Generating Pseudo-labels Adaptively for Few-shot Model-Agnostic Meta-Learning

Guodong Liu
guodl@hust.edu.cn
Tongling Wang
wangtl@hust.edu.cn
Shuoxi Zhang
zhangshuoxi@hust.edu.cn
Kun He
brooklet60@hust.edu.cn

[1] School of Computer Science and Technology, Huazhong University of Science and Technology.
Wuhan, China
[2] Hopcroft Center on Computing Science, Huazhong University of Science and Technology,
Wuhan, China.

Abstract

Model-Agnostic Meta-Learning (MAML) is a famous few-shot learning method that has inspired many follow-up efforts, such as ANIL and BOIL. However, as an inductive method, MAML is unable to fully utilize the information of query set, limiting its potential of gaining higher generality. To address this issue, we propose a simple yet effective method that generates pseudo-labels adaptively and could boost the performance of the MAML family. The proposed methods, dubbed Generative Pseudo-label based MAML (GP-MAML), GP-ANIL and GP-BOIL (when combined with MAML, ANIL and GP-BOIL respectively), leverage statistics of the query set to improve the performance on new tasks. Specifically, we adaptively add pseudo labels and pick samples from the query set, then re-train the model using the picked query samples together with the support set. The GP series can also use information from the pseudo query set to re-train the network during the meta-testing, while some transductive methods, such as Transductive Propagation Network (TPN), struggle to achieve this goal. Experiments show that all our methods, GP-MAML, GP-ANIL and GP-BOIL, can boost the performance of the corresponding model considerably, and achieve competitive performance as compared to the state-of-the-art baselines. Our code is available at https://github.com/JHL-HUST/GP-MAML.

1 Introduction

Labeled classification models have achieved remarkable achievements in various domains, such as classification on images, texts, audio, or videos. Training these classification models often necessitates a large amount of labeled data. However, it is extremely hard or even impossible to obtain labeled data in some fields, such as medical imaging, military applications. Such challenge leads to *Few-Shot Learning (FSL)*, that aims to train a model with limited training data. From the perspective of the solution approach, FSL models roughly fall into three categories, *i.e.*, optimization-based [3, 8, 9, 10, 11], metric-based [2, 6, 7, 14, 15, 17, 18], and model-based [13]. Since the model-based method is not the focus of our work, we will mainly discuss the first two categories.

The most typical optimization-based method, Model-Agnostic Meta-Learning (MAML) [3], divides the training data into two parts, *i.e.*, support set and query set. Under this setting, the training process consists of two stages: inner loop and outer loop. The inner loop effectively optimizes the initial parameters for unseen tasks with limited labeled support training data, whereas the outer loop can access the loss for optimization. The second category is metric-based, aiming to learn deep embedding with strong generalization ability.

However, as an inductive method, MAML is unable to fully utilize data from the query set to improve the performance. To tackle these difficulties, we take the pseudo query data into account. We investigate the impact of pseudo query data on the performance of the MAML family and observe that: (1) the classifier of MAML family is highly sensitive to the pseudo query data; (2) the feature extractor of MAML family is highly adaptable to the pseudo query data; and (3) the imbalance of pseudo query data may have negative impact.

Motivated by these observations, we study the method of labeling and picking query data as the pseudo labeled data, and propose a new method called the Generative Pseudo-label based MAML (GP-MAML). We generate pseudo-labels by using label propagation with adaptive picking for MAML and its two typical variants, ANIL [10] and BOIL [9], resulting in three new methods, GP-MAML, GP-ANIL, and GP-BOIL. We do not use any data augmentation techniques. As a result, our model can perform targeted parameter updates based on the pseudo query data in both meta-training and meta-testing phases.

The main contributions of this work are summarized as follows:

- To our knowledge, this work is the first to incorporate label propagation used in transductive methods to generate pseudo-labels for MAML, a typical inductive method for few-shot learning.
- We propose to use *adaptive picking* to select instances from the pseudo query set to balance the number of samples for each class, leading to higher performance for models that are even sensitive to these pseudo data.
- We apply our Generative Pseudo-label method (GP) to two typical variants of MAML, and improve their performance, demonstrating the applicability of our approach.
- The evaluation on three benchmark datasets shows that our method outperforms all existing MAML-based methods and achieves competitive performance in comparison with the state-of-the-art few-shot learning methods.

2 Related Work

This section reviews some popular methods on few-shot learning for related works according to the above categories. We first present induction versus transduction, and then introduce optimization-based methods and metric-based methods.

Induction versus Transduction. Induction is a type of reasoning from observed training cases to the general rules, which is then applied to test cases. In the context of few-shot learning, it is hard for the inductive methods [3, 10] to generalize from a small set of training data and adapt to unseen tasks. Another approach to achieve better improvements with limited training data, called transduction or transductive inference, is to consider the relationships between instances in the test set to make predictions on them as a whole. The earliest transductive method was introduced by Vapnik [16], whose purpose is to reduce the classification loss on the specific test set.

Optimization-based methods. The optimization-based methods aim to quickly learn the parameters to be optimized when the model comes across new tasks. As the most far-reaching optimization-based method, MAML [3] trains the model's initial parameters so that the model would have a good performance on new tasks after only a small number of gradient updates. ANIL (Almost No Inner Loop) [10] only updates the classifier parameters of MAML in the inner loop and almost removes the inner loop without reduction in performance. Jaehoon *et al.* [9] show that the success of the MAML family is attributed to the reuse of high-quality features from the meta-initialized parameters and introduce a simple yet effective algorithm called BOIL (Body Only update in Inner Loop).

Metric-based methods. This stream of methods converts the query image and support image to the same embedding space. It then classifies the query images by calculating the distance or similarity among the embedding features. Vinyals *et al.* [17] train a weighted nearest neighbor classifier by the support set and update the model according to the performance on query set. Snell *et al.* [14] introduce Prototypical Networks, which calculate a prototype for each class in the support set. Liu *et al.* [7] propose TPN (Transductive Propagation Network) that propagates labels from labeled support instances to unlabeled query instances by learning a graph construction module that exploits the manifold structure in the data.

In contrast with the above approaches, our method provides a transductive way to make fully use of query set data in the inductive method. To the best of our knowledge, we are the first to incorporate the label propagation to the MAML family and take the pseudo query data into account. Experiments show our framework outperforms all existing methods based on MAML.

3 Methodology

In this section, we first introduce in detail the two most related methods, the MAML series and TPN, and then present our proposed GP-MAML.

3.1 MAML and Its Variants

Model-Agnostic Meta-Learning (MAML) tries to learn an effective meta-initialization so that the model can start from this initialization and achieve excellent results with limited data training. MAML is trained on many tasks $\mathcal{T}$ sampled from the dataset distribution $p(\mathcal{T})$. Each task $\mathcal{T}_i$ consists of a support set spt_i and a query set qry_i. The support set samples under the N-way-K-shot setting (each support set consists of N classes and each class has K samples), the query set contains unlabeled data for evaluation.

The training procedure of MAML includes two stages: the inner loop and the outer loop. In the inner loop, the model first computes the task-specific loss $\mathcal{L}_{spt_i}(f_\theta)$ on the support set, where f_θ is a neural network parameterized by θ. Then MAML conducts a task-specific gradient update on the model:

$$\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{spt_i}(f_\theta). \quad (1)$$

In the outer loop, MAML first computes the loss $\mathcal{L}_{qry_i}\left(f_{\theta_i'}\right)$ of each query set based on the updated parameters of the inner loop. Then the outer loss of all the tasks in a batch is

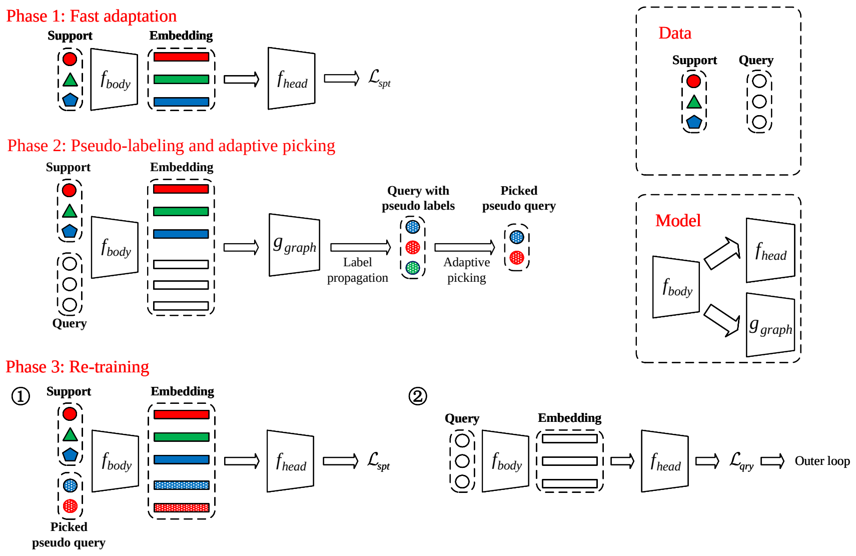


Figure 1: The overall framework of GP-MAML. It contains three phases: fast adaptation, pseudo-labeling and adaptive picking, and re-training. In phase 1, the few-shot model updates its parameters through the support set. In phase 2, we adopt the label propagation of TPN to label the query set, which will be filtered by adaptive picking to select the pseudo query data. In phase 3, the picked pseudo query data will be added to the support set for re-training, then the loss of the query set is calculated as in MAML.

calculated as $\sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{qry_i}\left(f_{\theta_i'}\right)$. The meta-initialized parameters are then updated across the sampled tasks using the gradient descent method with respect to parameters θ:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{qry_i}\left(f_{\theta_i'}\right). \tag{2}$$

There are two main variants based on MAML, *i.e.*, ANIL [10] and BOIL [9]. Both of them split the model parameters θ into two parts, θ_{body} and θ_{head}, representing parameters for feature extractor and classifier, respectively. They have the same outer loop as MAML does. Yet in the inner loop, ANIL only updates the head parameters θ_{head} (Equation 3), while BOIL only updates the body parameters θ_{body} (Equation 4).

$$\theta'_{head_i} = \theta_{head} - \alpha \nabla_{\theta_{head}} \mathcal{L}_{spt_i}\left(f_{\theta_{body},\theta_{head}}\right), \; \theta'_{body_i} = \theta_{body}, \tag{3}$$

$$\theta'_{body_i} = \theta_{body} - \alpha \nabla_{\theta_{body}} \mathcal{L}_{spt_i}\left(f_{\theta_{body},\theta_{head}}\right), \; \theta'_{head_i} = \theta_{head}. \tag{4}$$

3.2 Transductive Propagation Network

Transductive Propagation Network (TPN) [7] is a transductive metric-based method in which label propagation [19] is critical to the success. TPN uses the convolutional neural network f to perform feature extraction on the support set and the query set for a task. The resulting feature maps $f(\mathbf{x}_i), \mathbf{x}_i \in spt \cup qry$ are used to calculate the Gaussian similarity between the samples:

$$W_{ij} = \exp\left(-\frac{1}{2} d\left(\frac{f(\mathbf{x}_i)}{\sigma_i}, \frac{f(\mathbf{x}_j)}{\sigma_j}\right)\right), \tag{5}$$

where $d(\cdot,\cdot)$ is the Euclidean distance, $\sigma_i = g(f(\mathbf{x}_i))$ is an example-wise length-scale parameter calculated by the graph construction module g. Label propagation is based on the obtained W_{ij}. Firstly, it applies the normalized graph Laplacians on W as follows:

$$S = D^{-1/2} W D^{-1/2}, \tag{6}$$

where D is a diagonal matrix. Then, TPN defines a label matrix Y with $Y_{ij} = 1$ if $\mathbf{x}_i$ is from the support set and labeled as $\mathbf{y}_i = j$. The solution for the predicted labels is as follows:

$$F^* = (I - \alpha S)^{-1} Y, \tag{7}$$

where $\alpha \in (0,1)$ is a hyper-parameter controlling the amount of propagated information, and I is the identity matrix [19].

Algorithm 1 The Adaptive Picking Method

```
Input: $CM$: confidence measure of query set
Input: query set $qry = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$, where $\mathbf{y}_i \in \{1, \ldots, C\}$
Output: the picked pseudo query set $P$
1: Calculate the number of picked samples in each category: $k = \min_{i=1}^{C} \sum_{j=1}^{N} I(\mathbf{y}_j = i)$
2: for $i$ in $\{1, \ldots, C\}$ do
3:     Pick the top $k$ confidence samples in category $i$ to join $P$
4: return $P$
```

3.3 The Proposed GP-MAML

As an inductive method, MAML can not fully utilize the data of the query set to achieve better performance. Even for transductive methods, it is also hard to update their parameters during meta-testing according to the query set. To tackle these issues, we introduce Generative Pseudo-label (GP) into MAML, making it possible to leverage the statistic information of the query set in both meta-training and meta-testing.

Our framework is illustrated in Fig. 1. The classifier f_{head} and the graph construction module g_{graph} share the same feature extractor f_{body}. Our training strategy consists of three phases:

Phase 1: Fast adaptation. We use the feature extractor f_{body} to extract features of the support set and classify these features using f_{head}. Then we update the parameters of f_{body} using the loss $\mathcal{L}_{spt}$.

Phase 2: Pseudo-labeling and adaptive picking. We first use label propagation to label the query set. Specifically, we use the updated f_{body} obtained in phase 1 to re-extract features of the support set and the query set. The resulting features are embedded into a new manifold structure using the graph construction module g_{graph}, which is then used for label propagation. We take the label propagation result of confidence measure (CM) as input and label the data in the query set. However, the pseudo query data can not be directly used to expand the support set because the number of samples in different classes of the pseudo query data varies greatly. Thus we propose *adaptive picking*, as shown in Algorithm 1, by taking the CM and pseudo query data as input. Then we calculate the smallest sample size (say k) among all categories and pick the top k samples for each category according to CM.

Algorithm 2 The GP-MAML algorithm

```
Require: $p(\mathcal{T})$: distribution over tasks
Require: $\alpha, \beta, \gamma$ : hyperparameters on step size
Require: $f$: backbone network model with parameters $\theta_{body}$, $\theta_{head}$
Require: $g$: graph construction model with parameter $\theta_{graph}$

1:  Randomly initialize $\theta_{body}$, $\theta_{head}$, $\theta_{graph}$
2:  while not done do
3:      Sample a batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4:      for all $\mathcal{T}_i$ do
5:          $spt_i, qry_i \leftarrow \mathcal{T}_i$
6:          Evaluate $\nabla_{\theta_{body}} \mathcal{L}_{spt_i}\left(f_{\theta_{body},\theta_{head}}\right)$ with respect to support set examples
7:          Compute adapted body parameters with gradient descent:
                $\theta^{TPN}_{body_i} = \theta_{body} - \alpha \nabla_{\theta_{body}} \mathcal{L}_{spt_i}\left(f_{\theta_{body},\theta_{head}}\right)$
8:          Compute embedded feature maps with support set and query set examples and construct the weighted graph:
                $Graph_i = g_{\theta_{graph}}(f_{\theta^{TPN}_{body_i}}(spt_i, qry_i))$
9:          Obtain CM and pseudo query data of $\mathcal{T}_i$ using label propagation:
                $CM_i, PLabel_i = LabelPropagation(Graph_i)$
10:         Expand support set with picked pseudo query data:
                $spt'_i \leftarrow spt_i + AdaptivePicking(CM_i, PLabel_{query_i})$
11:         Re-update parameters with new support set examples:
                $\theta'_{body_i}, \theta'_{head_i} \leftarrow \theta_{body}, \theta_{head} - \beta \nabla_{\theta_{body},\theta_{head}} \mathcal{L}_{spt'_i}(f_{\theta_{body},\theta_{head}})$
12:     Update $\theta_{body}, \theta_{head} \leftarrow \theta_{body}, \theta_{head} - \beta \nabla_{\theta_{body},\theta_{head}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{qry_i}(f_{\theta'_{body_i},\theta'_{head_i}})$
13:     Update $\theta_{graph} \leftarrow \theta_{graph} - \gamma \nabla_{\theta_{graph}} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{qry_i}\left(g_{\theta_{graph}}\right)$
```

Phase 3: Re-training. All the samples chosen in phase 2, together with the original support set, are then used to re-train f_{body} and f_{head}. The inner loop of GP-MAML is then competed after calculating the loss $\mathcal{L}_{qry}$ of the original query set through the re-trained f_{body} and f_{head}. In the outer loop, we sum the loss $\mathcal{L}_{qry}$ of each task and update f_{body} and f_{head} with the original parameters as MAML does. Details of GP-MAML are presented in Algorithm 2. We also generate pseudo-labels for ANIL and BOIL by using label propagation with adaptive picking, denoted as GP-ANIL and GP-BOIL, respectively.

4 Experiments

In this section, following the line of MAML, ANIL, and BOIL, we provide practical details of the method presented in Section 3.3 and examine their effectiveness. We conduct a comprehensive experimental analysis of our GP-MAML, GP-ANIL, and GP-BOIL, and do ablation studies to verify the effectiveness of our proposed Adaptive Picking (AP) method.

4.1 Datasets

We choose three datasets for experiments.

MiniImageNet. The miniImageNet dataset [12] is the most popular few-shot learning benchmark. It is composed of $60,000$ images selected from the ImageNet dataset [5], with a total of 100 categories. Each category has 600 images, and the size of each image is 84×84. We follow the class splits used by Ravi and Larochelle [12], which include 64 classes for training, 16 classes for validation, and 20 classes for testing.

CIFAR-FS. The CIFAR-FS dataset is derived from the CIFAR100 dataset and contains 100 categories, each having 600 images, with a total of $60,000$ images. It is usually divided into training set (64 categories), validation set (16 categories), and testing set (20 categories).

FC100. The Few-shot CIFAR100 dataset (FC100) contains 20 superclasses (60 categories), including 12 superclasses in the training set, 4 superclasses (20 categories) in the validation set, and 4 superclasses (20 categories) in the testing set.

4.2 Experimental Setup

The backbone few-shot network F is a four convolutional-block network with 64 channels and a fully-connected layer, following Jaehoon *et al.*'s setting [9]. The graph construction module is composed of two convolutional blocks and two fully-connected layers, according to the TPN's setting [7], offering an example-wise scaling parameter. The hyper-parameter k of k-nearest neighbor graph is set to 20, and the α for label propagation is set to 0.99, as suggested in TPN. We use the batch size of 64 for all networks. The model is trained with $30,000$ episodes. The initial learning rate of the optimizer is e^{-3}, and drops by a factor of 10 after $10,000$ and $20,000$ episodes, respectively.

4.3 Results and Discussions

4.3.1 Improve the MAML family using the query set

For MAML, ANIL and BOIL, the model performs a fast adaptation through the support set, and the updated model is used to calculate the loss of the query set. However, as inductive methods, they can not fully utilize the data statistics from the query set to achieve better performance. Hence, we first conjecture that the model's performance would be enhanced if it could take the query data into account during the fast adaption process and make targeted updates for the query data. Specifically, we first utilize the model's own classifier to pseudo-label the query set data after fast adaptation, and then re-train the model with all the pseudo query data and the support set for meta-testing. Table 1 shows the experimental results.

Method	miniImageNet		CIFAR-FS		FC100	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML	48.24 ± 0.32	61.52 ± 1.95	57.57 ± 0.16	72.31 ± 0.05	36.64 ± 0.05	47.26 ± 0.13
ANIL	49.61 ± 0.27	64.90 ± 0.63	58.34 ± 0.03	72.34 ± 0.01	36.38 ± 0.16	47.16 ± 0.08
BOIL	49.82 ± 0.24	67.60 ± 0.01	58.69 ± 0.11	75.31 ± 0.02	38.80 ± 0.03	51.42 ± 0.20
MAML (*w. qry*)	34.89 ± 0.41	50.11 ± 0.14	43.71 ± 0.30	58.40 ± 0.43	25.24 ± 0.15	30.51 ± 0.28
ANIL (*w. qry*)	37.49 ± 0.58	48.41 ± 1.78	46.23 ± 0.25	59.52 ± 0.08	25.68 ± 0.12	30.97 ± 0.29
BOIL (*w. qry*)	$\mathbf{49.88\pm0.46}$	$\mathbf{68.08\pm0.15}$	$\mathbf{60.91\pm0.13}$	$\mathbf{76.32\pm0.03}$	$\mathbf{39.67\pm0.02}$	$\mathbf{52.02\pm0.21}$

Table 1: Comparison on miniImageNet, CIFAR-FS, and FC100 under the 5-way setting. *w. qry* indicates the accuracy when adding the query set to the support set after pseudo-labeling.

The results show that after adding the pseudo query data into the support set for fast adaptation, only BOIL maintains its performance, but MAML and ANIL perform poorly. The reason would be related to the *representation reuse* and *representation change* [9]. The meta-initialization of MAML and ANIL offers efficient *representation reuse* through the body before fast adaptation. Despite the fact that the meta-initialization of BOIL provides less efficient *representation reuse* compared to MAML and ANIL, the BOIL's body can update its parameters to extract more efficient representations through task-specific adaptation, which is called the *representation change*. This means BOIL is less sensitive to pseudo query data than MAML and ANIL because of the *representation change*.

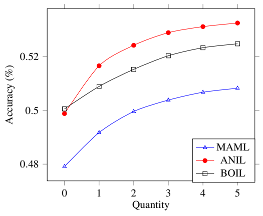
4.3.2 Make pseudo query data more balanced by a careful picking

During the experiments, we also observe that the number of samples in different pseudo query data categories varies greatly. So models that are very sensitive to the pseudo query data like MAML and ANIL can not directly use the pseudo query data. To tackle this issue, we propose an adaptive picking method to select as much pseudo query data as possible while maintaining a balanced number of samples for each category. The comparison results of using adaptive picking are shown in Table 2.

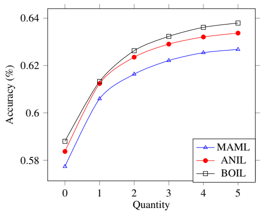
Method	miniImageNet		CIFAR-FS		FC100	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML	48.24 ± 0.32	61.52 ± 1.95	57.57 ± 0.16	72.31 ± 0.05	36.64 ± 0.05	47.26 ± 0.13
ANIL	49.61 ± 0.27	64.90 ± 0.63	58.34 ± 0.03	72.34 ± 0.01	36.38 ± 0.16	47.16 ± 0.08
BOIL	49.82 ± 0.24	67.60 ± 0.01	58.69 ± 0.11	75.31 ± 0.02	38.80 ± 0.03	51.42 ± 0.20
MAML(*w. AP*)	51.37 ± 0.41	63.82 ± 2.39	62.58 ± 0.27	75.08 ± 0.05	38.65 ± 0.02	48.92 ± 0.12
ANIL(*w. AP*)	$\mathbf{53.00\pm0.33}$	67.86 ± 0.71	63.44 ± 0.07	75.12 ± 0.03	38.37 ± 0.20	48.71 ± 0.12
BOIL(*w. AP*)	52.42 ± 0.29	$\mathbf{69.84\pm0.01}$	$\mathbf{63.51\pm0.11}$	$\mathbf{77.76\pm0.09}$	$\mathbf{40.76\pm0.01}$	$\mathbf{52.97\pm0.24}$

Table 2: Comparison results on miniImagenetNet, CIFAR-FS, FC100 under the 5-way setting. *w. AP* indicates the accuracy after adaptive picking be used to select pseudo query data to add to the support set.

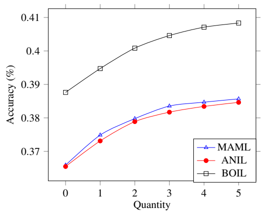
The results show that the pseudo query data can be well adapted to MAML, ANIL and BOIL after Adaptive Picking, We also calculate the accuracy of the model with an increasing number of samples picked for each category under the setting of Adaptive Picking. The results are shown in Fig. 2, indicating that more pseudo query data per class is beneficial.



(a) miniImageNet



(b) CIFAR-FS



(c) FC100

Figure 2: Performance for various number of selected pseudo query data per class. It shows that under the setting of adaptive picking, the model performance increases steadily with the increment on the number of pseudo-label samples.

4.3.3 Consider the relationship among samples

The above experiments show two key factors for better performance: the model's feature extractor and the method of labelling and picking pseudo query data. Based on these observations, we propose GP-MAML, GP-ANIL, and GP-BOIL. Specifically, when labeling the query set data, we use the body of MAML, ANIL, or BOIL and the graph construction module of TPN to get a weighted graph, and then use label propagation to label data in the query set all at once, which means we can make the query data more balanced by considering the relationship between query set samples. In the end, we use adaptive picking to select pseudo query data which is then used to re-train the model. The results shows that our GP-MAML, GP-ANIL and GP-BOIL outperform MAML, ANIL and BOIL, respectively. GP-BOIL outperforms others because the BOIL's body can update its parameters to extract more efficient representations through task-specific adaptation, which is called the *representation change*. Our model also outperforms TPN because our model can utilize information from the unlabeled query set to re-train the network in a targeted manner in meta-testing, but it is difficult for TPN to perform in this way. For MAML-based methods, the outer loop takes up most of the training time and memory. The outer loops of GP-MAML and MAML are the same. Hence both the training time and memory of GP-MAML are almost the same as that of MAML. For the MAML-based methods, only increasing the number of inner loops per outer loop has less impact on results. The results of our method are shown in Table 3.

5 Conclusion

In this work, we generated pseudo-labels by using label propagation with adaptive picking, introduced transductive methods to typical inductive methods, *i.e.*, the MAML series, and thereby improving their performance. We started by taking the pseudo query data into account, addressing the problem that inductive methods can not fully utilize information of the query set. Since the classifier of inductive methods is sensitive to the pseudo query data, we employed feature extractor and label propagation to label the query set. We also proposed a simple yet effective method called adaptive picking to select samples from distinct classes with balanced quantity. Experiments show that when MAML, ANIL, and BOIL are re-trained with pseudo-labeled data, they are all boosted with higher performance.

There are certain observations gained from our work. First, the query data can be well utilized by pseudo-labeling. Second, the imbalances in pseudo query data would have neg-

Method	miniImageNet		CIFAR-FS		FC100	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
MAML [3]	48.24 ± 0.32	61.52 ± 1.95	57.57 ± 0.16	72.31 ± 0.05	36.64 ± 0.05	47.26 ± 0.13
ANIL [10]	49.61 ± 0.27	64.90 ± 0.63	58.34 ± 0.03	72.34 ± 0.01	36.38 ± 0.16	47.16 ± 0.08
BOIL [9]	49.82 ± 0.24	67.60 ± 0.01	58.69 ± 0.11	75.31 ± 0.02	38.80 ± 0.03	51.42 ± 0.20
Reptile [8]	47.07 ± 0.26	62.74 ± 0.37	–	–	–	–
Prototypical Net [14]	49.42 ± 0.78	68.20 ± 0.66	55.50 ± 0.70	72.00 ± 0.60	35.30 ± 0.60	48.60 ± 0.60
BatchProx [20]	48.51 ± 0.92	64.15 ± 0.92	–	–	–	–
TPN [7]	54.41 ± 0.49	69.54 ± 0.33	63.53 ± 0.28	74.03 ± 0.90	38.11 ± 0.13	48.48 ± 0.27
Reptile + BN [8]	49.97 ± 0.32	65.99 ± 0.58	–	–	–	–
RelationNet + BN [15]	50.44 ± 0.82	65.32 ± 0.70	–	–	–	–
BatchProx + BN [20]	50.77 ± 0.90	67.43 ± 0.89	–	–	–	–
MAML + BN [3]	48.70 ± 1.84	63.11 ± 0.92	–	–	–	–
MAML++ + BN [1]	52.15 ± 0.26	68.32 ± 0.44	–	–	–	–
GP-MAML (Ours)	52.71 ± 0.20	68.06 ± 0.62	64.03 ± 0.06	75.60 ± 0.27	38.57 ± 0.36	48.50 ± 0.52
GP-ANIL (Ours)	**55.92** ± **0.50**	70.73 ± 0.59	65.66 ± 0.25	75.08 ± 2.57	38.95 ± 0.03	51.16 ± 0.19
GP-BOIL (Ours)	55.55 ± 0.10	**71.36** ± **0.12**	**66.55** ± **0.05**	**78.50** ± **0.34**	**41.80** ± **0.12**	**53.17** ± **0.04**

Table 3: Comparison on miniImageNet, CIFAR-FS, FC100 under the 5-way setting. GP-MAML, GP-ANIL and GP-BOIL outperform MAML, ANIL and BOIL, respectively. All accuracy results are averaged over 600 test episodes. *BN* means that information is shared between the test samples via batch normalization [4].

ative impact on models that are very sensitive to data. Third, the classifier of MAML is not appropriate for pseudo-labeling, and other ways based on the feature extractor of MAML should be considered. We will follow this line for further exploration in future works.

Acknowledgements

This work is supported by National Natural Science Foundation (U22B2017).

References

[1] Antreas Antoniou, Harrison Edwards, and Amos J. Storkey. How to train your MAML. In *7th International Conference on Learning Representations*, 2019.

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *7th International Conference on Learning Representations*, 2019.

[3] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[4] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: Annual Conference on Neural Information Processing Systems*, 2012.

[6] Chengcheng Liu, Dexing Zhong, and Huikai Shao. Few-shot palmprint recognition based on similarity metric hashing network. *Neurocomputing*, 456:540–549, 2021.

[7] 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 *7th International Conference on Learning Representations*, 2019.

[8] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *Computer Research Repository*, 2018.

[9] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. BOIL: towards representation change for few-shot learning. In *9th International Conference on Learning Representations*, 2021.

[10] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *8th International Conference on Learning Representations*, 2020.

[11] Aravind Rajeswaran, Chelsea Finn, Sham M. Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, 2019.

[12] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations*, 2017.

[13] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. One-shot learning with memory-augmented neural networks. *Computer Research Repository*, 2016.

[14] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*, 2017.

[15] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[16] Vladimir Vapnik. An overview of statistical learning theory. *IEEE Trans. Neural Networks*, 1999.

[17] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems*, 2016.

[18] Yunlong Yu, Dingyi Zhang, Sidi Wang, Zhong Ji, and Zhongfei Zhang. Local spatial alignment network for few-shot learning. *Neurocomputing*, 497:182–190, 2022.

[19] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16: Annual Conference on Neural Information Processing Systems*, 2003.

[20] Pan Zhou, Xiaotong Yuan, Huan Xu, Shuicheng Yan, and Jiashi Feng. Efficient meta learning via minibatch proximal update. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems*, 2019.