock(
        (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (downsample): Sequential(
          (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
          (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        )
      )
      (1): BasicBlock(
        (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
        (relu): ReLU(inplace=True)
        (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (7): AdaptiveAvgPool2d(output_size=(1, 1))
  )
  (projector): Sequential(
    (0): Linear(in_features=512, out_features=256, bias=False)
    (1): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Linear(in_features=256, out_features=128, bias=True)
  )
  (online_head): Linear(in_features=512, out_features=1, bias=True)
)
------------------
Num of Params = 11333825
```