SET-SIMCLR: SELF-SUPERVISED SET REPRESENTATION LEARNING FOR UNSUPERVISED META-LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised meta-learning (UML) combines the concepts of unsupervised learning and meta-learning, which aims at learning transferrable models for unseen downstream tasks, without any human supervisions. This conceptually resembles the main application of self-supervised learning, however, the meta-learning problem is often interpreted as a *set*-level problem. Motivated by this, we introduce self-supervised set-representation learning framework, Set-SimCLR for solving UML problem. Specifically, we learn set-representation by maximizing the agreement of two sets, where the positive sets are constructed with repeated applications of stochastic augmentations on a same example. At meta-test time, the weights of linear classifier is initialized with the learnt set-representations. We motivate our algorithmic design of Set-SimCLR with theoretical analysis on learning set-representation, and also show its empirical effectiveness in compared to the both of UML and instance-level self-supervised learning baselines.

1 INTRODUCTION

One of the most challenging and long-standing problems in machine learning is learning meaningful representation without human-supervision. Self-supervised learning (Chen et al., 2020a;b; Grill et al., 2020; Zbontar et al., 2021) is one of the most popular framework for learning visual representations from unlabeled data in a *task-agnostic* way. These methods rely on *proxy* objective whose goal is to maximize agreement between differently augmented views of the same data example in the latent space. The augmented views are easily obtained by sequentially applying pre-defined stochastic augmentations such as random crops, flips, color distortion, and Gaussian blur.

Meanwhile, meta-learning is an established popular framework for learning models that quickly adapt to on-the-fly tasks given a small number of examples (Finn et al., 2017; Snell et al., 2017). It shares the spirit of unsupervised learning in that both of them seek more efficient and effective learning procedure over learning from scratch. However, the essential difference between the two is that most meta-learning approaches have been built on the supervised learning scheme and require human-crafted task distributions to be applied in few-shot classification. In order to tackle the limitation, several works (Hsu et al., 2019; Khodadadeh et al., 2019; 2021; Lee et al., 2021) have proposed *unsupervised meta-learning* that combines unsupervised learning and meta-learning. They train a model with unlabeled data such that the model can adapt to unseen tasks with few labeled examples at meta-test time. This scenario conceptually resembles the main application of self-supervised learning where we transfer the learnt representation into various downstream tasks; this motivates us to develop a self-supervised learning method for few-shot classification.

Most of existing self-supervised learning have focused on the instance-level visual features. The importance of learning meaningful instance-level feature is clear for generalization on unseen tasks, however, meta-learning problem is often interpreted as a *set*-level problem. For example, Snell et al. (2017) takes an average of features belonging to the same class as a prototype (or a set-representation) for representing a set of the instances with the same class. Similarly, Gordon et al. (2019); Iakovleva et al. (2020) propose Bayesian framework to learn stochastic prototypes using multi-layer perceptron. Further, Triantafillou et al. (2019) propose to finetune the prototype with supervised loss. Inspired by the successes of set-representation in few-shot learning, we propose self-supervised set-representation learning framework for unsupervised meta-learning.

The underlying assumption of self-supervised learning methods is that two different views of an image shares the most of visual semantics. Built upon this idea, we construct two sets where each

set consists of different views of the same image and maximize the agreement between them. We dub this framework as Set-SimCLR. Concretely, we repeatedly apply stochastic augmentations to each image of mini-batch for multiple times and construct a set consisting of the augmented images. Then we divide the set by half into two sets which are considered to be a positive pair of sets. Given a positive pair, similar to Chen et al. (2020a), the other sets within the mini-batch are considered as negative examples. We encode the sets using attention-based set encoder (Vaswani et al., 2017; Lee et al., 2019) to obtain set-representations. The set-encoder is then trained to reduce the distance of positive sets and increase that of negative sets. At meta-test time, given few labeled data, each row of the weight for linear classifier is initialized with the learnt representation of a set composing of instances belonging to the same class.

We motivate our algorithmic design of Set-SimCLR based on theoretical analysis. Specifically, we study how the set-representation can potentially improve final performance and the reason why one use set-representation as an initializer of classifier weights. We then empirically validate the effectiveness of our Set-SimCLR by using four unsupervised meta learning methods and three instance-level self-supervised learning methods. We find that our method outperforms the baselines on six benchmark datasets, including Mini-ImageNet (Ravi & Larochelle, 2017), Tiny-ImageNet (Le & Yang, 2015), CIFAR100 (Krizhevsky et al., 2009), Aircraft (Maji et al., 2013), Stanford Cars (Krause et al., 2013), and CUB (Wah et al., 2011) datasets.

We summarize our contributions as follows:

- We introduce Set-SimCLR framework for solving unsupervised meta-learing problem, which learns instance- and set-level representations simultaneously for downstream tasks.
- We provide a theoretical motivation of our model Set-SimCLR and study how the set-representation potentially improve few-shot classification performance.
- The proposed Set-SimCLR outperforms the previous UML baselines and self-supervised learning baselines by significant margins in all datasets we consider.

2 Related Work

Unsupervised Meta-Learning There are two lines of works for meta-learning — gradient based methods such as MAML (Finn et al., 2017) and metric based ones namely ProtoTypical Network (Snell et al., 2017). However, critical limitation of such conventional meta-learning is that it requires a predefined distribution over tasks for meta-training. In order to tackle the limitation, several unsupervised meta-learning methods are proposed. CACTUs (Hsu et al., 2019) run off the shelf unsupervised embedding algorithm and episodically run clustering on the embedding space to construct a task, and then run the supervised meta-learning algorithm on the constructed task. UMTRA (Khodadadeh et al., 2019) assumes that each example within a mini-batch belong to the different class and leverage data augmentation to construct a task. LASIUM (Khodadadeh et al., 2021) trains generative models to harvest synthetic examples and construct a task by assuming that nearby latent codes produces in-class examples. Different from the work have focused on generating pseudo tasks, Meta-GMVAE (Lee et al., 2021) introduces a Mixture of Gaussian priors by performing Expectation-Maximization during the meta-training and the meta-test. In this work, we focus on self-supervised learning as a base framework for solving unsupervised meta-learning.

Set-Representation Zaheer et al. (2017) propose DeepSets which is a permutation invariant or equivariant neural architecture for set encoding. It independently processes each element and aggregate all the elements with min, max, mean or sum operation. In order to tackle the lack of expressiveness of Deepsets, Set Transformer (Lee et al., 2019) leverages self-attention to model pairwise interaction of elements in a set. Instead of aiming to design more expressive neural architecture for set encoding, several works are proposed to learn set representation by minimizing a distance between an input set and trainable reference set. RepSet (Skianis et al., 2020) leverages a bipartite matching to minimize some cost between them. OTKE (Mialon et al., 2020) first introduce ideas from optimal transport (OT) by aligning features of a set to a trainable reference distribution. WEGL (Kolouri et al., 2020) propose Wasserstein embedding as a pooling operator for set-representation. Similarly, POT dan Guo et al. (2022) minimize optimal transport distance between empirical distribution of set elements and distribution over global prototypes to improve set encoding. Note that the above work is not our competitor, and can be combined our self-supervised set representation learning framework.

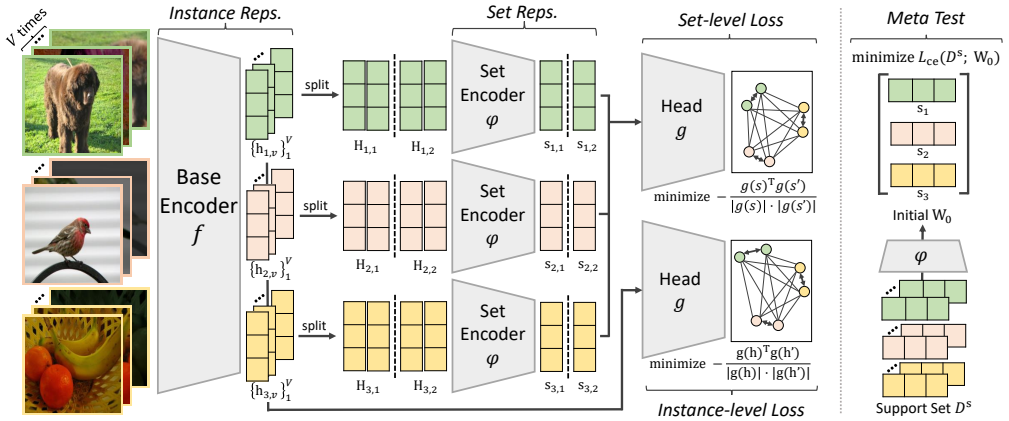

Figure 1: **(Left)**: A conceptual illustration of Set-SimCLR with three images. We first encode each augmented image into instance representation using the base encoder f. Then we partition the set of V augmented images into two sets and obtain set-representations with the set encoder φ. We finally compute set- and instance-level loss. **(Right)**: At meta-test, we use the set-representation of each class as an initializer of linear classifier weight.

Self-supervised Learning Recently, a large volume of works have proposed self-supervised learning methods where the core idea is the representation of differently augmented views of the same image should be similar. Note that here we introduce just a few of them which we consider as baselines in our experiments. SimCLR (Chen et al., 2020a;b) is one of the most representative contrastive framework where two views of the same images are pulled together while the negative pairs are repulsed. MOCO (He et al., 2020; Chen et al., 2020c; 2021) builds a dynamic feature dictionary using a queue and momentum encoder and learns from contrastive loss from the dictionary. Meanwhile, several works show that non-contrastive approaches can learn meaningful representation without a latent feature collapse. For example, Grill et al. (2020) leverages two identical networks where one of them is momentum encoder to encode different view of images and minimize a distance between positive pairs. Zbontar et al. (2021) computes cross-correlation matrix between different view of images and optimize it to be close to identity matrix. In this work, we exploit the effectiveness of self-supervised learning on unsupervised meta-learning, especially combined with our proposed set-representation learning.

3 METHOD

In this section, we describe the problem setting of unsupervised meta-learning and our approach, Set-SimCLR which is a self-supervised set-representation learning framework. The conceptual illustration of our framework is depicted in Figure 1.

3.1 PROBLEM STATEMENT

For unsupervised meta-learning, we can only access to an unlabeled dataset $\mathcal{D}^u = \{\mathbf{x}_i\}_{i=1}^U$ for meta-training. As in the most of meta-learning literature, we assume meta-test data also follows the same data distribution of unlabeled dataset $\mathcal{D}^u$, while having a different set of classes. At meta-test time, we are given a set of N-way S-shot classification tasks and each task consists of a support set $\mathcal{D}^s = \{(\mathbf{x}_i^s, y_i^s)\}_{i=1}^{N\times S}$ and a query set $\mathcal{D}^q = \{\mathbf{x}_i^q\}_{i=1}^{N\times Q}$. The final goal is to leverage the model trained on the unlabeled data to predict the labels of query set with the help of the support set.

3.2 SELF-SUPERVISED CONTRASTIVE REPRESENTATION LEARNING

Before introducing our method, we first describe one of the most successful self-supervised learning methods SimCLR. SimCLR is a contrastive learning framework which maximizes agreement between differently augmented views of the same data example in the latent space. Specifically, it first randomly sample a mini-batch of M images $\{\mathbf{x}_m\}_{m=1}^M$ and obtains two different views of each image using stochastic data augmentation, resulting in $2 \times M$ data points $\{(\mathbf{x}_{m,1}, \mathbf{x}_{m,2})\}_{m=1}^M$. There are two components: 1) base encoder $f(\cdot)$ (e.g. convolutional neural networks) that extracts feature representations, and 2) a relatively small neural network $g(\cdot)$, dubbed as projection head, maps the representation to the latent space where the contrastive loss is applied. By sequentially applying the base encoder $f(\cdot)$, and head $g(\cdot)$, the latent representation of each image is obtained as

$\mathbf{z}_{m,j} = g(f(\mathbf{x}_{m,j}))$ for $j = 1, 2$. Then the constrastive loss function for the mini-batch of M images is defined as

$$\mathcal{L}^{\text{SimCLR}}\left(\{(\mathbf{z}_{m,1}, \mathbf{z}_{m,2})\}_{m=1}^{M}\right) = \frac{-1}{2M}\sum_{m=1}^{M} \log \frac{\texttt{exp}(\texttt{sim}(\mathbf{z}_{m,1}, \mathbf{z}_{m,2})/\tau)}{\sum_{j=1}^{2}\sum_{k=1}^{M} \mathbb{1}_{[\mathbf{z}_{k,j} \neq \mathbf{z}_{m,1}]}\texttt{exp}(\texttt{sim}(\mathbf{z}_{m,1}, \mathbf{z}_{k,j})/\tau)} + \log \frac{\texttt{exp}(\texttt{sim}(\mathbf{z}_{m,2}, \mathbf{z}_{m,1})/\tau)}{\sum_{j=1}^{2}\sum_{k=1}^{M} \mathbb{1}_{[\mathbf{z}_{k,j} \neq \mathbf{z}_{m,2}]}\texttt{exp}(\texttt{sim}(\mathbf{z}_{m,2}, \mathbf{z}_{k,j})/\tau)} \tag{1}$$

where $\texttt{exp}$ is an exponential function, $\texttt{sim}$ is a measure of similarity (e.g., cosine similarity), and $\mathbb{1}_{[\mathbf{z}_{k,j} \neq \mathbf{z}_{m,1}]} \in \{0, 1\}$ is an indicator function evaluating to 1 if $\mathbf{z}_{k,j} \neq \mathbf{z}_{m,1}$ otherwise 0. The temperature τ is a hyperparameter controling the sharpness of distribution.

3.3 Self-Supervised Set Representation Learning with SimCLR

Existing self-supervised learning have focused on the instance-level visual features. The importance of the instance-level features is clear for generalization on unseen tasks, however, meta-learning problem is often interpreted as a *set*-level problem. For example, Snell et al. (2017) takes an average of features belonging to the same class as a prototype (or a set-representation), or Gordon et al. (2019); Iakovleva et al. (2020) propose Bayesian framework to learn stochastic prototypes using multi-layer perceptron. Further, Triantafillou et al. (2019) propose to finetune prototype with supervised loss. Inspired by the successes, we propose a novel self-supervised contrastive learning framework for learning set-representation.

Set Representation The underlying assumption of self-supervised learning methods is that two different views of an image share the most of visual semantics. We extend this idea to set level representation by constructing two sets where each set consists of different views of the same image and maximizing agreement between them. Specifically, we repeatedly apply stochastic augmentations to an image for $V \in \mathbb{N}$ times and construct a set of $\{\mathbf{x}_{m,v}\}_{v=1}^{V}$ for each image of a mini-batch, where V is even number. Then we independently encode each augmented image with the base encoder f to obtain instance-level feature representations $\mathbf{h}_{m,v} = f(\mathbf{x}_{m,v}) \in \mathbb{R}^d$ for $m = 1, \ldots, M$ and $v = 1, \ldots, V$, where d is the dimension of the feature representation. This results in M different sets of instance-level representations $\mathbf{H}_m = \{\mathbf{h}_{m,v}\}_{v=1}^{V}$ for $m = 1, \ldots, M$. Then we divide each set $\mathbf{H}_m$ by half to obtain positive pairs of sets, i.e., $\mathbf{H}_{m,1} = \{\mathbf{h}_{m,v}\}_{v=1}^{V/2}$ and $\mathbf{H}_{m,2} = \{\mathbf{h}_{m,v}\}_{v=V/2+1}^{V}$. and get a set representation by applying a set encoder to each set. Here any set encoders taking as an input of a set of vectors and outputs a vector can be deployed, but we empirically find the following implementation of set encoder works well:

$$\begin{aligned} \mathbf{H}_{m,j} &\in \mathbb{R}^{V/2 \times d} \\ \mathbf{T}_{m,j} &= \texttt{TransformerEncoder}(\mathbf{H}_{m,j}) \in \mathbb{R}^{V/2 \times d} \\ \mathbf{s}_{m,j} &= \texttt{MLP}(\texttt{concat}\,(\texttt{mean}(\mathbf{T}_{m,j}); \texttt{std}(\mathbf{T}_{m,j}); \texttt{max}(\mathbf{T}_{m,j}); \texttt{min}(\mathbf{T}_{m,j}))) \in \mathbb{R}^d \end{aligned} \tag{2}$$

for $j = 1, 2$, where we define a mapping $\varphi : \mathbf{H}_{m,j} \in \mathbb{R}^{V/2 \times d} \mapsto \mathbf{s}_{m,j} \in \mathbb{R}^d$. For $\texttt{TransformerEncoder}$, we use the multi-head self-attention mechanism proposed by Vaswani et al. (2017); Lee et al. (2019). Please see Appendix A for more detail. We take the row-wise operations on the outputs $\mathbf{T}_{m,j} \in \mathbb{R}^{V/2 \times d}$ of $\texttt{TransformerEncoder}$ to compute mean, standard deviation, maximum, and minimum (denoted as $\texttt{mean}$, $\texttt{std}$, $\texttt{max}$, and $\texttt{min}$), where each results in d-dimensional vector. Then we concatenate them, denoted as $\texttt{concat}$, which is $4d$-dimensional vector, and feed it into multi-layer perceptron $\texttt{MLP}$ to obtain the final set representation $\mathbf{s}_{m,j} \in \mathbb{R}^d$.

Contrastive Loss for Set Representation Learning We now obtain set representations $\mathbf{s}_{m,1}, \mathbf{s}_{m,2}$, by applying the set encoder in Equation. 2 to each set $\mathbf{H}_{m,1}$ and $\mathbf{H}_{m,2}$. Following self-supervised literature, we project this set representation into the latent space with the same head $g(\cdot)$ used for instance-level feature learning. Finally, we compute set level contrastive loss by plugging the projected set representations into Equation. 1, i.e., $\mathcal{L}^{\text{SimCLR}}(\{(g(\mathbf{s}_{m,1}), g(\mathbf{s}_{m,2})\}_{m=1}^{M})$. The difference between our loss and SimCLR loss is that instead of instance-level representation, we pull positive pair of set-level representation and repulse the negative pairs . Further, we find that sharing the projection head encourages the set-level and instance-level features to be consistent to each other,

which empirically shows better results. Then the final loss is combination of two the set-level and instance-level SimCLR losses as follows:

$$\mathcal{L}^{\text{SimCLR}}\left(\{(\mathbf{z}_{m,1},\mathbf{z}_{m,2})\}_{m=1}^{M}\right)+\lambda\mathcal{L}^{\text{SimCLR}}\left(\{(g(\mathbf{s}_{m,1}),g(\mathbf{s}_{m,2}))\}_{m=1}^{M}\right) \tag{3}$$

where λ is a hyperparameter that balances two different loss signals. We do not tune the λ and set it to 1 for all the experiments.

Linear Evaluation for Downstream Tasks We now describe how we utilize the learnt instance-level and set-level representations on downstream few-shot classification tasks. For a novel N-way S-shot task at meta-test time, we are given support set $\mathcal{D}^s=\{(\mathbf{x}_i^s,y_i^s)\}_{i=1}^{N\times S}$ and supposed to predict the labels of query set $\mathcal{D}^q=\{\mathbf{x}_i^q\}_{i=1}^{N\times Q}$. We first encode the support and query set using the encoder $f(\cdot)$ and obtain instance feature representations of support set $\{(\mathbf{h}_i^s,y_i^s)\}_{i=1}^{N\times S}$ and query set $\{\mathbf{h}_i^q\}_{i=1}^{N\times Q}$. For each class $c=1,\dots,N$, we encode $\mathbf{H}_c^s=\{\mathbf{h}_i^s\mid y_i^s=c\}$, a set of instances belonging to the class c, with the mapping $\varphi(\mathbf{H}_c^s)$ as described in Equation. 2. Let the obtained set representation of c-th class be $\mathbf{s}_c=\varphi(\mathbf{H}_c^s)\in\mathbb{R}^d$.

We then initialize a weight of a linear classifier with the set representations $\mathbf{s}_1,\dots,\mathbf{s}_N$ and train the classifier with support set while freezing the base encoder f, which is similar to the linear evaluation of self-supervised learning (Chen et al., 2020a). We find that this is much suitable for our few-shot setting than the strategy of finetuning the full model due to the risk of overfitting to few data. Specifically, we initializing the weight W_0 of the classifier by stacking the learnt set-representation s_c as row vectors and optimize it by minimizing cross-entropy loss as follows:

$$W_0^*=\arg\min_{W_0}\frac{1}{N\cdot S}\sum_{\mathbf{x}_i^s\in\mathcal{D}^s}\mathcal{L}_{\text{CE}}(\texttt{softmax}(W_0f(\mathbf{x}_i^s)),y_i^s),\quad W_0\leftarrow\begin{bmatrix}- & \mathbf{s}_1 & -\\ & \vdots & \\ - & \mathbf{s}_N & -\end{bmatrix}\in\mathbb{R}^{N\times d} \tag{4}$$

where $\mathcal{L}_{\text{CE}}$ denotes a cross-entropy loss. After the optimization, we predict a label for each data point in query set as follows:

$$y_i^q=\arg\max_c p_c^{(i)},\quad \text{where }\left(p_1^{(i)},\dots,p_N^{(i)}\right)^\top=W_0^*f(\mathbf{x}_i^q).$$

3.4 Theoretical Motivation

In this section, we provide theoretical motivations on several aspects of the algorithmic design. Namely, we study how the set-representation can potentially improve the final performance and why it makes sense to initialize W_0 with the set-representation and optimize it further to obtain W_0^* with the cross-entropy loss. In appendix B, we show that the proposed method is equivalent to the metric-based inference with the fine-tuning of the class prototypes s_c, where the initial class prototypes s_c are obtained with the set-representation by $s_c=\varphi(\mathbf{H}_c^s)$ and each input x is represented by instance-level representation $f(x)$. Thus, in the following, we discuss how such metric-based inference behaves with respect to the supervised loss in the downstream task.

To obtain theoretical insights, this section focuses on the binary classification without the head g and considers the following abstract data-generating process: each of the unknown labels y^+ and y^- is drawn independently from a uniform distribution U on $\{1,2\}$, and then each of the unlabeled positive examples x^+ and x^{++} is drawn from the conditional distribution $\mathcal{D}_{y^+}$ conditioned on the label y^+ while the negative example x^- is drawn from the conditional distribution $\mathcal{D}_{y^-}$. Accordingly, this hidden process forms the joint distribution $\mathcal{D}(x,y)=\mathcal{D}_y(x)U(y)$. In this setting, we can write the contrastive unsupervised loss L^{SimCLR} of the representation f and the corresponding supervised loss L_{s} of our classifier W_tf by $L^{\text{SimCLR}}(f)=\mathbb{E}_{y^+\sim U,y^-\sim U}\mathbb{E}_{x^+,x^{++}\sim\mathcal{D}_{y^+}^2,x^-\sim\mathcal{D}_{y^-}}[-\log(\frac{\exp(f(x^{++})^\top f(x^+))}{\exp(f(x^{++})^\top f(x^+))+\exp(f(x^{++})^\top f(x^-))})]$ and $L_{\text{s}}(f)=\mathbb{E}_{(x,y)\sim\mathcal{D}}[-\log(\frac{\exp((W_tf(x))_y)}{\sum_{k=1}^2\exp((W_tf(x))_k)})]$, where the matrix $W_t\in\mathbb{R}^{2\times d}$ is defined by $\lim_{t\to\infty}W_t=W_0^*$ and $W_t=[\varphi[\mathbf{H}_1^s]+(\Delta_t)_1,\ \varphi[\mathbf{H}_2^s]+(\Delta_t)_2]^\top$. Here, $\Delta_t=[(\Delta_t)_1,(\Delta_t)_2]^\top=W_t-W_0\in\mathbb{R}^{2\times d}$ is the elements added during the training with the support set. Importantly, y^+ and y^- can be the same as $y^+=y^-$ as we do not know the true labels in unsupervised loss. This is reflected by the fact that they are sampled from the same (unknown) probability measure on labels U. We also define

the corresponding supervised loss with the average pooling (instead of our set-representation) by $L_{\mathrm{s}}^{A}(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[-\log(\frac{\exp((Af(x))_y)}{\sum_{k=1}^{2}\exp((Af(x))_k)})]$, where $A = [\mathbb{E}_{x\sim\mathcal{D}_1}[f(x)],\ \mathbb{E}_{x\sim\mathcal{D}_2}[f(x)]]^\top$. We define the performance difference between the average pooling and our set-representation in terms of supervised loss by $\gamma = L_{\mathrm{s}}^{A}(f) - L_{\mathrm{s}}(f)$. Define the probability of y^+ and y^- being the same by $\mathbb{P}(y^+ = y^-) = \mathbb{E}_{y^+,y^-\sim U^2}[\mathbb{1}\{y^+ = y^-\}]$. Similarly, $\mathbb{P}(y^+ \neq y^-) = \mathbb{E}_{y^+,y^-\sim U^2}[\mathbb{1}\{y^+ \neq y^-\}]$. We define $c = \mathbb{P}(y^+ \neq y^-)^{-1}$ and $\zeta = \frac{\mathbb{P}(y^+=y^-)\log(2)}{\mathbb{P}(y^+\neq y^-)}$.

The following theorem provides a theoretical motivation on the use of the set-representation to increase γ at $t = 0$ and on the fine-tuning of W_0 to further increase γ at $t > 0$, in order to minimize the supervised loss L_{s} in the downstream task:

Theorem 1. *For any $f \in \mathcal{F}$, $L_{\mathrm{s}}(f) \leq cL^{\mathrm{SimCLR}}(f) - \zeta - \gamma$.*

Proof. The proof is deferred to appendix B. □

There are two ways to interpret the theorem. The first way is obtained by setting the probability measures for the expectations to be the empirical measures. Then, Theorem 1 is all about the empirical losses; i.e., $L^{\mathrm{SimCLR}}(f)$ is the empirical contrastive loss, L_{s} is the empirical supervised loss, and $\mathbb{P}(y^+ = y^-)$ is the rate of the positive and negative labels colliding in the empirical contrastive loss. In this interpretation, we can see that the γ is expected to be positive even at $t = 0$ since φ can recover the average pooling to obtain $L_{\mathrm{s}}^{A}(f) - L_{\mathrm{s}}(f) = 0$ and φ is optimized to be better than the average pooling. As we increase t, we can further increase γ to a certain degree until the correlation between the gradients on the support set and the unlabeled dataset is positive. Once the correlation becomes negative, γ starts to decrease, resulting into the overfitting to the support set by increasing the (unknown) supervised loss at the unlabeled data. Thus, it makes sense to train W_t by setting $\Delta_t \neq 0$ but we should regularize W_t so that γ does not decrease during the fine-tuning of W_t with the support set. The second interpretation is obtained by setting the probability measures to be the probability measures, with which we have qualitatively the same conclusion as discussed in appendix B. Therefore, in both interpretations, Theorem 1 provides a theoretical motivation on the use of the set-representation (to increase γ at $t = 0$) and on the fine-tuning of W_t (and W_0^*) with weight-decay (to further increase γ at $t > 0$).

4 Experiment

In this section, we empirically validate the effectiveness of our set representation learning framework on several downstream few-shot classification tasks.

4.1 Comparison to Unsupervised Meta-Learning

We first compare our method against existing unsupervised meta-learning methods.

Dataset We use the Mini-ImageNet dataset introduced by Ravi & Larochelle (2017), which is a subset of ILSVRC-2012 (Deng et al., 2009) consisting of 100 classes and each class contains 600 different images. We use 64 classes for self-supervised learning, 16 classes for meta-validation, and the remaining 20 classes for meta-test. Following the standard protocol of unsupervised meta-learning, we evaluate our method on 1000 randomly sampled tasks from the meta-test set.

Baselines We compare Set-SimCLR with two supervised meta-learning approaches which we consider as "*oracles*", and four different unsupervised meta-learning baselines as follows; **1) MAML (oracle) (Finn et al., 2017)**, **2) ProtoNets (oracle) (Snell et al., 2017)**, **3) CACTUs (Hsu et al., 2019)**, **4) UMTRA (Khodadadeh et al., 2019)**, **5) LASIUM (Khodadadeh et al., 2021)**, and **6) Meta-GMVAE (Lee et al., 2021)**. We provide the detailed explanation of baselines in Appendix C.

Implementation Details For the base encoder f, we stack five blocks of convolution (Conv5), where each block consists of convolutional operation with 64 channels followed by batch normalization (Ioffe & Szegedy, 2015), ReLU activation (Agarap, 2018), and max pooling. We set the dimension of output of the base encoder d to 256. The head network $g(\cdot)$ is two layered fully connected networks composing of affine transformation, batch normalization, Leaky ReLU (Xu et al., 2015) and another affine transformation. The last affine transformation reduces the dimension of input d by a quarter. We follow SimCLR (Chen et al., 2020a;b) for random augmentation, which is detailed in Appendix D. We apply the composed augmentations to 64 mini-batch images eight times (i.e., $M = 64, V = 8$), resulting in 4 elements in each set. We optimize the base encoder, set encoder, and head network for 400 epochs using Adam optimizer (Kingma & Ba, 2014) with constant learning

Table 1: Results for 5-way S-shot classification on Mini-ImageNet. We report mean and standard deviation of accuracy evaluated on 1000 episodes with 5 different runs. Note that we take the accuracy of the other baselines from the previous works (Hsu et al., 2019; Khodadadeh et al., 2019; 2021; Lee et al., 2021).

Method	**Clustering**	1-shot	5-shot	20-shot	50-shot
Training from Scratch	N/A	27.59	38.48	51.53	59.63
CACTUs-MAML	BiGAN	36.24	51.28	61.33	66.91
CACTUs-ProtoNets	BiGAN	36.62	50.16	59.56	63.27
CACTUs-MAML	ACAI/DC	39.90	53.97	63.84	69.64
CACTUs-ProtoNets	ACAI/DC	39.18	53.36	61.54	63.55
UMTRA	N/A	39.93	50.73	61.11	67.15
LASIUM-MAML-RO/N	N/A	40.19	54.56	65.17	69.13
LASIUM-ProtoNets-RO/N	N/A	40.05	52.53	59.45	61.43
Meta-GMVAE	N/A	42.82	55.73	63.14	68.26
Set-SimCLR (ours)	N/A	**43.55** $\pm.17$	**58.74** $\pm.21$	**69.09** $\pm.26$	**72.43** $\pm.16$
MAML (oracle)	N/A	46.81	62.13	71.03	75.54
ProtoNets (oracle)	N/A	46.56	62.29	70.05	72.04

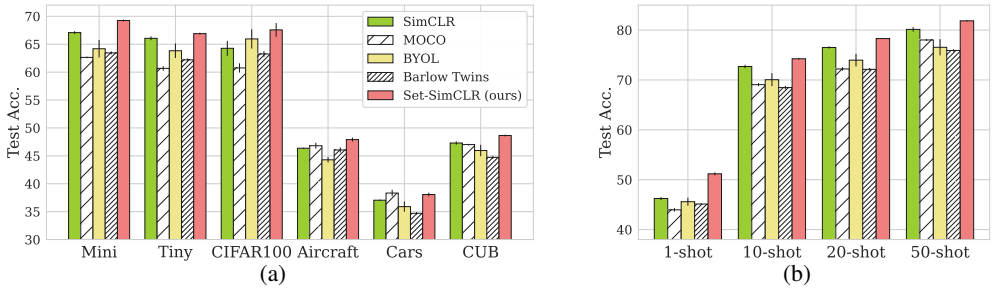

Figure 2: **(a)**: The 5-way 5-shot classification results on Aircraft, Stanford Cars, CIFAR100, CUB, Mini-ImageNet, and Tiny-ImageNet datasets. **(b)**: The 5-way 1, 10, 20, 25-shot classification results on Mini-ImageNet dataset. We report the mean and standard deviation with 5 different random seeds.

rate 0.001. For downstream tasks, We use scikit-learn (Pedregosa et al., 2011) package with default settings to optimize a linear classifier.

Results Table 1 shows the performance of the baselines and our Set-SimCLR for 5-way 1, 5, 20, and 50-shot classification on the Mini-ImageNet dataset. We can see that Set-SimCLR outperforms all the previous methods by considerable margins. For an instance, it achieves $+3.01\%, +3.92\%, +2.79\%$ performance improvement over the best performing baseline on 5-, 20-, 50-shot settings. However, the performance gain of $+0.73\%$ on 1-shot setting is comparatively smaller than others. This is because the set encoder is always trained on the set with cardinality of four and never learns representation of a set with single element during training. We further believe that we can improve the generalization performance on 1-shot setting by training the set encoder with singleton sets. For example, we divide 8 different view of images into a singleton set and a set with 7 elements. However, we do not target any specific settings since the fundamental assumption of unsupervised meta learning is that any piece of information about the tasks from meta-test is unknown at training time.

4.2 Comparison to Self-Supervised Learning

We now compare our method against instance-level self-supervised learning methods.

Dataset We also use the Mini-ImageNet dataset for training and evaluating models. Further, to verify the effectiveness of the proposed method on transfer learning scenario, we evaluate the trained models on the conventional meta-test split of Tiny-ImageNet (Le & Yang, 2015), CIFAR100 (Krizhevsky et al., 2009), Aircraft (Maji et al., 2013), Stanford Cars (Krause et al., 2013), and CUB (Wah et al., 2011) datasets. Here, Mini-, Tiny-ImageNet, and CIFAR100 are with general categories, meanwhile Aircraft, Stanford Cars, and CUB target a fine-grained classification. Since all the models are trained on 84×84 images from the source dataset Mini-ImageNet, we resize the image to 84×84 resolution for all datasets. Following UML literature, we evaluate our method on 1000 randomly sampled tasks from the meta-test set.

Baselines Although there are a vast number of self-supervised learning methods, in this work, we want to show the effectiveness of set-representation learning compared to learning instance

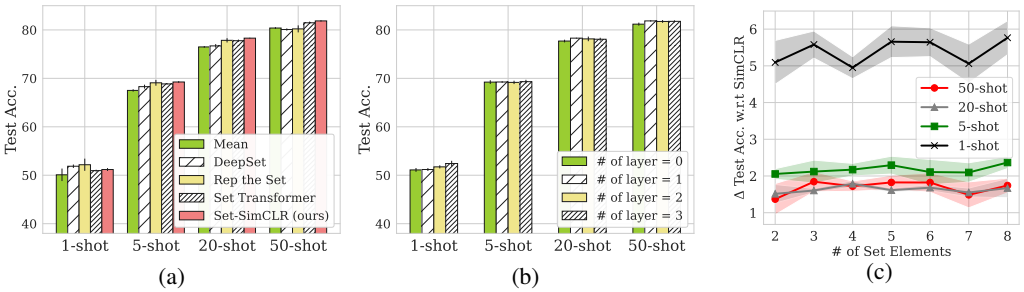

Figure 3: The results of ablation study on the 5-way 1, 5, 20, 50-shot classification using Mini-ImageNet. We study the effectiveness of **(a)**: different set encoder architectures, **(b)**: the depth of `TransformerEncoder` layers, and **(c)**: the number of set elements w.r.t SimCLR. We report the results over 3 different random seeds.

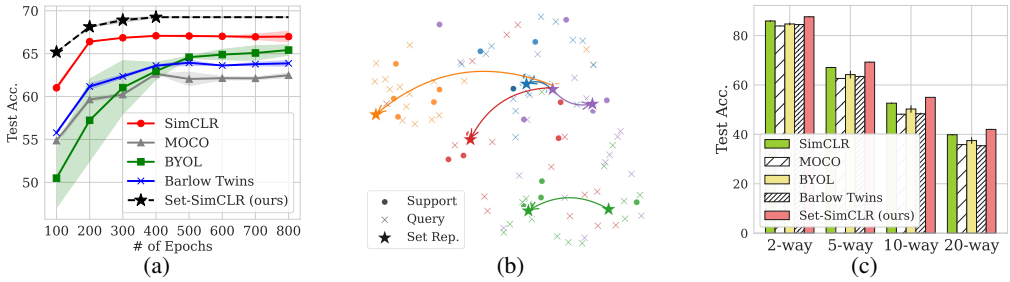

Figure 4: The analysis of the proposed Set-SimCLR on the Mini-ImageNet dataset. **(a)**: The 5-way 5-shot test accuracy of baselines as a function of training epochs. **(b)**: The t-sne visualization of our adaptation process on a 5-way 5-shot task. **(c)**: The 5-shot test accuracy of different ways. We report the results over 3 different runs.

representation. Thus, we choose following four representative self-supervised baselines as follows: **1) SimCLR (Chen et al., 2020a;b)**, **2) MOCO (He et al., 2020; Chen et al., 2020c; 2021)**, **3) BYOL (Grill et al., 2020)**, **4) Barlow Twins (Zbontar et al., 2021)**. We also provide the detailed explanation of the four self-supervised learning baselines in Appendix C.

Implementation Details For the base encoder f, we use ResNet-18 architecture (He et al., 2016) which is widely used for evaluating self-supervised learning methods and set d the dimension of the output of f to 512. For a fair comparison, we use the same architecture of head network g (described in UML experiments), for all self-supervised learning methods except for MOCO. MOCO does not use the head as firstly proposed in the original paper. We use the same random augmentations described in Appendix D. For our method Set-SimCLR, we apply the augmentations 8 times to the mini-batch of 64 images (i.e., $M = 64, V = 8$), resulting in 4 elements in each set, while performing the same augmentation twice on the mini-batch of 256 images (i.e., $M = 256, V = 2$) for the other baselines. For all the methods, we optimize the models for 400 epochs using Adam optimizer (Kingma & Ba, 2014). We do not use learning rate scheduling which is not effective for any methods in our experiments. We search for an adequate learning rate in $0.001, 0.0005, 0.0001$ for baselines and ours using a meta-validation split. We use scikit-learn (Pedregosa et al., 2011) package with default settings to optimize classifiers for downstream tasks.

Results Figure 2a shows the performance of the baselines and our Set-SimCLR for the 5-way 5-shot setting on the Mini-ImageNet, Tiny-ImageNet, CIFAR100, Aircraft, Stanford Cars, CUB datasets. We can see that Set-SimCLR outperforms all the self-supervised learning baselines by considerable margins, from $+0.84\%$ to $+2.17\%$. MOCO shows slightly better result ($+0.26\%$) than ours on the Stanford Cars dataset, however, the difference is not significant considering its standard deviation (0.57%). We further evaluate ours and baselines over different shots, e.g., 1-, 10-, 20, and 50-shot on the Mini-ImageNet dataset. Figure 2b shows the performance of the baselines and our Set-SimCLR, where our Set-SimCLR obtains outstanding performance gains of $4.95\%, 1.55\%, 1.79\%, 1.72\%$ over the best performing baselines on 1-, 10-, 20-, 50-shot settings.

4.3 Ablation Study and Analysis

In this section, we conduct ablation studies to verify a necessity of each components of Set-SimCLR such as architecture and depth of set encoder, and the number of set elements. We further provide analysis on our Set-SimCLR in comparison to self-supervised learning baselines.

Set Encoder Architecture We replace the architecture of the set encoder described in Eq. 2 with mean pooling, Deep Set (Zaheer et al., 2017), Rep the Set (Skianis et al., 2020), or Set Transformer (Lee et al., 2019). Figure 3a shows the 5-way 5-shot test accuracy of different set encoder architectures on the Mini-ImageNet dataset. We find that Rep the Set architecture works well on 1-shot setting, and our set encoder φ in Equation 2 shows slightly better performance on 5-, 20-, and 50-shot settings than the others. Note that our Set-SimCLR is set-representation learning framework that is agnostic to the choice of set encoder architecture. Furthermore, even with the worst architecture (mean pooling), it still shows slightly better performance than the self-supervised baselines in all settings.

The Depth of Set Encoder Other important compoment of our model is the number of `TransformerEncoder` layers in Equation 2. First, we start without `TransformerEncoder` layer, i.e., identity function, and increase the depth of the set encoder. Figure 3b shows the 5-way 5-shot test accuracy on the Mini-ImageNet dataset with varying the number of layers. We find that the set encoder with a single layer is the most effective on the overall settings.

The Number of Set Elements We then conduct experiments by varying the number of set elements (i.e., $V/2$), which is the most important hyperparameter for our Set-SimCLR. Figure 3c shows the 5-way 5-shot test accuracy improvement with different size of set over SimCLR, which is the best-performing baseline on the Mini-ImageNet dataset. We find that our Set-SimCLR shows consistent improvements over SimCLR for all the settings we consider.

Training Budgets Analysis It takes more time to train our Set-SimCLR than self-supervised baselines we consider, since ours requires multiple (more than two) stochastic augmentations to construct a set. One may wonder whether the baseline can be comparable or even better if we train it with similar computational budgets to ours. To address this question, we train the self-supervised learning baselines for 800 epochs, which is twice larger than the before, and observe test accuracy over training. Figure 4a shows the 5-way 5-shot test accuracy of self-supervised learning baselines on the Mini-ImageNet dataset. We find that the our Set-SimCLR evaluated at 400 epochs largely outperforms the self-supervised baselines for all the training budgets we consider.

Qualitative Analysis on Adaptation of Set Representation We now qualitatively analyze the adaptation process of our set-representation at meta-test time. To do so, we visualize the set-representations before and after the adaptation (i.e., each row of the classifier weight W_0 and W_0^*), support points, and query points. As shown in Figure 4b, we normalize all the point to be length 1 and project them to 2-dimensional space with T-SNE (Van der Maaten & Hinton, 2008). Here support points, query points, and set representation are denoted as a circle, cross, and star. We further denote arrows as the adaptation of set representation, and the color stands for each class. We find that the set-representation is not discriminative at the beginning, however, it represents each class very well after the adaptation. This shows the necessity of our proposed adaptation process of set-representation to achieve better performance on the downstream tasks.

Accuracy over Different Ways We finally conduct experiments to show the performance of each model with varying the way of meta-test tasks. Figure 4c shows the 2-, 5-, 10-, and 20-way 5-shot test accuracy of the self-supervised learning baselines and ours on the Mini-ImageNet dataset. We find that our Set-SimCLR consistently outperforms the baselines on all the way we consider here.

5 Conclusion

In this paper, we have proposed self-supervised set-representation learning framework for unsupervised meta-learning (UML). Our Set-SimCLR learns set-representation by maximizing the agreement between positive sets in latent space, where the positive sets are constructed with repeated stochastic augmentations of an image. Based on theoretical analysis, we have studied how the learnt set-representation can improve generalization ability and why it makes sense to initialize of W_0 with the learnt set-representation for downstream tasks. We further validate the empirical efficiency of proposed Set-SimCLR in respect to UML and self-supervised baselines using several benchmark few-shot classification datasets. Note that our main idea of constructing semantically similar sets with repeated augmentations is not limited to SimCLR framework. Based on this, we plan to expand our framework to various self-supervised learning methods to leverage their potential merits.

REFERENCES

Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149, 2018.

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

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020b.

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

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9640–9649, 2021.

Dan dan Guo, Long Tian, Minghe Zhang, Mingyuan Zhou, and Hongyuan Zha. Learning prototype-oriented set representations for meta-learning. In *International Conference on Learning Representations*, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pp. 248–255. IEEE Computer Society, 2009.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. Adversarial feature learning. In *International Conference on Learning Representations*, 2017.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2019.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. In *International Conference on Learning Representations*, 2019.

Ekaterina Iakovleva, Jakob Verbeek, and Karteek Alahari. Meta-learning with shared amortized variational inference. In *International Conference on Machine Learning*, pp. 4572–4582. PMLR, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pp. 448–456. PMLR, 2015.

Siavash Khodadadeh, Ladislau Boloni, and Mubarak Shah. Unsupervised meta-learning for few-shot image classification. *Advances in neural information processing systems*, 32, 2019.

Siavash Khodadadeh, Sharare Zehtabian, Saeed Vahidian, Weijia Wang, Bill Lin, and Ladislau Boloni. Unsupervised meta-learning through latent-space interpolation in generative models. In *International Conference on Learning Representations*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Soheil Kolouri, Navid Naderializadeh, Gustavo K Rohde, and Heiko Hoffmann. Wasserstein embedding for graph learning. *arXiv preprint arXiv:2006.09430*, 2020.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

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

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Dong Bok Lee, Dongchan Min, Seanie Lee, and Sung Ju Hwang. Meta-GMVAE: Mixture of gaussian VAE for unsupervised meta-learning. In *International Conference on Learning Representations*, 2021.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019.

Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

Grégoire Mialon, Dexiong Chen, Alexandre d'Aspremont, and Julien Mairal. A trainable optimal transport embedding for feature aggregation and its relationship to attention. *arXiv preprint arXiv:2006.12065*, 2020.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 3674–3683, 2020.

Konstantinos Skianis, Giannis Nikolentzos, Stratis Limnios, and Michalis Vazirgiannis. Rep the set: Neural networks for learning set representations. In *International conference on artificial intelligence and statistics*, pp. 1410–1420. PMLR, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pp. 12310–12320. PMLR, 2021.

A TRANSFORMER ENCODER

We describe the TransformerEncoder from Eq. 2 in more detail. Let $X \in \mathbb{R}^{n \times d}$ be a stack of n d-dimensional row vectors. Let $W_j^Q, W_j^K, W_j^V \in \mathbb{R}^{d \times d_k}$ be weight matrices for self-attention and let $b_j^Q, b_j^K, b_j^V \in \mathbb{R}^{d_k}$ be bias vectors for $j = 1, \ldots, 4$. For encoding the input X, we compute self-attention as follows:

$$\begin{aligned}
Q^{(j)} &= XW_j^Q + \mathbf{1}(b_{1,j}^Q)^\top \in \mathbb{R}^{n \times d_k} \\
K^{(j)} &= XW_j^Q + \mathbf{1}(b_{1,j}^K)^\top \in \mathbb{R}^{n \times d_k} \\
V^{(j)} &= XW_j^K + \mathbf{1}(b_{i,j}^V)^\top \in \mathbb{R}^{n \times d_k} \\
A^{(j)}(X) &= \text{LayerNorm}\left(Q_1^{(j)} + \text{softmax}\left(Q_1^{(j)}(K_1^{(j)})^\top / \sqrt{d_k}\right) V_1^{(j)}\right) \in \mathbb{R}^{n \times d_k} \\
O(X) &= \text{Concat}(A_1^{(1)}(X), \ldots, A_1^{(4)}(X)) \in \mathbb{R}^{n \times d}
\end{aligned} \tag{5}$$

where $\mathbf{1} = (1, \ldots, 1)^\top \in \mathbb{R}^n$ is a vector of ones, $d = 4d_k$, and softmax is applied for each row. After self-attention, we add a skip connection with layer normalization (Ba et al., 2016) as follows:

$$\texttt{TransformerEncoder}(X) = \text{LayerNorm}\left(O(X)\right) + \text{ReLU}\left(WO(X) + \mathbf{1}b^\top\right) \tag{6}$$

where $W \in \mathbb{R}^{d \times d}, b \in \mathbb{R}^d$.

B ON THEORETICAL MOTIVATION

B.1 ON THE RELATIONSHIP WITH METRIC-BASED INFERENCE

The metric-based inference using the instance-level representation h of x with the class prototypes s_c can be written by

$$\hat{y}(x) = \arg\min_c d(s_c, h).$$

By choosing the metric to be the negative dot product as $d(s_c, h) = -s_c^\top h$, we can write

$$\hat{y}(x) = \arg\min_c -s_c^\top h = \arg\min_c -\log \frac{\exp(s_c^\top h)}{\sum_k \exp(s_k^\top h)} = \arg\min_c \mathcal{L}_{\text{CE}}(\texttt{softmax}(W_0 f(x)), c),$$

where the second line follows from the fact that the output of $\arg\min_c$ does not change by adding the constant in c. In other words, the prediction of the metric-based inference with $d(s_c, h) = -s_c^\top h$ (without further fine-tuning) is equivalent to the proposed method at the initialization of W_0. Thus, the proposed method can be understood as the metric-based inference with the fine-tuning of the class prototypes s_c based on the support set, where the initial class prototypes s_c are obtained by the set-representation and each input x represented by instance-level representation h.

In this view, a naive approach of fine-tuning of the class prototypes s_c is to fine-tune the parameters of φ to minimize $-\log \frac{\exp(s_c^\top h)}{\sum_k \exp(s_k^\top h)}$ with the support set where $s_c = \varphi(\mathbf{H}_c)$. However, since φ has many parameters, changing φ allows to change $s_c = \varphi(\mathbf{H}_c)$ freely without restrictions on the space of s_c. Thus, instead of fine-tuning the parameters of φ, we can directly optimize the values of s_c by initializing $s_c = \varphi(\mathbf{H}_c)$ and untying s_c from $\varphi(\mathbf{H}_c)$. This is what is done in the proposed algorithm. This results in the faster computation and the well-behaving convex optimization when compared to the fine-tuning of parameters of φ.

B.2 ON INTERPRETATION OF THEOREM 1

There are two ways to interpret the theorem. The first interpretation is obtained by setting the probability measures for the expectations to be the empirical measures. The second interpretation is obtained by setting the probability measures for the expectations to be the true unknown probability measures. The first way is discussed in the main text. Thus, we now focus on the second interpretation. In the the second interpretation, Theorem 1 is all about the expected losses; i.e., $L^{\text{SimCLR}}(f)$ is the expected contrastive loss, L_{s} is the expected supervised loss, and $\mathbb{P}(y^+ = y^-)$ is the probability of

the positive and negative labels colliding. In this interpretation, we can see that the γ is still expected to be positive even at $t = 0$ since φ can act as an empirical approximation of the average pooling to obtain $L_{\mathrm{s}}^{A}(f) - L_{\mathrm{s}}(f) \approx 0$ and φ is optimized to be better than the average pooling. As we increase t, we tend to further increase the γ since we are reducing the empirical loss of the support set, and the generalization gap between the empirical loss of the support set and the expected loss can be bounded by standard generalization bounds. Once this generalization gap start to increase, γ starts to decrease, resulting into the overfitting to the support set. Thus, it still makes sense to train W_t by setting $\Delta_t \neq 0$ but we should regularize W_t so that the γ does not decrease during the fine-tuning of W_t with the support set. This is qualitatively the same conclusion as that of the first interpretation.

B.3 Proof of Theorem 1

Proof. Define the function ℓ by

$$\ell\left(q\right) = \log(1 + \exp(-q)).$$

Then,

$$\begin{aligned}
&\ell\left(f(x^{++})^\top (f(x^+) - f(x^-))\right) \\
&= \log(1 + \exp(-f(x^{++})^\top (f(x^+) - f(x^-))) \\
&= \log\left(\left(1 + \exp(-f(x^{++})^\top f(x^+) + f(x^{++})^\top f(x^-))\right) \times \frac{\exp(f(x^{++})^\top f(x^+))}{\exp(f(x^{++})^\top f(x^+))}\right) \\
&= \log\left(\frac{\exp(f(x^{++})^\top f(x^+)) + \exp(f(x^{++})^\top f(x^-)))}{\exp(f(x^{++})^\top f(x^+))}\right) \\
&= -\log\left(\frac{\exp(f(x^{++})^\top f(x^+))}{\exp(f(x^{++})^\top f(x^+)) + \exp(f(x^{++})^\top f(x^-)))}\right).
\end{aligned}$$

Thus, we have that

$$L^{\mathrm{SimCLR}}(f) = \mathbb{E}_{\substack{y^+\sim U \\ y^-\sim U}} \mathbb{E}_{\substack{x^+, x^{++}\sim \mathcal{D}_{y^+}^2 \\ x^-\sim \mathcal{D}_{y^-}}} \left[\ell\left(f(x^{++})^\top (f(x^+) - f(x^-))\right)\right].$$

Then, from the convexity of ℓ, Jensen's inequality, and the linearity of the expectation, we have that

$$L^{\mathrm{SimCLR}}(f) \geq \mathbb{E}_{\substack{y^+\sim U \\ y^-\sim U}} \mathbb{E}_{x^{++}\sim \mathcal{D}_{y^+}} \left[\ell\left(f(x^{++})^\top (\mathbb{E}_{x^+\sim \mathcal{D}_{y^+}}[f(x^+)] - \mathbb{E}_{x^-\sim \mathcal{D}_{y^-}}[f(x^-)])\right)\right].$$

By decomposing the expectation with sums of conditional expectations, conditioned on the event of $y^+ = y^-$ and its complement of $y^+ \neq y^-$,

$$L^{\mathrm{SimCLR}}(f) \geq \mathbb{P}(y^+ = y^-)\kappa_1 + \mathbb{P}(y^+ \neq y^-)\kappa_2 \tag{7}$$

where

$$\kappa_1 = \mathbb{E}_{\substack{y^+\sim U \\ y^-\sim U}} \left[\mathbb{E}_{x^{++}\sim \mathcal{D}_{y^+}} \left[\ell\left(f(x^{++})^\top (\mathbb{E}_{x^+\sim \mathcal{D}_{y^+}}[f(x^+)] - \mathbb{E}_{x^-\sim \mathcal{D}_{y^-}}[f(x^-)])\right)\right] \mid y^+ = y^-\right]$$

and

$$\kappa_2 = \mathbb{E}_{\substack{y^+\sim U \\ y^-\sim U}} \left[\mathbb{E}_{x^{++}\sim \mathcal{D}_{y^+}} \left[\ell\left(f(x^{++})^\top (\mathbb{E}_{x^+\sim \mathcal{D}_{y^+}}[f(x^+)] - \mathbb{E}_{x^-\sim \mathcal{D}_{y^-}}[f(x^-)])\right)\right] \mid y^+ \neq y^-\right].$$

For the first term, since $y^+ = y^-$ inside the loss ℓ, we have that

$$\kappa_1 = \ell(0) = \log(2) \tag{8}$$

For the second term,

$$\begin{aligned}
\kappa_2 &= \mathbb{E}_{y^+\sim U} \left[\mathbb{E}_{x^{++}\sim \mathcal{D}_{y^+}} \left[\ell\left(f(x^{++})^\top (\mathbb{E}_{x^+\sim \mathcal{D}_{y^+}}[f(x^+)] - \mathbb{E}_{x^-\sim \mathcal{D}_{r(y^+)}}[f(x^-)])\right)\right]\right] \\
&= \mathbb{E}_{(x,y)\sim \mathcal{D}} \left[\ell\left(g_\varphi(x)_y - g_\varphi(x)_{r(y)}\right)\right] + \tilde{\gamma}
\end{aligned} \tag{9}$$

where $g_\varphi(x) = W_t f(x)$, $g_A(x) = Af(x^{++})$, and $\tilde{\gamma} = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell\left(g_A(x)_y - g_A(x)_{r(y)}\right) - \ell\left(g_\varphi(x)_y - g_\varphi(x)_{r(y)}\right)]$. By defining r by

$$r(y) = \begin{cases} 1 & \text{if } y = 2 \\ 2 & \text{if } y = 1, \end{cases}$$

we have that

$$\begin{aligned} \ell\left(g(x)_y - g(x)_{r(y)}\right) &= \log\left(\left(1 + \exp(-g(x)_y + g(x)_{r(y)}\right) \times \frac{\exp(g(x)_y)}{\exp(g(x)_y)}\right) \\ &= -\log\left(\frac{\exp(g(x)_y)}{\exp(g(x)_y) + \exp(g(x)_{r(y)})}\right) \\ &= -\log\left(\frac{\exp(g(x)_y)}{\sum_{k=1}^{2}\exp(g(x)_k)}\right). \end{aligned} \tag{10}$$

By combining equation 9 and equation 10,

$$\kappa_2 = \mathbb{E}_{(x,y)\sim\mathcal{D}}\left[-\log\left(\frac{\exp(g_\varphi(x)_y)}{\sum_{k=1}^{2}\exp(g_\varphi(x)_k)}\right)\right] + \gamma, \tag{11}$$

By combining equation 7, equation 8, and equation 11, we have

$$\begin{aligned} L^{\text{SimCLR}}(f) \geq \mathbb{P}(y^+ \neq y^-)&\left(\mathbb{E}_{(x,y)\sim\mathcal{D}}\left[-\log\left(\frac{\exp(g_\varphi(x)_y)}{\sum_{k=1}^{2}\exp(g_\varphi(x)_k)}\right)\right] + \gamma\right) \\ &+ \mathbb{P}(y^+ = y^-)\log(2) \end{aligned}$$

This implies that

$$L_{\text{s}}(f) \leq cL^{\text{SimCLR}}(f) - \zeta\log(2) - \gamma.$$

□

C Details of Baselines

In this section, we detail the supervised meta-learning, unsupervised meta-learning, and instance-level self-supervised learning baselines. We first introduce two supervised meta-learning approaches which we consider as "*oracles*", and four different unsupervised meta-learning baselines as follows;

1) MAML (oracle) (Finn et al., 2017): Model Agnostic Meta Learning where it learns the initialization of the parameters of the model such that few steps of gradient descent on a support set leads to generalization on a query set. We compare against its performance reported in Hsu et al. (2019).

2) ProtoNets (oracle) (Snell et al., 2017): Euclidean distance-based meta-learning framework. It learns a metric embedding space where we perform prediction by computing a distance between class prototype and instances from query sets. We also compare against it using its performance reported in Hsu et al. (2019).

3) CACTUs (Hsu et al., 2019): Clustering to Automatically Construct Tasks for Unsupervised meta-learning. It automatically constructs tasks by clustering the unsupervised dataset in embedding space learned by ACAI (Berthelot et al., 2019), BiGAN (Donahue et al., 2017), or DeepCluster (Caron et al., 2018). Then it train either MAML or ProtoNets using the cluster indices as pseudo-labels.

4) UMTRA (Khodadadeh et al., 2019): Unsupervised Meta-learning with Tasks constructed by Random sampling and Augmentation. For constructing a K-way 1-shot task, it randomly samples K-way data points from unsupervised dataset and augments each data point. Then MAML is trained on the constructed tasks.

5) LASIUM (Khodadadeh et al., 2021): It trains generative models on the given unlabeled data and sample N different latent vector such that each pair-wise distance is greater than a predefined threshold. Each latent vector is fed into the generative model and decoded to a training instance belonging to distinct class. Then it adds some noise to each latent vector to generate S examples and

the generated ones are labeled with the class of the original latent vector. Finally, it trains MAML or ProtoNets using the synthetic N-way S-shot task.

6) Meta-GMVAE (Lee et al., 2021): Meta-level Gaussian Mixture Variational AutoEncoder. It learns a latent representation by matching set-level amortized variational posterior and task-specific multi-modal prior optimized by EM algorithm.

We then present the four representative self-supervised baselines used in our experiments as follows:
1) SimCLR (Chen et al., 2020a;b): It is a constrative learning framework which learns by maximizing agreement between differently augmented views of the same data example in the latent space.

2) MOCO (He et al., 2020; Chen et al., 2020c; 2021): It builds a dynamic feature dictionary using a queue and momentum encoder and learns to minimize contrastive loss from the dictionary.

3) BYOL (Grill et al., 2020): From pair views of an image, it learns visual representation by matching momentum encoder, which is exponentially moving average of the encoder.

4) Barlow Twins (Zbontar et al., 2021): This method measure the cross-correlation matrix between the feature representations of two different views and learns by making it close to identity matrix.

D RANDOM AUGMENTATION

Augmentation	Probability	Hyperparameters
`ResizedCrop`	1.0	size $= (84, 84)$, scale $= (0.08, 1.0)$, ratio $= (0.75, 1.3...)$
`HorizontalFlip`	0.5	N/A
`ColorJitter`	0.8	brightness $= 0.8$, contrast $= 0.8$, saturation $= 0.8$, hue $= 0.2$
`GrayScale`	0.2	N/A
`GaussianBlur`	0.5	kernel_size $= (85, 85)$, $\sigma\ U(0.1, 2.0)$

Table 2: The application probability and hyperparameters of each augmentation.

For random augmentation, we compose `ResizedCrop`, `HorizontalFlip`, `ColorJitter`, `GrayScale`, and `GaussianBlur`. The application probability and hyperparameters of each augmentation is shown in Table 2. Note that we perform `ResizedCrop` on a larger resolution of 224×224 images than the resolution of 84×84 images we target, which is found to be more effective. We implement the augmentation using Kornia framework (Riba et al., 2020).