Abstract

In learning to discover novel classes (L2DNC), we are given labeled data from seen classes and unlabeled data from unseen classes, and we train clustering models for the unseen classes. However, the rigorous definition of L2DNC is unexplored, which results in that its implicit assumptions are still unclear. In this paper, we demystify assumptions behind L2DNC and find that high-level semantic features should be shared among the seen and unseen classes. This naturally motivates us to link L2DNC to meta-learning that has exactly the same assumption as L2DNC. Based on this finding, L2DNC is not only theoretically solvable, but can also be empirically solved by meta-learning algorithms after slight modifications. This L2DNC methodology significantly reduces the amount of unlabeled data needed for training and makes it more practical, as demonstrated in experiments. The use of very limited data is also justified by the application scenario of L2DNC: since it is unnatural to label only seen-class data, L2DNC is sampling instead of labeling in causality. Therefore, unseen-class data should be collected on the way of collecting seen-class data, which is why they are novel and first need to be clustered.

1 Introduction

With the development of high-performance computing, we can train deep networks to achieve various tasks well [5, 36, 16, 37, 22]. However, the trained networks can only recognize the classes seen in the training set (i.e., known/seen classes), and cannot identify and cluster novel classes (i.e., unseen classes) like human beings. A prime example is that human can easily tell a novel animal category (e.g., okapi) after learning a few seen animal categories (e.g., horse and dog). Namely, human can effortlessly discover (cluster) novel categories of animals. Inspired by this fact, previous works formulated a novel problem called learning to discover novel classes (L2DNC) [12], where we train a clustering model using plenty of unlabeled novel-class and labeled known-class data.
(a) Experts annotate data (labelling in causality). 
(b) Experts sample data (sampling in causality).

Figure 1: Two ways to obtain data in L2DNC: (a) labeling in causality, e.g., we first obtain unlabeled images and then hire experts to label them and (b) sampling in causality, e.g., we are given a label set, and then sample images regarding these labels. In (a), experts can go through all images and find out novel classes. However, the novel classes (like cars) might be totally different from known classes (like animals), which makes L2DNC a theoretically unsolvable problem. In this paper, we revisit L2DNC from (b), where novel-class data are collected on the same way of sampling known-class data. In this view, L2DNC can be theoretically solved, since novel classes and known classes are highly related. The yellow rectangles represent the identified novel classes.

However, there exists two issues in L2DNC: L2DNC might not be a theoretically solvable problem. For example, if novel classes are completely different from known classes, then it is unrealistic to use the known classes (like animals) to help precisely cluster novel classes (like cars, Figure 1(a)). Moreover, L2DNC might not be a realistic problem in some scenarios where novel classes might only be seen once or twice. This does not satisfy the assumptions considered in existing L2DNC methods. These issues naturally motivate us to find out when L2DNC can be theoretically solved and what assumptions are considered behind L2DNC.

In this paper, we revisit L2DNC and find that L2DNC will be a well-defined problem if L2DNC is sampling in causality, i.e., novel and known classes are sampled together (Figure 1(b)). In this sampling process, data are often obtained because of a given purpose, and novel classes and known classes are obtained in the same scenario. For instance, botanists sample plant specimens for research purposes in the forests. Except for the plants they are interested in (i.e., known classes), they also find scarce plants never seen before (i.e., finding novel classes) [34]. Since a trip to forests is relatively costly and toilsome, botanists had better sampled these scarce plants passingly for future research. From this example, it can be seen that botanists will have plenty of labeled data with known classes, and few unlabeled data with novel classes. Since both data are sampled together from the same scenario (e.g., plants in the forests), it is reasonable to leverage knowledge of known classes to help cluster novel classes, which is like “discovering novel categories” happened in our daily life.

Therefore, we argue that the key assumption behind L2DNC is that, known classes and novel classes share high-level semantic features. For example, known classes and novel classes are different plants but all of them have the leaf, the stem, and the roots [39]. After demystifying the key assumption behind L2DNC, we find that this assumption is exactly the same as the underlying assumption of meta-learning [3, 27]. Hence, we naturally link L2DNC to meta-learning.

Based on the definition of meta-learning, we formulate the L2DNC problem rigorously. As a result, L2DNC is not only theoretically solvable (Theorem 1), but can also be empirically addressed by wisely modified meta-learning approaches. More importantly, based on our formulation and meta-learning mechanism, we can focus on a more challenging problem setting: L2DNC given very limited data (L2DNCL), i.e., how to cluster novel classes given very limited data, which is the most challenging part in L2DNC and previous methods cannot address this situation well (Figure 2). Recall the issues mentioned previously, it is clear that both issues are well addressed after finding the key assumption behind L2DNC.
and linking L2DNC to meta-learning.

The key difference between meta-learning and L2DNC lies in their inner-tasks. In meta-learning, the inner-task is a classification task, while in L2DNC, it is a clustering task. Thus, we can modify the training strategies of the inner-tasks of meta-learning methods such that they can discover novel classes, i.e., meta discovery (MEDI). Specifically, we first propose a novel method to sample training tasks for meta-learning methods. In sampled tasks, labeled and unlabeled data share high-level semantic features and the same clustering rule (Figure 3). Then, based on this novel sampling method, we realize MEDI using two representative meta-learning methods: model-agnostic meta-learning (MAML) [6] and prototypical network (ProtoNet) [35]. Figure 2 demonstrates that existing L2DNC methods cannot address L2DNC tasks well; while our methods (i.e., MEDI-MAML and MEDI-PRO) can perform much better.

We conduct experiments on four benchmarks and compare our method with five competitive baselines [26, 12, 13, 9, 8]. Empirical results show that our method outperforms these baselines significantly when novel-class data are very limited. Moreover, we provide a new practical application to the prosperous meta-learning community [3], which lights up a novel road for L2DNC.

2 Problem Formulation

This section presents a plain definition and a rigorous definition for the L2DNC problem.

A plain definition. In L2DNC, we have a dataset consisting of labeled known-class data $S^l = \{(x^i_l, y^i_l): i = 1, \ldots, n^l\}$ and unlabeled novel-class data $S^u = \{x^i_u: i = 1, \ldots, m\}$, where $0 < m \ll n^l$. The label space of $S^l$ and the label space of $S^u$ are disjoint but $S^l$ and $S^u$ share high-level semantic features. The number of classes in $S^l$ and the number of classes in $S^u$ are $K^l$ and $K^u$, respectively. Our goal is to leverage the prior knowledge of $S^l$ to assign the data in $S^u$ into a certain number of clusters (denoted as $C^u$) according to their categories. In the following, we give a rigorous definition of L2DNC according to the definition of meta-learning.

A rigorous definition. Let $T^* = (\mathcal{D}^l, \ell, \mathcal{H})$ be the task space, where $\mathcal{D}^l$ is a distribution over domain $\mathcal{X} \times \mathcal{C}$, $\mathcal{H} = \{h : \mathcal{X} \to \mathcal{C}\}$ is the hypothesis space, $\mathcal{C}$ contains indices of categories (including known and novel categories), and the loss function $\ell : \mathcal{H} \times \mathcal{X} \to \mathbb{R}^+$ means the loss incurred by predicting an output $h(x)$ when the ground-truth cluster index is $c$. The rigorous definition of the L2DNC problem is given as follows.

Problem 1 (L2DNC). Given the training tasks $T = \{T_i = (\mathcal{D}_i^l, \ell, \mathcal{H})\}_{i=1}^n$ drawn from a task distribution $P(T^*)$, meta-samples $S^l = \{S^l_{i, tr} \cup S^l_{i, ts}\}_{i=1}^n$ are drawn from the training tasks $T$, where $S^l_i \sim (\mathcal{D}^l_i)^m$ and...
We can cluster these objects using at least three rules, i.e., colors, shapes and frames. Motivated by the rule of clustering, when addressing the L2DNCL problem, we need to sample inner-level tasks, where data share the same rule (Algorithm 1).

Figure 3: When sampling tasks for MEDI, we need to care about clustering rules.

\( S_{1:tr}^{1:m} \sim (D_{tr}^{1:m}) \) are the training set and the test set of the task \( T_i \) with the sizes \( m \) and \( k \), respectively, and each task \( T_i \) can output an inner-task clustering algorithm \( A(S^1) : \mathcal{X}^m \rightarrow \mathcal{H} \). In L2DNCL, we aim to propose a meta-algorithm \( A \) to train an inner-task clustering algorithm \( A(S^1) \) with the meta-samples \( S^1 \). The trained \( A(S^1) \) should have a good performance on the new task \( T_{ne} = (D_{ne}^{1:k}, \ell, \mathcal{H}) \sim P(T^*) \) where we can only observe features (i.e., \( x \in \mathcal{X} \)) from the distribution \( D_{ne}^{1:k} \). Specifically, the trained \( A(S^1) \) should learn a hypothesis \( h^* = A(S^1)(S_{ne}^{1:tr}) : \mathcal{X} \rightarrow C \) with \( S_{ne}^{1:tr} \) such that \( h^*(x) \) is the right cluster index of \( x \), where \( S_{ne}^{1:tr} \) is the feature set of \( S_{ne}^{1:tr} \sim (D_{ne}^{1:m}) \) and \( (x, c) \sim D_{ne}^{1:k} \).

Remark 1. Compared to meta-learning, L2DNCL aims to train an inner-task clustering algorithm \( A(S^1) : \mathcal{X}^m \rightarrow \mathcal{H} \) rather than a classification algorithm often used in meta learning. Besides, in L2DNCL, we can only observe features from the new task, while we can observe the labeled data from the new task in the meta-learning. Since we can only observe few data from the new task in L2DNCL, \( m \) is a very small number.

3 Meta Discovery (MEDI) for L2DNCL

According to Problem 1, the key difference between meta-learning and L2DNCL lies in their inner-tasks. In meta-learning, the inner-task is a classification task, while in L2DNCL, the inner-task is a clustering task. Thus, we can modify the training strategies of the inner-tasks of meta-learning methods such that they can discover novel classes, i.e., meta discovery (MEDI).

In L2DNCL, sampling training tasks \( T \) from the distribution of known classes \( P(T^*) \) is important, which affects the final results directly. Since the clustering rules may not be unique (Figure 3), different data take on various rules. If we sample data with different rules to compose a training task, these data will influence each other, resulting in misleading the training procedures of the inner-tasks. Namely, sampling independently and identically (like MAML) cannot contribute to obtaining a good performance when addressing the L2DNCL problem. Thus, in MEDI, it is a key to propose a new task sampler that takes care of clustering rules.

3.1 CATA: Clustering-rule-aware Task Sampler

From the perspective of multi-view learning [2], data usually contain different feature representations. Namely, data have multiple views. However, there are always one view or a few views that are dominate for each instance, and these dominated views are similar in the high-level semantic meaning [20, 23, 24]. Therefore, we propose to use dominated views to replace with clustering rules. To this end, we propose a novel task-sampling method called clustering-rule-aware task sampler (CATA, Algorithm 1), which is based on a multi-view network that contains a feature extractor \( G : \mathcal{X} \rightarrow \mathbb{R}^h \) and \( K \) classifiers \( \{F_i : \mathbb{R}^h \rightarrow \mathcal{Y}\}_{i=1}^{K} \) (Figure 4).

The feature extractor \( G \) provides the shared data representations for \( K \) different classifiers \( \{F_i\}_{i=1}^{K} \). Each classifier classifies data based on its own view. The feature extractor \( G \) learns from all gradients.
We consider two representative meta-learning methods, MAML and ProtoNet to solve the L2DNCL problem. Namely, we will realize MEDI using MAML (MEDI-MAML) and ProtoNet (MEDI-PRO).

3.2 Two Realizations of MEDI

We consider two representative meta-learning methods, MAML and ProtoNet to solve the L2DNCL problem. Namely, we will realize MEDI using MAML (MEDI-MAML) and ProtoNet (MEDI-PRO).

Algorithm 1 Clustering-rule-aware task sampler (CATA)

\begin{algorithm}
\caption{Clustering-rule-aware task sampler (CATA)}
\begin{algorithmic}
\State \textbf{Input:} known-class data $S^i$, feature extractor $G$, classifiers $\{F_i\}_{i=1}^{K}$, learning rates $\omega_1$, $\omega_2$.
\State \textbf{Initialize} $\theta_G$ and $\{\theta_{F_i}\}_{i=1}^{K}$.
\For{$t = 1, \ldots, T$} \Comment{Clustering-rule-aware task sampler (CATA)}
\State 2: Compute $\nabla_{\theta_G} L_S$ and $\{\nabla_{\theta_{F_i}} L_S\}_{i=1}^{K}$ using $S^i$ and $L_S$ in Eq. (1);
\State 3: Update $\theta_G = \theta_G - \omega_1 \nabla_{\theta_G} L_S(\theta_G, \{\theta_{F_i}\}_{i=1}^{K})$ and $\theta_{F_i} = \theta_{F_i} - \omega_2 \nabla_{\theta_{F_i}} L_S(\theta_G, \{\theta_{F_i}\}_{i=1}^{K}), i = 1, \ldots, K$;
\EndFor
\State 4: Compute $F_i(G(x))$ to obtain $\{F_i(y|x)\}_{i=1}^{K}$ for each $x \in S^i$;
\State 5: Compose $V_i = \{x : V(x) = i\}$ using $V$ in Eq. (2), $i = 1, \ldots, K$;
\State \textbf{Output:} $\{V_i\}_{i=1}^{K}$.
\end{algorithmic}
\end{algorithm}

from $\{F_i\}_{i=1}^{K}$. To ensure that different classifiers have different views, we constrain the weight vector of the first fully connected layer of each classifier to be orthogonal. Take $F_i$ and $F_j$ as an example, we add the term $|W_i^TW_j|$ to the sampler’s loss function, where $W_i$ and $W_j$ denote the weight vectors of the first fully connected layer of $F_i$ and $F_j$ respectively. $|W_i^TW_j|$ tending to 0 means that $F_i$ and $F_j$ are nearly independent [31]. Therefore, the loss function of our sampler is defined as follows,

$$L_S(\theta_G; \{\theta_{F_i}\}_{i=1}^{K}) = \frac{1}{NK} \sum_{j=1}^{K} \sum_{i=1}^{N} \ell_{ce}(F_j \circ G(x_i), y_i) + \frac{2\lambda}{K(K-1)} \sum_{i \neq j} |W_i^TW_j|,$$

where $\ell_{ce}$ is the standard cross-entropy loss function and $\lambda$ is a trade-off parameter.

After we obtain the well-trained feature extractor $G$ and classifiers $\{F_i\}_{i=1}^{K}$, we input a training data point $x$ to our sampler and then we will get the probabilities that $x$ belongs to class $y$ in each classifier, i.e. $\{P_i(y|x)\}_{i=1}^{K}$, where $y$ is the label of $x$. Therefore, the view which $x$ belongs to is defined as

$$V(x) = \arg\max_i P_i(y|x).$$

Now we have assigned a data point to $K$ subsets according to their views, i.e. $\{V_i = \{x \in \mathcal{X} : V(x) = i\}\}_{i=1}^{K}$. Then, we can directly randomly sample a certain number of data (e.g., $N$-way, $K$-obs) from one subset to compose a training task $T_i$. According to the proportion of the number of data in each subset, we sample training tasks with different frequencies.

3.2 Two Realizations of MEDI

We consider two representative meta-learning methods, MAML and ProtoNet to solve the L2DNCL problem. Namely, we will realize MEDI using MAML (MEDI-MAML) and ProtoNet (MEDI-PRO).

MEDI-MAML. Here, another feature extractor $\Psi$ is given to obtain an embedding of data, following a classifier $g$ with the softmax layer. As novel-class data come from the same distribution as known-class data, the feature extractor $\Psi$ should be applicable to the known and novel classes as well. The key idea is that similar data should belong to the same class. For data pair $(x_i, x_j)$, we denote by the symbol $s_{ij} = 1$ if they come from the same class; otherwise, $s_{ij} = 0$.

Following [8], we adopt a more robust pairwise similarity called ranking statistics. For $z_i = \Psi(x_i)$ and $z_j = \Psi(x_j)$, we rank the values of $z_i$ and $z_j$ by the magnitude. Then we check if the indices of the values of top-$k$ ranked dimensions are the same. Namely, $s_{ij} = 1$ if they are the same, and $s_{ij} = 0$ otherwise.

We use the pairwise similarities $\{s_{ij}\}_{1 \leq i, j \leq N}$ as pseudo labels to train our feature extractor $\Psi$ and classifier $g$. Therefore, we convert L2DNCL from a clustering problem to a classification problem. As mentioned above, $g$ is a classifier with the softmax layer, so the inner product $g(z_i)^T g(z_j)$ is the cosine
CATA is a novel sampling method of meta-learning for L2DNCL. Here we show the inference process of assigning labeled data of known classes to three different views with the well-trained \(G\) and \(\{F_i\}_{i=1}^3\). \(V(x)\) is the voting function defined as Eq. (2). The weights of the first layers of \(F_1\), \(F_2\), and \(F_3\) are constrained to orthogonal mutually.

**Figure 4:** The structure of the clustering-rule-aware task sampler (CATA).

similarity between \(x_i\) and \(x_j\), which serves as the score for whether \(x_i\) and \(x_j\) belong to the same class. After sampling training tasks \(\{T_i\}_{i=1}^n\), we will train a model by the inner algorithm that optimizes the following binary cross-entropy (BCE) loss function:

\[
L_{T_i}(\theta_{g\varphi}) = -\frac{1}{N^2} \sum_{i=1}^N \sum_{j=1}^N \left[ s_{ij} \log(g(z_i)^T g(z_j)) + (1 - s_{ij}) \log(1 - g(z_i)^T g(z_j)) \right].
\]  

(3)

Entire procedures of L2DNCL by MAML are shown in Algorithm 2. Following MAML, the parameters of clustering algorithm \(A\) are trained by optimizing the following loss function:

\[
L_A(\theta_{g\varphi}) = \sum_{i=1}^n L_{T_i}(\theta_{g\varphi} - \alpha \nabla_{\theta_{g\varphi}} L_{T_i}(\theta_{g\varphi})),
\]  

(4)

where \(\alpha > 0\) is the learning rate of the inner-algorithm. Then we conduct the meta-optimization to update the parameters of cluster algorithm \(A\) as follows:

\[
\theta_{g\varphi} \leftarrow \theta_{g\varphi} - \eta \nabla_{\theta_{g\varphi}} L_A(\theta_{g\varphi}),
\]  

(5)

where \(\eta > 0\) denotes the meta learning rate.

**MEDI-PRO.** Following [35], we denote \(f: \mathcal{X} \rightarrow \mathbb{R}^M\) as an embedding function, which maps data to their representations. In training task \(T_i\), the mean vector of representations of data from class-\(s\) (i.e., \(S_{1,s}^{l, \text{tr}}\)) is defined as prototype \(c_k\):

\[
c_{i,s}(S_{1,s}^{l, \text{tr}}) = \frac{1}{|S_{1,s}^{l, \text{tr}}|} \sum_{(x_i^l, y_i) \in S_{1,s}^{l, \text{tr}}} f(x_i^l).
\]  

(6)

Here, we define a distance \(d: \mathbb{R}^M \times \mathbb{R}^M \rightarrow [0, +\infty)\) to measure the distance between data from the test set and the prototype. In this paper, the Euclidean distance as \(d\). Then, we produce a distribution over classes for a test data point \(x\) based on softmax over the distances to the prototypes in the embedding space:

\[
p(y = s|x) = \frac{\exp(-d(f(x), c_s))}{\sum_{s'} \exp(-d(f(x), c_{s'}))},
\]  

(7)
We train the embedding function by optimizing the negative log-probability, i.e., $-\log p(y = s|x)$. So the loss function of ProtoNet is defined as follows:

$$\mathcal{L}_{\mathcal{T}_i} = -\frac{1}{K} \sum_{s \in [K]} \sum_{x \in S_{s}^{l,ts}} \log p(y = s|x),$$

where $[K^n]$ denotes the number $K^n$ of classes selected from $\{1, \ldots, K^l\}$. $S_{s}^{l,ts}$ is the test set of labeled data of class-$s$ from task $\mathcal{T}_i$. The full procedures of training $f$ are shown in Algorithm 3. After training the embedding function $f$ well, we use the training set of $S^n$ to obtain the prototypes. For $x$ in the test set of $S^n$, we compute the distance between $x$ and each prototype, and then the class corresponding to the nearest prototype is the class of $x$.

4 Theoretical Guarantees for L2NDCL

This section presents that L2DNCL is theoretically solvable (Theorem 1) according to the meta learning theory. Based on Problem 1, we aim to minimize the following risk in L2DNCL:

$$\mathcal{R}(A(S^n), P(T^*)) = \mathbb{E}_{\mathcal{T} \sim P(T^*)} \mathbb{E}_{S \sim (D^n)\rightarrow} \mathbb{E}_{(x,c) \sim D^l} \ell(A(S^n) (S^{X,l}, x),$
where $S^{X, i}$ denotes the feature set of $S^i \sim (D^i)^m$. $R(A(S^i), P(T^*))$ is the expectation of the generalized error w.r.t. the task distribution $P(T^*)$ and can measure the performance of each inner-task clustering algorithm. In practice, the meta-clustering algorithm of L2DNCL is optimized by minimizing the average of the empirical error on the training tasks, called the empirical multi-task error:

$$\hat{R}(A(S^i), S^i) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{k} \sum_{(x_{ij}, z_{ij}) \in S^{\text{ts}}_i} \ell(A(S^i)(S^{X, 1, \text{tr}}_i), x_{ij}),$$

(10)

where $S^{X, 1, \text{tr}}_i$ denotes the feature set of $S^{\text{tr}}_i \cup S^{\text{ts}}_i \sim (D^i)^m$. Then, the generalization bound of inner-task clustering algorithm $A(S^i)$ of meta-based L2DNCL algorithms can be obtained from the uniform stability parameter $\beta$ of the meta-algorithm $A$.

**Definition 1** (Uniform Stability [27]). A meta-algorithm $A$ has uniform stability $\beta$ w.r.t. the loss function $\ell$ if the following holds for any meta-samples $S$ and $\forall i \in \{1, \ldots, n\}$, $\forall T \sim \hat{P}(T)$, $\forall S^{\text{tr}} \sim D^m$, $\forall S^{\text{ts}} \sim D^k$:

$$|\hat{L}(A(S)(S^{\text{tr}}), S^{\text{ts}}) - \hat{L}(A(S)(S^{\text{tr}}), S^{\text{ts}})| \leq \beta,$$

where

$$\hat{L}(A(S)(S^{\text{tr}}), S^{\text{ts}}) = \frac{1}{k} \sum_{z_{ij} \in S^{\text{ts}}_i} \hat{\ell}(A(S)(S^{\text{tr}}), z_{ij}).$$

**Theorem 1.** For any task distribution $P(T^*)$ and meta-samples $S^i$ with $n$ tasks, if a meta-algorithm $A$ has uniform stability $\beta$ w.r.t. a loss function $\ell$ bounded by $M$, then the following statement holds with probability at least $1 - \delta$ for any $\delta \in (0, 1)$:

$$R(A(S^i), P(T^*)) \leq \hat{R}(A(S^i), S^i) + \epsilon(n, \beta),$$

(11)

where $\epsilon(n, \beta) = 2\beta + (4n\beta + M)\sqrt{\frac{\log 1/\delta}{2n}}$.

By Theorem 1, the generalization bound depends on the number of the training tasks $n$ and the uniform stability parameter $\beta$. If $\beta < O(1/\sqrt{n})$, we have $\epsilon(n, \beta) \to 0$ as $n \to \infty$. Hence, given a sufficiently small $\beta$, the error $R(A(S^i), P(T^*))$ converges to training error $\hat{R}(A(S^i), S^i)$ as the number of training tasks $n$ grows. The proof of Theorem 1 can be found in Appendix A. Theorem 1 indicates that L2DNCL can be theoretically solved if we can control the uniform stability of a meta-algorithm (like MAML did via support-query learning [3]).

**Remark 2.** The only difference between L2DNC and L2DNCL is the number of novel-class data, which does not affect the results from Theorem 1. Thus, L2DNCL can be theoretically solved if data used in L2DNC are sampled in causality (Figure 1(b)).

5 Related Work and Discussion

Our proposal is mainly related to L2DNC and meta-learning. We briefly summarize the most representative works and discuss the relationship between these works and our work.

**Learning to discover novel classes.** L2DNC was proposed in recent years, aiming to cluster unlabeled novel-class data according to their underlying categories. Compared with unsupervised learning [1], L2DNC also requires labeled known-class data to help cluster novel-class data. The pioneering methods include the KL-Divergence-based contrastive loss (KCL) [12], the meta classification likelihood (MCL) [13], deep transfer clustering (DTC) [9], and the rank statistics (RS) [8]. Detailed introduction of these four
methods is in Appendix B. Comparing existing works [12, 46, 45] with ours, we aim to cluster unlabeled data when their quantity is few.

**Meta-learning.** Meta-learning is also known as learning-to-learn, which trains a meta-model over a wide variety of learning tasks [13, 41, 30, 4]. In meta-learning, we often assume that data share the same high-level features, which ensures that meta-learning can be theoretically addressed [27]. According to [11], there are three common approaches to meta-learning: optimization-based [6], model-based [32], and metric-based [35]. Review of meta learning methods is in Appendix B.

**Sampler for meta-learning.** Tasks in meta-learning are heterogeneous in some scenarios, which can not be handled via globally sharing knowledge among data. Therefore, it is crucial to address the task-sampling problem in meta-learning. [44] assigned many tasks that are randomly sampled from different clusters using their similarities, and only used the most related task cluster for training. This method solves the task-sampling problem in the view of tasks. [21] proposed a greedy class-pair based sampling method, which selects difficult tasks according to the class-pair potentials. This method solves the sampling problem in the view of classes. In our paper, we propose CATA (Section 3.1) based on clustering rules regarding data, which is in the view of data.

**Clustering via meta-learning.** In [15], researchers trained a recurrent model that learns how to cluster given multiple types of training datasets. Compared with MEDI, [15] learns a clustering model with multiple unlabeled datasets, while MEDI learns a clustering model with labeled known data and (abundant or few) unlabeled novel data from the same dataset.

6 Experiments

In this section, we test the efficacy of our methods and possible baselines on four datasets.

**Datasets.** To evaluate the performance of our methods, we conduct experiments on four popular image classification benchmarks, including CIFAR-10 [18], CIFAR-100 [18], SVHN [28], and OmniGlot [19]. Detailed introductions and partitions of known classes and novel classes of these four datasets is in Appendix C. Following the protocol of few-shot learning [40, 47, 25, 29], for SVHN and CIFAR-10, we perform the few-obsv tasks of 5-way 1-obsv and 5-way 5-obsv, and we perform the few-obsv tasks of 20-way 1-obsv and 20-way 5-obsv for CIFAR-100 and OmniGlot.

**Baselines.** To verify the performance of our meta-based L2DNCL methods (i.e., MEDI-MAML and MEDI-PRO), we compare them with 5 competitive baselines, including K-means [26], KCL [12], MCL [13], DTC [9], and RS [8]. We modify these baselines by only reducing the amount of novel-class data,
Table 1: Ablation Study on four datasets.

| Methods         | MM w/o C | MM | MP w/o C | MP |
|-----------------|----------|----|----------|----|
| SVHN (5-way)    |          |    |          |    |
| 5-obsv          | 47.3±0.3 | 40.1±0.3 | 61.0±0.5 | 60.5±0.2 |
| 1-obsv          | 42.7±0.3 | 39.7±0.4 | 52.8±0.4 | 50.2±0.3 |
| CIFAR-10 (5-way)|          |    |          |    |
| 5-obsv          | 45.3±0.1 | 42.7±0.3 | 58.5±0.2 | 57.9±0.1 |
| 1-obsv          | 41.3±0.4 | 40.3±0.3 | 51.7±0.2 | 47.6±0.2 |
| CIFAR-100 (20-way)|       |    |          |    |
| 5-obsv          | 42.0±0.4 | 39.9±0.3 | 45.5±0.3 | 44.1±0.1 |
| 1-obsv          | 39.2±0.2 | 37.1±0.3 | 38.8±0.4 | 37.0±0.2 |
| OmniGlot (20-way)|      |    |          |    |
| 5-obsv          | 82.1±0.4 | 80.1±0.4 | 98.4±0.2 | 96.7±0.3 |
| 1-obsv          | 77.3±0.4 | 78.5±0.4 | 94.6±0.3 | 91.2±0.2 |

In this table, we report the ACC (%) ± standard deviation of ACC (%) on four datasets, where MM and MP represent MEDI-MAML and MEDI-PRO, respectively, and C represents CATA and w/o represents “without”. It is clear that CATA improves the ACC.

Evaluation metric. For a clustering problem, we use the average clustering accuracy (Acc) to evaluate the performance of clustering, which is defined as follows,

$$\max_{\phi \in L} \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{\bar{y}_i = \phi(y_i)\},$$

where \(\bar{y}_i\) and \(y_i\) denote the ground-truth label and assigned cluster indices respectively. \(L\) is the set of mappings from cluster indices to ground-truth labels.

Results on CIFAR-10. As shown in Figures 2(a) and 2(b), MEDI-MAML and MEDI-PRO outperform all baselines significantly, and the ACC of MEDI-PRO is much higher than that of MEDI-MAML. The main reason is that MEDI-PRO makes full use of the labels of known-class data in the training process, while MEDI-MAML does not. MEDI-MAML only uses the labels of known-class data in the sampling process. Besides, K-means performs better than it on other datasets significantly. The reason is that clustering rules contained in CIFAR-10 are suitable for K-means.

Results on SVHN. Figures 2(d) and 2(c) show that our methods still outperform all baselines. In the task of 5-way 1-obsv, RS performs as well as MEDI-MAML (Figure 2(d)). The reason is that RS trains the embedding network with self-supervised learning under 1-obsv case, which partly overcomes this problem by data augment.

Results on CIFAR-100. It is clear that we outperform all baselines. Differ from tasks on other datasets, MEDI-MAML performs equally even a little better than MEDI-PRO shown in Figure 5(a). The reason is that the amount of known classes is relatively large and the data distribution of CIFAR-100 is complex, so we cannot accurately compute prototypes with very limited data.

Results on OmniGlot. As shown in Figures 5(c) and 5(d), our methods still have the highest ACC. We find that the Acc of K-means are merely 1.15% and 1.30%. Namely, K-means hardly works on OmniGlot. The number of novel classes of OmniGlot (659) is too large for K-means.

Results analysis of L2DNC with abundant novel-class data. Other than L2DNCL, MEDI can also address L2DNC efficiently. Results of L2DNC are shown in Table 2 (Appendix E). Table 2 shows that MEDI-MAML is comparable with the representative methods but cannot outperform the RS. Compared with RS, MEDI-MAML samples many inner-tasks for training, while RS uses the whole data. Incomplete data makes MEDI-MAML unable to learn the global distribution of novel classes. MEDI-PRO is not as well as MEDI-MAML on L2DNC tasks. As the absence of labels of novel class data, we cannot finetune
the model used for calculating data embedding, which is trained by known-class data. Although this model cannot adapt to novel classes, we can calculate more accurately prototypes with abundant novel-class data. Hence, with MEDI-PRO, the results of L2DNC are obviously better than the results of L2DNCL.

**Ablation study.** To verify the effectiveness of CATA, we conduct ablation study by removing CATA from MEDI-MAML and MEDI-PRO. According to Table 6, CATA significantly improves the performance of MEDI-MAML and MEDI-PRO. However, there exists an abnormal phenomenon in OmniGlot, i.e., MM w/o C outperforms MM in the task of 20-way 1-obs. Although we need 16 (= |S_l| + |S_t| = 1 + 15) data for each class in an inner-task, the total amount of data for each class is only 20. Therefore, there are not enough data for CATA to sample, which makes CATA cannot improve the ACC of MEDI-MAML.

7 Conclusions

In this paper, we study an important problem called learning to discover novel classes (L2DNC) and demystify the key assumptions behind this problem. We find that L2DNC is sampling instead of labeling in causality, and, furthermore, data in the L2DNC problem should share high-level semantic features. This finding motivates us to link L2DNC to meta-learning, since meta-learning also assumes that the high-level semantic features are shared between seen and unseen classes. To this end, we propose to discover novel classes in a meta-learning way, i.e., the meta discovery (MEDI). After realizing MEDI using MAML and ProtoNet, we find that meta-learning based methods outperform all existing baselines when addressing a more challenging problem, L2DNC given very limited data (L2DNCL) where only few novel-class data can be observed. Moreover, based on this finding, L2DNC/L2DNCL can be theoretically solved, which lights up a novel road for L2DNC/L2DNCL.

References

[1] Horace B Barlow. Unsupervised learning. *Neural computation*, 1(3):295–311, 1989.
[2] Avrim Blum and Tom M. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.
[3] Jiaxin Chen, Xiao-Ming Wu, Yanke Li, Qimai Li, Li-Ming Zhan, and Fu-Lai Chung. A closer look at the training strategy for modern meta-learning. In *NeurIPS*, 2020.
[4] Xiaohan Chen, Zhangyang Wang, Siyu Tang, and Krikamol Muandet. MATE: plugging in model awareness to task embedding for meta learning. In *NeurIPS*, 2020.
[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
[7] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *ICLR*, 2018.
[8] Kai Han, Sylvestre-Alvise Rebuffi, Sébastien Ehrhardt, Andrea Vedaldi, and Andrew Zisserman. Automatically discovering and learning new visual categories with ranking statistics. In *ICLR*, 2020.
[9] Kai Han, Andrea Vedaldi, and Andrew Zisserman. Learning to discover novel visual categories via deep transfer clustering. In *ICCV*, 2019.
[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.

[11] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. arXiv:2004.05439, 2020.

[12] Yen-Chang Hsu, Zhaoyang Lv, and Zsolt Kira. Learning to cluster in order to transfer across domains and tasks. In ICLR, 2018.

[13] Yen-Chang Hsu, Zhaoyang Lv, Joel Schlosser, Phillip Odom, and Zsolt Kira. Multi-class classification without multi-class labels. In ICLR, 2019.

[14] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In ICML, 2015.

[15] Yibo Jiang and Nakul Verma. Meta-learning to cluster. arXiv:1910.14134, 2019.

[16] Yongcheng Jing, Xiao Liu, Yukang Ding, Xinchao Wang, Errui Ding, Mingli Song, and Shilei Wen. Dynamic instance normalization for arbitrary style transfer. In AAAI, 2020.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

[18] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[19] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. Science, 350(6266):1332–1338, 2015.

[20] Yingming Li, Ming Yang, and Zhongfei Zhang. A survey of multi-view representation learning. IEEE Transactions on Knowledge and Data Engineering, 31(10):1863–1883, 2019.

[21] Chenghao Liu, Zhihao Wang, Doyen Sahoo, Yuan Fang, Kun Zhang, and Steven C. H. Hoi. Adaptive task sampling for meta-learning. In ECCV, 2020.

[22] Feng Liu, Wenkai Xu, Jie Lu, Guangquan Zhang, Arthur Gretton, and Danica J. Sutherland. Learning deep kernels for non-parametric two-sample tests. In ICML, 2020.

[23] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, Xuanyi Dong, and Chengqi Zhang. Isometric propagation network for generalized zero-shot learning. In ICLR, 2021.

[24] Lu Liu, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Attribute propagation network for graph zero-shot learning. In AAAI, 2020.

[25] 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 ICLR, 2019.

[26] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967.

[27] Andreas Maurer. Algorithmic stability and meta-learning. Journal of Machine Learning Research, 6:967–994, 2005.

[28] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning, 2011.
[29] Minseop Park, Sachoon Kim, Jungtaek Kim, Yanbin Liu, and Seungjin Choi. Taeml: Task-adaptive ensemble of meta-learners. In NeurIPS Workshop on Meta learning, 2018.

[30] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In ICLR, 2017.

[31] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In ICML, 2017.

[32] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. Meta-learning with memory-augmented neural networks. In ICML, 2016.

[33] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[34] EJ A’Court Smith. Discovery of remains of plants and insects. Nature, 11(266):88–88, 1874.

[35] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In NeurIPS, 2017.

[36] Jie Song, Yixin Chen, Xinchao Wang, Chengchao Shen, and Mingli Song. Deep model transferability from attribution maps. In NeurIPS, pages 6179–6189, 2019.

[37] Jie Song, Yixin Chen, Jingwen Ye, Xinchao Wang, Chengchao Shen, Feng Mao, and Mingli Song. DEPARA: deep attribution graph for deep knowledge transferability. In CVPR, 2020.

[38] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1):1929–1958, 2014.

[39] Dirk R. Walters. Vascular Plant Taxonomy. Kendall/Hunt Publishing Company, 1977.

[40] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. Generalizing from a few examples: A survey on few-shot learning. ACM Computing Surveys, 53(3):1–34, 2020.

[41] Ying Wei, Peilin Zhao, and Junzhou Huang. Meta-learning hyperparameter performance prediction with neural processes. In Proceeding of the Thirty-eighth International Conference on Machine Learning, 2021.

[42] Junyuan Xie, Ross B. Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In ICML, 2016.

[43] Huaxiu Yao, Longkai Huang, Linjun Zhang, Ying Wei, Li Tian, James Zou, Junzhou Huang, and Zhenhui Li. Improving generalization in meta-learning via task augmentation. In Proceeding of the Thirty-eighth International Conference on Machine Learning, 2021.

[44] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. In ICML, 2019.

[45] Zhun Zhong, Enrico Fini, Subhankar Roy, Zhiming Luo, Elisa Ricci, and Nicu Sebe. Neighborhood contrastive learning. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

[46] Zhun Zhong, Linchao Zhu, Zhiming Luo, Shaozi Li, Yi Yang, and Nicu Sebe. Openmix: Reviving known knowledge for discovering novel visual categories in an open world. In CVPR, 2021.

[47] Imtiaz Masud Ziko, Jose Dolz, Eric Granger, and Ismail Ben Ayed. Laplacian regularized few-shot learning. In ICML, 2020.
A Proof of Theorem 1

The proof of Theorem 1 mainly follows [3]. Given training meta-samples $S = \{S_i^r \cup S_i^s\}_{i=1}^n$, we modify $S$ by replacing the $i$-th element to obtain $S' = \{S_i^r \cup S_i^s, \ldots, S_i^r \cup S_{i-1}^s, S_i^r \cup S_{i+1}^s, \ldots, S_n^r \cup S_n^s\}$, where the replacement sample $S'_i$ is assumed to be drawn from $D$ and is independent from $S$. In the same way, given a training set $S = \{z_1, \ldots, z_{i-1}, z_i, z_{i+1}, \ldots, z_n\}$, we modify $S$ by replacing the $i$-th element to obtain $S_i = \{z_1, \ldots, z_{i-1}, z'_i, z_{i+1}, \ldots, z_n\}$.

Lemma 1 (McDiarmid Inequality). Let $S$ and $S_i$ defined as above, let $F : \mathcal{Z}^n \rightarrow \mathbb{R}$ be any measurable function for which there exits constants $c_i$ ($i = 1, \ldots, n$) such that

$$\sup_{S \in \mathcal{Z}^n, z' \in \mathcal{Z}} |F(S) - F(S')| \leq c_i,$$

then

$$P_S[F(S) - E_S[F(S)] \geq \epsilon] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2}\right).$$

Theorem 1’. For any task distribution $P(T^*)$ and meta-samples $S^i$ with $n$ tasks, if a meta-algorithm $A$ has uniform stability $\beta$ w.r.t. a loss function $\ell$ bounded by $M$, then the following statement holds with probability of at least $1 - \delta$ for any $\delta \in (0, 1)$:

$$\mathcal{R}(A(S^i), P(T^*)) \leq \tilde{\mathcal{R}}(A(S^i), S^i) + \epsilon(n, \beta),$$

where $\epsilon(n, \beta) = 2\beta + (4n\beta + M)\sqrt{\frac{\log(1/\delta)}{2n}}$.

Proof. Let $F(S^i) = \mathcal{R}(A(S^i), P(T^*)) - \tilde{\mathcal{R}}(A(S^i), S^i)$ and $F(S^i):=\mathcal{R}(A(S^i), P(T^*)) - \tilde{\mathcal{R}}(A(S^i), S^i)$. We have

$$|F(S^i) - F(S^i)| \leq |\mathcal{R}(A(S^i), P(T^*)) - \mathcal{R}(A(S^i), P(T^*))| + |\mathcal{R}(A(S^i), S^i) - \mathcal{R}(A(S^i), S^i)|.$$  \hspace{1cm} (14)

The first term in Eq. (14) can be written as

$$|\mathcal{R}(A(S^i), P(T^*)) - \mathcal{R}(A(S^i), P(T^*))| \leq |\mathcal{R}(A(S^i), P(T^*)) - \mathcal{R}(A(S^i), P(T^*))|$$

$$+ |\mathcal{R}(A(S^i), P(T^*)) - \mathcal{R}(A(S^i), P(T^*))|.$$  \hspace{1cm} (14)

We can upper bound the first term in Eq. (14) by studying the variation when a sample set $S^i$ of training task $D_i$ is deleted,

$$|\mathcal{R}(A(S^i), P(T^*)) - \mathcal{R}(A(S^i), P(T^*))|$$

$$\leq E_{T \sim P(T^*)} E_{S \sim (D^i)} \max_{(x, c) \sim D^i} |\ell(A(S^i)(S^X, x)) - \ell(A(S^i)(S^X, x))|$$

$$\leq \sup_{T \sim P(T^*)} \max_{S \sim (D^i)} \max_{(x, c) \sim D^i} |\ell(A(S^i)(S^X, x)) - \ell(A(S^i)(S^X, x))|$$

$$\leq \beta.$$  \hspace{1cm} (14)

Similarly, we have $|\mathcal{R}(A(S^i), P(T^*)) - \mathcal{R}(A(S^i), P(T^*))| \leq \beta$. So the first term of Eq. (14) is upper
bounded by $2\beta$. The second factor in Eq. (14) can be guaranteed likewise as follows,

$$
|\hat{\mathcal{R}}(A(S^i), S^i) - \hat{\mathcal{R}}(A(S^{i,j}), S^{i,j})| \\
\leq \frac{1}{n} \sum_{q \neq i} \frac{1}{k} \sum_{(x_{ij}, c_{ij}) \in S^{i,ts}_q} |\ell(A(S^i)(S^{X,1,tr}_q), x_{ij}) - \ell(A(S^{i,j})(S^{X,1,tr}_q), x_{ij})| \\
+ \frac{1}{nk} \sum_{(x_{ij}, c_{ij}) \in S^{i,ts}_i} \ell(A(S^i)(S^{X,1,tr}_i), x_{ij}) - \sum_{(x_{ij}, c_{ij}) \in S^{i,1,ts}_i} \ell(A(S^{i,j})(S^{X,1,tr}_i), x_{ij}) \\
\leq 2\beta + \frac{M}{n}.
$$

Hence, $|F(S^i) - F(S^{i,j})|$ satisfies the condition of Lemma 1 with $c_i = 4\beta + \frac{M}{n}$. It remains to bound $E_S'[F(S^i)] = E_S'[\mathcal{R}(A(S^i), P(T^*))] - E_S'[\hat{\mathcal{R}}(A(S^i), S^i)]$. The first term can be written as follows,

$$
E_S'[\mathcal{R}(A(S^i), P(T^*))] = E_{S^i,S^{X,1,tr}_i,S^{i,1,tr}_i}[1/n \sum_{i=1}^n \sum_{(x_{ij}, c_{ij}) \in S^{i,ts}_i} \ell(A(S^i)(S^{X,1,tr}_i), x_{ij})].
$$

Similarly, the second term is,

$$
E_S'[\hat{\mathcal{R}}(A(S^i), S^i)] = E_{S^i}[1/n \sum_{i=1}^n \sum_{(x_{ij}, c_{ij}) \in S^{i,ts}_i} \ell(A(S^i)(S^{X,1,tr}_i), x_{ij})] \\
= E_{S^i}[1/k \sum_{(x_{ij}, c_{ij}) \in S^{i,ts}_i} \ell(A(S^i)(S^{X,1,tr}_i), x_{ij})] \\
= E_{S^i,S^{X,1,tr}_i,S^{i,1,tr}_i}[1/k \sum_{(x_{ij}, c_{ij}) \in S^{i,ts}_i} \ell(A(S^i)(S^{X,1,tr}_i), x_{ij})].
$$

Hence, $E_S'[F(S^i)]$ is upper bounded by $2\beta$,

$$
E_S'[\mathcal{R}(A(S^i), P(T^*))] - E_S'[\hat{\mathcal{R}}(A(S^i), S^i)] \\
= E_{S^i,S^{X,1,tr}_i,S^{i,1,tr}_i}[1/k \sum_{(x_{ij}, c_{ij}) \in S^{i,ts}_i} \ell(A(S^i)(S^{X,1,tr}_i), x_{ij}) - 1/k \sum_{(x_{ij}, c_{ij}) \in S^{i,1,ts}_i} \ell(A(S^{i,j})(S^{X,1,tr}_i), x_{ij})] \\
\leq 2\beta.
$$

Plugging the above inequality in Lemma 1, we obtain

$$
P_{S^i}[\mathcal{R}(A(S^i), P(T^*)) - \hat{\mathcal{R}}(A(S^i), S^i) \geq 2\beta + \epsilon] \leq \exp\left(-\frac{2\epsilon^2}{\sum_{i=1}^n (4\beta + \frac{M}{n})^2}\right).
$$

Finally, setting the right side of the above inequality to $\delta$, the following result holds with probability of $1 - \delta$,

$$
\mathcal{R}(A(S^i), P(T^*)) \leq \hat{\mathcal{R}}(A(S^i), S^i) + 2\beta + (4n\beta + M)\sqrt{\frac{\log(1/\delta)}{2n}}.
$$

\[\square\]
Table 2: Results of L2DNC with abundant novel class data. In this table, we report the ACC (%) ± standard deviation of ACC (%) of baselines and our methods (MEDI-MAML and MEDI-PRO) given abundant novel class data. We still evaluate these methods on four benchmarks (SVHN, CIFAR-10, CIFAR-100, and OmniGlot).

| Methods       | K-means | KCL  | MCL  | DTC   | RS    | MEDI-MAML | MEDI-PRO |
|---------------|---------|------|------|-------|-------|-----------|----------|
| SVHN          | 42.6±0.0| 21.4±0.6| 38.6±10.8| 60.9±1.6| 95.2±0.2| 93.1±2.1 | 77.1±0.8 |
| CIFAR-10      | 65.5±0.0| 66.5±3.9| 64.2±0.1 | 87.5±0.3 | 91.7±0.9 | 92.3±0.9 | 73.2±1.9 |
| CIFAR-100     | 56.6±1.6| 14.3±1.3| 21.3±3.4 | 56.7±1.2 | 75.2±4.2 | 69.8±1.3 | 58.3±2.2 |
| OmniGlot      | 77.2     | 82.4     | 83.3     | 89.0     | 89.1     | 88.6±0.7 | 98.4±0.2 |

B Detailed Related Work

Learning to discover novel classes. L2DNC is proposed in recent years, aiming to cluster unlabeled novel-class data according to their underlying categories. Compared with unsupervised learning [1], L2DNC also requires labeled known-class data to help cluster novel-class data. The pioneering methods include KLD-based contrastive loss (KCL) [12], meta classification likelihood (MCL) [13], deep transfer clustering (DTC) [9], and rank statistics (RS) [8]. In this paper, we update and present the results of our methods regarding L2DNC.

In KCL [12], a method based on pairwise similarity is introduced. They first pre-trained a similarity prediction network on labeled data of known classes and then use this network to predict the similarity of each unlabeled data pair, which acts as the supervision information to train the main model. Then, MCL [13] changed the loss function of KCL (the KL-divergence based contrastive loss) to the meta classification likelihood loss.

In DTC [9], they first learned a data embedding with metric learning on labeled data, and then they employed the DEC [42] to learn the cluster assignments on unlabeled data.

In RS [8], they used the rank statistics to predict the pairwise similarity of data. To keep the performance on data of known classes, they pre-trained the data embedding network with self-supervised learning method [7] on both labeled data and unlabeled data.

Meta-Learning. Meta-learning is also known as learning-to-learn, which train a meta-model over a large variety of learning tasks [30]. In meta-learning, we often assume that data share the same high-level features, which ensures that meta-learning can be theoretically addressed [27]. According to [11], there are three common approaches to meta-learning: optimization-based [6], model-based [32], and metric-based [35].

Optimization-based methods include those where the inner-level task is literally solved as an optimization problem, and focus on extracting meta knowledge required to improve optimization performance. In model-based methods, the inner learning step is wrapped up in the feed-forward pass of a single model. Metric-based methods perform non-parametric learning at the inner-task level by simply comparing validation points with training points and predicting the label of matching training points. Since meta-learning and the L2DNCL have the same assumption that data share the same high-level semantic features (introduced in Section 1), we link L2DNCL to meta-learning problem, providing a way to formulate and analyze the L2DNCL.

C Dataset Introductions and Splits

CIFAR-10 dataset contains 60,000 images with sizes of 32 × 32. Following [9], for L2DNCL, we select the first five classes (i.e. airplane, automobile, bird, cat, and deer) as known classes and the rest of classes as
novel classes. The amount of data from each novel class is no more than 5. CIFAR-100 dataset contains 100 classes. Following [8], we select the first 80 classes as known classes and select the last 20 classes as novel classes.

SVHN contains 73,257 training data and 26,032 test data with labels 0-9. Following [9], we select the first five classes (0-4) as known classes and select the (5-9) as novel classes. OmniGlot constains 1,632 handwritten characters from 50 different alphabets. Following [13], we select all the 30 alphabets in background set (964 classes) as known classes and select each of the 20 alphabets in evaluation set (659 classes) as novel classes.

D Implementation Details

We implement all methods by PyTorch 1.7.1 and Python 3.7.6, and conduct all the experiments on two NVIDIA RTX 3090 GPUs.

**CATA.** We use ResNet-18 [10] as the feature extractor and use three fully-connected layers with softmax layer as the classifier. We also use BN layer [14] and Dropout [38] in network layers. In this paper, we select the number of views $K = 3$ for all four datasets. In other words, there are three classifiers following by the feature extractor. Both the feature extractor and 3 classifier use Adam [17] as their optimizer. The number of training steps is 50 and the learning rates of feature extractor and classifiers are 0.01 and 0.001 respectively. We use the tradeoff $\lambda$ of 1/3.

**MEDI-MAML for L2DNCL.** We use VGG-16 [33] as the feature extractor for all four datasets. We use SGD as meta-optimizer and general gradient descent as inner-optimizer for all four datasets. For all experiments, we sample 1000 training tasks by CATA for meta training and finetune the meta-algorithm after every 200 episodes with data of novel classes. The meta learning rate and inner learning are 0.4 and 0.001 respectively. We use a meta batch size (the amount of training tasks per training step) of 16\8 for {CIFAR-10,SVHN}\{CIFAR-100,Omniglot}. In addition, we choose $k$ to be 10 which is suitable for all datasets. For each training task, we update the corresponding inner-algorithm by 10 steps.

**MEDI-PRO for L2DNCL.** We use a neural network of four convolutional blocks as the embedding function for all datasets following [35]. Each block comprises a 64-filter $3 \times 3$ convolution, BN layer [14], a ReLU function and a $2 \times 2$ max-pooling layer. We use the same embedding function for embedding both training data and test data. For all experiments, we train the models via Adam [17], and we use an initial learning rate of 0.001 and cut the learning rate in half every 20 steps. We train the embedding function for 200 steps with 1000 training tasks sampled by CATA.

E Results of L2DNC

In this section, we show the results of L2DNC with abundant novel-class data in Table 2. Detailed analysis is in Section 6.

F Complexity Analysis

We give a brief analysis of time complexity for each algorithm. As MEDI-MAML and MEDI-PRO are two-step methods, we first analyze the sampling algorithm CATA, and then analyze the main parts of MEDI-MAML and MEDI-PRO.
CATA  The time complexity of CATA is $O(E \times D/B \times T)$, where $F$ is number of training tasks, $E$ is number of epochs, $D$ is size of dataset, $B$ is meta batch size, and $T$ is the time complexity of each iteration. We can future decompose $O(T) = O(L \times n)$, where $L$ is the average time complexity of each layer, and $n$ is number of layers. Then, we can decompose $O(L) = O(M \times N \times K^2 \times H \times W)$, where $M$ and $N$ are numbers of channels of input and output, $K$ is size of convolutional kernel, and $H$ and $W$ are height and weight of feature space.

MEDI-MAML (Main part)  The time complexity of MEDI-MAML is $O(F \times E \times D/B \times T)$, where $F$ is number of training tasks, $E$ is number of epochs, $D$ is size of dataset, $B$ is meta batch size, and $T$ is the time complexity of each iteration. We can future decompose $O(T) = O(L \times n)$, where $L$ is the average time complexity of each layer, and $n$ is number of layers. Then, we can decompose $O(L) = O(M \times N \times K^2 \times H \times W)$, where $M$ and $N$ are numbers of channels of input and output, $K$ is size of convolutional kernel, and $H$ and $W$ are height and weight of feature space.

MEDI-PRO (Main part)  The time complexity of MEDI-PRO is $O(F \times E \times D/B \times T)$, where $F$ is number of training tasks, $E$ is number of epochs, $D$ is size of dataset, $B$ is meta batch size, and $T$ is the time complexity of each iteration. We can future decompose $O(T) = O(L \times n)$, where $L$ is the average time complexity of each layer, and $n$ is number of layers. Then, we can decompose $O(L) = O(M \times N \times K^2 \times H \times W)$, where $M$ and $N$ are numbers of channels of input and output, $K$ is size of convolutional kernel, and $H$ and $W$ are height and weight of feature space.

G  Running Time

In this section, we list the running time of MEDI-MAML and MEDI-PRO for all four datasets. The results are shown in Table 3.

| Methods | SVHN (5-way) | CIFAR-10 (5-way) | CIFAR-100 (20-way) | OmniGlot (20-way) |
|---------|--------------|-----------------|-------------------|------------------|
|         | 5-obs | 1-obs | 5-obs | 1-obs | 5-obs | 1-obs | 5-obs | 1-obs |
| MM      | 5.3h  | 4.6h | 3.1h  | 2.7h | 3.1h | 2.8h | 3.0h | 2.7h |
| MP      | 6.2h  | 6.0h | 2.1h  | 1.6h | 2.0h | 1.6h | 1.2h | 1.0h |

H  Link to Datasets

SVHN dataset can be downloaded from [http://ufldl.stanford.edu/housenumbers/train.tar.gz](http://ufldl.stanford.edu/housenumbers/train.tar.gz) and [http://ufldl.stanford.edu/housenumbers/test.tar.gz](http://ufldl.stanford.edu/housenumbers/test.tar.gz).

CIFAR-10 and CIFAR-100 datasets can be downloaded from [https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz](https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz) and [https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz](https://www.cs.toronto.edu/~kriz/cifar-100-python.tar.gz).

OmniGlot dataset can be downloaded from [https://github.com/brendenlake/omniglot/tree/master/python](https://github.com/brendenlake/omniglot/tree/master/python).