The More, The Better? Active Silencing of Non-Positive Transfer for Efficient Multi-Domain Few-Shot Classification

Xingxing Zhang*
Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, THBI Lab, Tsinghua University
Beijing, China
xxzhang2020@mails.tsinghua.edu.cn

Zhizhe Liu*
Institute of Information Science, Beijing Jiaotong University
Beijing, China
zhzliu@bjtu.edu.cn

Weikai Yang
School of Software, BNRist Center, Tsinghua University
Beijing, China
vicayang496@gmail.com

Liyuan Wang
Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, THBI Lab, Tsinghua University
Beijing, China
wly19@mails.tsinghua.edu.cn

Jun Zhu†
[1]Dept. of Comp. Sci. & Tech., Institute for AI, BNRist Center, THBI Lab, Tsinghua University, Beijing, China
[2]Peng Cheng Laboratory, Pazhou Laboratory (Huangpu)
Guangzhou, China
dcszj@tsinghua.edu.cn

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; *Image representations.*

KEYWORDS

multi-domain, pre-trained knowledge, subset selection, efficient few-shot learning

ACM Reference Format:
Xingxing Zhang, Zhizhe Liu, Weikai Yang, Liyuan Wang, and Jun Zhu. 2022. The More, The Better? Active Silencing of Non-Positive Transfer for Efficient Multi-Domain Few-Shot Classification. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3503161.3548349

1 SINGLE-DOMAIN FEW-SHOT CLASSIFICATION

In this section, we discuss previous work on few-shot classification in the single-domain setting. It recognizes samples from several novel classes using only a few labeled samples, relying on **an extra labeled dataset** from other classes (*aka.* base classes). A wide variety of advanced methods have been proposed and significantly improved the few-shot classification performance on multiple benchmark datasets. In general, these typical methods can be roughly divided into three groups, *i.e.*, fine-tuning based methods [3, 5, 16, 24, 29], meta-learning based methods [9, 10, 14, 18, 19, 21, 22, 28], and metric-learning based methods [12, 13, 15, 16, 20, 23, 26, 27, 30].

Fine-tuning based methods [3, 5, 16, 24, 29] follow the standard paradigm of transfer learning, and consist of two stages, *i.e.*, pre-training with base classes and fine-tuning with novel classes. Of note, because of the extreme scarcity of training samples in the target few-shot task, the pre-trained embedding parameters are generally fixed in fine-tuning to avoid over-fitting. Representative methods contain Baseline [3], Baseline++ [3], RFS-simple [3], *etc.* The main difference between them is that they use different classifiers (*e.g.*, linear classifier or logistic regression) in fine-tuning. Although achieving good accuracy, the cross-entropy loss used in pre-training may make the model overfit on base classes, thus lacking generalization ability for novel classes. Meta-learning based methods [9, 10, 14, 18, 19, 21, 22, 28] normally perform a meta-learning paradigm on a sequence of few-shot tasks constructed from base classes, aiming to make the learned across-task meta-knowledge adapt to a new few-shot task. MAML [9] is one popular representative method by training a model's initial parameters with one or a few gradient steps. Other representative methods include Versa [10], R2D2 [1], LEO [19], MTL [22], ANIL [18], OVE [21], *etc.* Compared with those early meta-learning based methods that learn from scratch, the recent methods, such as MTL [22] and LEO [19], enjoy considerable improvements with the pre-training technique. Different from the two-loop structure of meta-learning based methods, metric-learning based methods [12, 13, 15, 16, 20, 23, 26, 27, 30] directly compare the distances between the query samples and given shots through one single feed-forward pass. ProtoNet [20] is a typical one, which takes the mean vector of shots as its corresponding prototype representation, and then compare the relationships between the query image and prototypes. Other representative methods include MatchingNet [26], RelationNet [23], DN4 [15], *etc.* Note that since there are no data-dependent parameters in the classifier (*i.e.*, 1-NN), the metric-learning based methods do not have the fine-tuning procedure, thus enjoying an efficient test stage. In

*Equal contribution.
†Corresponding author.

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548349

fact, the few-shot classifier in our proposal is also based on metric learning, where we assign an input sample to the closest centroid using cosine similarity.

2 SOLVING BASE CLASS SELECTION PROBLEM

The base class selection problem is formulated as

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \sum_{j=1}^{C}\sum_{i=1}^{B} d_{ij} z_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{B} z_{ij} = 1,\ \forall j;\ z_{ij} \in [0,1],\ \forall i,j;\ \sum_{i=1}^{B} \max_{j} z_{ij} \le \tau, \end{aligned} \tag{1}$$

where $\tau \in [0, B]$ is the desired number of selected base classes, $\boldsymbol{D} = \{d_{ij}\}_{i=1,\cdots,B}^{j=1,\cdots,C}$, and $\boldsymbol{Z} = \{z_{ij}\}_{i=1,\cdots,B}^{j=1,\cdots,C}$. In essence, problem (1) is a convex relaxation of problem (2) via Lagrange multiplier, to minimize the total representation cost given a selection 'budget' τ.

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \sum_{j=1}^{C}\sum_{i=1}^{B} d_{ij} z_{ij} \ + \ \lambda \sum_{i=1}^{B} \max_{j} z_{ij} \\ \text{s.t.} \quad & \sum_{i=1}^{B} z_{ij} = 1,\ \forall j;\ z_{ij} \in [0,1],\ \forall i,j, \end{aligned} \tag{2}$$

The reason is as follows. Augmenting the last inequality constraint of problem (1) to the objective function via the Lagrange multiplier $\lambda \in \mathbb{R}$, we can write the Lagrangian function of problem (1) as

$$\begin{aligned} \mathcal{L}(\boldsymbol{Z}) &= \sum_{j=1}^{C}\sum_{i=1}^{B} d_{ij} z_{ij} \ + \ \lambda\Big(\sum_{i=1}^{B} \max_{j} z_{ij} - \tau\Big) \\ &= \sum_{j=1}^{C}\sum_{i=1}^{B} d_{ij} z_{ij} \ + \ \lambda \sum_{i=1}^{B} \max_{j} z_{ij} - \lambda\tau, \end{aligned} \tag{3}$$

where $\boldsymbol{Z}$ is subject to the probability simplex constraints. Thus, minimizing $\mathcal{L}(\boldsymbol{Z})$ is equivalent to solving problem (2). Besides, τ can represent the size of selected subset, since $\sum_{i=1}^{B} \max_j z_{ij}$ approximately measures the number of nonzero rows in $\boldsymbol{Z}$, and the selected subset is indexed by the nonzero rows in $\boldsymbol{Z}$.

Next, we show how to solve problem (1) using the ADMM approach in [8, 31]. To do so, an auxiliary matrix $\boldsymbol{A} = \{a_{ij}\}_{i=1,\cdots,B}^{j=1,\cdots,C}$ is introduced to problem (1) to rewrite the problem as

$$\begin{aligned} \min_{\boldsymbol{Z},\boldsymbol{A}} \quad & \sum_{j=1}^{C}\sum_{i=1}^{B} d_{ij} a_{ij} + \frac{\mu}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2 \\ s.t. \quad & \mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}};\ \boldsymbol{A} \ge \mathbf{0};\ \boldsymbol{Z} = \boldsymbol{A};\ \|\boldsymbol{Z}\|_{1,\infty} \le \tau, \end{aligned} \tag{4}$$

where we define $\|\boldsymbol{Z}\|_{1,\infty} = \sum_{i=1}^{B} \|[z_{i1}, \cdots, z_{iC}]\|_\infty = \sum_{i=1}^{B} \max_j z_{ij}$, and $\mu > 0$ is a penalty parameter. Notice that problem (1) and problem (4) are equivalent, hence finding the same optimal solution $\boldsymbol{Z}^*$.

Augmenting the last equality constraint in problem (4) to the objective function via the Lagrange multiplier matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{B\times C}$, we can write the Lagrangian functions of problem (4) as

$$\mathcal{L}(\boldsymbol{Z}) = \frac{\mu}{2}\left\|\boldsymbol{Z} - \left(\boldsymbol{A} - \frac{\boldsymbol{\Lambda}}{\mu}\right)\right\|_F^2 + h_1(\boldsymbol{A}, \boldsymbol{\Lambda}) \tag{5}$$

Algorithm 1 Base Class Selection using ADMM

Input: $\boldsymbol{D}$; τ; μ; ε; maxIter.
Initialize: $k = 0$; $\boldsymbol{Z}^{(0)} = \boldsymbol{A}^{(0)} = \boldsymbol{I}$; $\boldsymbol{\Lambda}^{(0)} = \mathbf{0}$.
while (error1 $> \varepsilon$ or error2 $> \varepsilon$) and ($k <$maxIter) **do**
  Update $\boldsymbol{Z}$ by the algorithm in [2, 4]
  $\boldsymbol{Z}^{(k+1)} = \arg\min_{\boldsymbol{Z}} \left\|\boldsymbol{Z} - \left(\boldsymbol{A}^{(k)} - \frac{\boldsymbol{\Lambda}^{(k)}}{\mu}\right)\right\|_F^2$,
  $s.t. \quad \|\boldsymbol{Z}\|_{1,p} \le \tau$;
  Update $\boldsymbol{A}$ by the algorithm in [6]
  $\boldsymbol{A}^{(k+1)} = \arg\min_{\boldsymbol{A}} \left\|\boldsymbol{A} - \left(\boldsymbol{Z}^{(k+1)} + \frac{\boldsymbol{\Lambda}^{(k)} - \lambda_3 \boldsymbol{D}}{\mu}\right)\right\|_F^2$, $s.t. \quad \mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}}, \quad \boldsymbol{A} \ge \mathbf{0}$;
  $\boldsymbol{\Lambda}^{(k+1)} = \boldsymbol{\Lambda}^{(k)} + \mu(\boldsymbol{Z}^{(k+1)} - \boldsymbol{A}^{(k+1)})$;
  $error1 = \left\|\boldsymbol{Z}^{(k+1)} - \boldsymbol{Z}^{(k)}\right\|_\infty$;
  $error2 = \left\|\boldsymbol{Z}^{(k+1)} - \boldsymbol{A}^{(k+1)}\right\|_\infty$;
  $k \leftarrow k + 1$;
end while
Output: Optimal solution $\boldsymbol{Z}^* = \boldsymbol{Z}^{(k)}$.

and

$$\mathcal{L}(\boldsymbol{A}) = \frac{\mu}{2}\left\|\boldsymbol{A} - \left(\boldsymbol{Z} + \frac{\boldsymbol{\Lambda} - \boldsymbol{D}}{\mu}\right)\right\|_F^2 + h_2(\boldsymbol{Z}, \boldsymbol{\Lambda}), \tag{6}$$

where the functions $h_1(\cdot)$ and $h_2(\cdot)$ do not depend on $\boldsymbol{Z}$ and $\boldsymbol{A}$, respectively. We can obtain the solution of minimizing $\mathcal{L}(\boldsymbol{Z})$ via Euclidean projection onto the ℓ_1 ball [2, 4]. Minimizing $\mathcal{L}(\boldsymbol{A})$ subject to the probability simplex constraints $\left\{\mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}}; \boldsymbol{A} \ge \mathbf{0}\right\}$ can be done using the algorithm in [6].

Algorithm 1 shows the ADMM iterations that consist of minimizing $\mathcal{L}(\boldsymbol{Z})$ and $\mathcal{L}(\boldsymbol{A})$, and updating $\boldsymbol{\Lambda}$.

Complexity Analysis. Our implementation results in a memory and computational time complexity which are of the order of the number of elements in $\boldsymbol{D}$. In addition, it allows for parallel implementation, which can further reduce the computational time. More specifically,

- Minimizing the Lagrangian function in Eq. (5) with respect to $\boldsymbol{Z}$ can be done in $O(BC)$ computational time. Notice that we can perform the minimization in problem (5) via B independent smaller optimization programs over the B rows of $\boldsymbol{Z}$. Thus, having P parallel processing resources, we can reduce the computational time to $O(\lceil B/P \rceil C)$.
- Minimizing the Lagrangian function in Eq. (6) with respect to $\boldsymbol{A}$ subject to the probability simplex constraints $\left\{\mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}}; \boldsymbol{A} \ge \mathbf{0}\right\}$ can be done with $O(B\log(B)C)$ computational time (or $O(BC)$ expected time using the randomized algorithm in [6]). Notice that we can minimize Eq. (6) via C independent smaller optimization programs over the C columns of $\boldsymbol{A}$. Thus, having P parallel processing resources, we can reduce the computational time to $O(B\log(B)\lceil C/P \rceil)$ (or $O(B\lceil C/P \rceil)$ expected time using the randomized algorithm in [6]).
- The update on Λ has $O(BC)$ computational time and can be performed, respectively, by B or C independent updates over rows or columns, hence having $O(\lceil B/P \rceil C)$ or $O(B\lceil C/P \rceil)$ computational time when using P parallel processing resources.

As a result, the proposed ADMM implementation for base class selection can be performed in $O(B\log(B)C)$ computational time. In fact, our complexity is approximately equal to $O(B)$ since $C \in \{1, \cdots, 10\}$ and $B \gg C$. We can reduce the computational time to

$O(\lceil BC/P \rceil \log(B))$ using P parallel resources. This provides significant improvement with respect to standard convex solvers, such as CVX [11], which typically have cubic or higher complexity in the problem size.

3 SOLVING BASE LEARNER SELECTION PROBLEM

We formulate the base learner selection problem as

$$\begin{aligned} \min_{\mathbf{Z}} \quad & \sum_{k=1}^{K}\sum_{i=1}^{n_s} z_{ki} d_{ki} + \alpha_1 \sum_{k=1}^{K} \lambda_k \max_i z_{ki} \\ & + \alpha_2 \sum_{1\le k<l\le K} \mu_{kl} \max_i z_{ki} \cdot \max_i z_{li} \\ \text{s.t.} \quad & z_{ki} \ge 0,\ \forall k,i; \quad \sum_{k=1}^{K} z_{ki} = 1,\ \forall i, \end{aligned} \tag{7}$$

where $\boldsymbol{D} = \{d_{ki}\}_{k=1,\cdots,K}^{i=1,\cdots,n_S}$, and $\boldsymbol{Z} = \{z_{ki}\}_{k=1,\cdots,K}^{i=1,\cdots,n_S}$.

Similar to base class selection, we adopt ADMM to solve problem (7). To do so, an auxiliary matrix $\boldsymbol{A} = \{a_{ki}\}_{k=1,\cdots,K}^{i=1,\cdots,n_S}$ is introduced to problem (7) to rewrite the problem as

$$\begin{aligned} \min_{\boldsymbol{Z},\boldsymbol{A}} \quad & \sum_{k=1}^{K}\sum_{i=1}^{n_S} a_{ki} d_{ki} + \frac{\mu}{2}\|\boldsymbol{Z}-\boldsymbol{A}\|_F^2 + \|\lambda \boldsymbol{Z}\|_{1,\infty} \\ & + \mathrm{Tr}(\Omega^{\mathrm{T}}(\boldsymbol{Z}_\infty \boldsymbol{Z}_\infty^{\mathrm{T}})) \\ s.t. \quad & \mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}};\ \boldsymbol{A} \ge \mathbf{0};\ \boldsymbol{Z} = \boldsymbol{A}, \end{aligned} \tag{8}$$

where $\mu > 0$ is a penalty parameter. We define the row vector $z_k = [z_{k1}, \cdots, z_{kn_S}]$ (*i.e.*, the k-th row in the matrix $\boldsymbol{Z}$), $\lambda = [\lambda_1, \cdots, \lambda_K]$, and $\|\boldsymbol{Z}\|_{1,\infty} = \sum_{k=1}^{K} \|z_k\|_\infty = \sum_{k=1}^{K} \max_i z_{ki}$. Then, we have $\|\lambda \boldsymbol{Z}\|_{1,\infty} = \sum_{k=1}^{K} \|[\lambda_k z_k\|_\infty = \sum_{k=1}^{K} \lambda_k \max_i z_{ki}$. Further, we define the column vector $\boldsymbol{Z}_\infty = [\|z_1\|_\infty, \cdots, \|z_K\|_\infty]$, and $\Omega = \{\mu_{kl}\}_{k=1,\cdots,K}^{l=1,\cdots,K}$. Then $\mathrm{Tr}(\Omega^{\mathrm{T}}(\boldsymbol{Z}_\infty \boldsymbol{Z}_\infty^{\mathrm{T}})) = \sum_{1\le k<l\le K} \mu_{kl} \max_i z_{ki} \cdot \max_i z_{li}$, where $\mathrm{Tr}(\cdot)$ denotes the trace operator. Notice that problem (7) and problem (8) are equivalent, hence finding the same optimal solution $\boldsymbol{Z}^*$.

Augmenting the last equality constraint in problem (8) to the objective function via the Lagrange multiplier matrix $\boldsymbol{\Lambda} \in \mathbb{R}^{K\times n_S}$, we can write the Lagrangian functions of problem (8) as

$$\begin{aligned} \mathcal{L}(\boldsymbol{Z}) = & \frac{\mu}{2}\left\|\boldsymbol{Z} - \left(\boldsymbol{A} - \frac{\boldsymbol{\Lambda}}{\mu}\right)\right\|_F^2 + \|\lambda \boldsymbol{Z}\|_{1,\infty} \\ & + \mathrm{Tr}(\Omega^{\mathrm{T}}(\boldsymbol{Z}_\infty \boldsymbol{Z}_\infty^{\mathrm{T}})) + h_1(\boldsymbol{A}, \boldsymbol{\Lambda}), \end{aligned} \tag{9}$$

and

$$\mathcal{L}(\boldsymbol{A}) = \frac{\mu}{2}\left\|\boldsymbol{A} - \left(\boldsymbol{Z} + \frac{\boldsymbol{\Lambda} - \boldsymbol{D}}{\mu}\right)\right\|_F^2 + h_2(\boldsymbol{Z}, \boldsymbol{\Lambda}), \tag{10}$$

where the functions $h_1(\cdot)$ and $h_2(\cdot)$ do not depend on $\boldsymbol{Z}$ and $\boldsymbol{A}$, respectively. Solving the minimization of Eq. (9) is more difficult than that of Eq. (5) due to the last two regularizer. Here, we address this problem via the efficient projections onto a non-negative max-heap [17]. Minimizing the Lagrangian function in Eq. (10) subject to the probability simplex constraints $\left\{\mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}}; \boldsymbol{A} \ge \mathbf{0}\right\}$ can be done using the algorithm in [6].

Algorithm 2 Base Learner Selection using ADMM

Input: $\boldsymbol{D}$; λ; Ω; μ; ε, maxIter.
Initialize: $k = 0$, $\boldsymbol{Z}^{(0)} = \boldsymbol{A}^{(0)} = \boldsymbol{I}$; $\boldsymbol{\Lambda}^{(0)} = \mathbf{0}$.
while (error1 $> \varepsilon$ or error2 $> \varepsilon$) and ($k <$maxIter) **do**
  Update $\boldsymbol{Z}$ by the algorithm in [17]
  $\boldsymbol{Z}^{(k+1)} = \arg\min_{\boldsymbol{Z}} \frac{\mu}{2}\left\|\boldsymbol{Z} - \left(\boldsymbol{A}^{(k)} - \frac{\boldsymbol{\Lambda}^{(k)}}{\mu}\right)\right\|_F^2 + \|\lambda \boldsymbol{Z}\|_{1,\infty} + \mathrm{Tr}(\Omega^{\mathrm{T}}(\boldsymbol{Z}_\infty \boldsymbol{Z}_\infty^{\mathrm{T}}))$;
  Update $\boldsymbol{A}$ by the algorithm in [6]
  $\boldsymbol{A}^{(k+1)} = \arg\min_{\boldsymbol{A}} \left\|\boldsymbol{A} - \left(\boldsymbol{Z}^{(k+1)} + \frac{\boldsymbol{\Lambda}^{(k)} - \lambda_3 \boldsymbol{D}}{\mu}\right)\right\|_F^2, s.t. \quad \mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}}, \quad \boldsymbol{A} \ge \mathbf{0}$;
  $\boldsymbol{\Lambda}^{(k+1)} = \boldsymbol{\Lambda}^{(k)} + \mu(\boldsymbol{Z}^{(k+1)} - \boldsymbol{A}^{(k+1)})$;
  $error1 = \left\|\boldsymbol{Z}^{(k+1)} - \boldsymbol{Z}^{(k)}\right\|_\infty$;
  $error2 = \left\|\boldsymbol{Z}^{(k+1)} - \boldsymbol{A}^{(k+1)}\right\|_\infty$;
  $k \leftarrow k + 1$;
end while
Output: Optimal solution $\boldsymbol{Z}^* = \boldsymbol{Z}^{(k)}$.

Algorithm 2 shows the ADMM iterations that consist of minimizing $\mathcal{L}(\boldsymbol{Z})$ and $\mathcal{L}(\boldsymbol{A})$, and updating $\boldsymbol{\Lambda}$.

Complexity Analysis. Our implementation results in a memory and computational time complexity which are of the order of the number of elements in $\boldsymbol{D}$. In addition, it allows for parallel implementation, which can further reduce the computational time. More specifically,

- Minimizing the Lagrangian function in Eq. (9) with respect to $\boldsymbol{Z}$ can be done in $O(Kn_S)$ computational time. Notice that we can perform the minimization in Eq. (9) via K independent smaller optimization programs over the K rows of $\boldsymbol{Z}$. Thus, having P parallel processing resources, we can reduce the computational time to $O(\lceil K/P \rceil n_S)$.
- Minimizing the Lagrangian function in Eq. (10) with respect to $\boldsymbol{A}$ subject to the probability simplex constraints $\left\{\mathbf{1}^{\mathrm{T}}\boldsymbol{A} = \mathbf{1}^{\mathrm{T}}; \boldsymbol{A} \ge \mathbf{0}\right\}$ can be done with $O(K\log(K)n_S)$ computational time (or $O(Kn_S)$ expected time using the randomized algorithm in [6]). Notice that we can solve Eq. (10) via n_S independent smaller optimization programs over the n_S columns of $\boldsymbol{A}$. Thus, having P parallel processing resources, we can reduce the computational time to $O(K\log(K)\lceil n_S/P \rceil)$ (or $O(K\lceil n_S/P \rceil)$ expected time using the randomized algorithm in [6]).
- The update on Λ has $O(Kn_S)$ computational time and can be performed, respectively, by K or n_S independent updates over rows or columns, hence having $O(\lceil K/P \rceil n_S)$ or $O(K\lceil n_S/P \rceil)$ computational time when using P parallel processing resources.

As a result, the proposed ADMM implementation for base learner selection can be performed in $O(K\log(K)n_S)$ computational time. In fact, our complexity is approximately to $O(K)$ and even $O(1)$ since K and n_S are generally very small. In our experiments, $K = 8$ and $n_S \le 100$. We can reduce the computational time to $O(\lceil Kn_S/P \rceil \log(K))$ using P parallel resources. This provides significant improvement with respect to standard convex solvers, such as CVX [11], which typically have cubic or higher complexity in the problem size.

4 IMPLEMENTATION DETAILS

To conduct the selection of base classes defined in problem (1), for each testing dataset, we first compute the distance between a base

class b_i and a novel class c_j based on the pre-trained BERT model $g(\cdot)$, where the pre-trained BERT model is bert-base-uncased[1] containing 12-layers, 768 hidden and 12-heads. The BERT embedding of each class is calculated based on the average of each word embedding. Then, we can select the appropriate number of base classes based on the parameter τ. Specially, if the number of selected class is less than one-tenth of the training classes in a base dataset, we drop this part of the data without training the corresponding base learner. This is because the base learners obtained by training with such a small amount of data are often unreliable. After the selection of base classes, for each testing dataset, we train the corresponding base learner using the labeled sample of selected base classes. We train multiple ResNet-18 on selected base classes (a single ResNet per base dataset). For optimization, we use SGD with momentum and adjust the learning rate using cosine annealing. As presented in Tab. 1, the training parameters of the base learner for each base dataset are consistent with the SUR [7], including learning rate, batch size and so on. In addition, we use early stopping to prevent overfitting. For the selection of base learners, the max-iteration is set 1000 and the weight of each base learner is normalized by min-max.

5 EXPERIMENTAL RESULTS

A1: More Is Not Always Better. In original manuscript, we have presented the forward knowledge transfer (FKT(712)) that learning an extra 7 base domains has on each testing domain (*i.e.*, Figure 4). FKT(712) is de facto accuracy difference of two types of few-shot classifiers using the base training sets of ILSVRC-2012 and META-DATASET, denoted as accuracy(712) and accuracy(all) respectively. There are 8 compared methods, including seven single-domain few-shot classification methods (*i.e.*, k-NN, Finetune, MatchingNet, ProtoNet, fo-MAML, RelationNet, and ProtoMAML) [25], and one multi-domain few-shot classification method (*i.e.*, SUR [7]). Here, we further provide the corresponding accuracy of each method (*i.e.*, accuracy(712) and accuracy(all)) for ease of comparison (see Tab. 2). By comparing the top and bottom results in Tab. 2, we can observe that for each method, employing as more base classes and domains as possible is not always better for adaptation and generalization to a new domain. In particular, the recent multi-domain few-shot classification method, SUR, only achieves a significant improvement on Omniglot, Aircraft, Quick Draw, and Fungi datasets by employing more base classes and domains, while no obvious improvement and even reduce on the remaining 6 testing datasets.

A2: Comparison on Accuracy. In Figure 6 of original manuscript, as we change the number of selected classes (*i.e.*, τ) on META-DATASET, we have showed the performance change using different approaches, including two adapted SOTA approaches (*i.e.*, 'SUR w/ sel' and 'URT w/ sel'), two own baselines, and our AS3. To further provide the corresponding forward knowledge transfer using our proposed metric (FKT(τ)), we first present the prediction accuracy using each approach in various cases in Tab. 3, and then visualize this metric (FKT(τ)) in a bar plot (see Fig. 1). As expected, except Baseline1, by selecting only half of base classes, all these approaches can achieve higher or comparable performance to the case of using all base classes. Thus, base class selection can indeed silence non-positive transfer in few-shot classification. In addition, for 7/13 testing datasets, only 500 base classes (*i.e.*, 1/6 of all base classes) are enough to obtain a promising result. Further, the accuracy improvements on SUR and URT just verify that our base class selection process is general, where the base class selection can be combined with current SOTA few-shot classification approaches in a play-and-plug way. In particular, the results about Baseline1 is reasonable, since random selection is totally irrelevant with the testing domain. In summary, data is not always the more, the better in multi-domain few-shot classification. We should avoid redundant computation for improving efficiency without accuracy loss.

Ablation Study and Discussion. The base class selection relies on the semantic encoding of the class names. However, this information may be unavailable in another setting, where few-shot target domain may not provide the actual names of classes. To improve applicability, we can also select base classes relying on data, where we compute the class similarity in problem (1) using data (pseudo-)centroids instead of class names. Fig. 2 presents the accuracy comparison using class names and sample real-centroids on 9 testing domains (*i.e.*, ILSVRC-2012, Omniglot, Aircraft, CUB-200-2011, Describable Textures, Quick Draw, Fungi, VGG Flower and MSCOCO, indexed from '0' to '8'). where using class names usually achieves better accuracy since it may involve more priors with Bert, and meanwhile, takes less time due to only one embedding extraction for each class.

[1] https://huggingface.co/transformers/pretrained_models.html

REFERENCES

[1] Luca Bertinetto, Joao F Henriques, Philip Torr, and Andrea Vedaldi. 2019. Meta-learning with differentiable closed-form solvers. In *Proc. ICLR*.

[2] C. Chaux, P.L. Combettes, J.C. Pesquet, and V. Wajs. 2007. A variational formulation for frame-based inverse problems. *Inverse Problems* 23, 4 (Jun. 2007), 1495–1518.

[3] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2018. A Closer Look at Few-shot Classification. In *Proc. ICLR*.

[4] P.L. Combettes and V.R. Wajs. 2006. Signal recovery by proximal forward-backward splitting. *SIAM J. Multiscale Model. Simul.* 4, 4 (Jul. 2006), 1168–1200.

[5] Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. 2019. A Baseline for Few-Shot Image Classification. In *Proc. ICLR*.

[6] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. 2008. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proc. ICML*. 272–279.

[7] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. 2020. Selecting relevant features from a multi-domain representation for few-shot classification. In *Proc. ECCV*. 769–786.

[8] Ehsan Elhamifar, Guillermo Sapiro, and S Shankar Sastry. 2015. Dissimilarity-based sparse subset selection. *IEEE Trans. Pattern Anal. Mach. Intell.* 38, 11 (2015), 2182–2197.

[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. ICML*. 1126–1135.

[10] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. 2018. VERSA: Versatile and efficient few-shot learning. In *Proc. NeurIPS*. 1–9.

[11] Michael Grant and Stephen Boyd. 2014. CVX: Matlab software for disciplined convex programming, version 2.1.

[12] Dahyun Kang, Heeseung Kwon, Juhong Min, and Minsu Cho. 2021. Relational Embedding for Few-Shot Classification. In *Proc. ICCV*. 8822–8833.

[13] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. 2015. Siamese neural networks for one-shot image recognition. In *Proc. ICMLW*, Vol. 2. Lille, 0.

[14] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. 2019. Meta-learning with differentiable convex optimization. In *Proc. CVPR*. 10657–10665.

[15] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. 2019. Revisiting local descriptor based image-to-class measure for few-shot learning. In *Proc. CVPR*. 7260–7268.

[16] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. 2020. Negative margin matters: Understanding margin in few-shot classification.

Base training dataset	learning rate	weight decay	Max iter.	annealing freq.	batch size	class num.
ImageNet	3×10^{-2}	7×10^{-4}	480,000	48,000	64	712
Omniglot	3×10^{-2}	7×10^{-4}	50,000	3,000	16	883
Aircraft	3×10^{-2}	7×10^{-4}	50,000	3,000	8	70
Birds	3×10^{-2}	7×10^{-4}	50,000	3,000	16	140
Textures	3×10^{-2}	7×10^{-4}	50,000	1,500	32	33
Quick Draw	3×10^{-2}	7×10^{-4}	480,000	48,000	64	241
Fungi	3×10^{-2}	7×10^{-4}	480,000	15,000	32	994
VGG Flower	3×10^{-2}	7×10^{-4}	50,000	1,500	8	71

Table 1: Training hyper-parameters of individual feature networks about each base training domain in META-DATASET.

Method	ILSVRC	Omniglot	Aircraft	CUB	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO
k-NN	41.0	37.1	46.8	50.1	66.4	32.1	36.2	83.1	44.6	30.4
Finetune	45.8	60.9	68.7	57.3	69.1	42.6	38.2	85.5	66.8	34.9
MatchingNet	45.0	52.3	49.0	62.2	64.2	42.9	34.0	80.1	47.8	35.0
ProtoNet	50.5	60.0	53.1	68.8	66.6	49.0	39.7	85.3	47.1	41.0
fo-MAML	45.5	55.6	56.3	63.6	68.1	44.0	32.1	81.7	50.9	35.3
RelationNet	34.7	45.4	40.7	49.5	53.0	43.3	30.6	68.8	33.7	29.2
ProtoMAML	49.5	63.4	56.0	68.7	66.5	51.5	40.0	87.2	48.8	43.7
SUR	56.3	67.5	50.4	71.7	70.2	52.4	39.1	84.3	50.1	52.8

Method	ILSVRC	Omniglot	Aircraft	CUB	Textures	Quick Draw	Fungi	VGG Flower	Traffic Sign	MSCOCO
k-NN	38.6	74.6	65.0	66.4	63.6	44.9	37.1	83.5	40.1	29.6
Finetune	43.1	71.1	72.0	59.9	69.1	47.1	38.2	85.3	66.7	35.2
MatchingNet	36.1	78.3	69.2	56.4	61.8	60.8	33.7	81.9	55.6	28.8
ProtoNet	44.5	79.6	71.1	67.0	65.2	64.9	40.3	86.9	46.9	39.9
fo-MAML	37.8	83.9	76.4	62.4	64.2	59.7	33.5	79.9	42.9	29.4
RelationNet	30.9	86.6	69.7	54.1	56.6	61.8	32.6	76.1	37.5	27.4
ProtoMAML	46.5	82.7	75.2	69.9	68.3	66.8	42.0	88.7	52.4	41.7
SUR	56.1	93.1	85.4	71.4	71.5	81.30	63.1	82.8	53.4	52.4

Table 2: Few-shot classification results on 10 testing domains of META-DATASET, using models trained on the base training set of ILSVRC-2012 only (top) and trained on all 8 base training datasets of META-DATASET (bottom).

In *Proc. ECCV*. 438–455.
[17] Jun Liu, Shuiwang Ji, Jieping Ye, et al. 2009. SLEP: Sparse learning with efficient projections. *Arizona State University* 6, 491 (2009), 7.
[18] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. 2020. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. In *Proc. ICLR*.
[19] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. 2019. Meta-Learning with Latent Embedding Optimization. In *Proc. ICLR*.
[20] Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Proc. NeurIPS*. 4080–4090.
[21] Jake Snell and Richard Zemel. 2021. Bayesian Few-Shot Classification with One-vs-Each Pólya-Gamma Augmented Gaussian Processes. In *Proc. ICLR*.
[22] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. 2019. Meta-transfer learning for few-shot learning. In *Proc. CVPR*. 403–412.
[23] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. 2018. Learning to compare: Relation network for few-shot learning. In *Proc. CVPR*. 1199–1208.
[24] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. 2020. Rethinking few-shot image classification: a good embedding is all you need?. In *Proc. ECCV*. 266–282.
[25] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. 2020. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. In *Proc. ICLR*.
[26] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Proc. NeurIPS*. 3630–3638.
[27] Davis Wertheimer, Luming Tang, and Bharath Hariharan. 2021. Few-Shot Classification With Feature Map Reconstruction Networks. In *Proc. CVPR*. 8012–8021.
[28] Weijian Xu, Huaijin Wang, Zhuowen Tu, et al. 2020. Attentional Constellation Nets for Few-Shot Learning. In *Proc. ICLR*.
[29] Shuo Yang, Lu Liu, and Min Xu. 2021. Free Lunch for Few-shot Learning: Distribution Calibration. In *Proc. ICLR*.
[30] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. 2020. DeepEMD: Few-Shot Image Classification With Differentiable Earth Mover's Distance and Structured Classifiers. In *Proc. CVPR*. 12203–12213.
[31] Xingxing Zhang, Zhenfeng Zhu, Yao Zhao, Dongxia Chang, and Ji Liu. 2018. Seeing All From a Few: ℓ_1-Norm-Induced Discriminative Prototype Selection. *IEEE Trans. on Neural Netw. Learn Syst.* 30, 7 (2018), 1954–1966.

Figure 1: Comparison of AS3 to two adapted SOTA approaches and two own baselines on 13 testing datasets, in terms of FKT(τ).

Dataset	Selected classes number	SUR w/ sel	URT w/ sel	Baseline1	Baseline2	AS3
ILSVRC	$\tau = 500$	50.7 ± 1.3	51.3 ± 1.1	37.5 ± 1.0	43.1 ± 1.4	50.1 ± 1.3
	$\tau = 1000$	54.4 ± 1.1	55.2 ± 1.1	46.8 ± 1.0	43.6 ± 1.1	54.2 ± 1.1
	$\tau = 1500$	56.5 ± 1.1	57.7 ± 1.1	49.8 ± 1.1	46.2 ± 1.1	56.3 ± 1.1
	all classes	56.3 ± 1.1	55.7 ± 1.1	56.5 ± 1.0	44.8 ± 1.0	56.5 ± 1.0
Omniglot	$\tau = 500$	93.2 ± 1.0	93.2 ± 0.5	87.8 ± 0.9	93.3 ± 1.0	93.3 ± 1.0
	$\tau = 1000$	94.0 ± 0.5	93.4 ± 0.4	91.5 ± 0.6	90.4 ± 0.7	93.9 ± 0.5
	$\tau = 1500$	94.0 ± 0.5	94.5 ± 0.4	91.4 ± 0.6	92.1 ± 0.6	93.9 ± 0.5
	all classes	93.1 ± 0.5	94.4 ± 0.4	93.1 ± 0.5	91.9 ± 0.6	93.1 ± 0.5
Aircraft	$\tau = 500$	87.9 ± 0.6	88.4 ± 0.5	54.8 ± 0.9	83.8 ± 0.8	87.9 ± 0.5
	$\tau = 1000$	86.6 ± 0.6	87.0 ± 0.5	54.6 ± 0.9	83.0 ± 0.8	86.7 ± 0.6
	$\tau = 1500$	88.0 ± 0.6	88.0 ± 0.5	73.1 ± 0.7	80.9 ± 0.9	87.8 ± 0.6
	all classes	85.4 ± 0.7	85.8 ± 0.6	85.1 ± 0.7	77.1 ± 0.9	85.1 ± 0.7
CUB	$\tau = 500$	72.2 ± 1.2	74.7 ± 0.8	44.4 ± 4.1	68.0 ± 1.2	72.5 ± 1.1
	$\tau = 1000$	73.3 ± 1.1	78.3 ± 0.8	55.0 ± 1.2	65.9 ± 1.1	74.7 ± 1.0
	$\tau = 1500$	75.0 ± 1.0	78.4 ± 0.8	61.7 ± 1.3	66.8 ± 1.1	76.1 ± 1.1
	all classes	71.4 ± 1.0	76.3 ± 0.8	76.3 ± 0.9	62.1 ± 1.1	76.3 ± 0.9
Textures	$\tau = 500$	63.7 ± 0.8	62.9 ± 0.7	58.6 ± 0.8	65.3 ± 0.9	64.5 ± 0.9
	$\tau = 1000$	67.0 ± 0.9	67.3 ± 0.7	61.5 ± 0.8	63.1 ± 0.9	67.0 ± 0.9
	$\tau = 1500$	69.9 ± 0.7	67.3 ± 0.8	66.2 ± 0.7	65.4 ± 0.8	71.5 ± 0.8
	all classes	71.5 ± 0.8	71.8 ± 0.7	72.1 ± 0.8	62.7 ± 0.7	72.1 ± 0.8
Quick Draw	$\tau = 500$	81.1 ± 0.7	81.5 ± 0.6	70.7 ± 0.8	77.8 ± 0.8	80.4 ± 0.7
	$\tau = 1000$	81.5 ± 0.6	81.2 ± 0.6	74.0 ± 0.8	75.7 ± 0.8	80.4 ± 0.7
	$\tau = 1500$	81.3 ± 0.6	81.6 ± 0.6	76.5 ± 0.6	74.8 ± 0.8	80.9 ± 0.6
	all classes	81.3 ± 0.6	82.5 ± 0.6	80.9 ± 0.6	75.1 ± 0.7	80.8 ± 0.6
Fungi	$\tau = 500$	55.3 ± 2.2	52.6 ± 1.0	34.7 ± 0.9	55.1 ± 2.1	55.1 ± 2.1
	$\tau = 1000$	64.3 ± 1.7	63.8 ± 0.9	46.9 ± 1.0	55.6 ± 1.7	64.1 ± 1.6
	$\tau = 1500$	64.3 ± 1.0	63.6 ± 0.9	49.3 ± 1.0	50.7 ± 1.1	64.0 ± 0.9
	all classes	63.1 ± 1.0	63.5 ± 1.0	63.5 ± 0.9	46.1 ± 0.9	63.5 ± 0.9
VGG Flower	$\tau = 500$	81.0 ± 0.9	84.0 ± 0.7	78.8 ± 0.7	81.9 ± 0.8	79.9 ± 0.9
	$\tau = 1000$	84.7 ± 0.9	87.9 ± 0.6	80.8 ± 0.9	86.5 ± 0.8	84.2 ± 1.0
	$\tau = 1500$	83.4 ± 0.8	87.3 ± 0.6	79.4 ± 0.9	86.1 ± 0.7	82.9 ± 0.9
	all classes	82.8 ± 0.7	88.2 ± 0.6	83.3 ± 0.8	84.5 ± 0.7	83.3 ± 0.8
Traffic Signs	$\tau = 500$	45.3 ± 1.2	40.3 ± 1.0	45.3 ± 1.1	44.5 ± 1.2	43.9 ± 1.3
	$\tau = 1000$	51.5 ± 1.2	43.5 ± 1.0	49.3 ± 1.1	46.1 ± 1.2	51.1 ± 1.2
	$\tau = 1500$	50.9 ± 1.1	44.8 ± 1.0	49.7 ± 1.0	47.1 ± 1.0	50.1 ± 1.1
	all classes	53.4 ± 1.0	51.1 ± 1.1	50.5 ± 1.1	44.7 ± 1.0	50.5 ± 1.1
MSCOCO	$\tau = 500$	48.4 ± 1.3	45.7 ± 1.1	37.7 ± 1.0	46.0 ± 1.3	47.9 ± 1.3
	$\tau = 1000$	46.9 ± 1.1	47.3 ± 1.1	44.4 ± 1.0	42.8 ± 1.1	46.8 ± 1.1
	$\tau = 1500$	49.7 ± 1.0	48.5 ± 1.0	49.2 ± 1.1	43.9 ± 1.1	49.6 ± 1.0
	all classes	52.4 ± 1.1	52.2 ± 1.1	51.7 ± 1.1	42.7 ± 1.0	51.7 ± 1.1
MNIST	$\tau = 500$	85.8 ± 0.7	86.7 ± 0.5	90.3 ± 0.5	90.2 ± 0.6	85.3 ± 0.7
	$\tau = 1000$	92.8 ± 0.6	90.7 ± 0.5	87.1 ± 0.6	93.1 ± 0.6	91.6 ± 0.6
	$\tau = 1500$	94.2 ± 0.5	88.0 ± 0.5	94.4 ± 0.4	94.7 ± 0.4	93.3 ± 0.5
	all classes	94.3 ± 0.4	94.8 ± 0.4	92.7 ± 0.6	93.1 ± 0.6	92.7 ± 0.6
CIFAR10	$\tau = 500$	58.6 ± 1.0	56.2 ± 0.8	51.4 ± 0.8	58.4 ± 1.0	47.7 ± 1.4
	$\tau = 1000$	61.6 ± 0.9	60.7 ± 0.8	57.3 ± 0.8	59.3 ± 0.9	61.9 ± 0.9
	$\tau = 1500$	63.0 ± 0.9	61.7 ± 0.8	58.4 ± 0.9	59.4 ± 0.9	63.1 ± 1.0
	all classes	66.8 ± 0.9	67.3 ± 0.8	67.4 ± 0.9	59.0 ± 0.8	67.4 ± 0.9
CIFAR100	$\tau = 500$	49.3 ± 1.3	49.7 ± 1.1	39.2 ± 1.1	47.0 ± 1.3	40.0 ± 1.5
	$\tau = 1000$	52.3 ± 1.1	46.5 ± 1.1	45.6 ± 1.0	47.8 ± 1.2	52.4 ± 1.2
	$\tau = 1500$	52.0 ± 1.1	51.2 ± 1.1	48.0 ± 1.0	47.5 ± 1.1	51.7 ± 1.1
	all classes	56.6 ± 1.0	56.9 ± 1.0	56.8 ± 1.0	45.7 ± 1.0	56.8 ± 1.0

Table 3: Comparison of AS3 to two adapted SOTA approaches and two own baselines on 13 testing datasets, as a function of the number of selected base classes (*i.e.*, τ). 'ALL' represents using all 3144 base classes in META-DATASET without selection.

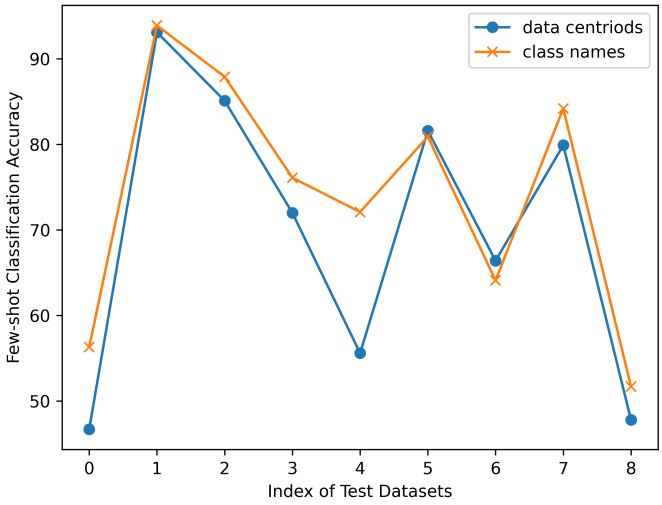

Figure 2: Accuracy comparison using class names and sample real-centroids.