Appendix for Mutual Centralized Learning in Few-shot Classifications

A. Proof of Eqn.(5)

Proof of periodic: Consider the eigenvalue $\lambda$ of $P$ by the determinant equation

$$\det(\lambda I - P) = \det\begin{pmatrix} \lambda I & -P_{Sq} \\ -P_{qS} & \lambda I \end{pmatrix} = \det(\lambda^2 I - P_{Sq} P_{qS}),$$

(A.1)

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

Since both $P_{Sq}$ and $P_{qS}$ are column-normalized matrices, their product is still column-stochastic that can be proved by:

$$e_{N^r}^T P_{Sq} P_{qS} = e_{r}^T P_{qS} = e_{N^r}^T$$

(A.2)

where $e_{i:j}$ is a vector of ones with different length indicated by its subscript. $N^r$ and $r$ are the cardinalities of $S$ and $q$, respectively.

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

Proof for even periods: We give the limit of matrix $P^{2t}$ for the extremely large number of $t$ as follows:

$$\lim_{t \to \infty} P^{2t} = \lim_{t \to \infty} \left( P_{Sq} P_{qS} \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right)^t = \left( \lim_{t \to \infty} P_{Sq} P_{qS} ight)^t \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

(A.3)

Since we have shown in Eqn.(A.2) that $P_{Sq} P_{qS}$ is also column-stochastic, we use $\pi(S)$ to denote its stationary distribution vector by equation $\lim_{t \to \infty} P_{Sq} P_{qS} = \pi(S) e_{N^r}^T$.

By analogy, the infinity power of $P_{Sq} P_{qS}$ could also reach a similar stationary distribution $\pi(q)$ with equation $\lim_{t \to \infty} P_{qS} P_{Sq} = \pi(q) e_{r}^T$.

Substituting the two stationary vectors into Eqn.(A.3), we can prove the stationary distributions of $P$ for the even periods in Eqn.(5).

Proof for odd periods: From the definition of matrix product, we first have

$$\lim_{t \to \infty} P^{2t-1} = \lim_{t \to \infty} P^{2t+1} = P \lim_{t \to \infty} P^{2t} = P \left( \begin{pmatrix} 0 & P_{Sq} \pi(q) e_{N^r}^T \\ P_{qS} \pi(S) e_{r}^T & 0 \end{pmatrix} \right)$$

(A.4)

Next, according to the definition of $\pi(q)$ and $\pi(S)$, we can get

$$\pi(q) e_{r}^T = P_{qS} \left( \lim_{t \to \infty} P_{Sq} P_{qS} \right)^t \begin{pmatrix} 0 \\ 0 \end{pmatrix} \begin{pmatrix} \pi(q) e_{N^r}^T \\ \pi(S) e_{r}^T \end{pmatrix}$$

(A.5)

If we right matrix product of $e_{r}$ on both sides of Eqn.(A.5), we have

$$\pi(q) e_{r}^T e_{r} = P_{qS} \pi(S) e_{N^r}^T P_{Sq} e_{r}$$

(A.6)

Since $e_{r}^T e_{r} = r$ and $e_{N^r}^T P_{Sq} e_{r} = \sum_I \sum_J [P_{Sq}]_{i,j} = r$, Eqn.(A.6) can be simplified by dividing the same scalar $r$ on both sides:

$$\pi(q) = P_{qS} \pi(S).$$

(A.7)

By analogy, a symmetric equation $\pi(S) = P_{Sq} \pi(q)$ can also be easily proved in the same way as from Eqn.(A.5) to Eqn.(A.7).

Substituting $\pi(S) = P_{Sq} \pi(q)$ and $\pi(q) = P_{qS} \pi(S)$ into Eqn.(A.4), we can prove the stationary distributions of $P$ for the odd periods in Eqn.(5).

B. Proof of Lemma 1

We have shown in Appendix A that there exists an eigenvalue $\lambda = 1$ for the column-stochastic matrix $P$ with equation $Px = x$. If we interpret the transition matrix as an adjacency matrix for the directed bipartite graph, the eigenvector centrality of that graph is $x$. 
We split the eigenvector $x$ into $x_S$, $x_q$ for the bipartite vertex set $q$, $S$ respectively and the single-mode eigenvector centralities of the single vertex set can therefore be formulated by:

$$
\hat{x}_S = \sum_{s \in S} x_s S \quad \hat{x}_q = \sum_{q \in q} x_q q
$$  \hfill (B.1)

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

$$
\begin{pmatrix}
P_{sq}P_{qs} & 0 \\ 0 & P_{qs}P_{sq}
\end{pmatrix}
\begin{pmatrix}
\hat{x}_S \\ \hat{x}_q
\end{pmatrix}
=
\begin{pmatrix}
\hat{x}_S \\ \hat{x}_q
\end{pmatrix}
$$  \hfill (B.2)

Consider the first row of $P^2$ matrix product with $x$ in Eqn.(B.2), we have $P_{sq}P_{qs}x_S = \hat{x}_S$. Since $\pi(S)$ is the eigenvector of $P_{sq}P_{qs}$ of eigenvalue 1 with probability constraint $e^T\pi(S) = 1$, $\pi(S)$ is exactly equivalent to the single-mode eigenvector centrality $\hat{x}_q$ of the bipartite graph.

C. Proof of Eqn.(6)

Eqn.(6) to prove:

$$
\Pr(\hat{y} = c) = \lim_{t \to \infty} \mathbb{E}_{z \in \mathbb{S}} \left[ \sum_{k=1}^{t} \mathbb{I}[X_k \in S'] \right]_{X_0 = z} = \sum_{s \in S'} [\pi(S)]_s
$$

We first define $\Pr(\hat{y} = c) \triangleq \lim_{t \to \infty} \Pr(t)$. Since we have proved that Markov process of transition matrix $P$ is of 2 period in Appendix A, the proof of Eqn.(6) is thus equivalent to prove:

$$
\lim_{t \to \infty} \Pr(2t) = \lim_{t \to \infty} \Pr(2t - 1) = \sum_{s \in S'} [\pi(S)]_s
$$  \hfill (C.1)

Proof for even period: From the definition, we have

$$
\Pr(2t) = \frac{1}{N_r + r} \sum_{s \in S'} \sum_{k=1}^{2t} \sum_{s \in S'} [P^k]_{sz}
$$

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

Taking Eqn.(C.2) to the extreme, we have

$$
\lim_{t \to \infty} \Pr(2t) = \frac{1}{N_r + r} \sum_{s \in S'} \left( \sum_{k=1}^{2t} \sum_{z \in q} [P^{2t}]_{sz} + \sum_{z \in q} [P^{2t-1}]_{sz} \right)
$$  \hfill (C.3)

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

Proof for odd period: From the definition, we have

$$
\Pr(2t - 1) = \frac{1}{N_r + r} \sum_{s \in S'} \sum_{k=1}^{2t-1} \sum_{z \in q} [P^k]_{sz}
$$

$$
= \frac{1}{N_r + r} \left( \sum_{s \in S'} \sum_{z \in q} [P^t]_{sz} + \sum_{z \in q} [P^{t-1}]_{sz} \right)
$$  \hfill (C.4)

where $\omega$ equals $(N_r + r)(t - \frac{N_r}{N_r + r})$. Take Eqn.(C.4) to the extreme, we have

$$
\lim_{t \to \infty} \Pr(2t - 1) = \frac{1}{N_r + r} \sum_{s \in S'} \sum_{z \in q} [P^t]_{sz} + \sum_{z \in q} [P^{t-1}]_{sz} = \sum_{z \in q} [\pi(S)]_s
$$  \hfill (C.5)

where $\lim_{t \to \infty} t - \frac{N_r}{N_r + r} \sum_{z \in q} P_{sz} = 0$ is ignored when $t$ approaches the infinity.

D. Pseudo codes

We use block-wise inversion in Eqn.(14) that is more computational-efficient than directly inverting Eqn.(10) when the number of support classes $N$ is large. The corresponding Pytorch-code can be found above where the whole calculation is performed in parallel via batched matrix multiplication and inversion.
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
- beta: scaled similarity parameter
- alpha: Katz attenuation factor
- alpha_2: the square of alpha

@: the matrix multiplication operator in Pytorch

```python
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, N * r]
    St = S.transpose(-2, -1)  # [q, N * r, r]
    # column-wise softmax probability
    P_sq = torch.softmax(gamma * St, dim=-2)
    P_qs = torch.softmax(beta * S, 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)
    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.

### E. Implementation details

**Preprocessing:** During training on CUB, miniImageNet and tieredImageNet, images are randomly cropped to $92 \times 92$ and then resized into $84 \times 84$. For meta-iNat and tiered-meta-iNat, images are randomly padded then cropped into $84 \times 84$. Unlike previous methods, we only random horizontal flip the image during training.

During inference, images are center cropped to $92 \times 92$ and then resized into $84 \times 84$ for CUB, miniImageNet and tieredImageNet. For meta-iNat and tiered-meta-iNat, images are already $84 \times 84$ and are fed into the models directly.

**Network backbones:** We use two backbones in our experiments: Conv-4 and ResNet-12. Conv-4 contains four convolutional blocks, each of which consists of a $3 \times 3$ convolution, a BatchNorm, a LeakyReLU(0.2) and an additional $2 \times 2$ max-pooling. ResNet-12 consist four residual blocks, each with three convolutional layers, with LeakyReLU(0.1) and $2 \times 2$ max-pooling on the main stem. Given the image of input size $84 \times 84$, Conv-4 outputs a feature map of size $5 \times 5 \times 64$ while ResNet-12 outputs that of size $5 \times 5 \times 64$.

**Re-implemented baselines:** We re-implement ProtoNet [9] and RelationNet [10] in our unified framework as two global feature based baselines for our centrality plugin. We borrow most of codes from their official implementations and introduce slight modifications to improve their performances inspired their subsequent work [13, 14]. For ProtNet, we introduce a fixed temperature scaling with $1/64$ before in softmax function. For RelationNet, we change the original MSELoss to CrossEntropyLoss.

**Pre-training:** We use the pre-training + meta-training procedure for ResNet-12 backbones on miniImageNet and tieredImageNet like most of the methods in the literature [6, 13, 15]. We follow the same pre-training technique from the dense features based FRN [13] to learn spatially distinctive
We did not use pre-training for Conv-4, the number of samples per epoch of Conv-4 is far larger than ResNet-12 for convergences. In each episode, besides $K$ support images in each class, 15 query images will also be selected from each class.

For Conv-4, we adopt Adam optimizer with initial learning rate of 1e-3 to train for 30 epochs (on both datasets) and reduce it by a factor of 10 every 10 epochs.

For ResNet-12, we adopt SGD with initial learning rate of 1e-3 to train for 300, 400 and 300 epochs. We set 0.5 learning rate decay every 20 epochs. Both Conv-4 and ResNet-12 are trained with 10-way for both 1-shot and 5-shot models.

**F. Cross-Domain Few-shot Classification**

We also evaluate on the challenging cross-domain setting proposed by [3], where models trained on miniImageNet base classes are directly evaluated on test classes from CUB. We use the same test class split as in [13] for fair comparisons, which is much harder than the test class split in [3].

As shown in Table 1, our MCL outperforms previous state-of-the-art methods by large margins of 3.9% on the 1-shot task and 5.4% on the 5-shot task, respectively.

**G. Evaluation without Meta-training**

Given that an increasing number of methods simply use standard supervised learning to pre-train the feature extractor and then use their methods directly for evaluation without meta-training [3, 4], we also evaluate our methods under this setting with the same pre-trained feature extractor we used in the Table 1. As shown in Table S2, global feature based methods are likely to misclassify images under the extremely 1-shot scenario, where the significant intra-class variations would inevitably drive the image-level embedding from the same category far apart. In contrast, dense feature based methods provide more information across categories that shows promising performances in that scenario.

Our end-to-end Katz centrality based MCL outperforms previous methods by a margin of 1.8% and 1.3% on 1-shot and 5-shot tasks, respectively. It is interesting to note that our MCL plugin help centralize the task-relevant local features.
### H. Ablation on parameters $\alpha$, $\gamma$ and $\beta$

#### Ablation on Katz attenuation factor $\alpha$

Ablation studies on the Katz attenuation factor $\alpha$ (Table S3) show that different $\alpha$ in bidirectional random walks, as discussed in Sec.5, lead to different centrality with $\alpha=0$ is more influenced by the in-degree topology while that with $\alpha>0$ is more influenced by the in-degree paths. It could be observed that the optimal $\alpha$ differs across various FSL tasks (that differ with shot on different datasets). In the experiments, we use $\alpha=0.999$ to approximate single-mode eigenvector centrality in Eqn.(6) and simply use $\alpha=0.5$ to represent general Katz centrality in Eqn.(8).

#### Ablation on scaling parameters $\gamma$ and $\beta$

Ablation studies of MCL ($\alpha=0.999$) on the scaling parameters $\gamma$ and $\beta$ (Table S4) show different $\gamma$ and $\beta$ in Eqn.(1) and Eqn.(2), respectively. Experiments are conducted on 5-way 1-shot miniImageNet with VanillaFCN.

#### I. Additional Plugin Experiments

We have shown that our proposed centrality weighted pooling has a consistent performance gain (especially in the extreme 1-shot scenario) over global average pooling on ProtoNet [9] and RelationNet [10] by concentrating on more task-relevant local features. Besides Table 1, 2, 4 and S2, we give additional results in Table S5 to show that MCL can be easily plugged into those methods that need no extra parameters except for the feature extractor.

The experiments are conducted by the following rules: all comparing methods are evaluated with the same pre-trained backbone ResNet-12 as in Sec.C without meta-training; we only use centrality weighted pooling to aggregate local features from query image as different methods have different operations on support features; we fixed the parameters $\gamma=20$ and $\beta=10$ to MCL plugins for all comparing methods.

As shown in Table S5, our proposed centrality plugin consistently improves the performances of all the global feature based methods without bells and whistles.

#### J. Additional Visualization

The additional Grad-CAM visualizations as in Table 5 of the main paper are presented as follows: Table S6 illustrates end-to-end MCL-Katz like in Table 5(b). Table S7 illustrates MCL plugin on ProtoNet [9] like in Table 5(a).
Table S6. Additional Grad-CAM visualizations of MCL-katz on 5-way 1-shot FSL tasks (miniImageNet) with ResNet-12. Formatting follows Table 5(b): query images are placed in the first column for each task; ground truth support images are placed in the second column; the four images on the far right column of each task are from the confounding support classes.
Table S7. Additional ProtoNet+MCL Grad-CAM visualizations on 5-way 1-shot FSL tasks (miniImageNet) with ResNet-12. Formatting follows Table 5(a): query images are placed in the first column for each task; ground truth support images are placed in the second column; the four images on the far right column of each task are from the confounding support classes. The top row in each task is from ProtoNet while the bottom row is from ProtoNet+MCL.
References


