LEARNING TO AFFILIATE: MUTUAL CENTRALIZED LEARNING FOR FEW-SHOT CLASSIFICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

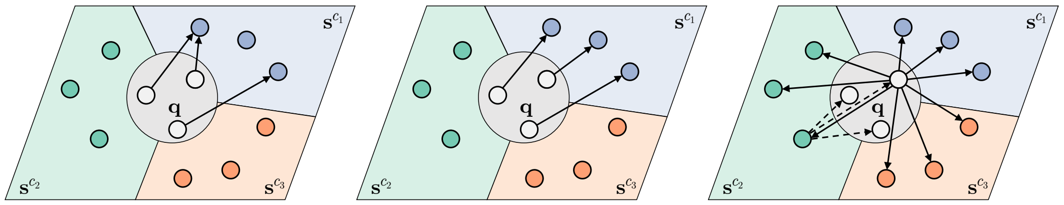
Few-shot learning (FSL) aims to learn a classifier that can be easily adapted to accommodate new tasks not seen during training, given only a few examples. To handle the limited-data problem in few-shot regimes, recent methods tend to collectively use a set of local features to densely represent an image instead of using a mixed global feature. They generally explore a unidirectional query-to-support paradigm in FSL, *e.g.*, find the nearest/optimal support feature for each query feature and aggregate these local matches for a joint classification. In this paper, we propose a new method *Mutual Centralized Learning* (MCL) to fully affiliate the two disjoint sets of dense features in a bidirectional paradigm. We associate each local feature with a particle that can bidirectionally random walk in a discrete feature space by the affiliations. To estimate the class probability, we propose the features' accessibility that measures the expected number of visits to the support features of that class in a Markov process. We relate our method to learning a centrality on an affiliation network and demonstrate its capability to be plugged in existing methods by highlighting centralized local features. Experiments show that our method achieves the state-of-the-art on both *mini*ImageNet and *tiered*ImageNet.

1 INTRODUCTION

Few-shot classification aims to learn a classifier that can be readily adapted to novel classes given just a small number of labeled instances. To address this problem, a line of previous literature adopts metric-based methods (Vinyals et al., 2016; Snell et al., 2017; Sung et al., 2018) that learn a global image representation in an appropriate feature space and use a distance metric to predict their labels.

Recent approaches (Zhang et al., 2020; Li et al., 2019; Lifchitz et al., 2019) have demonstrated that the significant intra-class variations would inevitably drive the image-level embedding from the same category far apart in a given metric space under low-data regimes. In contrast, densely representative local features can provide transferrable information across categories that have shown promising performances in the few-shot scenario. Among those methods illustrated in Figure 1, Li et al. (2019) find the nearest neighbor support feature for each query feature and accumulates all the local matches in a Naive-Bayes way to represent an image-to-class similarity; Zhang et al. (2020) use the earth mover distance to compare the complex structured representations composed of local features. They generally use the distances/similarities between two sets of dense features and perform classifications in a query-to-support paradigm by either aggregating those local nearest matches or introducing the optimal matching flows for each class.

In this work, we consider an extra support-to-query paradigm as a complement to fully affiliate two disjoint sets of dense features. The potential of mutual affiliations stems from the intuition that, except for using query features to find related support features, it is also plausible to estimate the relevance of query features according to the support features. To achieve this, we use the local similarity matrix as a basis to formulate the directional query-to-support and support-to-query affiliation matrices. We associate each local feature with a particle that could bidirectionally random walk in a discrete feature space by the mutual affiliations. The prediction probability for each class is then estimated by the features' accessibility (the expected number of visits to support features of that class) in a time-homogeneous Markov process. The motivation behind our method is that support features sharing the same class label with the query image would be frequently visited by their mutual closeness.



$\mathbf{Pr}_{\mathrm{NBNN}}(c \mid \mathbf{q})$
$\propto \exp\left(\sum_{q\in\mathbf{q}} \phi(q, \mathrm{NN}_{\mathbf{s}^c}(q))\right)$
(a) Naive-Bayes Nearest Neighbor

$\mathbf{Pr}_{\mathrm{EMD}}(c \mid \mathbf{q})$
$\propto \exp\left(\sum_{q\in\mathbf{q}} -d(q, \mathrm{EMD}_{\mathbf{s}^c}(q))\right)$
(b) Earth Mover's Distance

$\mathbf{Pr}_{\mathrm{MCL}}(c \mid \mathbf{q})$
$\propto \sum_{z\in\mathbf{q}\cup\mathbf{S}} \mathbb{E}\left[\sum_{t=1}^{\infty} \mathbb{1}[X_t \in \mathbf{s}^c] \middle| X_0 = z\right]$
(c) Bidirectional Random Walks

Figure 1: Comparisons of three methods in the 3-way 1-shot scenario where every colored pane indicates an image and particles within each pane represents the dense features of that image. The solid lines indicate a query-to-support paradigm while the dotted lines indicate a support-to-query paradigm. Among them, (a) Li et al. (2019) accumulates local similarities $\phi(\cdot,\cdot)$ between all query dense features and their nearest support dense features in a Naive-Bayes way as an image-to-class similarity; (b) Zhang et al. (2020) finds the optimal matching flow that have the minimum earth mover's distance $d(\cdot,\cdot)$; (c) We use the features' accessibility (*i.e.*, the expected number of visits to support features of each class) in a bidirectional Markov random walk $\{X_t\}$ to measure the image-to-class relevance. A major difference is that we consider the mutual affiliations between dense features instead of following the unidirectional query-to-support paradigm.

The contributions are as follows: (1) We propose to learn mutual affiliations between the query and support features instead of following the unidirectional query-to-support paradigm in FSL. (2) We introduce the features' accessibility to FSL and demonstrate that traditional transductive methods could be easily adapted to the inductive setting if we treat dense features from a single query image as a set of unlabeled data. (3) We propose a novel bidirectional random walks in FSL and draw its connection to the single-mode eigenvector centrality of an affiliation network (4) The underlying centrality investigated in this work can be plugged in existing global feature based methods like ProtoNet and RelationNet by highlighting centralized local features instead of average pooling.

2 Related Work

Dense Feature based FSL. A branch of method focuses on learning image-to-image similarities based on the dense features. Among them, DC (Lifchitz et al., 2019) proposes to make predictions for each local characteristic and average their output probabilities. DeepEMD (Zhang et al., 2020) adopts the earth mover's distance as a metric to compute a structural distance between dense features to determine the image relevance. DN4 (Li et al., 2019) uses the top-k nearest neighbors between two dense features in a Naive-Bayes way to represent image-level similarities. FRN (Wertheimer et al., 2021) reconstructs the dense query features from support dense features of a given class to predict their relevances. Our MCL is also among the family of dense feature based frameworks. A major difference is that we consider their mutual affiliations instead of the unidirectional match up.

Graphical models in FSL. Another branch of methods learns graphs on FSL. Among them, Garcia & Bruna (2017) build graph neural network (GNN) from a collection of images. Ding et al. (2020) use the degree-centrality as part of graph features to adjust the weight of graph vertex in transductive FSL. Our proposed feature accessibility of bidirectional random walks is well-related to the eigenvector centrality. Differently, we use the single-mode centrality of bipartite data as a noval criterion for the inductive classification.

From Transductive to Inductive. Liu et al. (2019) first introduced the well-known label spreading (Zhou et al., 2003) to the transductive FSL that transfers information from labeled data to unlabeled one, and shows substantial improvements over inductive methods. In this paper, we first demonstrate it feasible to adopt label spreading for inductive FSL by treating the query dense features as numerous unlabeled data and using features' accessibility as a criterion for a joint prediction. Different from label spreading, we further consider the set of query and support features as bipartite data where the self-reinforcements (*i.e.*, query-to-query and support-to-support random walks) are avoided.

3 Method

3.1 Problem Formulation

Let $\mathcal{S}$ denote the set of support images with corresponding class labels in a N-way K-shot episode, the goal is to predict a class label for a single query image $\mathcal{Q}$. We follow the recent dense feature based methods to learn r number of d-dimensional features $f_\theta(\cdot) \in \mathbb{R}^{d\times r}$ to represent a set of local characteristics for each image. We use $\mathbf{q} = \{q_1, ..., q_r\}$ to denote the set of dense features for a single query image and use the K-shot averaged features $\mathbf{s}^c = \{s^c_1, ..., s^c_r\}$ to represent the set of dense features from support class c. The bold font of $\{\mathbf{q}, \mathbf{s}\}$ indicates a set of dense features while normal font $\{q, s\}$ indicates a single channel-dimensional feature vector. We use $\mathbf{S} = \bigcup_{c\in C} \mathbf{s}^c$ to represent the union of features from all supporting classes.

3.2 Learning Mutual Affiliations through Bidirectional Random Walks

Our goal is to formulate and study an algorithm to find the label of a query image in a few-shot classification task, given dense query features $\mathbf{q}$ and the union set $\mathbf{S}$ of support features from all the classes. For simplicity of exposition, we denote $|\mathbf{q}| = r$, $|\mathbf{S}| = Nr$ as the cardinality of sets $\mathbf{q}$ and $\mathbf{S}$ respectively.

We start by defining local similarities between the two disjoint sets of dense features $\mathbf{q}, \mathbf{S}$. To be specific, given vectors v_1 and v_2 from the two feature sets respectively, we use the the *scaled cosine similarity* $\phi_\gamma(v_1, v_2) = \gamma\langle v_1/\|v_1\|, v_2/\|v_2\|\rangle$ to measure their closeness where $\langle\cdot,\cdot\rangle$ denotes the Frobenius inner product and $\gamma \in \mathbb{R}^+$ is a scaling parameter.

Following the query-to-support convention in previous work, we affiliate each query feature q to support features $s \in \mathbf{S}$ with a random walk probability $p_{sq} = \exp(\phi_\gamma(s, q))/\sum_{s'\in\mathbf{S}} \exp(\phi_\gamma(s', q))$ on the discrete feature space based on the relative closeness. The query-to-support probability matrix $\mathbf{P_{Sq}} = (p_{sq})_{s\in\mathbf{S}, q\in\mathbf{q}}$ that indicates the probabilities for each $q \in \mathbf{q}$ random walking to $s \in \mathbf{S}$ can be formulated by

$$\mathbf{P_{Sq}} = \mathbf{\Phi}_\gamma \mathbf{D}^{-1} \tag{1}$$

where $\mathbf{\Phi}_\gamma \in \mathbb{R}^{|\mathbf{S}|\times|\mathbf{q}|}$ is the exponential scaled similarity matrix with element entry $[\mathbf{\Phi}_\gamma]_{sq} = \exp(\phi_\gamma(s, q))$. $\mathbf{D}$ is a diagonal matrix with its (j, j)-value to be the sum of the j-th column of $\mathbf{\Phi}_\gamma$.

Unlike existing methods that simply consider the query-to-support matches in a unidirectional paradigm, we introduce an extra support-to-query relation as a complement to fully affiliate these two feature sets. Formally, we affiliate support feature s to query features $q \in \mathbf{q}$ by a similar random walk probability $p_{qs} = \exp(\phi_\tau(q, s))/\sum_{q'\in\mathbf{q}} \exp(\phi_\tau(q', s))$. The analogous support-to-query probability matrix $\mathbf{P_{qS}} = (p_{qs})_{q\in\mathbf{q}, s\in\mathbf{S}}$ is given by

$$\mathbf{P_{qS}} = \mathbf{\Phi}_\tau \mathbf{W}^{-1} \tag{2}$$

where $\mathbf{\Phi}_\tau \in \mathbb{R}^{|\mathbf{q}|\times|\mathbf{S}|}$ and $[\mathbf{\Phi}_\tau]_{qs} = \exp(\phi_\tau(q, s))$. $\mathbf{W}$ is the similar diagonal normalization matrix.

We consider a support feature random walks from $\mathbf{S}$ to $\mathbf{q}$ and then walks back to $\mathbf{S}$ as bidirectional random walks starting from $\mathbf{S}$. The probable locations for each support feature appearing at $\mathbf{S}$ after walking and walking back are given by each column of probability matrix $\mathbb{R}^{|\mathbf{S}|\times|\mathbf{S}|} \ni \mathbf{\Psi_S} = \mathbf{P_{Sq}}\mathbf{P_{qS}}$. By analogy, query features could also random walk from $\mathbf{q}$ to $\mathbf{S}$ and then walk back to appear at $\mathbf{q}$ with a similar probability matrix $\mathbb{R}^{|\mathbf{q}|\times|\mathbf{q}|} \ni \mathbf{\Psi_q} = \mathbf{P_{qS}}\mathbf{P_{Sq}}$.

Although the local similarity matrices $\mathbf{\Phi}_\gamma$ and $\mathbf{\Phi}_\tau$ would be symmetric if we ignore their different scaling parameters γ and τ, the probability matrices $\mathbf{P_{qS}}$ and $\mathbf{P_{Sq}}$ are both column-normalized and thus directional.

3.3 Estimating Class Labels by the Long-term Features' Accessibility

We attack the classification problem based on the bidirectional random walks that encodes the mutual affiliations between the two dense feature sets $\mathbf{q}, \mathbf{S}$. The motivation of our method is straightforward: support features would be frequently visited in the long term of bidirectional random walks if they shared the same class label with the query image. In other words, support class that owns most querying characteristics would be the predicted class of the query image due to their mutual closeness.

To formulate it, we associate each local feature with a particle z that is allowed to bidirectionally random walk in a discrete feature space $\mathbf{z} = \mathbf{q} \cup \mathbf{S}$ and assume these discrete random walks time-homogeneous in a Markov process $\{X_t\}$. Formally, the probability from z_j to z_i is

$$\mathbf{Pr}(X_{t+1} = z_i | X_t = z_j) = \begin{cases} \exp(\phi_\gamma(z_i, z_j)) / \sum_{s \in \mathbf{S}} \exp(\phi_\gamma(s, z_j)) & z_i \in \mathbf{S}, z_j \in \mathbf{q} \\ \exp(\phi_\tau(z_i, z_j)) / \sum_{q \in \mathbf{q}} \exp(\phi_\tau(q, z_j)) & z_i \in \mathbf{q}, z_j \in \mathbf{S} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and the probability for all particles in the discrete feature space is given by the entry of

$$\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{P_{Sq}} \\ \mathbf{P_{qS}} & \mathbf{0} \end{pmatrix} \tag{4}$$

where $\mathbf{P}$ is an anti-diagonal matrix that consists of the column-normalized sub-matrices $\mathbf{P_{Sq}}$ and $\mathbf{P_{qS}}$ defined in Eqn.(1) and Eqn.(2) respectively. The subscript of sub-matrices serves both as the matrix size and as an indication for the particle with which it is associated. For example, $\mathbf{P_{Sq}}$ is of size $|\mathbf{S}| \times |\mathbf{q}|$ and indicates the probabilities from query features $\mathbf{q}$ to support features $\mathbf{S}$.

It can be proved (in Appendix B) that the Markov chain with anti-diagonal transition matrix $\mathbf{P}$ is periodic and its stationary distributions are of period 2:

$$\lim_{t \to \infty} \mathbf{P}^{2t} = \begin{pmatrix} \boldsymbol{\pi}(\mathbf{S}) \boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\pi}(\mathbf{q}) \boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}} \end{pmatrix} \qquad \lim_{t \to \infty} \mathbf{P}^{2t-1} = \begin{pmatrix} \mathbf{0} & \boldsymbol{\pi}(\mathbf{S}) \boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}} \\ \boldsymbol{\pi}(\mathbf{q}) \boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} \tag{5}$$

where $\boldsymbol{\pi}(\mathbf{S}) \in \mathbb{R}^{|\mathbf{S}|}$, $\boldsymbol{\pi}(\mathbf{q}) \in \mathbb{R}^{|\mathbf{q}|}$ are the stationary distributions of aforementioned bidirectional random walks in Section 3.2 with equations $\boldsymbol{\pi}(\mathbf{S}) = \boldsymbol{\Psi}_{\mathbf{S}} \boldsymbol{\pi}(\mathbf{S})$ and $\boldsymbol{\pi}(\mathbf{q}) = \boldsymbol{\Psi}_{\mathbf{q}} \boldsymbol{\pi}(\mathbf{q})$ respectively. $\boldsymbol{e}_{|\cdot|}$ is a vector of ones with different length indicated by its subscript.

To perform classification, we first assume particles uniformly distributed in the discrete finite space $\mathbf{z} = \mathbf{q} \cup \mathbf{S}$. Then, we use the expected number of visits from all particles $z \in \mathbf{z}$ to support features $\mathbf{s}^c \subset \mathbf{S}$ of class c in the long term of bidirectional random walks $\{X_t\}$ as a measure:

$$\begin{aligned} \mathbf{Pr}(\tilde{y} = c) &\propto \lim_{t \to \infty} \sum_{z \in \mathbf{z}} \mathbb{E} \left[\sum_{k=1}^{t} \mathbb{1}[X_k \in \mathbf{s}^c] \middle| X_0 = z \right] \\ &= \lim_{t \to \infty} \frac{1}{t} \sum_{k=1}^{t} \left[\frac{1}{|\mathbf{S}| + |\mathbf{q}|} \sum_{s \in \mathbf{s}^c} \left(\sum_{s' \in \mathbf{S}} \left[\mathbf{P}^{2k}\right]_{ss'} + \sum_{q \in \mathbf{q}} \left[\mathbf{P}^{2k-1}\right]_{sq} \right) \right] \\ &= \frac{1}{|\mathbf{S}| + |\mathbf{q}|} \sum_{s \in \mathbf{s}^c} \left(\sum_{s' \in \mathbf{S}} \left[\lim_{t \to \infty} \mathbf{P}^{2t} \right]_{ss'} + \sum_{q \in \mathbf{q}} \left[\lim_{t \to \infty} \mathbf{P}^{2t-1} \right]_{sq} \right) = \sum_{s \in \mathbf{s}^c} [\boldsymbol{\pi}(\mathbf{S})]_s \end{aligned} \tag{6}$$

where $\mathbb{1}[\cdot]$ is an indicator function that equals 1 if its argument is true and zero otherwise. $[\cdot]_{ij}$ indicates the entry of the matrix from particle j to particle i and $[\cdot]_i$ indicates the entry of the vector that associated to feature i. It should be noted that we give the derivation when the Markov chain length t is even in the first equality of Eqn.(6) and prove that the odd t will reach the same result in Appendix B. The second equality is from the *absorbing* of the periodic Markov chain where the power of matrix get saturated to Eqn.(5) when the value of t increases. Since $\boldsymbol{\pi}(\mathbf{S})$ is the stationary distribution of column-stochastic $\boldsymbol{\Psi}_{\mathbf{S}}$ under the probability constraint $\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} \boldsymbol{\pi}(\mathbf{S}) = 1$, $\mathbf{Pr}(\tilde{y})$ is a valid distribution for class predictions.

3.4 REINTERPRETATION AS A GRAPH CENTRALITY

If we interpret the Markov transition matrix $\mathbf{P}$ as an adjacency matrix $\{a_{v_1,v_2}\}$ of a directed bipartite graph $G := (V = \{\mathbf{q}, \mathbf{S}\}, E)$, we find that the accessibility of support features in the time-homogeneous bidirectional random walk would be equivalent to learning a *single-mode eigenvector centrality* on graph G. The bipartite graph is also called the *affiliation network* in social network analysis (Bonacich, 1972; 1991; Borgatti & Everett, 1997) that models two types of entities "*actors*" and "*society*" related by affiliation of the former in the latter. The concept of centrality in social network analysis is generally used to investigate the acquaintanceships among people that often stem from one or more shared affiliations.

To see this in graph theory, we start with a brief overview of the eigenvector centrality that reflects score x of vertex v for both $q \in \mathbf{q}$ and $s \in \mathbf{S}$ in the affiliation network with adjacency $\{a_{v_1,v_2}\}$:

$$x_q = \frac{1}{\lambda} \sum_{v \in \mathbb{V}(q)} a_{q,v} x_v = \frac{1}{\lambda} \sum_{v \in \mathbf{S}} a_{q,v} x_v \qquad x_s = \frac{1}{\lambda} \sum_{v \in \mathbb{V}(s)} a_{s,v} x_v = \frac{1}{\lambda} \sum_{v \in \mathbf{q}} a_{s,v} x_v \tag{7}$$

where $\mathbb{V}(\cdot)$ is a set of neighbors for the given vertex and λ is a constant. With a small rearrangement, Eqn.(7) can be rewritten in vector notation with an eigenvector equation $\mathbf{P}\mathbf{x} = \lambda\mathbf{x}$. The additional requirement that all the entries of the eigenvector be non-negative implies (by the Perron–Frobenius theorem Perron (1907)) that only the greatest eigenvalue results in the desired centrality measure. For the adjacency matrix defined by the Markov transition matrix in our method, the largest eigenvalue λ of the column-stochastic matrix $\mathbf{P}$ is 1.

Single-mode centrality (Borgatti & Everett, 1997) is a special form of graph centrality that measures the extent to which nodes in one vertex set are relatively central only to other nodes in the same vertex set on bipartite graph. For example, the single-mode centrality for different s in $\mathbf{S}$ is defined by $\hat{\mathbf{x}}_\mathbf{S} = \mathbf{x}_\mathbf{S} / \sum_{s \in \mathbf{S}} x_s$. Lemma 1 (proved in Appendix C) shows that features' accessibility $\boldsymbol{\pi}(\mathbf{S})$ in Eqn.(6) is equivalent to the single-mode eigenvector centrality of support set $\mathbf{S}$ on bipartite data.

Lemma 1. *Assume G is the affiliation network of bipartite data $\{\mathbf{q}, \mathbf{S}\}$ with the adjacency matrix defined by the anti-diagonal Markov transition matrix $\mathbf{P}$ in Eqn.(4). The single-mode eigenvector centrality $\hat{\mathbf{x}}_\mathbf{S} = \mathbf{x}_\mathbf{S} / \sum_{s \in \mathbf{S}} x_s$ of vertex set $\mathbf{S}$ is equivalent to the features' accessibility $\boldsymbol{\pi}(\mathbf{S})$ on $\mathbf{S}$ in a long term of time-homogeneous Markov process.*

Based on this interpretation, it is straightforward to consider the attenuation (damping) factor α on the affiliation network motivated by the well-known Katz centrality, a well-known variant of eigenvector centrality in graph theory (Katz, 1953). The features' accessibility of Markov bidirectional random walks with attenuation α for few-shot classifications is defined by

$$\begin{aligned} \mathbf{Pr}_{\text{Katz}}(\tilde{y} = c) &\propto \sum_{z \in \mathbf{z}} \mathbb{E}\left[\sum_{t=1}^{\infty} \alpha^t \mathbb{1}[X_t \in \mathbf{s}^c] \middle| X_0 = z \right] \\ &= \frac{1}{\epsilon} \sum_{t=1}^{\infty} \sum_{s \in \mathbf{s}^c} \left(\sum_{s' \in \mathbf{S}} \left[\alpha^{2t} \mathbf{P}^{2t}\right]_{ss'} + \sum_{q \in \mathbf{q}} \left[\alpha^{2t-1} \mathbf{P}^{2t-1}\right]_{sq} \right) \end{aligned} \tag{8}$$

where $\epsilon = (|\mathbf{S}| + |\mathbf{q}|) \sum_{t=1}^{\infty} \alpha^t = (|\mathbf{S}| + |\mathbf{q}|)\alpha/(1-\alpha)$ is a constant for a valid distribution.

Although we simply consider the single-mode centrality on $\mathbf{S}$ for an end-to-end classification purpose, it is also beneficial to learn its conjugate centrality $\boldsymbol{\pi}(\mathbf{q})$ for $\mathbf{q}$ on the affiliation network. We will show that both centralities could serve as plug-and-play for finding centralized local characteristics in existing methods (hence the term Mutual Centralized Learning, MCL).

3.5 Mutual Centralized Learning

The algorithm of Mutual Centralized Learning (MCL) in Eqn.(6, 8) involves the computation of Markov stationary distribution $\boldsymbol{\pi}(\mathbf{S})$ with equation $\boldsymbol{\pi}(\mathbf{S}) = \boldsymbol{\Psi}_\mathbf{S} \boldsymbol{\pi}(\mathbf{S})$. Theoretically, the $\boldsymbol{\pi}(\mathbf{S})$ is the eigenvector of $\boldsymbol{\Psi}_\mathbf{S}$ with the eigenvalue 1 under a probability constraint $\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} \boldsymbol{\pi}(\mathbf{S}) = 1$.

The above constraints could lead to a solution of $\boldsymbol{\pi}(\mathbf{S})$ by solving the overdetermined linear system

$$\begin{pmatrix} \boldsymbol{\Psi}_\mathbf{S} - \boldsymbol{I} \\ \boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} \end{pmatrix} \boldsymbol{\pi}(\mathbf{S}) = \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \tag{9}$$

where $\boldsymbol{I}$ is an identity matrix and $\mathbf{0}$ is a vector of zeros. Although we can solve it by various methods like pseudo-inverse or QR decomposition and back substitution, it is empirically found time-consuming as these operators are either numerically unstable or not paralleled well in modern deep-learning packages.

To handle it, we present a fast solution by the graph centrality. We first calculate the closed-form Katz centrality (Katz, 1953) and then solve single-mode Katz centrality in Eqn.(8) by

$$\mathbf{x}^{\text{Katz}} = ((\boldsymbol{I} - \alpha\mathbf{P})^{-1} - \boldsymbol{I})\boldsymbol{e} \tag{10}$$

$$\mathbf{Pr}_{\text{Katz}}(\tilde{y} = c) = \frac{\sum_{s \in \mathbf{s}^c} \mathbf{x}_s^{\text{Katz}}}{\sum_{s' \in \mathbf{S}} \mathbf{x}_{s'}^{\text{Katz}}} \tag{11}$$

where $\boldsymbol{e}$ is vector of ones with length $|\mathbf{S}| + |\mathbf{q}|$. Since Katz centrality degrades to the eigenvector centrality when α approaches 1 (Katz, 1953), we can obtain a fast approximation of Eqn.(6) with a large $\alpha = 0.999$:

$$\mathbf{x}^{\mathrm{Eigen}} = \lim_{\alpha \to 1} ((\boldsymbol{I} - \alpha \mathbf{P})^{-1} - \boldsymbol{I})\boldsymbol{e} \approx ((\boldsymbol{I} - 0.999\mathbf{P})^{-1} - \boldsymbol{I})\boldsymbol{e} \tag{12}$$

$$\mathbf{Pr}_{\mathrm{MCL}}(\tilde{y} = c) = \frac{\sum_{s \in \mathbf{s}^c} \mathbf{x}_s^{\mathrm{Eigen}}}{\sum_{s' \in \mathbf{S}} \mathbf{x}_{s'}^{\mathrm{Eigen}}} \tag{13}$$

We give the whole picture of proposed MCL with episode loss for few-shot classifications in Appendix A and demonstrate that Eqn.(10) can be time efficiently solved in $O(r^3)$ by the block-wise inversion in Eqn.(F.2).

3.6 Graph Centrality as Plugin for Global Feature based FSL.

Since the underlying centrality learned by MCL reveals the different importance of local features on an affiliation network, it is thus plausible to plug it into global feature based methods by replacing their native global average pooling (GAP) with the centrality weighted pooling as follows:

1. Remove the GAP layer to get feature map outputs (a set of dense features) for each image.
2. Calculate single-mode centrality $\boldsymbol{\pi}(\mathbf{S})$ and its conjugate $\boldsymbol{\pi}(\mathbf{q})$ by Eqn.(12)
3. Use $\boldsymbol{\pi}(\mathbf{q})$ as graph centrality weights on query dense features and weighted accumulate them to a single feature vector for each image.
4. Unlike the centrality $\boldsymbol{\pi}(\mathbf{q})$ of query features that are from the single query image, $\boldsymbol{\pi}(\mathbf{S})$ are calculated for support dense features from all classes. Thus, we execute an extra class-wise normalization step to obtain normalized centrality weights for each supporting class.
5. Use the class-wise normalized centrality to accumulate dense features for every supporting class like in step 3 and each class will be represented by a single feature vector.

Once we get the centrality weighted vector representations for the query image and support classes, it is straightforward to perform traditional global feature based ProtoNet (Snell et al., 2017) and RelationNet (Sung et al., 2018) as before.

4 Experiments

4.1 Experiment Setups

We conduct experiments on two widely-used FSL datasets and three fine-grained datasets: (1) ***mini*ImageNet** (Vinyals et al., 2016) contains 600 images per class over 100 classes. We follow the split used by Sachin & Hugo (2017) that takes 64, 16 and 20 classes for train/val/test respectively; (2) ***tiered*ImageNet** (Ren et al., 2018) is much larger compared to *mini*ImageNet with 608 classes. The 351, 97 and 160 classes are used for train/val/test respectively. (3) **CUB** (Welinder et al., 2010) consists 11,788 images from 200 bird classes. 100/50/50 classes are used for train/val/test and each image is first cropped to a human-annotated bounding box. (4) **meta-iNat** (Wertheimer & Hariharan, 2019) is a benchmark of animal species in the wild. We follow the same class split proposed by (Wertheimer & Hariharan, 2019) that uses 908/227 classes for training/evaluation respectively. (5) **tiered meta-iNat** (Wertheimer & Hariharan, 2019) is a more difficult version of meta-iNat where a large domain gap is introduced. The 354 test classes are populated by insects and arachnids, while the remaining 781 classes (mammals, birds *et al.*) form the training set.

Backbone networks. We conduct experiments with both widely-used four layer convolutional Conv-4 (Vinyals et al., 2016) and deep ResNet-12 (Sun et al., 2019) backbones. As is commonly implemented in the state-of-the-art FSL literature, we adopt a pre-training stage for the ResNet-12 before the episode meta-training while directly meta-train from scratch for the simple Conv-4.

Dense feature extractor $f_\theta(\cdot)$. For a fair comparison with the previous dense feature based methods, we explore three dense feature extractors in our experiments (details can be found in Appendix E): (1) *VanillaFCN* simply treats the feature map output of fully convolutional network as dense features. (2) *PyramidFCN* applies pyramid structures to extract dense features of different scales. (3) *PyramidGrid* crops image into grid patches of different scales and encodes each patch to a feature vector.

Method		Conv-4				ResNet-12			
		*mini*ImageNet		*tiered*ImageNet		*mini*ImageNet		*tiered*ImageNet	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Global features	DSN	51.78	68.99	53.22	71.06	62.64	78.83	67.39	82.85
	MetaOptNet	52.87	68.76	54.71	71.76	62.64	78.63	65.99	81.56
	FEAT	55.15	71.61	-	-	66.78	82.05	70.80	84.79
	RelationNet	52.12	66.90	54.33	69.95	60.97	75.12	64.71	78.41
	RelationNet+MCL	*54.50*	*70.63*	*57.73*	*74.46*	*61.70*	*75.53*	*65.93*	*80.27*
	ProtoNet	52.32	69.74	53.19	72.28	62.67	77.88	68.48	83.46
	ProtoNet+MCL	*54.31*	*69.84*	*56.67*	*74.36*	*64.40*	*78.60*	*70.62*	*83.84*
Dense features (*VanillaFCN*)	DN4	54.66	71.26	56.86	72.16	65.35	81.10	69.60	83.41
	DeepEMD	52.15	65.52	50.89	66.12	65.91	82.41	71.16	83.95
	FRN	54.87	71.56	55.54	74.68	65.90	81.92	70.56	85.47
	Label Spreading	52.24	67.68	54.56	70.08	65.00	80.07	71.12	83.89
	MCL (ours)	55.38	70.02	57.63	74.25	66.75	83.37	71.76	85.68
	MCL-Katz (ours)	**55.55**	**71.74**	**57.78**	**74.77**	**66.91**	**83.53**	**72.01**	**86.02**
Dense features (*PyramidFCN*)	DN4	54.54	70.94	57.05	72.90	63.54	79.04	71.10	84.22
	DeepEMD	50.67	64.94	51.26	65.64	66.27	82.41	70.76	84.20
	FRN	54.40	70.75	57.30	**75.58**	65.94	81.97	70.56	85.44
	Label Spreading	53.38	68.69	55.21	71.22	65.71	75.78	71.00	78.01
	MCL (ours)	55.13	70.77	57.93	74.36	**67.38**	**84.06**	72.01	86.31
	MCL-Katz (ours)	**55.77**	**71.24**	**58.20**	74.73	66.94	83.85	**72.13**	**86.32**
Dense features (*PyramidGrid*)	DN4	57.17	70.91	56.71	70.92	67.86	80.08	71.29	82.60
	DeepEMD	55.68	70.75	55.88	70.06	67.83	81.32	73.13	84.18
	FRN	55.80	71.52	55.68	72.87	67.00	82.20	71.42	85.58
	Label Spreading	55.12	68.43	56.05	72.39	67.18	81.07	73.18	85.19
	MCL (ours)	57.50	73.03	57.57	73.81	**69.03**	**85.11**	**73.62**	**86.29**
	MCL-Katz (ours)	**57.88**	**74.03**	**57.63**	**73.96**	68.96	84.71	73.38	86.21

Table 1: Few-shot classification accuracy (%) on *mini*ImageNet and *tiered*ImageNet. The confidence intervals are all below 0.25 for the 10,000 episodes evaluation. Results for DN4, DeepEMD and FRN (except for their larger shot training) are our reimplementations for a fair and thorough comparison under different dense feature extractor settings. Results of *italic* font indicates the performance of MCL as a plug-and-play for global features based methods. Results of blue fonts are the second-placed results in the last three panes, respectively.

4.2 Few-shot Classification Results

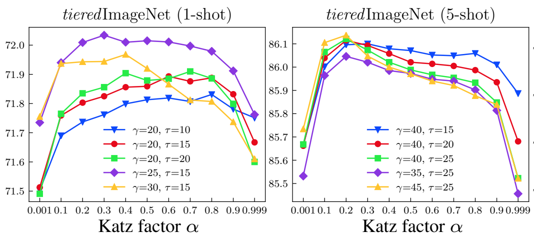
Table 1 details the comparisons of proposed MCL/MCL-Katz with competitive global feature based DSN (Simon et al., 2020), MetaOptNet (Lee et al., 2019), FEAT (Ye et al., 2020) as well as dense feature based DN4 (Li et al., 2019), DeepEMD (Zhang et al., 2020) and FRN (Wertheimer et al., 2021) on *mini-*/*tiered*ImageNet. Results on three fine-grained benchmarks can be found in Table 2.

Besides these methods, we also exploit the well-known label spreading in dense feature based inductive FSL as a baseline to demonstrate that traditional transductive methods could be easily adaptable to inductive settings if we treat vectors of feature maps as a set of unlabeled data and use the features' accessibility as a criterion in few-shot classification.

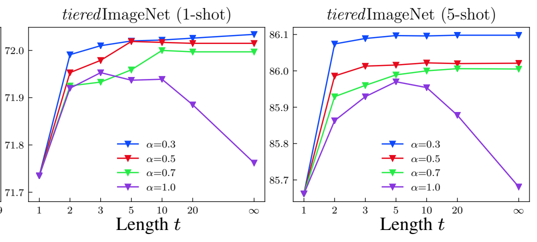
Comparisons with dense feature based methods. The last three panes of Table 1 illustrate that proposed bidirectional random walks consistently outperforms unidirectional DN4's nearest neighboring, DeepEMD's optimal matching and FRN's latent feature reconstructions under different settings. Although the optimal matching flow is unidirectional in DeepEMD, they adopt a bidirectional cross-reference attention that encodes the mutual relevance to a certain extent. If we consider dense features as *nodes* and their similarities as *edges* in the bipartite graph, a major difference between DeepEMD's attention and our method is that they treat nodes equally to derive different attention on edges while we use the fixed edges to derive the centrality of nodes in the graph.

Method	CUB		meta-iNat		*tiered* meta-iNat	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
ProtoNet (Snell et al., 2017)	63.73	81.50	55.34	76.43	34.34	57.13
Covar. pool (Wertheimer & Hariharan, 2019)	-	-	57.15	77.20	36.06	57.48
DSN (Simon et al., 2020)	66.01	85.41	58.08	77.38	36.82	60.11
CTX (Doersch et al., 2020)	69.64	87.31	60.03	78.80	36.83	60.84
DN4 (Li et al., 2019)	73.42	**90.38**	62.32	79.76	43.82	64.17
FRN (Wertheimer et al., 2021)	73.48	88.43	62.42	80.45	43.91	63.36
MCL (ours)	74.16	87.72	**64.66**	**81.31**	43.96	**64.61**
MCL-Katz (ours)	**76.19**	89.44	63.92	81.09	**44.01**	64.24

Table 2: Fine-grained 5-way few-shot classification (%) results with Conv-4. Results of all comparing methods (except for DN4) are drawn from FRN with larger shot training. The confidence intervals are all below 0.25 for 10,000 episodes evaluation.



(a) Ablation on parameters γ, τ and α.



(b) Ablation on the length of Markov process $\{X_t\}$.

Figure 2: Ablative of 5-way FSL results on *tiered*ImageNet with ResNet-12 and *VanillaFCN* feature extractor. (a) studies the influences from different choices of parameters. (b) compares performances between infinite length of Markov process and those from varying finite lengths.

Different choices of scaling hyper-parameters γ and τ. We use two hyper-parameters γ, τ in matrices $\mathbf{\Phi}_\gamma \in \mathbb{R}^{|\mathbf{S}|\times|\mathbf{q}|}$ and $\mathbf{\Phi}_\tau \in \mathbb{R}^{|\mathbf{q}|\times|\mathbf{S}|}$ respectively. With the column-wise normalization in Eqn.(1,2), γ and τ can be interpreted as the reciprocal of temperatures in softmax-like random walk probability. Thus, a large scaling parameter will have a hard random walk probability that leads to a concentrated centrality in the affiliation network. However, an extremely large parameter (*e.g.*, one-hot probability when γ, τ approach infinity) would inevitably bias the episodic training due to the potential gradient explosion. In the experiments, we carefully select γ and τ by the validation performance for pretrained ResNet12 backbone. Figure 2(a) details different combinations of γ and τ. Since the cardinality $|\mathbf{S}|$ is larger than $|\mathbf{q}|$, we empirically use a larger γ than τ in the experiments as we need a harder probability in random walks from query to the large union set of support features.

Influence of Katz attenuation factor α. Intuitively, the contributions of distant nodes should be penalized by attenuation in the long term of random walks. For a small value of α, the contribution given by paths longer than one rapidly declines, and thus the centrality is mainly influenced by short paths (mostly in-degrees). When α is large, long paths are devalued smoothly, and the centrality would be more influenced by endogenous topology. We show the ablative on α in Figure 2(a) that, for an extremely small $\alpha \to 0$, the classification will be only determined by γ in a unidirectional query-to-support walk where the contributions from paths longer than one just vanished. We show more quantitative comparisons between unidirectional and bidirectional random walks in Table 4. The performance is improved by considering mutual affiliations for an increasing value of α. In this work, we simply use $\alpha = 0.5$ for MCL-Katz in Eqn.(8) and use $\alpha = 0.999$ for MCL in Eqn.(6).

Long-term *v.s.* finite short-term features' accessibility in Markov process. Figure 2(b) shows the ablative results of using short-term features' accessibility from various finite steps of the Markov process. We see that the performance gains by the long term of random walks gradually saturate for the increasing values of t as contributions from far distant nodes get vanished due to the exponential decreasing of α^t in MCL-Katz. With appropriate α, we see using the long-term features' accessibility for classifications consistently outperforms those from the finite short-term's accessibility.

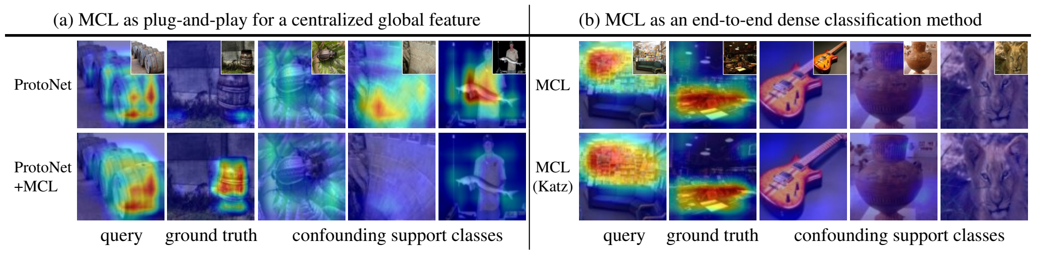

(a) MCL as plug-and-play for a centralized global feature
(b) MCL as an end-to-end dense classification method
ProtoNet
ProtoNet +MCL
MCL
MCL (Katz)
query
ground truth
confounding support classes
query
ground truth
confounding support classes

Table 3: Grad-CAM visualizations of query and support images in 4-way 1-shot classifications. The left pane (a) illustrates MCL as plug-and-play on *tiered*ImageNet. The right pane (b) illustrates MCL as an end-to-end dense classification method on *mini*ImageNet. Images at the second column of each task are from the ground truth while the others are from the confounding support classes.

	$f_\theta(\cdot)$	*mini-*		*tiered-*	
		1-shot	5-shot	1-shot	5-shot
uni-	*VanillaFCN*	54.88	69.98	55.31	68.87
MCL	*VanillaFCN*	55.38	70.02	57.63	74.25
MCL-Katz	*VanillaFCN*	**55.55**	**71.74**	**57.78**	**74.77**
uni-	*PyramidFCN*	54.92	69.17	55.69	69.26
MCL	*PyramidFCN*	55.13	70.77	57.93	74.36
MCL-Katz	*PyramidFCN*	**55.77**	**71.24**	**58.20**	**74.73**
uni-	*PyramidGrid*	56.61	71.40	55.12	69.19
MCL	*PyramidGrid*	57.50	73.03	57.57	73.81
MCL-Katz	*PyramidGrid*	**57.88**	**74.03**	**57.63**	**73.96**

Table 4: Comparisons between the *unidirectional* random walks and *bidirectional* MCL on *mini-*/*tiered*ImageNet with Conv-4.

	+MCL		*mini-*		*tiered-*	
	q	**s**	1-shot	5-shot	1-shot	5-shot
ProtoNet			52.32	69.74	53.19	72.28
ProtoNet	✓		53.74	69.61	55.36	73.17
ProtoNet		✓	53.97	69.74	56.60	74.18
ProtoNet	✓	✓	**54.31**	**69.84**	**56.67**	**74.36**
RelationNet			52.12	66.90	54.33	69.95
RelationNet	✓		53.25	66.79	54.46	70.71
RelationNet		✓	54.37	70.57	57.68	74.24
RelationNet	✓	✓	**54.50**	**70.63**	**57.73**	**74.46**

Table 5: Ablation of MCL as plug-and-play that individually applied on query features **q** and support features **s**. The experiments are conducted with Conv-4 and *VanillaFCN* on *mini-*/*tiered*ImageNet.

Graph centrality as plugin. The first pane in Table 1 shows that our proposed centrality could provide at most 3.4% and 4.5% performance gains for global feature based ProtoNet and RelationNet respectively. Table 5 details that the improvements are consistent if we solely replace the global average pooling with proposed centrality weighted pooling on the query and support dense features.

Qualitative visualizations of the feature centralization. To get a deeper understanding of our centrality based methods, we visualize the Grad-CAM (Selvaraju et al., 2017) of the last convolutional layer in ResNet-12. Table 3(a) first shows that ProtoNet fails to identify the most relevant regions of interests in the task. With centrality weighted pooling, ProtoNet+MCL identifies those regions and removing the confounding areas in predictions. Table 3(b) shows that the most centralized support features in MCL are not only from the ground-truth but also mutually affiliate with the task-relevant objects of the query images. It qualitatively validates the underlying idea of our methods that support features would be frequently visited in the long term of bidirectional random walks by the mutual affiliations if they shared the same class with the query image.

5 Conclusions

We present a novel dense feature based framework: Mutual Centralized Learning (MCL) to highlight the mutual affiliations of bipartite dense features in FSL. We introduce a novel features' accessibility criterion and demonstrate classic transductive methods like label spreading could be easily adapted to the inductive setting if we treat dense features from a single image as the set of unlabeled data. We propose bidirectional random walk to learn mutual affiliations in FSL and prove its features' accessibility in a long time-homogeneous Markov process is equivalent to the single-mode eigenvector centrality of an affiliation network. We show that such centrality could not only serve as a noval end-to-end classification criterion but also as a plugin in existing methods. Experimental results demonstrate MCL achieves the state-of-the-art on both *mini*ImageNet and *tiered*ImageNet.

REFERENCES

Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of mathematical sociology*, 2(1):113–120, 1972.

Phillip Bonacich. Simultaneous group and individual centralities. *Social networks*, 13(2):155–168, 1991.

Stephen P Borgatti and Martin G Everett. Network analysis of 2-mode data. *Social networks*, 19(3): 243–269, 1997.

Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. Graph prototypical networks for few-shot learning on attributed networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 295–304, 2020.

Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. *arXiv preprint arXiv:2007.11498*, 2020.

Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.

Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

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*, pp. 10657–10665, 2019.

Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7260–7268, 2019.

Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. Dense classification and implanting for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9258–9267, 2019.

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. In *International Conference on Learning Representations*, 2019.

Oskar Perron. Zur theorie der matrices. *Mathematische Annalen*, 64(2):248–263, 1907.

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. In *International Conference on Learning Representations*, 2018.

Ravi Sachin and Larochell Hugo. Optimization as a model for few-shot learning. *ICLR*, 2017.

Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.

Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *CVPR*, pp. 4136–4145, 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, pp. 4077–4087, 2017.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 403–412, 2019.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, pp. 1199–1208, 2018.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 3637–3645, 2016.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6558–6567, 2019.

Davis Wertheimer, Luming Tang, and Bharath Hariharan. Few-shot classification with feature map reconstruction networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8012–8021, 2021.

Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*, pp. 8808–8817, 2020.

Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Differentiable earth mover's distance for few-shot learning. *arXiv e-prints*, pp. arXiv–2003, 2020.

Dengyong Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, volume 16, pp. 321–328, 2003.

Appendices

A Algorithm for Episode Loss

Algorithm 1 Episodic Loss for N-way K-shot Mutual Centralized Learning.

```
Notations:
    $C_{\mathrm{train}}$: the set of training classes.
    $C_{\mathrm{episode}}$: the set of randomly selected classes in each episode.
    $N_Q$: the number of selected query examples per class in each episode.
    $\mathcal{D}_{\mathrm{train}}$: the set of training data $\{(x_0, y_0), (x_1, y_1), ...\}$ where $x$ is an image and $y$ is its label.
    $\mathcal{D}_{\mathrm{c}}$: the subset of $\mathcal{D}_{\mathrm{train}}$ that belongs to class $c$
    $\mathcal{Q}_{\mathrm{episode}}$: the set of query data in each episode.
    $\mathcal{S}^c_{\mathrm{episode}}$: the set of support data of class $c$ in each episode.
    $\mathbf{q}$: the set of dense features of a single query image.
    $\mathbf{s}^c$: the set of dense features of class $c$ in each episode.
    $\mathbf{S}$: the set of dense features of all classes in each episode.
Require:
    $\textsc{RandomSample}(\{\cdot\}, k)$: random sampling $k$ elements from set $\{\cdot\}$ without replacement.
    $f_\theta(x)$: function to dense features given the input image $x$ (e.g., feature map from VanillaFCN).
    $\mathbb{1}[\cdot]$: indicator function that equals one if its argument is true and zero otherwise
Procedure:
 1: $\mathcal{Q}_{\mathrm{episode}} \leftarrow \{\}$
 2: $C_{\mathrm{episode}} \leftarrow \textsc{RandomSample}(C_{\mathrm{train}}, N)$    ▷ Select $N$-way classes for episode
 3: for $c$ in $C_{\mathrm{episode}}$ do
 4:     $\mathcal{S}^c_{\mathrm{episode}} \leftarrow \textsc{RandomSample}(\mathcal{D}_c, K)$    ▷ Select $K$-shot support examples per class
 5:     $\mathbf{s}^c \leftarrow 0$
 6:     for $(x, y) \in \mathcal{S}^c_{\mathrm{episode}}$ do
 7:         $\mathbf{s}^c \leftarrow \mathbf{s}^c + f_\theta(x)/K$    ▷ Average support features from $K$-shot images of the same class
 8:     end for
 9:     $\mathcal{Q}_{\mathrm{episode}} \leftarrow \mathcal{Q}_{\mathrm{episode}} \cup \textsc{RandomSample}(\mathcal{D}_c \setminus \mathcal{S}^c_{\mathrm{episode}}, N_Q)$ ▷ Select $N_Q$ queries per class
10: end for
11: $\mathbf{S} \leftarrow \bigcup_{c \in C_{\mathrm{episode}}} \mathbf{s}^c$    ▷ Union support features of different classes within an episode
12: $\mathcal{L} \leftarrow 0$    ▷ Initialize episodic loss
13: for $(x, y)$ in $\mathcal{Q}_{\mathrm{episode}}$ do
14:     $\mathbf{q} \leftarrow f_\theta(x)$
15:     Compute matrix $\mathbf{\Phi}_\gamma$, $\mathbf{\Phi}_\tau$, and their normalization matrices $\mathbf{D}$ and $\mathbf{W}$ by Eqn.(1, 2)
16:     Construct transition matrix $\mathbf{P}$ by Eqn.(4)
17:     Compute complete Katz centrality or approximated eigenvector centrality by Eqn.(10, 12)
18:     Compute probability $\mathbf{Pr}(\tilde{y} = c)$ for class $c$ by Eqn.(13) for MCL (or Eqn.(11) for MCL-Katz).
19:     $\mathcal{L} \leftarrow \mathcal{L} - \frac{1}{N N_Q} \left[ \sum_{c \in C_e} \mathbb{1}[y = c] \log \mathbf{Pr}(\tilde{y} = c) \right]$    ▷ Negative log-likelihood loss
20: end for
```

B Proof of Eqn.(5)

Eqn.(5): Given the anti-diagonal Markov transition matrix $\mathbf{P} = \begin{pmatrix} \mathbf{0} & \mathbf{P}_{\mathbf{Sq}} \\ \mathbf{P}_{\mathbf{qS}} & \mathbf{0} \end{pmatrix}$ that is composed of two column-normalized submatrices $\mathbf{P}_{\mathbf{Sq}}$, $\mathbf{P}_{\mathbf{Sq}}$, the stationary distribution is periodic with equation:

$$\lim_{t\to\infty} \mathbf{P}^{2t} = \begin{pmatrix} \boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\pi}(\mathbf{q})\boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}} \end{pmatrix} \qquad \lim_{t\to\infty} \mathbf{P}^{2t-1} = \begin{pmatrix} \mathbf{0} & \boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}} \\ \boldsymbol{\pi}(\mathbf{q})\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} & \mathbf{0} \end{pmatrix}$$

where $\boldsymbol{\pi}(\mathbf{S})$ is the stationary vector of $\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}$ and $\boldsymbol{\pi}(\mathbf{q})$ is the stationary vector of $\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}}$. $\boldsymbol{e}_{|\cdot|}$ denotes a vector of ones with different length indicated by its subscript.

Proof of periodic: Consider the eigenvalue λ of $\mathbf{P}$ with determinant

$$\det(\lambda \boldsymbol{I} - \mathbf{P}) = \det \begin{pmatrix} \lambda \boldsymbol{I} & -\mathbf{P}_{\mathbf{Sq}} \\ -\mathbf{P}_{\mathbf{qS}} & \lambda \boldsymbol{I} \end{pmatrix} = \det(\lambda^2 \boldsymbol{I} - \mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}) \tag{B.1}$$

it can be found that the eigenvalues of $\mathbf{P}$ are the square roots of eigenvalues of $\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}$.

Since the subblocks of $\mathbf{P}_{\mathbf{Sq}}$ and $\mathbf{P}_{\mathbf{qS}}$ are both column-normalized matrices as defined in Eqn.(1,2), their product is column-stochastic that can be proved by:

$$\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} \mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}} = \boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}} \mathbf{P}_{\mathbf{qS}} = \boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} \tag{B.2}$$

We know (by the definition of stochastic matrix) that $\lambda = 1$ is the largest eigenvalue of $\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}$, and its uniqueness is guaranteed since there is no zero entry in both $\mathbf{P}_{\mathbf{Sq}}$ and $\mathbf{P}_{\mathbf{qS}}$. According to Eqn.(B.1), we get another eigenvalue $\lambda = -1$ for stochastic matrix $\mathbf{P}$. From the Perron–Frobenius theorem that the period of $\mathbf{P}$ equals to the number of eigenvalue whose absolute value is equal to the spectral radius of $\mathbf{P}$, we can get its stationary distribution is of period 2.

Proof of stationary distribution for even periods: Consider the even power of matrix $\mathbf{P}$ in extreme

$$\lim_{t\to\infty} \mathbf{P}^{2t} = \lim_{t\to\infty} \begin{pmatrix} \mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}} \end{pmatrix}^t = \begin{pmatrix} \lim_{t\to\infty} [\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}]^t & \mathbf{0} \\ \mathbf{0} & \lim_{t\to\infty} [\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}}]^t \end{pmatrix} \tag{B.3}$$

We have shown in Eqn.(B.2) that $\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}$ is column-stochastic and use $\boldsymbol{\pi}(\mathbf{S})$ to represent its stationary distribution vector with equation $\lim_{t\to\infty} [\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}]^t = \boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}}$. By analogy, the infinity power of $\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}}$ could also reach a similar stationary $\boldsymbol{\pi}(\mathbf{q})$. Thus, we can derive the left part of Eqn.(5) by substituting the two stationary vectors into Eqn.(B.3).

Proof of stationary distribution for odd periods: From the definition, we have

$$\lim_{t\to\infty} \mathbf{P}^{2t-1} = \lim_{t\to\infty} \mathbf{P}^{2t+1} = \mathbf{P} \lim_{t\to\infty} \mathbf{P}^{2t} = \begin{pmatrix} \mathbf{0} & \mathbf{P}_{\mathbf{Sq}}\boldsymbol{\pi}(\mathbf{q})\boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}} \\ \mathbf{P}_{\mathbf{qS}}\boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} & \mathbf{0} \end{pmatrix} \tag{B.4}$$

According to the definition of $\boldsymbol{\pi}(\mathbf{q})$ and $\boldsymbol{\pi}(\mathbf{S})$, we can get

$$\boldsymbol{\pi}(\mathbf{q})\boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}} = \lim_{t\to\infty} [\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}}]^t = \mathbf{P}_{\mathbf{qS}} \left(\lim_{t\to\infty} [\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}]^t \right) \mathbf{P}_{\mathbf{Sq}} = \mathbf{P}_{\mathbf{qS}}\boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}}\mathbf{P}_{\mathbf{Sq}} \tag{B.5}$$

if we right matrix product of $\boldsymbol{e}_{|\mathbf{q}|}$ on both sides of Eqn.(B.5), we have

$$\boldsymbol{\pi}(\mathbf{q})\boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}}\boldsymbol{e}_{|\mathbf{q}|} = \mathbf{P}_{\mathbf{qS}}\boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}}\mathbf{P}_{\mathbf{Sq}}\boldsymbol{e}_{|\mathbf{q}|} \tag{B.6}$$

Since $\boldsymbol{e}_{|\mathbf{q}|}^{\mathrm{T}}\boldsymbol{e}_{|\mathbf{q}|} = |\mathbf{q}|$ and $\boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}}\mathbf{P}_{\mathbf{Sq}}\boldsymbol{e}_{|\mathbf{q}|} = \sum_i \sum_j [\mathbf{P}_{\mathbf{Sq}}]_{ij} = |\mathbf{q}|$, Eqn.(B.6) can be simplified by dividing the same scalar $|\mathbf{q}|$ on both sides:

$$\boldsymbol{\pi}(\mathbf{q}) = \mathbf{P}_{\mathbf{qS}}\boldsymbol{\pi}(\mathbf{S}) \tag{B.7}$$

By analogy, a symmetric equation $\boldsymbol{\pi}(\mathbf{S}) = \mathbf{P}_{\mathbf{Sq}}\boldsymbol{\pi}(\mathbf{q})$ can also be easily proved in the same way from Eqn.(B.5) to Eqn.(B.7). Substituting both $\boldsymbol{\pi}(\mathbf{S}) = \mathbf{P}_{\mathbf{Sq}}\boldsymbol{\pi}(\mathbf{q})$ and $\boldsymbol{\pi}(\mathbf{q}) = \mathbf{P}_{\mathbf{qS}}\boldsymbol{\pi}(\mathbf{S})$ into Eqn.(B.4) gives the right part of Eqn.(5).

C Proof of Lemma 1.

Proof: We have shown in Appendix B that there exists an eigenvalue $\lambda = 1$ for the column-stochastic matrix $\mathbf{P}$ with equation $\mathbf{Px} = \mathbf{x}$. If we interprete the transition matrix as an adjacency matrix for the directed bipartite graph, the eigenvector centrality of that graph is $\mathbf{x}$.

We split the eigenvector $\mathbf{x}$ into $\mathbf{x_S}$, $\mathbf{x_q}$ for the bipartite vertex set $\mathbf{q}, \mathbf{S}$ respectively and the single-mode eigenvector centralities of the single vertex set can therefore be formulated by:

$$\hat{\mathbf{x}}_\mathbf{S} = \frac{\mathbf{x_S}}{\sum_{s\in\mathbf{S}} x_s} \qquad\qquad \hat{\mathbf{x}}_\mathbf{q} = \frac{\mathbf{x_q}}{\sum_{q\in\mathbf{q}} x_q} \tag{C.1}$$

If we left matrix product $\mathbf{P}$ on both sides of $\mathbf{Px} = \mathbf{x}$, we will have $\mathbf{P}^2\mathbf{x} = \mathbf{P}(\mathbf{Px}) = \mathbf{Px} = \mathbf{x}$. To write it in matrix notation, we have

$$\underbrace{\begin{pmatrix} \mathbf{P_{Sq}P_{qS}} & \mathbf{0} \\ \mathbf{0} & \mathbf{P_{qS}P_{Sq}} \end{pmatrix}}_{\mathbf{P}^2} \underbrace{\begin{pmatrix} \mathbf{x_S} \\ \mathbf{x_q} \end{pmatrix}}_{\mathbf{x}} = \underbrace{\begin{pmatrix} \mathbf{x_S} \\ \mathbf{x_q} \end{pmatrix}}_{\mathbf{x}} \tag{C.2}$$

Consider the first row of $\mathbf{P}^2$ matrix product with $\mathbf{x}$ in Eqn.(C.2), we have $\mathbf{P_{Sq}P_{qS}x_S} = \mathbf{x_S}$. Since $\boldsymbol{\pi}(\mathbf{S})$ is the eigenvector of $\mathbf{P_{Sq}P_{qS}}$ of eigenvalue 1 with probability constraint $\sum_{s\in\mathbf{S}}[\boldsymbol{\pi}(\mathbf{S})]_s = 1$, $\boldsymbol{\pi}(\mathbf{S})$ is exactly equivalent to the single-mode eigenvector centrality $\hat{\mathbf{x}}_\mathbf{S}$ in Eqn.(C.1).

By analogy, if we consider the matrix product between the second row of $\mathbf{P}^2$ and $\mathbf{x}$, we can prove $\boldsymbol{\pi}(\mathbf{q})$ equivalent to the conjugate single-mode eigenvector centrality $\hat{\mathbf{x}}_\mathbf{q}$ of the bipartite graph G.

D Supplementary proof of Eqn.(6)

Eqn.(6):

$$\mathbf{Pr}(\tilde{y} = c) = \lim_{t\to\infty} \frac{\sum_{z\in\mathbf{z}} \mathbb{E}\left[\sum_{k=1}^{t} \mathbb{1}[X_k \in \mathbf{s}^c] \middle| X_0 = z\right]}{\sum_{z\in\mathbf{z}} \mathbb{E}\left[\sum_{k=1}^{t} \mathbb{1}[X_k \in \mathbf{S}] \middle| X_0 = z\right]} = \sum_{s\in\mathbf{s}^c} [\boldsymbol{\pi}(\mathbf{S})]_s$$

Let $\mathbf{Pr}(\tilde{y} = c) \triangleq \lim_{t\to\infty} \mathbf{Pr}(t)$. Since we have shown $\{X_t\}$ is of 2 period in Appendix B, the proof of Eqn.(6) is thus equivalent to prove:

$$\lim_{t\to\infty} \mathbf{Pr}(2t) = \lim_{t\to\infty} \mathbf{Pr}(2t-1) = \sum_{s\in\mathbf{s}^c} [\boldsymbol{\pi}(\mathbf{S})]_s \tag{D.1}$$

Proof of even period: $\lim_{t\to\infty} \mathbf{Pr}(2t) = \sum_{s\in\mathbf{s}^c} [\boldsymbol{\pi}(\mathbf{S})]_s$ (most of which has been shown in Eqn.(6))

From the definition, we have

$$\begin{aligned} \mathbf{Pr}(2t) &= \frac{\frac{1}{|\mathbf{z}|}\sum_{z\in\mathbf{z}}\sum_{k=1}^{2t}\sum_{s\in\mathbf{s}^c} \left[\mathbf{P}^k\right]_{sz}}{\frac{1}{|\mathbf{z}|}\left(|\mathbf{q}|t + |\mathbf{S}|t\right)} \\ &= \frac{1}{t}\sum_{k=1}^{t}\left[\frac{1}{|\mathbf{z}|}\sum_{s\in\mathbf{s}^c}\left(\sum_{z\in\mathbf{S}}\left[\mathbf{P}^{2k}\right]_{sz} + \sum_{z\in\mathbf{q}}\left[\mathbf{P}^{2k-1}\right]_{sz}\right)\right] \end{aligned} \tag{D.2}$$

where $|\mathbf{q}|t$ is the number of visits from particles in $\mathbf{q}$ to support features in $\mathbf{S}$ after $2t$ steps of Markov bidirectional random walk. $|\mathbf{S}|t$ is the number of visits starting from particles in $\mathbf{S}$ to support features in $\mathbf{S}$. The second equality is derived from the diagonal/anti-diagonal property of $\mathbf{P}^{2k}$/$\mathbf{P}^{2k-1}$ respectively where the sub-matrices $\mathbf{0}$ are ignored in summation.

Taking it to the extreme, we have

$$\begin{aligned}\lim_{t\to\infty}\mathbf{Pr}(2t) &= \frac{1}{|\mathbf{z}|}\sum_{s\in\mathbf{s}^c}\left(\sum_{z\in\mathbf{S}}\left[\lim_{t\to\infty}\mathbf{P}^{2t}\right]_{sz}+\sum_{z\in\mathbf{q}}\left[\lim_{t\to\infty}\mathbf{P}^{2t-1}\right]_{sz}\right)\\ &= \sum_{s\in\mathbf{s}^c}\left[\boldsymbol{\pi}(\mathbf{S})\right]_s\end{aligned}\tag{D.3}$$

where the first equality in Eqn.(D.3) is derived from the *absorbing* of periodic Markov chain and the second equality is from the substitution of Eqn.(5).

Proof of odd period: $\lim_{t\to\infty}\mathbf{Pr}(2t-1)=\sum_{s\in\mathbf{s}^c}\left[\boldsymbol{\pi}(\mathbf{S})\right]_s$

From the definition, we have

$$\begin{aligned}\mathbf{Pr}(2t-1) &= \frac{\frac{1}{|\mathbf{z}|}\sum_{z\in\mathbf{z}}\sum_{k=1}^{2t-1}\sum_{s\in\mathbf{s}^c}\left[\mathbf{P}^k\right]_{sz}}{\frac{1}{|\mathbf{z}|}\left(|\mathbf{q}|t+|\mathbf{S}|(t-1)\right)}\\ &= \frac{1}{|\mathbf{z}|}\frac{1}{t-\frac{|\mathbf{S}|}{|\mathbf{z}|}}\sum_{s\in\mathbf{s}^c}\left[\sum_{z\in\mathbf{q}}\mathbf{P}_{sz}+\sum_{k=2}^{t}\left(\sum_{z\in\mathbf{S}}\left[\mathbf{P}^{2k-2}\right]_{sz}+\sum_{z\in\mathbf{q}}\left[\mathbf{P}^{2k-1}\right]_{sz}\right)\right]\end{aligned}\tag{D.4}$$

Take Eqn.(D.4) to the extreme, we have

$$\begin{aligned}\lim_{t\to\infty}\mathbf{Pr}(2t-1) &= \frac{1}{|\mathbf{z}|}\sum_{s\in\mathbf{s}^c}\left(\sum_{z\in\mathbf{S}}\left[\lim_{t\to\infty}\mathbf{P}^{2t-2}\right]_{sz}+\sum_{z\in\mathbf{q}}\left[\lim_{t\to\infty}\mathbf{P}^{2t-1}\right]_{sz}\right)\\ &= \sum_{s\in\mathbf{s}^c}\left[\boldsymbol{\pi}(\mathbf{S})\right]_s\end{aligned}\tag{D.5}$$

where $\lim_{t\to\infty}\frac{1}{t-\frac{|\mathbf{S}|}{\mathbf{z}}}\sum_{z\in\mathbf{q}}\mathbf{P}_{sz}=0$ is ignored when t approaches the infinity.

E Details of Experiment Settings.

E.1 Backbone Network.

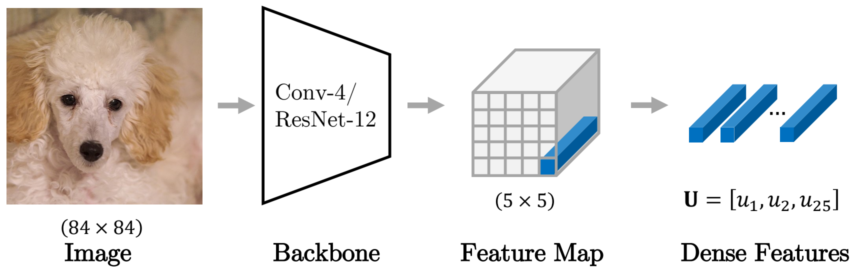
Conv-4. It contains four convolutional blocks, each of which consists of a convolutional layer, a batch normalization layer and a Leaky ReLU layer of parameter 0.2. Besides, for each convolutional blocks, an additional 2×2 max-pooling layer is appended, respectively. Given the image of size 84×84, Conv-4 outputs a feature map of size $5\times 5\times 64$.

ResNet-12. We use the same ResNet backbone as in previous literature. Although it is sometimes call ResNet-10 (*e.g.*, in DeepEMD) as they ignore the count of the last fully connected layer, they are indeed the same architecture. The network is composed of four residual blocks, each having three 3×3 convolutional layers with batch normalization and ReLU activation function. Each block is followed by 2×2 max-pooling. The shortcut connections have a convolutional layer to adapt to the right number of channels. The numbers of channels for each block are 64, 160, 320, 640 respectively. Given the image of size 84×84, ResNet-12 outputs a feature map of size $5\times 5\times 640$.

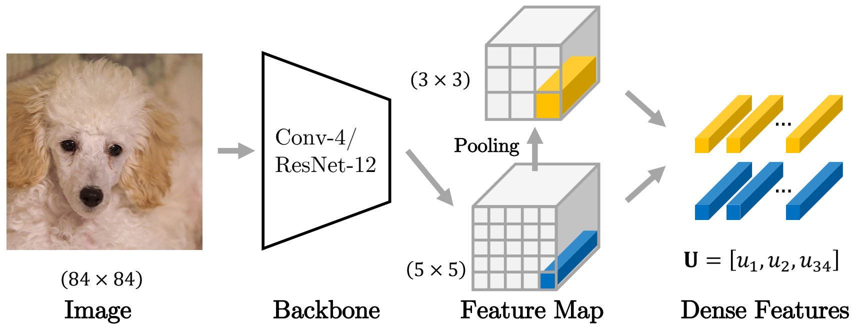
E.2 Dense Feature Extractor.

We compare three kinds of dense feature extractor in our work to demonstrate the effectiveness of our methods. We didn't investigate too much design choices in each extractors, *e.g.*, the size of grids in PyramidGrid, but mainly focus on their native designs as illustrated in Figure 3:

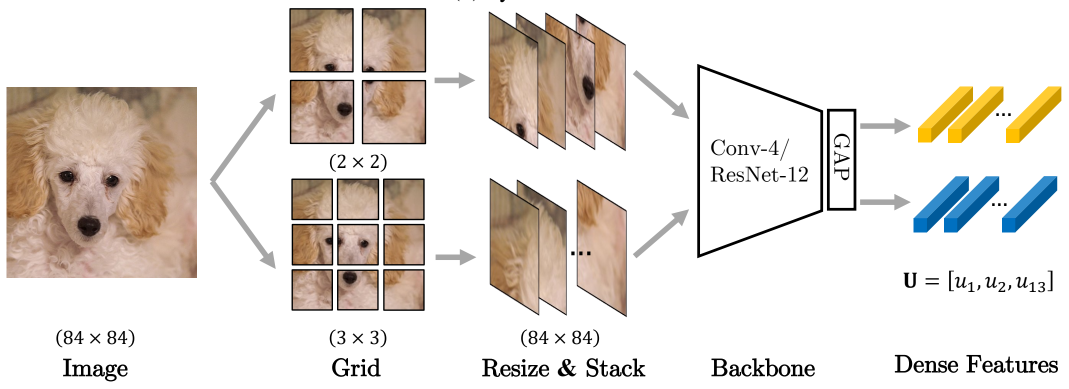
VanillaFCN simply treats the feature map output of the fully convolutional network as a set of dense features. For examples, there are 25 64-dimensional dense feature vectors for Conv-4 backbone and 25 640-dimensional dense feature vectors for ResNet-12 backbone.



(a) VanillaFCN



(b) PyramidFCN



(c) PyramidGrid

Figure 3: Dense feature extractors to extract local embeddings of the input image. (a) *VanillaFCN* simply treats the feature map output of fully convolutional network as dense features. (b) *PyramidFCN* applies pyramid structures on top of the native feature map to extract dense features of different scales. (c) *PyramidGrid* crops image into grid patches of different scales and encodes each patch to a feature vector with the embedding network. "GAP" indicates a single global average pooling layer.

PyramidFCN appends a feature pyramid structure at the end of VanillaFCN to extract local features from multiple image scales. Concretely, we use an extra adaptative average pooling layer of output size 3×3 to obtain 34 ($3 \times 3 + 5 \times 5$) channel-dimensional dense features for each image in our experiment.

PyramidGrid first crops the image evenly into an $H \times W$ grid before sending it to the embedding backbone and each image patch in the grid cell is encoded by the network individually. We add a global average pooling (GAP) layer at the end of the backbone so that each image patch will generate a feature vector. The feature vectors generated by all the patches constitute the set of dense features for each image. Like in PyramidFCN, we also adopt an pyramid structure of size $3 \times 3 + 2 \times 2$ in the experiments.

E.3 Training Procedure.

Conv-4 is trained from scratch with gradient descent. Specifically, the training is conducted on 30 epochs for both *mini*ImageNet and *tiered*ImageNet. Each epoch composes 20,000 episodes for training. In each episode, 15 and 10 query images will also be selected from each class for the 1-shot and 5-shot settings, respectively. In other words, for a 5-way 1-shot task, there will be 5 support images and 75 query images in one training episode. To train the Conv-4 model, we adopt Adam algorithm with an initial learning rate 1×10^{-3} and reduce it by 0.1 every 10 epochs.

ResNet-12. The networks are first pre-trained from scratch in a fully supervised manner like previous literature, *i.e.*, minimizing cross-entropy loss on the train split of a dataset. In meta-training stage, we use 30 epochs for *mini*ImageNet and 60 epochs for *tieredImageNet* respectively. Each epochs composes 200 episodes for the meta-training. We adopt the SGD with an initial learning rate 5×10^{-4} and reduce it by half every 10 epochs.

For data augmentation in both Conv-4 and ResNet-12, images are resized to 92×92, randomly cropped to 84×84 and randomly flipped along horizontal axis before used for training.

F Computational Speed and Scalability.

There are multiple ways to solve $\boldsymbol{\pi}(\mathbf{S})$ in Eqn.(6) and we discuss their computational speed and scalabilities in this section. Formally, solving the stationary distribution $\boldsymbol{\pi}(\mathbf{S})$ with probability constraints is equivalent to solving the $\mathbf{Ax} = \mathbf{b}$ like overdetermined linear system in Eqn.(9):

$$\underbrace{\begin{pmatrix} \boldsymbol{\Psi}_{\mathbf{S}} - \boldsymbol{I} \\ \boldsymbol{e}_{|\mathbf{S}|}^{\mathrm{T}} \end{pmatrix}}_{\mathbf{A}} \underbrace{\boldsymbol{\pi}(\mathbf{S})}_{\mathbf{x}} = \underbrace{\begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}}_{\mathbf{b}} \tag{F.1}$$

Here, we compare four different ways in solving this linear system:

1. Closed-form left pseudo inverse: $\mathbf{x}$ in Eqn.(F.1) has a closed-form solution $\mathbf{x} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$.

2. Deep learning pseudo-inverse package like `torch.pinverse`.

3. QR decomposition and back substitution: We first find $\mathbf{A} = \mathbf{QR}$. The solution $\mathbf{x}$ then can be expressed as $\mathbf{x} = \mathbf{R}^{-1}(\mathbf{Q}^{\mathrm{T}}\mathbf{b})$. Back substitution can be used to quickly and accurately find $\mathbf{x}$without explicitly inverting $\mathbf{R}$. In practice, we use the `torch.qr` and `torch.triangular_solve`.

4. Katz centrality approximation as introduced in Eqn.(13). At first glance, the most expensive step of MCL-Katz in Eqn.(10) is inverting $\boldsymbol{I} - \alpha\mathbf{P}$. Directly inverting this matrix costs $O(N^3r^3)$ where N is the number of supporting classes and r is the number of dense features for each image input. Fortunately, $\boldsymbol{I}-\alpha\mathbf{P}$ is a very special matrix that equals a diagonal matrix minus an anti-diagonal matrix. In this case, we can reformulate its inversion by:

$$\begin{aligned} (\boldsymbol{I} - \alpha\mathbf{P})^{-1} &= \begin{bmatrix} \boldsymbol{I} & -\alpha\mathbf{P}_{\mathbf{Sq}} \\ -\alpha\mathbf{P}_{\mathbf{qS}} & \boldsymbol{I} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \boldsymbol{I} + \alpha^2\mathbf{P}_{\mathbf{Sq}}(\boldsymbol{I} - \alpha^2\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}})^{-1}\mathbf{P}_{\mathbf{qS}} & \alpha\mathbf{P}_{\mathbf{Sq}}(\boldsymbol{I} - \alpha^2\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}})^{-1} \\ \alpha(\boldsymbol{I} - \alpha^2\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}})^{-1}\mathbf{P}_{\mathbf{Sq}} & (\boldsymbol{I} - \alpha^2\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}})^{-1} \end{bmatrix} \end{aligned} \tag{F.2}$$

where the reformulation is owed to the blockwise matrix inversion. The most expensive step becomes a $(\boldsymbol{I} - \alpha^2\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}) \in \mathbb{R}^{r\times r}$ matrix inversion and the time complexity is reduced to $O(r^3)$. The Pytorch pseudo-codes for Katz approximation can be found in Code 1.

It should be noted that those four mentioned MCL implementations won't have any differences in inference (except for their speed) but different in training. Among them, the closed-form left pseudo-inverse is comparably time efficient but may has numerical problem in back-propagation where extremely small values in inversion matrix cause gradient explosion; The SVD-based `torch.pinverse` would amplify the round-off error when dealing with the ill-conditioned matrix. Tricks like `clip_grad_norm` or pre-detect those ill-conditioned cases could slightly help mitigate the problem but we still found some performance drops. QR based method is a numerically stable

	Time per episode (ms)
VanillaFCN	622.3
ProtoNet	11.31
ProtoNet + MCL	16.75
MCL (closed-form)	15.43
MCL (torch.pinverse)	122.5
MCL (QR decomposition)	352.5
MCL (Katz approximation)	15.34

Table 6: Speed comparison between different on ResNet-12. Each class owns 15 query images in an 5-way 1-shot episode.

feature resolution r	*VanillaFCN* (ms)	DN4 (ms)	FRN (ms)	MCL (ms)
5×5	186.9	3.85	59.61	5.94
6×6	251.1	4.52	60.85	6.59
7×7	331.6	5.43	61.46	7.28
8×8	427.0	6.01	61.94	7.78
10×10	669.3	8.93	63.26	11.61
12×12	961.6	13.08	71.64	17.69

Table 7: Speed comparison for different input image resolutions in 5-way 1-shot FSL tasks with ResNet-12. Each class contains 4 query images in inference due to the limited GPU memory for large image resolutions.

way to directly solve the overdetermined system and doesn't suffer any of aforementioned problem. However, as shown in Table 6, it is quite time-consuming throughout the experiment. As a comparison, Katz approximation is not only time-efficient but also numerical stable. We provide thorough speed test for different cardinality r in Table 7 to show the scalability of proposed MCL.

```
# support of tensor shape [N, d, r]:
#     N-way FSL, each class owns r number of d-dimensional dense features
# query of tensor shape [q, d, r]:
#     q query examples, each of them owns r dense features.
#
# gamma: scaled similarity parameter
# tau: scaled similarity parameter
# alpha: Katz attenuation factor
# alpha_2: the square of alpha
#
# @: the matrix multiplication operator in Pytorch
def inner_cosine(query, support):
    N, d, r = support.shape
    q = len(query)
    query = query / query.norm(2, dim=-1, keepdim=True)
    support = support / support.norm(2, dim=-1, keepdim=True)

    support = support.unsqueeze(0).expand(q, -1, -1, -1)
    query = query.unsqueeze(1).expand(-1, N, -1, -1)
    S = query_xf.transpose(-2, -1)@support_xf
    S = S.permute(0, 2, 1, 3).contiguous().view(q, r, N * r)
    return S

def MCL_Katz_approx(query, support):
    N, d, r = support.shape
    q = len(query)
    S = inner_cosine(query, support) # [q, r, Nr]
    St = S.transpose(-2, -1) # [q, Nr, r]

    # column-wise softmax probability
    P_sq = torch.softmax(gamma * S, dim=-2)
    P_qs = torch.softmax(tau * St, dim=-2)
    # From the derivations in Eqn.(F.2)
    inv = torch.inverse(
        torch.eye(r)[None].repeat(q, 1, 1) - alpha_2 * P_qs@P_sq
    ) # [q, r, r]
    katz = (alpha_2 * P_sq@inv@P_qs).sum(-1) + (alpha * P_sq@inv).sum(-1)
    katz = katz / katz.sum(-1, keepdim=True)
    predicts = katz.view(q, N, r).sum(-1)
    return predicts
```

Code 1: Pytorch pseudo-code for 1-shot MCL (Katz approximation) in a single episode. The whole calculation is performed in parallel via batched matrix multiplication and inversion.

G THE ADVANTAGE OF BIDIRECTIONAL RANDOM WALK

In our proposed random walks, there are r particles random walking from query dense features to the support ones. If we only consider a unidirectional random walk, different starting q will end in support dense features with different probabilities. Since those r query dense features are actually from the same query image, an extra approach is needed for a joint prediction. Since the features' accessibility of unidirectional random walks is equivalent to average the probabilities from all starting query features, we consider it violates the intuition that different query dense features should own different importance.

In this paper, we proposed the bidirectional paradigm such that features' accessibility will converge to a Markov stationary distribution after infinite steps of random walks. An interesting property of stationary distribution is, whatever the initial data distribution is (*i.e.*, the distribution of q), they will converge to the same stationary distribution that is only determined by the transition matrix $\mathbf{P}$. If all starting q reach to the same stationary distribution, it is quite straightforward to treat that distribution as a joint prediction of $\mathbf{q}$.

Another interesting interpretation of $\boldsymbol{\pi}(\mathbf{S})$ in Eqn.(6) is from the equation proved in Appendix B that $\boldsymbol{\pi}(\mathbf{S}) = \mathbf{P}_{\mathbf{Sq}}\boldsymbol{\pi}(\mathbf{q})$ where $\boldsymbol{\pi}(\mathbf{q})$ is equivalent to measuring the importance of different $q \in \mathbf{q}$ (the single-mode centrality of query dense features) and $\mathbf{P}_{\mathbf{Sq}}\boldsymbol{\pi}(\mathbf{q})$ can be interpreted as an centrality weighted accumulation of different unidirectional random walk probabilities from different starting q.