Appendix for Mutual Centralized Learning in Few-shot Classifications

A. Proof of Eqn.(5)

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

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

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

$$\boldsymbol{e}_{Nr}^{\mathrm{T}}\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}} = \boldsymbol{e}_{r}^{\mathrm{T}}\mathbf{P}_{\mathbf{qS}} = \boldsymbol{e}_{Nr}^{\mathrm{T}} \tag{A.2}$$

where $\boldsymbol{e}_{|\cdot|}$ is a vector of ones with different length indicated by its subscript. Nr and r are the cardinalities of $\mathbf{S}$ and $\mathbf{q}$, respectively.

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.(A.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 prove its stationary distribution is of period 2.

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

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

Since we have shown in Eqn.(A.2) that $\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}$ is also column-stochastic, we use $\boldsymbol{\pi}(\mathbf{S})$ to denote its stationary distribution vector by equation $\lim\limits_{t\to\infty} [\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}]^t = \boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{Nr}^{\mathrm{T}}$.

By analogy, the infinity power of $\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}}$ could also reach a similar stationary distribution $\boldsymbol{\pi}(\mathbf{q})$ with equation $\lim\limits_{t\to\infty} [\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}}]^t = \boldsymbol{\pi}(\mathbf{q})\boldsymbol{e}_{r}^{\mathrm{T}}$.

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

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

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

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

$$\begin{aligned}\boldsymbol{\pi}(\mathbf{q})\boldsymbol{e}_{r}^{\mathrm{T}} &= \lim_{t\to\infty} [\mathbf{P}_{\mathbf{qS}}\mathbf{P}_{\mathbf{Sq}}]^t \\ &= \mathbf{P}_{\mathbf{qS}} \left(\lim_{t\to\infty} [\mathbf{P}_{\mathbf{Sq}}\mathbf{P}_{\mathbf{qS}}]^t \right) \mathbf{P}_{\mathbf{Sq}} \\ &= \mathbf{P}_{\mathbf{qS}}\boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{Nr}^{\mathrm{T}}\mathbf{P}_{\mathbf{Sq}}.\end{aligned} \tag{A.5}$$

If we right matrix product of $\boldsymbol{e}_r$ on both sides of Eqn.(A.5), we have

$$\boldsymbol{\pi}(\mathbf{q})\boldsymbol{e}_{r}^{\mathrm{T}}\boldsymbol{e}_r = \mathbf{P}_{\mathbf{qS}}\boldsymbol{\pi}(\mathbf{S})\boldsymbol{e}_{Nr}^{\mathrm{T}}\mathbf{P}_{\mathbf{Sq}}\boldsymbol{e}_r \tag{A.6}$$

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

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

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 as from Eqn.(A.5) to Eqn.(A.7).

Substituting $\boldsymbol{\pi}(\mathbf{S}) = \mathbf{P}_{\mathbf{Sq}}\boldsymbol{\pi}(\mathbf{q})$ and $\boldsymbol{\pi}(\mathbf{q}) = \mathbf{P}_{\mathbf{qS}}\boldsymbol{\pi}(\mathbf{S})$ into Eqn.(A.4), we can prove the stationary distributions of $\mathbf{P}$ 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 $\mathbf{P}$ with equation $\mathbf{P}\mathbf{x} = \mathbf{x}$. If we interpret 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:

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

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

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

Consider the first row of $\mathbf{P}^2$ matrix product with $\mathbf{x}$ in Eqn.(B.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 $e_{Nr}^{\mathrm{T}}\boldsymbol{\pi}(\mathbf{S}) = 1$, $\boldsymbol{\pi}(\mathbf{S})$ is exactly equivalent to the single-mode eigenvector centrality $\hat{\mathbf{x}}_\mathbf{S}$ in Eqn.(B.1).

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

C. Proof of Eqn.(6)

Eqn.(6) to prove:

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

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

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

Proof for even period: From the definition, we have

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

where rt is the number of visits from particles in $\mathbf{q}$ to support features in $\mathbf{S}$ after $2t$ steps of Markov bidirectional random walk. Nrt 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 Eqn.(C.2) to the extreme, we have

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

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

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

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

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

where $\lim_{t\to\infty}\frac{1}{t-\frac{Nr}{Nr+r}}\sum_{z\in\mathbf{q}}\mathbf{P}_{sz} = 0$ 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
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 * 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
    ) # [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.

E. Implementation details

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

During inference, images are center cropped to 92×92 and then resized into 84×84 for CUB, *mini*ImageNet and *tiered*ImageNet. For meta-iNat and *tiered*-meta-iNat, images are already 84×84 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 3×3 convolution, a BatchNorm, a LeakyReLU(0.2) and an additional 2×2 max-pooling. ResNet-12 consist four residual blocks, each with three convolutional layers, with LeakyReLU(0.1) and 2×2 max-pooling on the main stem. Given the image of input size 84×84, 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 *mini*ImageNet and *tiered*ImageNet 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

Model	Backbone	1-shot	5-shot
Baseline [3]	ResNet-10	-	65.57 ± 0.70
Baseline++ [3]	ResNet-18	-	62.04 ± 0.76
MetaOptNet [5]	ResNet-12	44.79 ± 0.75	64.98 ± 0.68
MatchingNet+FT [11]	ResNet-10	36.61 ± 0.53	55.23 ± 0.83
RelationNet+FT [11]	ResNet-10	44.07 ± 0.77	59.46 ± 0.71
GNN+FT [11]	ResNet-10	47.47 ± 0.75	66.98 ± 0.68
Neg-margin [7]	ResNet-18	-	69.30 ± 0.73
Centroid *et al*. [1]	ResNet-18	46.85 ± 0.75	70.37 ± 1.02
FRN [13]	ResNet-12	51.60 ± 0.21	72.97 ± 0.18
ProtoNet† [9]	ResNet-12	40.05 ± 0.18	55.29 ± 0.19
ProtoNet+MCL	ResNet-12	42.02 ± 0.19	64.76 ± 0.20
MCL	ResNet-12	**55.48 ± 0.22**	75.93 ± 0.18
MCL-Katz	ResNet-12	53.22 ± 0.22	**77.39 ± 0.18**

Table S1. Few-shot classifications (%) in the cross-domain setting: *mini*ImageNet→CUB. †: our re-implemented results in our unified framework that share the same dataloader and training strategies.

Methods	$f_\theta(\cdot)$	1-shot	5-shot
Baseline [3]	*Global feature*	53.99 ± 0.20	78.78 ± 0.15
Baseline++ [3]		55.03 ± 0.20	78.79 ± 0.15
ProtoNet [9]		53.99 ± 0.20	79.70 ± 0.15
ProtoNet+MCL		58.62 ± 0.20	80.99 ± 0.14
DN4 [6]	*VanillaFCN*	58.95 ± 0.19	78.01 ± 0.15
FRN [13]		59.71 ± 0.19	74.05 ± 0.15
MCL		60.94 ± 0.20	80.20 ± 0.14
MCL-Katz		**61.55 ± 0.20**	**81.09 ± 0.14**

Table S2. Few-shot classifications (%) without episodic meta-training. All of the comparing methods are evaluated in our unified framework with the same pre-trained ResNet-12 backbone.

dense features, *e.g*., run 350 epochs of batch size 128 on *mini*ImageNet, using SGD with initial learning rate 0.1 and decaying by a factor of 10 at epochs 200 and 300.

Meta-training on *mini*-/*tiered*ImageNet: We follow the earliest dense feature based DN4 [6] that randomly sample 20,000/200 episodes in an epoch for Conv-4/ResNet-12 backbone, respectively. Since we didn not use pre-training for Conv-4, the number of episodes 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 5e-4 (40 epochs on *mini*ImageNet and 60 epochs on *tiered*ImageNet) and cut it by half every 10 epochs.

Unlike the latest dense feature based FRN that adopts larger way during meta-training (*e.g*., 25-way for training 1-shot models and 20-way for 5-shot models), we did not use that setting in widely used *mini*-/*tiered*ImageNet as we thought it would be unfair in comparing with other methods.

Meta-training on fine-grained datasets: We follow the latest fine-grained few-shot classification setting from FRN [13] as most of the comparing performances in Table 2 are from them. Although practitioners agree on the train/validation/test split ratio (*i.e*., 100/50/50) on CUB dataset, there is no official class split. In our experiments, we use the same train/val/test class split as in [13] for a fair comparison.

For CUB, we train all our Conv-4 models for 1200 epochs using SGD with Nesterov momentum of 0.9 and an initial learning rate of 0.1. The learning rate decreases by a factor of 10 at epoch 400 and 800. ResNet-12 backbone trains for 600 epochs and scale down the learning rate by 10 at epoch 300, 400 and 500. Conv-4 backbone is trained with standard 20-way for 5-shot models and is trained with 30-way for 1-shot models like [13], while ResNet-12 backbone is trained with 10-way for the both shots models.

For meta-iNat and *tiered*-meta-iNat, we train our Conv-4 and ResNet-12 models for 100 epochs using Adam with initial learning rate 1e-3. 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 *mini*ImageNet 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 S1, 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 interested to note that our MCL plugin help centralize the task-relevant local features

	5-way 1-shot	5-way 5-shot
$\alpha \approx 0$ (unidirectional)	66.60 $\pm$ 0.20	81.76 $\pm$ 0.13
$\alpha = 0.1$	67.13 $\pm$ 0.20	83.20 $\pm$ 0.13
$\alpha = 0.3$	67.28 $\pm$ 0.20	83.89 $\pm$ 0.13
$\alpha = 0.5$ (MCL-Katz)	**67.51** $\pm$ **0.20**	83.99 $\pm$ 0.13
$\alpha = 0.7$	67.27 $\pm$ 0.20	**84.05** $\pm$ **0.13**
$\alpha = 0.9$	67.41 $\pm$ 0.20	83.96 $\pm$ **0.13**
$\alpha = 0.999$ (MCL)	67.36 $\pm$ 0.20	83.63 $\pm$ 0.13

Table S3. Ablation studies on the Katz attenuation factor α on *mini*ImageNet with *VanillaFCN* ResNet-12.

	$\beta = 5.0$	$\beta = 10.0$	$\beta = 20.0$	$\beta = 50.0$
$\gamma = 5.0$	66.12	66.15	65.87	65.39
$\gamma = 10.0$	67.21	67.20	66.78	66.23
$\gamma = 20.0$	67.53	**67.36**	66.70	66.00
$\gamma = 50.0$	67.05	66.77	65.84	65.08

Table S4. Ablation studies of MCL ($\alpha = 0.999$) on the scaling parameters γ and β in Eqn.(1) and Eqn.(2), respectively. Experiments are conducted on 5-way 1-shot *mini*ImageNet with *VanillaFCN*.

Method	*mini*ImageNet		*tiered*ImageNet	
	1-shot	5-shot	1-shot	5-shot
Baseline [3]	53.99	78.78	67.75	85.23
+MCL	56.21+2.22	80.44+1.66	68.53+0.78	85.75+0.52
Baseline++ [3]	55.03	78.79	61.86	84.31
+MCL	56.93+1.90	79.93+1.14	63.00+1.14	84.61+0.30
ProtoNet [9]	56.50	79.68	64.27	84.01
+MCL	58.62+2.12	80.99+1.30	66.48+2.21	84.83+0.82
MatchingNet [12]	58.41	79.52	68.68	85.16
+MCL	59.85+1.44	80.79+1.27	69.45+0.97	85.49+0.33
R2D2 [2]	59.82	78.97	70.22	85.30
+MCL	60.86+1.04	80.65+1.68	70.64 +0.42	85.84+0.54
MetaOptNet [5]	59.59	79.77	69.55	85.25
+MCL	60.66+1.07	81.27+1.50	70.11+0.55	85.76+0.51
DSN [8]	58.70	79.01	69.13	85.14
+MCL	60.08+1.38	80.66+1.65	69.59+0.46	85.61+0.47
Neg-cosine [7]	58.82	79.63	69.44	84.94
+MCL	60.20+1.38	81.04+1.41	69.85+0.41	85.31+0.37

Table S5. Few-shot classifications (%) before and after applying centrality weighted pooling. Experiment settings follow Table S2.

in ProtoNet [9] by a large margin of **4.6**% in 1-shot task and **1.2**% in 5-shot task without bell and whistles.

H. Ablation on parameters α, γ and β

Ablation on Katz attenuation factor α. Table S3 shows results with different α in bidirectional random walks. As discussed in Sec.5, the centrality with large α is more influenced by the endogenous topology while that with small α is more influenced by the in-degree paths. It could be observed in Table S3 that, the optimal α differs across various FSL tasks (that with different shot on different datasets).

In the experiments, we use α=0.999 to approximate single-mode eigenvector centrality in Eqn.(6) and simply use α=0.5 to represent general Katz centrality in Eqn.(8).

Ablation on scaling parameters γ and β. Table S4 shows MCL results with different scaling parameters γ and β. As discussed in Sec.5 that a larger scaling parameter (*i.e.*, a smaller temperature in softmax-like random walk probability) will lead to a more concentrated eigenvector centrality. However, a larger scaling parameter would inevitably bias the meta-training. Thus, there exists a trade-off to select optimal parameters according to each task.

In the experiments, we find its empirically effective to select γ and β according to their pre-trained models (similar to Table S4). In most cases, we use γ=20, β=10 and γ=40, β=20 for 1-shot and 5-shot tasks, respectively.

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.G 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 γ=20 and β=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).

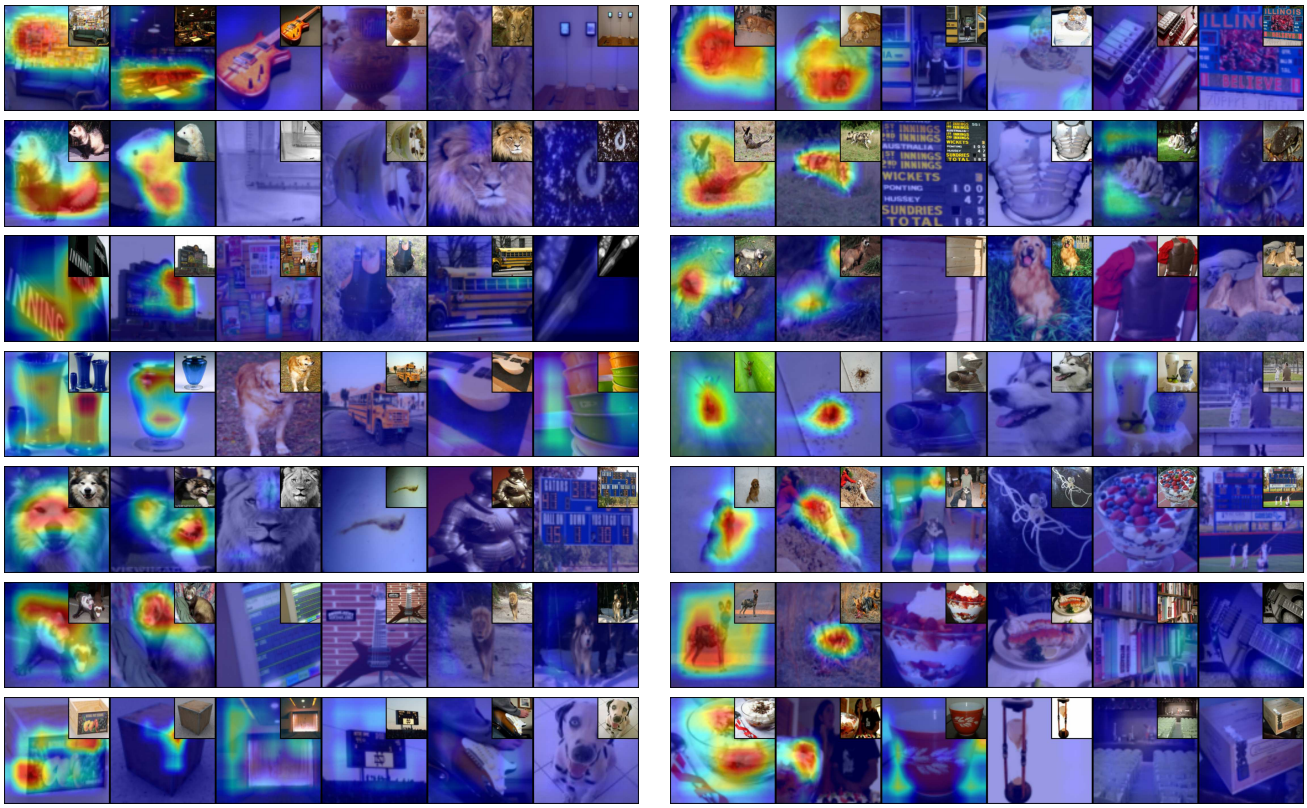

Table S6. Additional Grad-CAM visualizations of MCL-katz on 5-way 1-shot FSL tasks (*mini*ImageNet) with ResNet-12. Formatting follows Table 5(b): query images are placed in the first column for each task; ground truth support images are placed in the second column; the four images on the far right column of each task are from the confounding support classes.

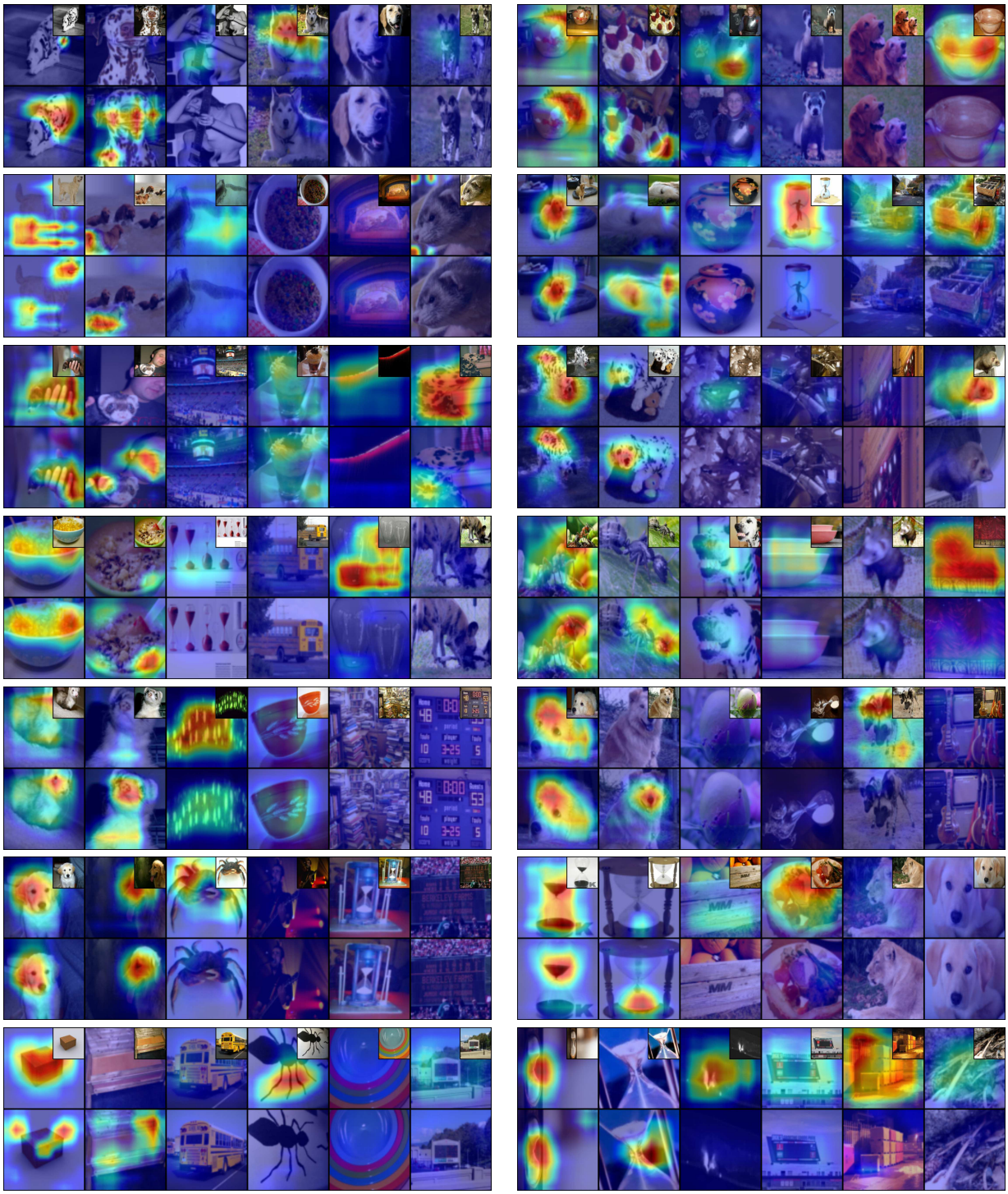

Table S7. Additional **ProtoNet+MCL** Grad-CAM visualizations on 5-way 1-shot FSL tasks (*mini*ImageNet) with ResNet-12. Formatting follows Table 5(a): query images are placed in the first column for each task; ground truth support images are placed in the second column; the four images on the far right 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

[1] Arman Afrasiyabi, Jean-François Lalonde, and Christian Gagn'e. Associative alignment for few-shot image classification. In *European Conference on Computer Vision*, pages 18–35. Springer, 2020. 4

[2] Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2018. 5

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *International Conference on Learning Representations*, 2019. 4, 5

[4] Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *International Conference on Learning Representations*, 2019. 4

[5] 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*, pages 10657–10665, 2019. 4, 5

[6] 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*, pages 7260–7268, 2019. 3, 4

[7] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *European Conference on Computer Vision*, pages 438–455. Springer, 2020. 4, 5

[8] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *CVPR*, pages 4136–4145, 2020. 5

[9] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087, 2017. 3, 4, 5

[10] 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*, pages 1199–1208, 2018. 3, 5

[11] Hung-Yu Tseng, Hsin-Ying Lee, Jia-Bin Huang, and Ming-Hsuan Yang. Cross-domain few-shot classification via learned feature-wise transformation. In *International Conference on Learning Representations*, 2020. 4

[12] 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*, pages 3637–3645, 2016. 5

[13] 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*, pages 8012–8021, 2021. 3, 4

[14] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*, pages 8808–8817, 2020. 3

[15] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Differentiable earth mover's distance for few-shot learning. *arXiv e-prints*, pages arXiv–2003, 2020. 3