Self-Attention Message Passing for Contrastive Few-Shot Learning

Anonymous WACV 2023 APPLICATIONS TRACK submission

Paper ID ****

Abstract

As humans, we have the natural ability to rapidly learn patterns and constantly make connections between the commonalities of multiple disparate objects and concepts. Inspired by this human process, this paper focuses on unsupervised learning from an abundance of unlabeled data followed by a few-shot fine-tuning on a downstream classification task. To this end, we develop a contrastive self-supervised scheme to enable refinement of features by incorporating message-passing networks with CNNs, giving the network a strong inductive prior and the ability to share features and learn commonalities between images. We also propose a lightweight optimal-transport-based task-awareness module, further increasing the flexibility of our method.

1. Introduction

In recent years, deep learning models have become larger and increasingly demand more data to perform their tasks satisfactorily. However, few-shot learning has been garnering increasing interest recently because it underscores a fundamental gap between smart human adaptability and data-hungry supervised and unsupervised deep learning methods. To address this challenge, few-shot classification is cast as a task to predict class labels for a set of unlabeled data points (*query set*) given only a small set of labeled data points (*support set*). Typically, the query and support data points are drawn from the same distribution.

Few-shot classification methods typically comprises two sequential phases: (i) *pre-training* on a large dataset of "base" classes, regardless of the training being supervised or unsupervised. This is followed by (ii) *fine-tuning* on an unseen dataset consisting of "novel" classes. Normally, the classes used in the pre-training and fine-tuning are mutually exclusive. In this paper, our focus is on the self-supervised (also sometimes interchangeably called "unsupervised" in the literature) setting where we have no access to the actual class labels of the "base" dataset.

To this end, various methods have been proposed and broadly classified into two different approaches. The first approach relies on the use of *meta-learning* and episodic training that involves the creation of synthetic "tasks" to mimic the subsequent episodic fine-tuning phase [1, 17, 23–25, 29, 54]. The second method follows a *transfer learning* approach, where the network trained non-episodically to learn optimal representations in the pre-training phase, which is then followed by an episodic fine-tuning phase [15, 33, 43]. In this method, a feature extractor (encoder) is trained using a form of metric learning to capture the structure of unlabeled data. Next, a simple predictor (conventionally a linear layer) is utilized in conjunction with the pre-trained feature extractor for quick adaptation to the novel classes in the fine-tuning phase. The better the feature extractor captures the global structure of the unlabeled data, the less the predictor requires training samples, and the faster it adapts itself to the unseen classes in the fine-tuning phase (also the testing phase).

Furthermore, supervised approaches that follow the episodic training paradigm may include a certain degree of *task awareness*. Such approaches exploit the information available in the query set during the training or testing phases [3, 11, 55] to alleviate the model's sample bias. As a result of this, task awareness allows the model to learn task-specific embeddings by better aligning the features of the support and query samples. We also see a set of supervised approaches that do not rely purely on a convolutional feature extractor. Instead, these approaches also use graphs and graph neural networks [26, 39, 53, 56]. Using a graph neural network (GNN) can help model instance-level and class-level relationships. GNN's can also help propagate labels by using a task-agnostic classifier, and such methods have been shown to work quite effectively when compared against their standard convolutional counterparts [26, 39, 53, 56]. However, graph-based methods have eluded the unsupervised setting.

Several recent studies have questioned the necessity of meta-learning for few-shot classification [4, 7, 15, 33, 41, 43, 60]. They report competitive performance on few-shot benchmarks without episodic training or few-shot task-based experiences during training. These methods follow the second approach and aim to solve the few-shot learning problem by fine-tuning a pre-trained feature extractor with a standard

cross-entropy loss. Some of these methods [13, 33, 43] in the space demonstrate that the transfer learning approach outperforms meta-learning based methods in standard in-domain and cross-domain settings, where the training and novel classes come from totally different distributions.

Many of the aforementioned unsupervised methods use a form of self-supervised learning called *contrastive learning*. Although these methods work pretty well, they overlook the fact that a given batch of images may contain multiple images belonging to the same class. Contrastive learning methods typically treat every image in a batch as its own class. The only other images that share the class are the augmentations of the image in question. Such methods enforce similarity between representations of an image and its augmentations, while enforcing dissimilarity between all other images through a *contrastive loss*. However, it is plausible that within a randomly sampled batch of images and their augmentations, there could be several images that, in reality, belong to the same class. By applying the contrastive loss, the network may inadvertently learn different representations for such images and classes. Recent methods such as SimCLR [8] use large batch sizes to approximate the data distribution to avoid this problem. In other words, there is a failure to look beyond single instances to learn the representations of images. To overcome this pitfall in the pre-training phase, we propose `SAMP-CLR` - using a form of *graph attention* with traditional feature extractors. This allows us to utilize the contrastive training scheme, while simultaneously learning refined representations by looking beyond single-image instances. In the few-shot downstream tasks, using graph attention also allows for a degree of task awareness to be induced in the feature extractor by using query samples to better align feature representations. We propose an *optimal transport* based algorithm during the fine-tuning stage that aligns the distributions of the support and query samples to improve downstream adaptability of the pre-trained encoder, in a way that requires no additional parameters. Our **contributions** can be summarized as: *(i)* we propose a graph-based contrastive learning approach that helps learn representations while accounting for intra-batch relationships, *(ii)* we propose an optimal transport based fine-tuning phase that enhances feature refinement, which in turn helps in stabilizing and improving test time performance without the need for additional trainable parameters, *(iii)* we present a detailed analysis of both the above points to demonstrate the superiority and robustness of our methods, *(iv)* we show that our unsupervised method is able to match or beat the performance of some of the latest supervised methods.

2. Related Work

Self-Supervised Learning. Self-supervised learning (SSL) is an umbrella term for a set of unsupervised methods that obtain supervisory signals from within the data itself, more often than not by leveraging the underlying structure in the data. The general technique of self-supervised learning is to predict any unobserved or hidden part (or property) of the input from any observed or unhidden part. Several latest advances in the SSL space have made waves by eclipsing their fully supervised counterparts. Some examples of seminal works include SimCLR [8], BYOL [19], SWaV [6], MoCo [21], and SimSiam [9]. Our pre-training method `SAMP-CLR` is inspired by SimCLR [8] and ProtoTransfer [33].

Metric Learning. Metric learning aims to learn a representation function that maps the data into an embedding space. The distance between objects in the embedding space must preserve their similarity (or dissimilarity) - similar objects are closer, while dissimilar objects are farther. For example, unsupervised methods based on some form of contrastive loss, such as SimCLR [8] or NNCLR [16], guide objects belonging to the same class to be mapped to the same point and those from different classes to be mapped to different points whose distances are larger than a margin. This process generally involves taking two crops of the same image and encouraging the network to emit an identical representation for the two while ensuring that the representations remain different from all other images in a given batch. Note that in an unsupervised setting, each image in a batch is its own class. Metric learning methods in both supervised and unsupervised settings have been shown to work quite well for few-shot learning. AAL-ProtoNets [1], ProtoTransfer [33], CACTUS [23], and certain types of graph neural networks [39] are excellent examples that use metric learning for few-shot learning.

Graph Neural Networks for FSL. Since the first use of graphs for FSL in [39], there have been several advancements and continued interest in using graphs for supervised FSL. In [39], each node corresponds to one instance (labeled or unlabeled) and is represented as the concatenation of a feature embedding and a label embedding. The final layer of their model is a linear classifier layer that directly outputs the prediction scores for each unlabeled node. There has also been a rise in methods using transduction. TPN [32] is one of those methods that uses graphs to propagate labels from labeled samples to unlabeled samples. Although methods like EGNN [26] make use of edge and node features, earlier methods focused only on using node features. Graphs are attractive because they can model intra-batch relations and can be extended for transduction, as evidenced in [26, 32] Along with transduction and relation modeling, graphs are highly potent as task adaptation modules. HGNN [56] is an example in which a graph is not used for label propagation Instead, it is used to refine and adapt feature embeddings To the best of our knowledge, it must be noted that most graph-based methods have been applied in the supervised FSL setting and that we are the first to use it in any form for

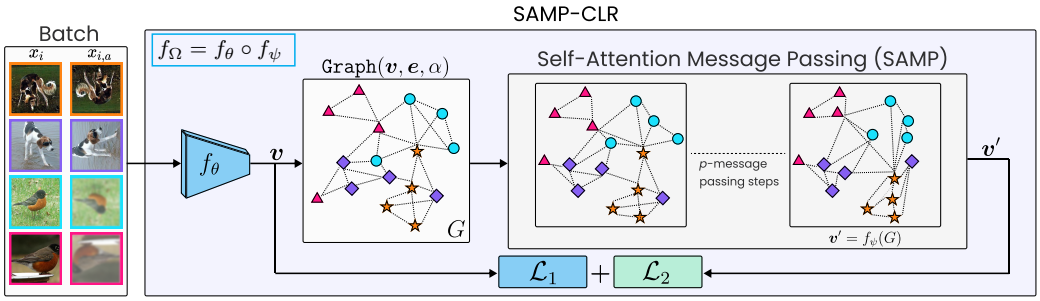

Figure 1: SAMP-CLR schematic view and pre-training procedure. In the figure, $x_{i,a}$ is an image sampled from the augmented set A. The p-message passing steps refine the features extracted using a CNN encoder.

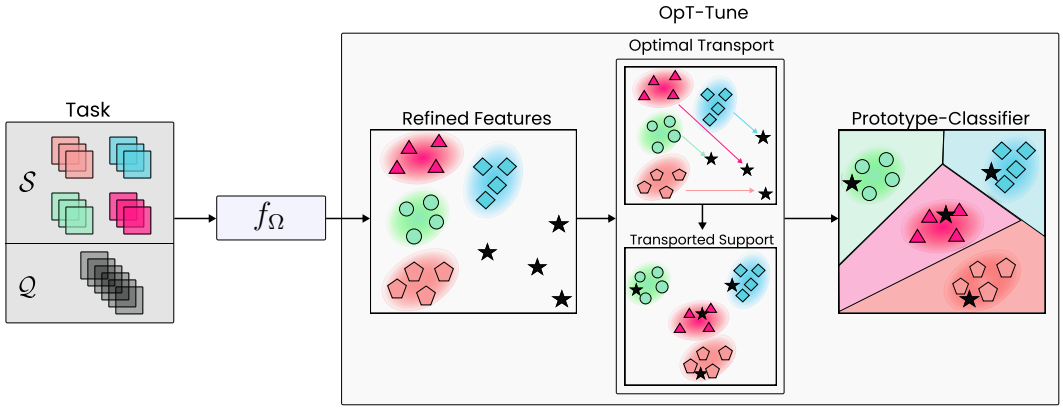

Figure 2: Features extracted from the pre-trained CNN are used to build a graph. Features refined using the pre-trained GAT layer(s). Optimal Transport based fine-tuning algorithm aligns support features with query features.

unsupervised FSL.

3. Proposed Method (SAMPTransfer)

In this section, we first describe our problem formulation. We then discuss the two phases of the proposed approach, SAMPTransfer: self-supervised pre-training (SAMP-CLR) and the optimal transport based few-shot supervised fine-tuning (OpT-Tune). The mechanics of the proposed pre-training procedure is shown in Figure 1, and the conceptual functioning of the fine-tuning procedure is illustrated in Figure 2.

3.1. Preliminaries

Let us denote the training data of size D as $\mathcal{D}_{\text{tr}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{D}$ with $(\boldsymbol{x}_i, y_i)$ representing an image and its class label, respectively. In the pre-training phase, we take L random samples from $\mathcal{D}_{\text{tr}}$ and augment each sample A times by randomly sampling augmentation functions $\zeta^a(.), \forall a \in [A]$ from the set $\mathcal{A}$. This results in a mini-batch of size $B = (A+1)L$ total samples. Note that we omit the data labels in the pre-training phase for the sake of unsupervised learning. Following the pre-training phase, we fine-tune our model episodically [47] on a set of randomly sampled tasks $\mathcal{T}_i$ from the test dataset $\mathcal{D}_{\text{tst}} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{D'}$ of size D'. The tasks $\mathcal{T}_i$ are constructed by drawing K labeled and Q unlabeled random samples from N different classes, which is called a (N way, K shot task), denoted by (N, K). These NK labeled samples make up the support set $\mathcal{S}$, from which the model learns, and the other NQ unlabeled samples make up the query set $\mathcal{Q}$, on which the model performs inference.

3.2. Self-Attention Message Passing (SAMP)

Our network architecture comprises a convolutional (CNN) feature extractor f_θ and a message passing network based on self-attention, f_ψ. The CNN feature extractor f_θ, parameterised by θ, is used to extract features $\boldsymbol{v} = f_\theta(\boldsymbol{x})$, where $\boldsymbol{v} \in \mathbb{R}^{B \times d}$ is the set of features and $\boldsymbol{x} \in \mathbb{R}^{B \times C \times H \times W}$ is a batch of images. To help refine the features and use batch-level relations, we create a graph $G = \text{Graph}(\boldsymbol{v}, \boldsymbol{e}, \alpha)$ where $\boldsymbol{v}$ is treated as a set of initial node features, $\boldsymbol{e}$ is the pairwise distance between all nodes based on a relevant distance metric, and α is a threshold on the values in $\boldsymbol{e}$ that determines if two nodes will be connected or not. Following this, we use a **s**elf-**a**ttention **m**essage **p**assing neural network (SAMP) f_ψ, parameterised by ψ, to refine the initial feature vectors by exchanging and amalgamating information between all connected node pairs

Algorithm 1: SAMP-CLR

Require: $\mathcal{D}_{\text{tr}}$, $\mathcal{A}$, f_θ, f_ψ, Ω, α, β, γ, $d[\cdot,\cdot]$, $d'[\cdot,\cdot]$

while *not done* **do**
  Sample minibatch $\{\boldsymbol{x}_i\}_{i=1}^{L}$
  forall $i \in \{1,\ldots,L\}$ **do**
    forall $a \in \{1,\ldots,A\}$ **do**
      $\tilde{\boldsymbol{x}}_{i,a} = \zeta^a(\boldsymbol{x}_i)$; $\zeta^a \sim \mathcal{A}$.
    end
  end
  $\boldsymbol{z}, \tilde{\boldsymbol{z}} = f_\theta\left(\{\boldsymbol{x}_i\}_{i=1}^{L}\right), f_\theta\left(\{\tilde{\boldsymbol{x}}_{i,\,a}\}_{i=1,a=1}^{L,A}\right)$
  $\boldsymbol{v} = \left[\{\boldsymbol{z}_i\}_{i=1}^{L}, \{\tilde{\boldsymbol{z}}_{i,\,a}\}_{i=1,a=1}^{L,A}\right]$
  $\boldsymbol{e} = \{d'(v_i, v_j)\;\forall\, v_i, v_j \in \boldsymbol{v}\}$
  $G = \texttt{Graph}(\boldsymbol{v}, \boldsymbol{e}, \alpha)$
  $\boldsymbol{v}' = f_\psi(G)$
  $\boldsymbol{z}', \tilde{\boldsymbol{z}}' = \boldsymbol{v}'[:L], \boldsymbol{v}'[L:]$
  let $\ell(i,a) = -\log \frac{\exp\left(-d\left[\tilde{\boldsymbol{z}}_{i,a}, \boldsymbol{z}_i\right]\right)}{\sum_{k=1}^{L}\exp\left(-d\left[\tilde{\boldsymbol{z}}_{i,a}, \boldsymbol{z}_k\right]\right)}$
  let $r(i,a) = -\log \frac{\exp\left(-d\left[\tilde{\boldsymbol{z}}'_{i,a}, \boldsymbol{z}'_i\right]\right)}{\sum_{k=1}^{L}\exp\left(-d\left[\tilde{\boldsymbol{z}}'_{i,a}, \boldsymbol{z}'_k\right]\right)}$
  $\mathcal{L}_1 = {}^1\!/_{LA}\sum_{i=1}^{L}\sum_{q=1}^{A}\ell(i,a)$
  $\mathcal{L}_2 = {}^1\!/_{LA}\sum_{i=1}^{L}\sum_{q=1}^{A}r(i,a)$
  $\mathcal{L} = \beta\mathcal{L}_1 + \mathcal{L}_2$
  $\Omega \leftarrow \Omega - \gamma\nabla_\Omega\mathcal{L}$
end

Algorithm 2: OpT-Tune

Require: f_Ω, $d[\cdot,\cdot]$, $\boldsymbol{x}_s$, $\boldsymbol{x}_q$

$\mathcal{Z}_s, \mathcal{Z}_q = f_\Omega\left([\boldsymbol{x}_s, \boldsymbol{x}_q]\right)$
$M = d(\mathcal{Z}_s, \mathcal{Z}_q)$
$\boldsymbol{\pi}^\star \leftarrow$ Solving eq. (2) using Sinkhorn-Knopp
$\hat{\boldsymbol{\pi}}^\star(i,j) \leftarrow {}^{\boldsymbol{\pi}^\star(i,j)}\!/_{\sum_j \boldsymbol{\pi}^\star(i,j)}, \forall i,j$
$\hat{\mathcal{Z}}_s := \hat{\boldsymbol{\pi}}^\star\mathcal{Z}_q \leftarrow$ Given by eq. (3)
return: $\hat{\mathcal{Z}}_s$

We shall refer to the feature extractor and the SAMP module collectively as $f_\Omega = f_\psi \circ f_\theta$ where $\Omega = \{\theta, \psi\}$ is the collection of all parameters. The SAMP layer(s), f_ψ operates on the graph G.

Message Passing Neural Network. To allow an effective exchange of information to refine initial node features $\boldsymbol{v}$, we make use of graph attention [46]. However, our implementation is slightly different from the standard graph attention defined in [46]. Standard graph attention makes use of a single weight matrix $\boldsymbol{W}$ that acts as a shared linear transformation for all nodes. Instead, we choose to use scaled dot product self-attention as defined in [40, 45]. The major benefit of this is that it affords the network more expressivity as shown in [5, 27]. The use of three separate representations (query, key, and value) instead of just a single weight matrix to linearly transform the data is key to modeling relationships between data points.

Message Passing. We apply the p message passing steps successively. In each step, we pass messages between the connected nodes of the graph and obtain updated features v_i^{p+1} for node i at the message passing step $p+1$ by aggregating the features v_j^p of all neighboring nodes $j \in \mathcal{N}_i$ at the message passing step p: $v_i^{p+1} = \sum_{j\in\mathcal{N}_i}\boldsymbol{W}^p v_j^p$ where $\boldsymbol{W}^p$ is the weight matrix of the message passing step p. The feature representation of an image i, is affected by the nodes present in its neighborhood $\mathcal{N}_i$.

Attention Weights on Messages. Of course, not all samples of a given mini-batch and neighborhood are equally informative in order to refine representations. Therefore, we multiply an attention score λ to each message passing step with the intention of allowing each sample to weigh the importance of other samples in the mini-batch: $v_i^{p+1} = \sum_{j\in N_i}\lambda_{ij}^p\boldsymbol{W}^p v_j^p$ where λ_{ij} is the attention score between nodes i and j. We employ scaled dot-product self-attention to compute the attention scores, leading to λ_{ij} at step p defined as: $\lambda_{ij}^p = \text{softmax}\left(\boldsymbol{W}_q^p v_i^p \left(\boldsymbol{W}_k^p v_j^p\right)^T/\sqrt{d}\right)$ where $\boldsymbol{W}_q^p$ is the weight matrix corresponding to the receiving node and $\boldsymbol{W}_k^p$ is the weight matrix corresponding to the sending node in the message passing step p.

To allow the MPNN to learn a diverse set of attention scores, we apply H scaled dot-product self-attention heads in every message passing step and concatenate their results. To this end, instead of using single weight matrices $\boldsymbol{W}_q^p, \boldsymbol{W}_k^p$ and $\boldsymbol{W}^p$, we use different weight matrices $\boldsymbol{W}_q^{p,h} \in \mathbb{R}^{d/H\times d}, \boldsymbol{W}_k^{p,h} \in \mathbb{R}^{d/H\times d}$ and $\boldsymbol{W}^{p,h} \in \mathbb{R}^{d/H\times d}$ for each attention head:

$$v_i^{p+1} = \text{cat}\left(\sum_{j\in N_i}\lambda_{ij}^{p,1}\boldsymbol{W}^{p,1}v_j^p, \ldots, \sum_{j\in N_i}\lambda_{ij}^{p,H}\boldsymbol{W}^{p,H}v_j^p\right)$$

where cat is the concatenation operator. Note that by using attention-head specific weight matrices, we reduce the dimensionality of all embeddings v_j^p by ${}^1\!/_H$. This is so that when the embeddings generated by all attention heads are concatenated, the resulting embedding v_i^{p+1} has the same dimension as the input embedding v_i^p.

3.3. Self-Supervised Pre-Training (SAMP-CLR)

The fact that we do not have access to the true class labels of the training data underscores the need to use a self-supervised pre-training scheme. As discussed briefly in Section 1, we build on the idea of employing contrastive learning for prototypical transfer learning with some inspiration from [33, 41]. Standard contrastive learning enforces embeddings of augmented images to be close to the embeddings of their source images in the representation space. The **key idea** of SAMP-CLR is not only to perform contrastive

learning on the source and augmented image embeddings but also to ensure that images in the mini-batch belonging to, potentially, the same classes have similar embeddings by enabling the model to look beyond single instances and their augmentations. This in turn allows the model to extract richer semantic information across multiple classes present in a mini-batch. To ensure that the SAMP layers learn to look for similar images across multiple instances in a mini-batch, we apply a contrastive loss on the SAMP refined features (generated by f_ψ), and on the standard convolutional features (generated by f_θ). Let us walk you through the process in more detail.

Algorithm 1 begins with **batch generation** (lines 2 to 7): each mini-batch consists of L random samples $\{\boldsymbol{x}_i\}_{i=1}^{L}$ from $\mathcal{D}_{\text{tr}}$, where $\boldsymbol{x}_i$ is treated as a 1 shot support sample for which we create A randomly augmented versions $\tilde{\boldsymbol{x}}_{i,a}$ as query samples (line 5), leading to a batch size of $B = (Q+1)L$. Then embeddings $\boldsymbol{z}$ and $\tilde{\boldsymbol{z}}$ are generated (lines 8 to 9) by passing the source images and augmented images, respectively, through a feature extraction network f_θ. The $\texttt{Graph}(\cdot)$ module (lines 10 to 11) then builds the graph $G = \texttt{Graph}(\boldsymbol{v}, \boldsymbol{e}, \alpha)$ where $\boldsymbol{v}$ is a node list made up of source image embeddings $\boldsymbol{z}$ and augmented image embeddings $\tilde{\boldsymbol{z}}$, $\boldsymbol{e}$ is the list of centered cosine similarities $d'[\cdot,\cdot]$ between all pairs of connected nodes, and α is a threshold that determines whether or not two nodes will be connected. The graph G, is passed through the SAMP layer(s) f_ψ (line 12) to allow for feature refinement that gives the refined node features $\boldsymbol{v}'$. This is still in the form of a node list and must be spliced (lines 13 - 14) appropriately to access the updated source image and augmented image embeddings, $\boldsymbol{z}'$ and $\tilde{\boldsymbol{z}}'$, respectively. We then apply contrastive losses (lines 15 to 18) $\mathcal{L}_1$ (between $\boldsymbol{z}$ and $\tilde{\boldsymbol{z}}$), as well as $\mathcal{L}_2$ (between $\boldsymbol{z}'$ and $\tilde{\boldsymbol{z}}'$). Here, $\mathcal{L}_1$ is the loss applied to the convolutional feature extractor, while $\mathcal{L}_2$ is the loss applied to the SAMP projector. $\mathcal{L}_1$ encourages the feature extractor to cluster embeddings of augmented query samples $\tilde{\boldsymbol{z}}$ around their prototypes (source embeddings) $\boldsymbol{z}$, which in turn provides a good and appropriate initial set of embeddings for the SAMP projector module to refine. $\mathcal{L}_2$ enforces the same constraints as $\mathcal{L}_1$ but for the embeddings generated by the SAMP layer. It enforces the SAMP projector to bring augmented query samples $\tilde{\boldsymbol{z}}'$ around their prototypes (source embeddings) $\boldsymbol{z}'$ while updating features based on their neighborhood in a graph. Both loss terms use a Euclidean distance metric in the embedding space denoted by $d[\cdot,\cdot]$. Finally, the complete loss $\mathcal{L} = \beta\mathcal{L}_1 + \mathcal{L}_2$, where β is the scaling factor for $\mathcal{L}_1$, is optimized with mini-batch stochastic gradient descent with respect to all parameters $\Omega = \{\theta, \psi\}$.

3.4. Supervised Fine-tuning (OpT-Tune)

Our supervised fine-tuning scheme is a two-stage process that makes use of a transportation stage followed by a prototypical fine-tuning and classification stage.

Optimal Transport Based Feature Alignment. We provide a basic intuition for optimal transport (OT); however, a more in-depth analysis and a thorough explanation can be found in [12, 36].

Let r and c be two probability vectors and let $\boldsymbol{\Pi}(r,c)$ be the set of nonnegative $NK \times NQ$ matrices for which all rows sum up to r and all columns sum up to c. If both r and c are accessible only through a finite set of samples $(r_1, \ldots, r_{NK})$ and $(c_1, \ldots, c_{NQ})$ respectively, we can write $\boldsymbol{\Pi}(r,c)$ as a *polytope* of r and c, representing a polyhedral set of $NK \times NQ$ matrices,

$$\boldsymbol{\Pi}(r,c) = \left\{\pi \in \mathbb{R}_{+}^{NK \times NQ} \mid \pi \mathbf{1}_{NQ} = r, \pi^\top \mathbf{1}_{NK} = c\right\}. \quad (1)$$

$\boldsymbol{\Pi}(r,c)$ is then essentially a collection of transport plans of which some are better than others. Our goal is to find the most optimal plan that can transport NK supports (r) to NQ queries (c). As given in [12] there is a probabilistic interpretation of $\boldsymbol{\Pi}(r,c)$: for A and B two multinomial random variables taking values in $\{1, \ldots, NK\}$ and $\{1, \ldots, NQ\}$, each with distribution r and c, respectively, the set $\boldsymbol{\Pi}(r,c)$ contains all possible *joint probabilities* of (A, B). Any matrix $\pi \in \boldsymbol{\Pi}(r,c)$ can be identified with a joint probability for (A, B) such that $p(A = i, B = j) = p_{ij}$.

Optimal Transport Between r and c. Given a cost matrix M, the cost of mapping r to c using a transport matrix (or joint probability) π can be quantified as $\langle \pi, M \rangle_F$. The problem in eq. (2) is then called an *optimal transport* (OT) problem between r and c given the cost matrix M

$$\boldsymbol{\pi}^\star = \underset{\pi \in \boldsymbol{\Pi}(r,c)}{\operatorname{argmin}} \langle \pi, M \rangle_F + \epsilon H(\boldsymbol{\pi}) \quad (2)$$

where $\boldsymbol{\pi}^\star$ is the optimal transportation plan and $\langle \cdot, \cdot \rangle_F$ is the Frobenius dot product. In our formulation, we use the Euclidean distance metric $d(z_i, z_j) = ||z_i - z_j||_2$ in the representation space to calculate the cost matrix M, where $M(i,j) = d(r_i, c_j)$. Equation (2) is then solved using the Sinkhorn-Knopp algorithm [12], which allows us to find an optimal plan $\boldsymbol{\pi}^\star$ in a reasonable amount of time. Sinkhorn-Knopp works on doubly stochastic matrices, π is doubly stochastic as stated in eq. (2). The optimum, $\boldsymbol{\pi}^\star$, is called *Wasserstein Metric*. From a probabilistic perspective, it is a distance between two probability distributions, sometimes also referred to as *earth mover distance* since it can be interpreted as how much "dirt" one must move to change one "landscape" (distribution) into another.

Transportation Stage. During testing, we receive an episode consisting of the support set $\mathcal{S}$ and the query set $\mathcal{Q}$ in the input space. Their representations generated from the pre-trained feature extractor f_Ω, are denoted as $\mathcal{Z}_s$ and $\mathcal{Z}_q$ *i.e.*, $\mathcal{Z}_s = f_\Omega(\mathcal{S})$ and $\mathcal{Z}_q = f_\Omega(\mathcal{Q})$. To adapt the support set $\mathcal{S}$ to the query set $\mathcal{Q}$ with cost matrix $M = d(\mathcal{Z}_s, \mathcal{Z}_q)$, we compute $\hat{\mathcal{Z}}_s$ the *projected mapping* of $\mathcal{Z}_s$ as follows:

$$\hat{\mathcal{Z}}_s = \hat{\boldsymbol{\pi}}^\star \mathcal{Z}_q \quad (3)$$

where $\hat{\boldsymbol{\pi}}^\star$ is the normalized transport plan:

$$\hat{\boldsymbol{\pi}}^\star(i,j) = \boldsymbol{\pi}^\star(i,j) / \sum_j \boldsymbol{\pi}^\star(i,j). \tag{4}$$

The *projected support* set $\hat{\mathcal{Z}}_s$ is an estimation of the support set in the query set's domain - in other words, it is a *barycentric mapping* of the support features $\mathcal{Z}_s$. Algorithm 2 shows this very process in a succinct manner.

Prototype Generation, Fine-tuning and Classification. The projected supports are used for the classification of the query points. To this end, following [33, 44] we concatenate f_Ω with a single layer nearest mean classifier (resulting in an architecture similar to ProtoNet [42]) f_ϕ and only fine-tune this last layer. In this stage, for each class $k \in \mathcal{C}$ in the support set, we compute the class prototype $\boldsymbol{c}_k$ for class k using the projected support set $\hat{\mathcal{Z}}_s$:

$$\boldsymbol{c}_k = {}^1/\left|\hat{\mathcal{Z}}_{s\,k}\right| \sum_{\hat{z} \in \hat{\mathcal{Z}}_{s\,k}} \hat{z}, \text{ for } k \in \mathcal{C}.$$

Following [33, 44], we initialize the classification layer f_ϕ with weights set to $\boldsymbol{W}_n = 2\boldsymbol{c}_k$ and biases set to $b_n = -\|\boldsymbol{c}_n\|^2$. To finetune this layer, we sample a subset of supports from the support set $\mathcal{S}$ and train f_ϕ with a standard cross-entropy loss; more details in Section 4.

4. Experimental Setup

In this section, we first discuss our experimental setup, and then we discuss our numerical results.

Datasets. We conduct several in-domain experiments to benchmark the performance of `SAMPTransfer`. For this purpose, we make use of several commonly adopted datasets, namely, *mini*-ImageNet [47] and *tiered*ImageNet [38] to compare against several other unsupervised few-shot learning methods. *Mini*-ImageNet contains 100 classes with 600 samples in each class. This equates to a total of $60,000$ images that we resize to 84×84 pixels. Of the 100 classes, we use 64 classes for training, 16 for validation, and 20 for testing. *Tiered*ImageNet is a larger subset of ILSVRC-12 [14] with 608 classes with a total of $779,165$ images of size 84×84. Of the 608 classes, we use 351 for training, 97 for validation, and 8 for testing. The augmentations in use during pre-training follow [2].

We also compare our method on the more challenging cross-domain few-shot learning (CDFSL) benchmark [20], which consists of several datasets. This benchmark has four datasets with increasing similarities to *mini*-ImageNet. In that order, we have grayscale chest X-ray images from ChestX [50], dermatological skin lesion images from ISIC2018 [10], aerial satellite images from EuroSAT [22] and crop disease images from CropDiseases [34]. We also used the Caltech-UCSD Birds (CUB) dataset [48] for further analysis of cross-domain performance. The CUB dataset is made up of $11,788$ images from 200 unique species of birds. We use 100 classes for training, 50 for both validation and testing.

Training. The Conv4 model follows [47] and has a single SAMP layer with four attention heads. We also use an alternate form of the Conv4 network where we change the number of filters from $(64, 64, 64, 64)$ to $(96, 128, 256, 512)$ and average pool the final feature map, giving us an embedding dimension of 512 instead of the standard 1600. We call this network Conv4b. It is pre-trained using `SAMP-CLR` on the respective training splits of the datasets, with an initial learning rate of 0.0005, annealed by a cosine scheduler via the Adam optimizer [28]. Experiments involving CDFSL benchmark follow [20, 33, 41], where we pre-train a ResNet-10 encoder using `SAMP-CLR` on *mini*-ImageNet images of size 224×224 for 400 epochs with the Adam optimizer and a constant learning rate of 0.0001. Similar to the Conv4(b) encoder, the ResNet-10 encoder also has a single SAMP layer with four attention heads.

The ResNet-12 architecture follows [30, 35, 43]. The network consists of 4 residual blocks, where each has 3 convolutional layers with a 3×3 kernel; a 2×2 max-pooling layer is applied after each of the first 3 blocks, and a global average-pooling layer is on top of the fourth block to generate embeddings. Similar to [30, 43] we make use of DropBlock regularization [18] and change the number of filters from $(64, 128, 256, 512)$ to $(64, 160, 320, 640)$.

During testing and validation, as defined in Section 3.4, we initialize and fine-tune a classifier layer for 15 iterations. For validation, we create 15 (N-way, K-shot) tasks using the validation split from which the validation accuracy and loss are calculated.

Evaluation scenarios and baseline. Our testing scheme uses 600 test episodes on which the pre-trained encoder (using `SAMP-CLR`) is fine-tuned (using `OpT-Tune`) and tested. All our results indicate 95% confidence intervals over 3 runs, each with 600 test episodes. Therefore, the standard deviation values are calculated according to the 3 runs to provide more concrete measures for comparison. For our in-domain benchmarks, we test on (5-way, 1-shot) and (5-way, 5-shot) classification tasks, while our cross-domain testing is done using (5-way, 5-shot) and (5-way, 20-shot) classification tasks following [20]. We compare our performance with a suite of recent unsupervised few-shot baselines such as U-MISo [58], C^{3}LR [41], Meta-GMVAE [29], and Revisiting UML [54] to name a few. Furthermore, we also compare with a set of supervised approaches (such as MetaQDA [59] and SimpleCNAPS [3]), the best performing of which are obviously expected to outperform ours and other unsupervised methodologies; however, we are reasonably close to their numbers.

Table 1: Accuracy (%± std.) for (N-way, K-shot) classification tasks. Style: **best** and second best.

Method(N, K)	Backbone	*mini*-ImageNet (5,1)	*mini*-ImageNet (5,5)
CACTUs-MAML [23]	Conv4	39.90 ± 0.74	53.97 ± 0.70
CACTUs-Proto [23]	Conv4	39.18 ± 0.71	53.36 ± 0.70
UMTRA [25]	Conv4	39.93	50.73
AAL-ProtoNet [1]	Conv4	37.67 ± 0.39	40.29 ± 0.68
AAL-MAML++ [1]	Conv4	34.57 ± 0.74	49.18± 0.47
UFLST [24]	Conv4	33.77 ± 0.70	45.03 ± 0.73
ULDA-ProtoNet [37]	Conv4	40.63 ± 0.61	55.41 ± 0.57
ULDA-MetaNet [37]	Conv4	40.71 ± 0.62	54.49 ± 0.58
U-SoSN+ArL [57]	Conv4	41.13 ± 0.84	55.39 ± 0.79
U-MlSo [58]	Conv4	41.09	55.38
ProtoTransfer [33]	Conv4	45.67 ± 0.79	62.99 ± 0.75
CUMCA [52]	Conv4	41.12	54.55
Meta-GMVAE [29]	Conv4	42.82	55.73
Revisiting UML [54]	Conv4	48.12 ± 0.19	65.33 ± 0.17
CSSL-FSL_Mini64 [31]	Conv4	48.53 ± 1.26	63.13 ± 0.87
C^{3}LR [41]	Conv4	47.92 ± 1.2	64.81 ± 1.15
`SAMPTransfer` (**ours**)	Conv4	55.75 ± 0.77	67.62 ± 0.66
`SAMPTransfer`* (**ours**)	Conv4b	**61.02** ± 1.0	**72.52** ± 0.68
Supervised Methods			
MAML [17]	Conv4	46.81 ± 0.77	62.13± 0.72
ProtoNet [42]	Conv4	46.44± 0.78	66.33± 0.68
MMC [38]	Conv4	50.41 ± 0.31	64.39 ± 0.24
FEAT [55]	Conv4	55.15	71.61
SimpleShot [51]	Conv4	49.69 ± 0.19	66.92 ± 0.17
Simple CNAPS [3]	ResNet-18	53.2 ± 0.9	70.8 ± 0.7
Transductive CNAPS [3]	ResNet-18	55.6 ± 0.9	73.1 ± 0.7
MetaQDA [59]	Conv4	56.41 ± 0.80	72.64 ± 0.62
Pre+Linear [33]	Conv4	43.87 ± 0.69	63.01 ± 0.71

Table 2: Accuracy (%± std.) for (N-way, K-shot) classification tasks. Style: **best** and second best.

Method(N, K)	*tiered*ImageNet (5,1)	*tiered*ImageNet (5,5)
C^{3}LR [41]	42.37 ± 0.77	61.77 ± 0.25
ULDA-ProtoNet [37]	41.60 ±0.64	56.28 ± 0.62
ULDA-MetaOptNet [37]	41.77 ± 0.65	56.78 ± 0.63
U-SoSN+ArL [57]	43.68 ± 0.91	58.56 ± 0.74
U-MlSo [58]	43.01 ± 0.91	57.53 ± 0.74
`SAMPTransfer` (**ours**)	45.25 ± 0.89	59.75 ± 0.66
`SAMPTransfer`* (**ours**)	**49.10** ± 0.94	**65.19** ± 0.82

5. Performance Evaluation

In-Domain Experiments. Table 1 summarizes our performance evaluation results on the *mini*-ImageNet dataset for (N-way, K-shot) scenarios with $N = 5$ and $K = 1, 5$. The top section compares the performance of the proposed approach (`SAMP-CLR`) with the most recent relevant unsupervised competitors. We outperform our closest competitors by about $7+\%$ and $2+\%$ in the (5-way, 1-shot) and (5-way, 5-shot) settings, respectively. More interestingly, our method matches or beats some of the supervised baselines (bottom section of the table) adopting a similar encoder architecture. Naturally, state-of-the-art supervised few-shot learning approaches have the advantage of accessing all the true labels and, thus, are always at an advantage. When it comes to *tiered*ImageNet, our approach using encoders identical to those previously used in Table 2, shows significant gains over the latest competitors such as U-MlSo [58] with a $6+\%$ improvement in both the (5-way 1 and 5 shot) settings.

Cross-Domain Experiments. So far, we have demonstrated that the proposed approach excels for in-domain scenarios. The next step is to assess the performance under more challenging cross-domain scenarios (Table 3) where we pre-train on a particular dataset in an unsupervised fashion and then fine-tune and test on a different dataset.

We focus on the CDFSL benchmark [20] to investigate the performance of `SAMPTransfer` in cross-domain scenarios. Here, we pre-train on *mini*-ImageNet and fine-tune on ChestX [50], ISIC2018 [10], EuroSAT [22], and CropDiseases [34]. We compare the performance against C^{3}LR[41], ProtoTransfer [33] along with its two variants using UMTRA [33] (also proposed in [33]), as well as ConFeSS [13] and ATA [49] - two of the latest methods dedicated to solving the cross-domain few-shot learning problem. Please note that we also compare with a couple of related supervised approaches from [20] as a reference. Our method consistently trades blows with ConFeSS [13], but scores higher in CropDiseases 5 and 20 shot tasks by $2+\%$ and about 1%, respectively. Except for EuroSAT, our method is consistently competitive ($\sim 1\%$ difference in accuracy) to the performance of ConFeSS in ChestX and ISIC. In ISIC, which is the second least similar dataset to *mini*-ImageNet, our method is better by $1+\%$ in the (5-way, 20-shot) setting. Note that `SAMPTransfer` outperforms another recent method ATA [49] in all CDFSL benchmark settings.

6. Ablation Study and Robustness Analysis

Table 4 shows the performance characteristics of the proposed method with various hyperparameters. In this section, we use the (5-way, 5-shot) *mini*-ImageNet benchmark to analyze the robustness of our method and demonstrate the importance of our design choices.

`OpT-Tune` is crucial. To show the effect on performance of using `OpT-Tune`, we perform experiments with `OpT-Tune` disabled. For a fair comparison, we use the same pre-trained models in the test runs with `OpT-Tune` enabled or disabled. The best performing model (a Conv4b) uses 1 SAMP layer with 4 attention heads and a reasonable batch size of 64, resulting in a score of 71.42% with `OpT-Tune` enabled. The same model, with `OpT-Tune` disabled, loses $7+\%$ accuracy. Even with `OpT-Tune` disabled, our method remains competitive with some of the

Table 3: Accuracy (%± std.) of (N-way, K-shot) classification on the CDFSL benchmark. Style: **best** and second best.

Method(N, K)	(5,5)	(5,20)	(5,5)	(5,20)	(5,5)	(5,20)	(5,5)	(5,20)
	ChestX		**ISIC**		**EuroSAT**		**CropDiseases**	
UMTRA-ProtoNet [33]	24.94 ± 0.43	28.04 ± 0.44	39.21 ± 0.53	44.62 ± 0.49	74.91 ± 0.72	80.42 ± 0.66	79.81 ± 0.65	86.84 ± 0.50
UMTRA-ProtoTune [33]	25.00 ± 0.43	30.41 ± 0.44	38.47 ± 0.55	51.60 ± 0.54	68.11 ± 0.70	81.56 ± 0.54	82.67 ± 0.60	92.04 ± 0.43
ProtoTransfer [33]	26.71 ± 0.46	33.82 ± 0.48	45.19 ± 0.56	59.07 ± 0.55	75.62 ± 0.67	86.80 ± 0.42	86.53 ± 0.56	95.06 ± 0.32
C^3LR [41]	26.00 ± 0.41	33.39 ± 0.47	45.93 ± 0.54	59.95 ± 0.53	80.32 ± 0.65	88.09 ± 0.45	87.90 ± 0.55	95.38 ± 0.31
`SAMPTransfer` (**ours**)	26.27 ± 0.44	**34.15** ± 0.50	47.60 ± 0.59	**61.28** ± 0.56	81.58 ± 0.63	88.52 ± 0.50	**91.74** ± 0.55	**96.36** ± 0.28
ConFeSS [13] (dedicated)	**27.09**	33.57	**48.85**	60.10	**84.65**	**90.40**	88.88	95.34
ATA [49] (dedicated)	24.43 ± 0.2	-	45.83 ± 0.3	-	83.75 ± 0.4	-	90.59 ± 0.3	-
ProtoNet [20] (sup.)	24.05 ± 1.01	28.21 ± 1.15	39.57 ± 0.57	49.50 ± 0.55	73.29 ± 0.71	82.27 ± 0.57	79.72 ± 0.67	88.15 ± 0.51
Pre+Mean-Cent. [20] (sup.)	26.31 ± 0.42	30.41 ± 0.46	47.16 ± 0.54	56.40 ± 0.53	82.21 ± 0.49	87.62 ± 0.34	87.61 ± 0.47	93.87 ± 0.68
Pre+Linear [20] (sup.)	25.97 ± 0.41	31.32 ± 0.45	48.11 ± 0.64	59.31 ± 0.48	79.08 ± 0.61	87.64 ± 0.47	89.25 ± 0.51	95.51 ± 0.31

Table 4: Effect of various parameters on accuracy (%± std.) over 600 (5-way, 5-shot) *mini*ImageNet test tasks.

Backbone	p	H	L	β	**OT**	**Accuracy**
Conv4b	1	4	64	1.0	✓	71.42 ± 0.73
Conv4b	1	4	64	0.7	✓	71.41 ± 0.71
Conv4b	1	8	64	1.0	✓	71.27 ± 0.75
Conv4b	1	8	64	0.7	✓	69.87 ± 0.72
Conv4b	2	1	64	0.7	✓	68.99 ± 0.71
Conv4b	2	4	64	0.7	✓	67.01 ± 0.69
Conv4	1	4	64	0.7	✓	69.61 ± 0.71
Conv4	1	4	64	1.0	✓	67.60 ± 0.62
Conv4	1	8	64	1.0	✓	63.59 ± 0.68
Conv4b	1	4	128	0.7	✓	72.52 ± 0.72
Conv4	1	4	128	0.7	✓	68.33 ± 0.71
Conv4b	1	4	64	0.7	✗	64.29 ± 0.63
Conv4b	1	4	128	0.7	✗	63.47 ± 0.64
Conv4	1	4	64	0.7	✗	66.73 ± 0.65

Table 5: Accuracy (%± std.) for (N-way, K-shot) classification on *mini*ImageNet with pre-training on CUB.

Training	**Testing**	(5,1)	(5,5)
ProtoTransfer [33]	ProtoTune	35.37 ± 0.63	52.38 ± 0.66
C^3LR [41]	ProtoTune	39.61 ± 1.11	55.53 ± 1.42
`SAMPTransfer` (**ours**)	OpT-Tune	**49.32** ± 0.75	**56.10** ± 0.60

latest methods in Table 1. This observation suggests that projecting the support embeddings in the domain of query embeddings is an efficient and straightforward method to incorporate task awareness and improve the quality of embeddings and, in turn, prototypes.

SAMP Layers and Attn. Heads. In Table 4, we also investigate the robustness of our method when the number of SAMP layers (p) and attention heads (H) vary. The best performance is achieved with a single SAMP layer with four attention heads. We see that increasing p can lead to a significant decrease in performance; however, increasing H leads to a small drop in performance. The observation made here is consistent with the observations made in [5, 46].

Embedding Dimension. We measure the performance of the model in relation to two commonly used embedding dimensions, 512 and 1600. As seen in Table 4, we observe that the network generally performs best with an embedding dimension of 512. Performance is significantly lower with an embedding dimension of size 1600. We hypothesize that this behavior can be attributed to the lower number of channels in the final feature map in a standard Conv4 network - limited to 64.

β and $\mathcal{L}_1$. The scaling factor β does not appear to be as important as the mere presence of $\mathcal{L}_1$ in the final loss $\mathcal{L}$. We observe that β values greater than 0.5 tend to have a minimal effect on performance. However, our best-performing models make use of $\beta = 0.7$, which appears to be the sweet spot regardless of the embedding dimension.

Cross-Domain Robustness. To further analyze the cross-domain performance characteristics of our method, in addition to Table 3, we trained a Conv4 model on CUB and tested it on tasks derived from *mini*-ImageNet. CUB consists of 200 classes of only birds, while *mini*-ImageNet consists of 64 classes, of which only 3 training classes are birds. Thus, CUB has a diminished class diversity compared to *mini*-ImageNet. Table 5 shows that our method retains better transfer accuracy than other competing methods when training classes are diversity constrained.

7. Conclusion

Inspired by the human mind's capacity to naturally make relevant connections between entities, we developed a contrastive learning scheme for unsupervised few-shot classification, where we supplemented a CNN's strong inductive prior with MPNNs to exploit intra-batch relations between images. Furthermore, we also showed that an optimal-transport-based task-awareness algorithm can aid in elevating the robustness of pre-trained models. We show that our approach (`SAMPTransfer`) offers appreciable performance improvements over its competitors in both in/cross-domain few-shot classification scenarios, setting new standards in the *mini*ImageNet, *tiered*ImageNet and CDFSL benchmarks.

References

[1] Antreas Antoniou and Amos Storkey. Assume, augment and learn: Unsupervised few-shot meta-learning via random labels and data augmentation. *arXiv preprint arXiv:1902.09884*, 2019. 1, 2, 7

[2] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019. 6

[3] Peyman Bateni, Jarred Barber, Jan-Willem van de Meent, and Frank Wood. Enhancing few-shot image classification with unlabelled examples. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2796–2805, 2022. 1, 6, 7

[4] Malik Boudiaf, Imtiaz Ziko, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. Information maximization for few-shot learning. *Advances in Neural Information Processing Systems*, 33:2445–2457, 2020. 1

[5] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021. 4, 8

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020. 2

[7] Da Chen, Yuefeng Chen, Yuhong Li, Feng Mao, Yuan He, and Hui Xue. Self-supervised learning for few-shot image classification. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1745–1749. IEEE, 2021. 1

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[9] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 2

[10] Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019. 6, 7

[11] Wentao Cui and Yuhong Guo. Parameterless transductive feature re-representation for few-shot learning. 2021. 1

[12] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013. 5

[13] Debasmit Das, Sungrack Yun, and Fatih Porikli. ConfeSS: A framework for single source cross-domain few-shot learning. In *International Conference on Learning Representations*, 2022. 2, 7, 8

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[15] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019. 1

[16] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9588–9597, 2021. 2

[17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 1, 7

[18] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31, 2018. 6

[19] 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. 2

[20] Yunhui Guo, Noel CF Codella, Leonid Karlinsky, John R Smith, Tajana Rosing, and Rogerio Feris. A New Benchmark for Evaluation of Cross-Domain Few-Shot Learning. *arXiv preprint arXiv:1912.07200*, 2019. 6, 7, 8

[21] 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*, pages 9729–9738, 2020. 2

[22] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification, 2017. 6, 7

[23] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. *arXiv preprint arXiv:1810.02334*, 2018. 1, 2, 7

[24] Zilong Ji, Xiaolong Zou, Tiejun Huang, and Si Wu. Unsupervised few-shot learning via self-supervised training. *arXiv preprint arXiv:1912.12178*, 2019. 7

[25] Siavash Khodadadeh, Ladislau Boloni, and Mubarak Shah. Unsupervised meta-learning for few-shot image classification. *Advances in neural information processing systems*, 32, 2019. 1, 7

[26] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11–20, 2019. 1, 2

[27] Jinwoo Kim, Tien Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *arXiv preprint arXiv:2207.02505*, 2022. 4

[28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[29] Dong Bok Lee, Dongchan Min, Seanie Lee, and Sung Ju Hwang. Meta-gmvae: Mixture of gaussian vae for unsupervised meta-learning. In *ICLR*, 2021. 1, 6, 7

[30] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex

optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 6

[31] Jianyi Li and Guizhong Liu. Few-shot image classification via contrastive self-supervised learning. *arXiv preprint arXiv:2008.09942*, 2020. 7

[32] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018. 2

[33] Carlos Medina, Arnout Devos, and Matthias Grossglauser. Self-supervised prototypical transfer learning for few-shot classification. *arXiv preprint arXiv:2006.11325*, 2020. 1, 2, 4, 6, 7, 8

[34] Sharada P Mohanty, David P Hughes, and Marcel Salathé. Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science*, 7:1419, 2016. 6, 7

[35] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31, 2018. 6

[36] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019. 5

[37] Tiexin Qin, Wenbin Li, Yinghuan Shi, and Yang Gao. Diversity helps: Unsupervised few-shot learning via distribution shift-based data augmentation. *arXiv preprint arXiv:2004.05805*, 2020. 7

[38] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018. 6, 7

[39] Victor Garcia Satorras and Joan Bruna Estrach. Few-shot learning with graph neural networks. In *International Conference on Learning Representations*, 2018. 1, 2

[40] Jenny Denise Seidenschwarz, Ismail Elezi, and Laura Leal-Taixé. Learning intra-batch connections for deep metric learning. In *International Conference on Machine Learning*, pages 9410–9421. PMLR, 2021. 4

[41] Ojas Kishore Shirekar and Hadi Jamali-Rad. Self-supervised class-cognizant few-shot classification. *arXiv preprint arXiv:2202.08149*, 2022. 1, 4, 6, 7, 8

[42] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 6, 7

[43] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? In *European Conference on Computer Vision*, pages 266–282. Springer, 2020. 1, 2, 6

[44] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. *arXiv preprint arXiv:1903.03096*, 2020. 6

[45] 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. 4

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. 4, 8

[47] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016. 3, 6

[48] 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. 6

[49] Haoqing Wang and Zhi-Hong Deng. Cross-domain few-shot classification via adversarial task augmentation. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1075–1081. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track. 7, 8

[50] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2097–2106, 2017. 6, 7

[51] Yan Wang, Wei-Lun Chao, Kilian Q Weinberger, and Laurens van der Maaten. Simpleshot: Revisiting nearest-neighbor classification for few-shot learning. *arXiv preprint arXiv:1911.04623*, 2019. 7

[52] Hui Xu, Jiaxing Wang, Hao Li, Deqiang Ouyang, and Jie Shao. Unsupervised meta-learning for few-shot learning. *Pattern Recognition*, 116:107951, 2021. 7

[53] Ling Yang, Liangliang Li, Zilun Zhang, Xinyu Zhou, Erjin Zhou, and Yu Liu. Dpgn: Distribution propagation graph network for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13390–13399, 2020. 1

[54] Han-Jia Ye, Lu Han, and De-Chuan Zhan. Revisiting Unsupervised Meta-Learning via the Characteristics of Few-Shot Tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2022. 1, 6, 7

[55] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8808–8817, 2020. 1, 7

[56] Tianyuan Yu, Sen He, Yi-Zhe Song, and Tao Xiang. Hybrid graph neural networks for few-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3179–3187, 2022. 1, 2

[57] Hongguang Zhang, Piotr Koniusz, Songlei Jian, Hongdong Li, and Philip HS Torr. Rethinking class relations: Absolute-relative supervised and unsupervised few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9432–9441, 2021. 7

[58] Hongguang Zhang, Hongdong Li, and Piotr Koniusz. Multi-level second-order few-shot learning. *IEEE Transactions on*

Multimedia, pages 1–1, 2022. 6, 7

[59] Xueting Zhang, Debin Meng, Henry Gouk, and Timothy M. Hospedales. Shallow bayesian meta learning for real-world few-shot recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 651–660, October 2021. 6, 7

[60] Imtiaz Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed. Laplacian regularized few-shot learning. In *International conference on machine learning*, pages 11660–11670. PMLR, 2020. 1