MetaNODE: Prototype Optimization as a Neural ODE for Few-Shot Learning

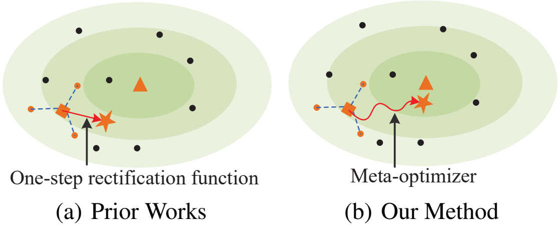
Baoquan Zhang, Xutao Li*, Shanshan Feng, Yunming Ye*, Rui Ye
Harbin Institute of Technology, Shenzhen
zhangbaoquan@stu.hit.edu.cn, {lixutao, victor_fengss, yeyunming}@hit.edu.cn, yerui_hitsz@163.com

Abstract

Few-Shot Learning (FSL) is a challenging task, *i.e.*, how to recognize novel classes with few examples? Pre-training based methods effectively tackle the problem by pre-training a feature extractor and then predicting novel classes via a cosine nearest neighbor classifier with mean-based prototypes. Nevertheless, due to the data scarcity, the mean-based prototypes are usually biased. In this paper, we attempt to diminish the prototype bias by regarding it as a prototype optimization problem. To this end, we propose a novel meta-learning based prototype optimization framework to rectify prototypes, *i.e.*, introducing a meta-optimizer to optimize prototypes. Although the existing meta-optimizers can also be adapted to our framework, they all overlook a crucial gradient bias issue, *i.e.*, the mean-based gradient estimation is also biased on sparse data. To address the issue, we regard the gradient and its flow as meta-knowledge and then propose a novel Neural Ordinary Differential Equation (ODE)-based meta-optimizer to polish prototypes, called MetaNODE. In this meta-optimizer, we first view the mean-based prototypes as initial prototypes, and then model the process of prototype optimization as continuous-time dynamics specified by a Neural ODE. A gradient flow inference network is carefully designed to learn to estimate the continuous gradient flow for prototype dynamics. Finally, the optimal prototypes can be obtained by solving the Neural ODE. Extensive experiments on miniImagenet, tieredImagenet, and CUB-200-2011 show the effectiveness of our method. Our code is available [1].

1 Introduction

With abundant annotated data, deep learning techniques have shown very promising performance for many applications, *e.g.*, image classification (He et al. 2016). However, preparing enough annotated samples is very time-consuming, laborious, or even impractical in some scenarios, *e.g.*, cold-start recommendation (Zheng et al. 2021) and medical diagnose (Prabhu et al. 2019). Few-shot learning (FSL), which aims to address the issue by mimicking the flexible adaptation ability of human to novel tasks from very few examples, has been proposed and received considerable attentions. Its main rationale is to learn meta-knowledge from base classes with sufficient labeled samples and then employ the meta-knowledge to perform class prediction for novel classes with scarce examples (Li et al. 2019a).



(a) Prior Works (b) Our Method

Figure 1: Pre-training based method estimates prototypes in an average manner, which suffers from a prototype bias issue. Prior works diminish the bias in a one-step manner (a). Our method addresses it in a meta-optimization manner (b). Here, orange and black points denote training and test samples, respectively. Orange square, star, and triangle denotes the mean-based, rectified, and real prototypes, respectively.

Previous studies primarily address the FSL problem using the idea of meta-learning, *i.e.*, constructing a large set of few-shot tasks on base classes to learn task agnostic meta-knowledge (Flennerhag et al. 2020). Recently, Chen et al. regard feature representation as meta-knowledge, and propose a simple pre-training method (Chen et al. 2020), which delivers more promising performance. In the method, they first pre-train a feature extractor on all base classes, and then perform novel class prediction via mean-based prototypes. However, this method suffers from a **prototype bias issue**, *i.e.*, the discrepancy between calculated mean and real prototypes. As illustrated in Figure 1, the mean-based prototype (orange square) is usually far away from real prototype (triangle). This kind of prototype bias is caused by the fact that scarce labeled samples cannot provide a reliable mean estimation for the prototypes (Liu, Song, and Qin 2020). To address the drawback, some prior works attempt to learn a one-step prototype rectification function from a large set of few-shot tasks (Liu, Song, and Qin 2020; Xue and Wang 2020; Zhang et al. 2021a), which is shown in Figure 1(a). However, characterizing the prototype bias with a one-step rectification function is too coarse to obtain accurate prototypes (as we will see in Table 3 of the Section 4.4).

In this paper, we propose a novel meta-learning based pro-

*Corresponding author.

[1] https://github.com/zhangbq-research/MetaNODE

totype optimization framework to rectify the prototype bias. In the framework, instead of using the one-step rectification manner, we consider the bias reduction as a prototype optimization problem and attempt to diminish the prototype bias with an optimization-based meta-learning method (called meta-optimizer). The idea behind such design is learning a novel Gradient Descent Algorithm (GDA) on base classes and then utlizing it to polish the prototypes via a few gradient update steps for novel classes. Specifically, we first pre-train a classifier on all base classes to obtain a good feature extractor. Then, given a few-shot task, as shown in Figure 1(b), we average the extracted features of all labeled samples as the initial prototype for each class. As a sequel, these prototypes will be further optimized to reduce the prototype bias through a meta-optimizer. Finally, we perform the class prediction via a nearest neighbor classifier.

The workhorse of our framework is the meta-optimizer, in which the mean-based prototypes will be further polished through GDA. Even though the existing meta-optimizer such as ALFA (Baik et al. 2020) and MetaLSTM (Ravi and Larochelle 2017) can also be utilized for this purpose, they all suffer from a common drawback, called **gradient bias issue**, *i.e.*, their gradient estimation is inaccurate on sparse data. The issue appears because all the existing meta-optimizers carefully model the hyperparameters (*e.g.*, initialization (Raghu et al. 2020) and regularization parameters (Baik et al. 2020; Flennerhag et al. 2020)) in GDA as meta-knowledge, but roughly estimate the gradient in an average manner with very few labeled samples, which is usually inaccurate. Given that the gradient estimation is inaccurate, more excellent hyperparameters are not meaningful and cannot lead to a stable and reliable prototype optimization.

To address the issue, we treat the gradient and its flow in GDA as meta-knowledge, and propose a novel Neural Ordinary Differential Equation (ODE)-based meta-optimizer to model the process of prototype optimization as continuous-time dynamics specified by a Neural ODE, called MetaNODE. The idea is inspired by the fact that the GDA formula is indeed an Euler-based discrete instantiation of an ODE (Bu, Xu, and Chen 2020), and the ODE will turn into a Neural ODE (Chen et al. 2018) when we treat its gradient flow as meta-knowledge. The advantage of such meta-optimizer is the process of prototype rectification can be characterized in a continuous manner, thereby producing more accurate prototypes for FSL. Specifically, in the meta-optimizer, a gradient flow inference network is carefully designed, which learns to estimate the continuous-time gradient flow for prototype dynamics. Then, given an initial prototype (*i.e.*, the mean-based prototype), the optimal prototype can be obtained by solving the Neural ODE for FSL.

Our main contributions can be summarized as follows:

- We propose a new perspective to rectify prototypes for FSL, by regarding the bias reduction problem as a prototype optimization problem, and present a novel meta-learning based prototype optimization framework.
- We identify a crucial issue of existing meta-optimizers, *i.e.*, gradient bias issue, which impedes their applicability to our framework. To address the issue, we propose a novel Neural ODE-based meta-optimizer (MetaNODE) by modeling the process of prototype optimization as continuous-time dynamics specified by a Neural ODE. Our MetaNODE optimizer can effectively alleviate the gradient bias issue and leads to more accurate prototypes.
- We conduct comprehensive experiments on both transductive and inductive FSL settings, which demonstrate the effectiveness of our method.

2 Related Work

2.1 Few-Shot Learning

FSL is a challenging task, aiming to recognize novel classes with few labeled samples. According to the test setting, FSL can be divided into two groups, *i.e.*, inductive FSL and transductive FSL. The former assumes that information from test data cannot be utilized when classifying the novel class samples while the latter considers that all the test data can be accessed to make novel class prediction.

In earlier studies, most methods mainly focus on inductive FSL setting, which can be roughly grouped into three categories. 1) Metric-based approaches. This line of works focuses on learning a task-agnostic metric space and then predicting novel classes by a nearest-centroid classifier with Euclidean or cosine distance such as (Nguyen et al. 2020; Snell, Swersky, and Zemel 2017; Li et al. 2019b). 2) Optimization-based approaches. The key idea is to model an optimization algorithm (*i.e.*, GDA) over few labeled samples within a meta-learning framework (von Oswald et al. 2021; Raghu et al. 2020), which is known as meta-optimizer, such as MetaLSTM (Ravi and Larochelle 2017) and ALFA (Baik et al. 2020). 3) Pre-training based approaches. This type of works mainly utilizes a two-phases training manner to quickly adapt to novel tasks, *i.e.*, pre-training and fine-tuning phases, such as (Liu et al. 2020; Shen et al. 2021; Chen et al. 2020; Zhang et al. 2021b; Li, Fu, and Gong 2021).

Recently, several studies explored transductive FSL setting and showed superior performance on FSL, which can be divided into two categories. 1) Graph-based approaches. This group of methods attempts to propagate the labels from labeled samples to unlabeled samples by constructing an instance graph for each FSL task, such as (Chen et al. 2021; Yang et al. 2020; Tang et al. 2021). 2) Pre-training based approaches. This kind of studies still focuses on the two-stages training paradigms. Different from inductive FSL methods, these methods further explore the unlabeled samples to train a better classifier (Boudiaf et al. 2020; Hu et al. 2020; Wang et al. 2020; Ziko et al. 2020) or construct more reliable prototypes (Xue and Wang 2020; Zhang et al. 2021a). In this paper, we also target at obtaining reliable prototypes for FSL. Different from these existing methods, we regard it as a prototype optimization problem and propose a new Neural ODE-based meta-optimizer for optimizing prototypes.

2.2 Neural ODE

Neural ODE, proposed by (Chen et al. 2018), is a continuous-time model, aiming to capture the evolution process of a state by representing its gradient flow with a neural network. Recently, it has been successfully used in various

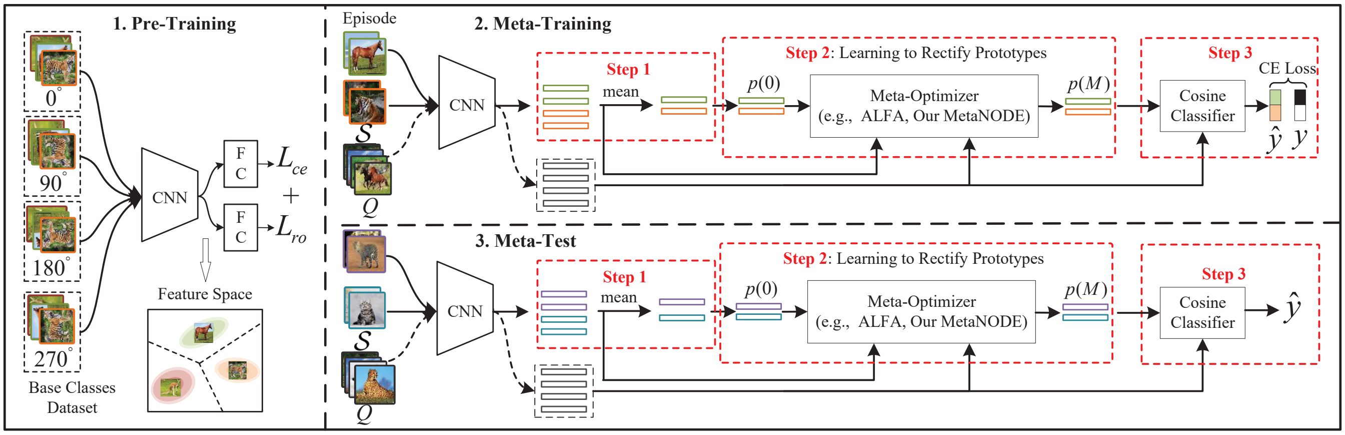


Figure 2: The meta-learning based prototype optimization framework, which rectifies prototypes in a meta-optimization manner.

domains, such as irregular time series prediction (Rubanova, Chen, and Duvenaud 2019), knowledge graph forecasting (Ding et al. 2021), MRI image reconstruction (Chen, Chen, and Sun 2020), and image dehazing (Shen et al. 2020). However, to our best knowledge, there is few previous works to explore it for FSL. In this paper, we propose a Neural ODE-based meta-optimizer to polish prototypes. Its advantage is that prototype dynamics can be captured in a continuous manner, which produces more accurate prototypes for FSL.

3 Methodology

3.1 Problem Definition

For a N-way K-shot problem, two datasets are given: a base class dataset $\mathcal{D}_{base}$ and a novel class dataset $\mathcal{D}_{novel}$. The base class dataset $\mathcal{D}_{base} = \{(x_i, y_i)\}_{i=0}^{B}$ is made up of abundant labeled samples, where each sample x_i is labeled with a base class $y_i \in \mathcal{C}_{base}$ ($\mathcal{C}_{base}$ denotes the set of base classes). The novel class dataset consists of two subsets: a training set $\mathcal{S}$ with few labeled samples (called support set) and a test set $\mathcal{Q}$ consisting of unlabeled samples (called query set). Here, the support set $\mathcal{S}$ is composed of N classes sampled from the set of novel class $\mathcal{C}_{novel}$, and each class only contains K labeled samples. Note that the base class set and novel class set are disjoint, *i.e.*, $\mathcal{C}_{base} \cap \mathcal{C}_{novel} = \emptyset$.

For transductive FSL, we regard all query samples $x \in \mathcal{Q}$ as unlabeled sample set $\mathcal{Q}'$. Our goal is to learn a classifier for query set $\mathcal{Q}$ by leveraging unlabeled sample set $\mathcal{Q}'$, support set $\mathcal{S}$, and base class dataset $\mathcal{D}_{base}$. However, for inductive FSL, the classifier is obtained only by leveraging $\mathcal{S}$ and $\mathcal{D}_{base}$. In the following subsections, we mainly focus on transductive FSL to introduce our method, and how to adapt to inductive FSL will be explained at the end of the section.

3.2 Prototype Optimization Framework

In this paper, we focus on addressing the prototype bias issue appearing in the pre-training FSL method (Chen et al. 2020). Different from existing one-step rectification methods (Liu, Song, and Qin 2020; Xue and Wang 2020), we regard the bias reduction as a prototype optimization problem and present a novel meta-learning based prototype optimization framework to rectify prototypes. Our idea is introducing a prototype meta-optimizer to learn to diminish the prototype bias. As shown in Figure 2, the framework consists of three phases, *i.e.*, pre-training, meta-training, and meta-test phases. Next, we detail on them, respectively.

Pre-Training. Following (Rodríguez et al. 2020), we first pretrain a feature extractor $f_{\theta_f}()$ with parameters θ_f by minimizing both a classification loss L_{ce} and an auxiliary rotation loss L_{ro} on all base classes. This aims to obtain a good image representation. Then, the feature extractor is frozen.

Meta-Training. Upon the feature extractor $f_{\theta_f}()$, we introduce a meta-optimizer $g_{\theta_g}()$ to learn to rectify prototypes in an episodic training paradigm (Vinyals et al. 2016). The idea behind such design is that learning task-agnostic meta-knowledge about prototype rectification from base classes and then applying this meta-knowledge to novel classes to obtain more reliable prototypes for FSL. The main details of the meta-optimizer will be elaborated in Section 3.3. Here, we first introduce the workflow depicted in Figure 2.

As shown in Figure 2, following the episodic training paradigm (Vinyals et al. 2016), we first mimic the test setting and construct a number of N-way K-shot tasks (called episodes) from base class dataset $\mathcal{D}_{base}$. For each episode, we randomly sample N classes from base classes $\mathcal{C}_{base}$, K images per class as support set S, and M images per class as query set Q. Then, we train the above meta-optimizer $g_{\theta_g}()$ to polish prototypes by minimizing the negative log-likelihood estimation on the query set $\mathcal{Q}$. That is,

$$\min_{\theta_g} \mathbb{E}_{(\mathcal{S},\mathcal{Q})\in\mathbb{T}} \sum_{(x_i,y_i)\in\mathcal{Q}} -log(P(y_i|x_i, \mathcal{S}, \mathcal{Q}', \theta_g)), \quad (1)$$

where $\mathbb{T}$ is the set of constructed N-way K-shot tasks and θ_g denotes the parameters of the meta-optimizer $g_{\theta_g}()$. Next we introduce how to calculate the class probability $P(y_i|x_i, \mathcal{S}, \mathcal{Q}', \theta_g)$, including the following three steps:

Step 1. We first leverage the feature extractor $f_{\theta_f}()$ to represent each image. Then we compute the mean-based prototype of each class k as initial prototype $p_k(0)$ at $t = 0$:

$$p_k(0) = \frac{1}{|\mathcal{S}_k|} \sum_{(x_i,y_i)\in\mathcal{S}_k} f_{\theta_f}(x_i), \quad (2)$$

where $\mathcal{S}_k$ is the support set extracted from class k and t denotes the iteration step (when existing meta-optimizers are employed) or the continous time (when our MetaNODE described in Section 3.3 is utilized). For clarity, we denote the prototype set $\{p_k(t)\}_{k=0}^{N-1}$ as the prototypes $p(t)$ of classifiers at iteration step/time t, *i.e.*, $p(t) = \{p_k(t)\}_{k=0}^{N-1}$.

Step 2: Unfortunately, the initial prototypes $p(0)$ are biased since only few support samples are available. To eliminate the bias, we view the prototype rectification as an optimization process. Then, given the initial prototypes $p(0)$, the optimal prototypes $p(M)$ can be obtained by leveraging the meta-optimizer $g_{\theta_g}()$ to optimize prototypes. That is,

$$p(M) = \Psi(g_{\theta_g}(), p(0), \mathcal{S}, \mathcal{Q}', t = M), \tag{3}$$

where M is the total iteration number/integral time and $\Psi()$ denotes the process of prototype optimization. Please refer to Section 3.3 for the details of the optimization process.

Step 3: Finally, we regard the optimal prototypes $p(M)$ as the final prototypes. Then, we evaluate the class probability that each sample $x_i \in \mathcal{Q}$ belongs to class k by computing the cosine similarity between x_i and $p(M)$. That is,

$$P(y = k|x_i, \mathcal{S}, \mathcal{Q}', \theta_g) = \frac{e^{\gamma \cdot < f_{\theta_f}(x_i), p_k(M) >}}{\sum_c e^{\gamma \cdot < f_{\theta_f}(x_i), p_c(M) >}}, \tag{4}$$

where $< \cdot >$ denotes the cosine similarity, and γ is a scale parameter. Following (Chen et al. 2019), $\gamma = 10$ is used.

Meta-Test. Its workflow is similar to the meta-training phase. The difference is that we remove the meta-optimizer training step defined in Eq. 1 and directly perform few-shot classification for novel classes by following Eqs. 2 $\sim$ 4.

3.3 Meta-Optimizer

In the FSL framework described in Section 3.2, the key challenge is how to design a meta-optimizer to polish prototypes. In this subsection, we first discuss the limitation of existing meta-optimizers when using them to polish prototypes. Then, a novel meta-optimizer, *i.e.*, MetaNODE, is presented.

Limitations. In the above framework, several existing meta-optimizers can be utilized to diminish prototype bias by setting the prototypes as their variables to be updated, such as MetaLSTM (Ravi and Larochelle 2017) and ALFA (Baik et al. 2020). However, we find that they suffer from a new drawback, *i.e.*, gradient bias issue. Here, we take ALFA as an example to illustrate the issue. Formally, for each few-shot task, let $L(p(t))$ be its differentiable loss function with prototype $p(t)$, and $\nabla L(p(t))$ be its prototype gradient, during performing prototype optimization. Then, the ALFA can be expressed as the following M-steps iterations, given the initial (*i.e.*, mean-based) prototypes $p(0)$. That is,

$$p(t+1) = p(t) - \eta(\nabla L(p(t)) + \omega p(t)), \tag{5}$$

where t is the iteration step (*i.e.*, $t = 0, 1, ..., M-1$), η is a learning rate, and ω denotes a weight of ℓ_2 regularization term infered by the meta-optimizer $g_{\theta_g}()$. Its goal is to improve fast adaptation with few examples by learning task-specific ℓ_2 regularization term. Though ALFA is effective (see Table 5), we find that it computes the gradient $\nabla L(p(t))$ in an average manner over few labeled samples $(x_i, y_i) \in \mathcal{S}$:

$$\nabla L(p(t)) = \frac{1}{|\mathcal{S}|} \sum_{(x_i, y_i) \in \mathcal{S}} \nabla L_{(x_i, y_i)}(p(t)), \tag{6}$$

where $|\cdot|$ denotes the size of a set and $\nabla L_{(x_i,y_i)}(p(t))$ is the gradient of the sample $(x_i, y_i) \in \mathcal{S}$. Such estimation is inaccurate, because the number of available labeled (support) samples (*e.g.*, K=1 or 5) is far less than the expected amount. As a result, the optimization performance of existing methods is limited. This is exactly the gradient bias issue mentioned in the section of introduction. Note that the unlabeled samples $x \in \mathcal{Q}'$ are not used in these methods because their gradients cannot be computed without labels.

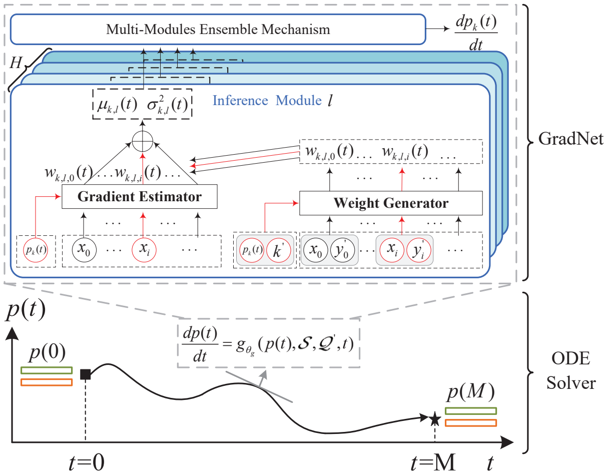

Multi-Modules Ensemble Mechanism
$\frac{dp_k(t)}{dt}$
H
Inference Module l
$\mu_{k,l}(t)$ $\sigma^2_{k,l}(t)$
$w_{k,l,0}(t) \ldots w_{k,l,i}(t) \ldots$
$w_{k,l,0}(t) \ldots w_{k,l,i}(t) \ldots$
Gradient Estimator
Weight Generator
$p_k(t)$
x_0
x_i
$p_k(t)$
k'
x_0
y_0'
x_i
y_i'
GradNet
$p(t)$
$p(0)$
$\frac{dp(t)}{dt} = g_{\theta_g}(p(t), \mathcal{S}, \mathcal{Q}', t)$
$p(M)$
ODE Solver
t=0
t=M
t

Figure 3: Illustration of MetaNODE, which characterizes the prototype optimization dynamics by using a Neural ODE.

MetaNODE. Recent studies (Bu, Xu, and Chen 2020) found that the iteration process of Gradient Descent Algorithm (GDA) can be viewed as an Euler discretization of an ordinary differential equation (ODE). That is,

$$\frac{\mathrm{d}p(t)}{\mathrm{d}t} = -\nabla L(p(t)), \tag{7}$$

where t is a continous variable (*i.e.*, time) and $\frac{\mathrm{d}p(t)}{\mathrm{d}t}$ denotes a continuous-time gradient flow of prototypes $p(t)$.

Inspired by this fact, to more finely rectify the prototypes, we propose to characterize the prototype dynamics by an ODE and consider the prototype rectification problem as the ODE initial value problem, where the initial and final status values correspond to the mean-based and optimal prototypes, respectively. To address the gradient bias issue appeared in Eq. 6, we view the prototype $p(t)$, support set $\mathcal{S}$, unlabeled sample set $\mathcal{Q}'$, and time t as inputs and then employ a neural network (*i.e.* the meta-learner $g_{\theta_g}()$) to directly estimate the continuous gradient flow $\frac{\mathrm{d}p(t)}{\mathrm{d}t}$. Then, the ODE turns into a Neural ODE, *i.e.*, $\frac{\mathrm{d}p(t)}{\mathrm{d}t} = g_{\theta_g}(p(t), \mathcal{S}, \mathcal{Q}', t)$.

Based on this notion, we design a novel Neural ODE-based Meta-Optimizer (called MetaNODE). Its advantage is that the prototype rectification dynamics can be captured in a continuous manner, thereby more finely diminishing the prototype bias. As shown in Figure 3, the MetaNODE consists of a Gradient Flow Inference Network (GradNet) and an ODE solver. The GradNet is regarded as the meta-learner $g_{\theta_g}()$, aiming to infer the continuous gradient flow $\frac{\mathrm{d}p(t)}{\mathrm{d}t}$ (see Section 3.4 for its details). Based on the GradNet $g_{\theta_g}()$ and initial prototypes $p(0)$, the optimal prototypes $p(M)$ can be obtained by evaluating the Neural ODE at the last time point $t = M$, *i.e.*, $p(M) = p(0) + \int_{t=0}^{M} g_{\theta_g}(p(t), \mathcal{S}, \mathcal{Q}', t)$, where the integral term is calculated by the ODE solvers. That is,

$$p(M) = ODESolver(g_{\theta_g}(), p(0), \mathcal{S}, \mathcal{Q}', t = M). \tag{8}$$

Following (Chen et al. 2018), we use Runge-Kutta method (Alexander 1990) as our ODE solver because it has relatively low integration error and computational complexity.

3.4 Gradient Flow Inference Network

In this section, we introduce how the GradNet $g_{\theta_g}()$ employed in MetaNODE is designed. Intuitively, different classes have distinct prototype dynamics. For many classes like animals and plants, the differences of their prototype dynamics may be quite large. To model the class diversities, as shown in Figure 3, instead of performing a single inference module, we design multiple inference modules with the same structure to estimate the prototype gradient $\frac{\mathrm{d}p(t)}{\mathrm{d}t}$. Here, the inference module consists of a gradient estimator and a weight aggregator. The former aims to predict the contributed gradient of each sample $x_i \in \mathcal{S} \cup \mathcal{Q}'$ for prototypes $p(t)$. The latter accounts for evaluating the importance of each sample and then combining their gradient estimations in a weighted mean manner. For clarity, we take inference module l and class k as an example to detail them.

Gradient Estimator. As we adopt the cosine-based classifier, the prototype is expected to approach the angle center of each class. To eliminate the impact of vector norm, we first transform the features $f_{\theta_f}(x_i)$ of each sample x_i to an appropriate scale by a scale layer $g_{\theta^l_{gs}}()$ with parameters θ^l_{gs}. Then, the gradient $d_{k,l,i}(t)$ is estimated by computing the difference vector between it and prototype $p_k(t)$. That is,

$$d_{k,l,i}(t) = g_{\theta^l_{gs}}(f_{\theta_f}(x_i) \| p_k(t)) \otimes f_{\theta_f}(x_i) - p_k(t), \tag{9}$$

where $\|$ is a concatenation operation of two vectors and $\otimes$ denotes an element-wise product operation.

Weight Generator. Intuitively, different samples make varying contributions to the gradient prediction of prototype $p_k(t)$. To this end, we design a weight generator to predict their weights. Specially, for each sample $x_i \in \mathcal{S} \cup \mathcal{Q}'$, we combine the prototype $p_k(t)$ and the sample x_i as a new feature. Then, the weight generating process involves a simple feed-forward mapping of the new features by an embedding layer $g_{\theta^l_{ge}}()$, followed by a relation layer $g_{\theta^l_{gr}}()$ with a multi-head based attention mechanism (Vaswani et al. 2017) and an output layer $g_{\theta^l_{go}}()$. Here, the relation layer aims to obtain a robust representation by exploring the pair-wise relationship between all samples and the output layer evaluates the contributed weight $w_{k,l,i}$. The above weight generating process can be summarized as Eq. 10. That is,

$$\begin{aligned} h_{k,l,i}(t) = g_{\theta^l_{ge}}(k' \| p_k(t) \| y'_i \| f_{\theta_f}(x_i) \| p_k(t) \otimes f_{\theta_f}(x_i)), \\ h'_{k,l,i}(t) = g_{\theta^l_{gr}}(\{h_{k,l,i}(t)\}_{i=0}^{|\mathcal{S} \cup \mathcal{Q}|-1}), \\ w_{k,l,i}(t) = g_{\theta^l_{go}}(h'_{k,l,i}(t)), \end{aligned} \tag{10}$$

where θ^l_{ge}, θ^l_{gr}, and θ^l_{go} denote model parameters; k' and y'_i denotes the one-hot label of prototype $p_k(t)$ and sample x_i, respectively. We replace the one-hot label of each unlabeled sample $x_i \in \mathcal{Q}'$ in a N-dim vector with value of $1/N$.

Finally, the gradient $\mu_{k,l}(t)$ and its estimation variance $\sigma^2_{k,l}(t)$ can be obtained in a weighted mean manner:

$$\begin{aligned} \mu_{k,l}(t) = \sum\nolimits_i w_{k,l,i}(t) \otimes d_{k,l,i}(t), \\ \sigma^2_{k,l}(t) = \sum\nolimits_i w_{k,l,i}(t) \otimes (d_{k,l,i}(t) - \mu_{k,l}(t))^2. \end{aligned} \tag{11}$$

Multi-Modules Ensemble Mechanism. We have obtained multiple gradient estimations for prototype $p_k(t)$, *i.e.*, $\{\mu_{k,l}(t)\}_{l=0}^{H-1}$, where H denotes the number of inference modules. Intuitively, the variance $\{\sigma^2_{k,l}(t)\}_{l=0}^{H-1}$ reflects the inconsistency of gradients contributed by all samples, *i.e.*, the larger variance implies greater uncertainty. Hence, to obtain a more reliable prototype gradient $\frac{\mathrm{d}p_k(t)}{\mathrm{d}t}$, we regard the variances as weights to combine these gradients $\mu_{k,l}(t)$:

$$\frac{\mathrm{d}p_k(t)}{\mathrm{d}t} = \beta \left[\sum_{l=0}^{H-1} (\sigma^2_{k,l}(t))^{-1} \right]^{-1} \left[\sum_{l=0}^{H-1} (\sigma^2_{k,l}(t))^{-1} \mu_{k,l}(t) \right], \tag{12}$$

where we employ a term of exponential decay with time t, *i.e.*, $\beta = \beta_0 \xi^{\frac{t}{M}}$ (β_0 and ξ are hyperparameters, which are all set to 0.1 empirically), to improve the model stability.

3.5 Adaption to Inductive FSL Setting

The above MetaNODE-based framework focuses on the transductive FSL, which can also be easily adapted to inductive FSL. Specifically, its workflow is similar to the process described in Sections 3.2 and 3.3. The only difference is the unlabeled sample set $\mathcal{Q}'$ is removed in the GradNet of Sections 3.4, and only using support set $\mathcal{S}$ to estimate gradients.

4 Performance Evaluation

4.1 Datasets and Settings

MiniImagenet. The dataset consists of 100 classes, where each class contains 600 images. Following the standard split in (Chen et al. 2020), we split the data set into 64, 16, and 20 classes for training, validation, and test, respectively.

TieredImagenet. The dataset is a larger dataset with 608 classes. Each class contains 1200 images. Following (Chen et al. 2020), the dataset is split into 20, 6, and 8 high-level semantic classes for training, validation, and test, respectively.

CUB-200-2011. The dataset is a fine-grained bird recognition dataset with 200 classes. It contains about 11,788 images. Following the standard split in (Chen et al. 2019), we split the data set into 100 classes, 50 classes, and 50 classes for training, validation, and test, respectively.

4.2 Implementation Details

Network Details. We use ResNet12 (Chen et al. 2020) as the feature extractor. In GradNet, we use four inference modules to estimate the gradient flow. For each module, we use a two-layer MLP with 512-dimensional hidden layer for the scale layer, a single-layer perceptron with 512-dimensional outputs for the the embedding layer, a multi-head attention module with 8 heads and each head contains 16 units for the relation layer, and a single-layers perceptron and a softmax

Setting	Method	Type	Backbone	miniImagenet		tieredImagenet	
				5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
Trans ductive	DPGN (Yang et al. 2020)	Graph	ResNet12	67.77 ± 0.32%	84.60 ± 0.43%	72.45 ± 0.51%	87.24 ± 0.39%
	EPNet (Rodríguez et al. 2020)	Graph	ResNet12	66.50 ± 0.89%	81.06 ± 0.60%	76.53 ± 0.87%	87.32 ± 0.64%
	MCGN (Tang et al. 2021)	Graph	Conv4	67.32 ± 0.43%	83.03 ± 0.54%	71.21 ± 0.85%	85.98 ± 0.98%
	ICI (Wang et al. 2020)	Pre-training	ResNet12	65.77%	78.94%	80.56%	87.93%
	LaplacianShot (Ziko et al. 2020)	Pre-training	ResNet18	72.11 ± 0.19%	82.31 ± 0.14%	78.98 ± 0.21%	86.39 ± 0.16%
	SIB (Hu et al. 2020)	Pre-training	WRN-28-10	70.0 ± 0.6%	79.2 ± 0.4%	-	-
	BD-CSPN (Liu, Song, and Qin 2020)	Pre-training	WRN-28-10	70.31 ± 0.93%	81.89 ± 0.60%	78.74 ± 0.95%	86.92 ± 0.63%
	SRestoreNet (Xue and Wang 2020)	Pre-training	ResNet18	61.14 ± 0.22%	-	-	-
	ProtoComNet (Zhang et al. 2021a)	Pre-training	ResNet12	73.13 ± 0.85%	82.06 ± 0.54%	81.04 ± 0.89%	87.42 ± 0.57%
	MetaNODE (ours)	Pre-training	ResNet12	***77.92*** ± ***0.99***%	***85.13*** ± ***0.62***%	***83.46*** ± ***0.92***%	***88.46*** ± ***0.57***%
In ductive	CTM (Li et al. 2019b)	Metric	ResNet18	62.05 ± 0.55%	78.63 ± 0.06%	64.78 ± 0.11%	81.05 ± 0.52%
	ALFA (Baik et al. 2020)	Optimization	ResNet12	59.74 ± 0.49%	77.96 ± 0.41%	64.62 ± 0.49%	82.48 ± 0.38%
	sparse-MAML (von Oswald et al. 2021)	Optimization	Conv4	56.39 ± 0.38%	73.01 ± 0.24%	–	–
	Neg-Cosine (Liu et al. 2020)	Pre-training	ResNet12	63.85 ± 0.81%	81.57 ± 0.56%	-	-
	P-Transfer (Shen et al. 2021)	Pre-training	ResNet12	64.21 ± 0.77%	80.38 ± 0.59%	-	-
	Meta-UAFS (Zhang et al. 2021b)	Pre-training	ResNet12	64.22 ± 0.67%	79.99 ± 0.49%	69.13 ± 0.84%	84.33 ± 0.59%
	RestoreNet (Xue and Wang 2020)	Pre-training	ResNet12	59.28 ± 0.20%	-	-	-
	ClassifierBaseline (Chen et al. 2020)	Pre-training	ResNet12	61.22 ± 0.84%	78.72 ± 0.60%	69.71 ± 0.88%	83.87 ± 0.64%
	MetaNODE (ours)	Pre-training	ResNet12	66.07 ± 0.79%	81.93 ± 0.55%	72.72 ± 0.90%	86.45 ± 0.62%

Table 1: Experiment results on miniImageNet and tieredImageNet. The best results are highlighted in bold.

Setting	Method	CUB-200-2011	
		5-way 1-shot	5-way 5-shot
Trans ductive	EPNet	82.85 ± 0.81%	91.32 ± 0.41%
	ECKPN	77.43 ± 0.54%	92.21 ± 0.41%
	ICI	87.87%	92.38%
	LaplacianShot	80.96%	88.68%
	BD-CSPN	87.45%	91.74%
	RestoreNet	76.85 ± 0.95%	-
	MetaNODE (ours)	***90.94*** ± ***0.62***%	***93.18*** ± ***0.38***%
In ductive	RestoreNet	74.32 ± 0.91%	-
	Neg-Cosine	72.66 ± 0.85%	89.40 ± 0.43%
	P-Transfer	73.88 ± 0.87%	87.81 ± 0.48%
	ClassifierBaseline	74.96 ± 0.86%	88.89 ± 0.43%
	MetaNODE (ours)	80.82 ± 0.75%	91.77 ± 0.49%

Table 2: Experiment results on CUB-200-2011.

layer for the output layer. ELU (Clevert, Unterthiner, and Hochreiter 2016) is used as the activation function.

Training details. Following (Chen et al. 2020), we use an SGD optimizer to train the feature extractor for 100 epochs. In the meta-training phase, we train the Neural ODE-based meta-optimizer 50 epochs using Adam with a learning rate of 0.0001 and a weight decay of 0.0005, where the learning rate is decayed by 0.1 at epochs 15, 30, and 40, respectively.

Evaluation. Following (Chen et al. 2020), we evaluate our method on 600 randomly sampled episodes (5-way 1/5-shot tasks) from the novel classes and report the mean accuracy together with the 95% confidence interval. In each episode, we randomly sample 15 images per class as the query set.

4.3 Experimental Results

We evaluate our MetaNODE-based prototype optimization framework and various state-of-the-art methods on general and fine-grained few-shot image recognition tasks. Among these methods, ClassifierBaseline, RestoreNet, BD-CSPN, SIB, and SRestoreNet are our strong competitors since they also focus on learning reliable prototypes for FSL.

General Few-Shot Image Recognition. Table 1 shows the results of various evaluated methods on miniImagenet and tieredImagenet. In transductive FSL, we found that (i) MetaNODE outperforms our competitors (*e.g.*, BD-CSPN, SIB, SRestoreNet) by around $2\% \sim 7\%$. This is because we rectify prototypes in a continuous manner; (ii) MetaNODE achieves superior performance over other state-of-the-art methods. Different from these methods, our method focuses on polishing prototypes instead of label propagation or loss evaluation. These results verify the superiority of MetaNODE; (iii) the performance improvement is more conspicuous on 1-shot than 5-shot tasks, which is reasonable because the prototype bias is more remarkable on 1-shot tasks.

In inductive FSL, our MetaNODE method outperforms ClassifierBaseline by a large margin, around $3\% \sim 5\%$, on both datasets. This means that our method introducing a Neural ODE to polish prototype is effective. MetaNODE outperforms our competitors (*i.e.*, RestoreNet) by around 7%, which validates the superiority of our manner to rectify prototypes. Besides, MetaNODE achieves competitive performance over other state-of-the-art methods. Here, (i) different from the metric and optimization methods, our method employs a pre-trained feature extractor and focuses on polishing prototype; (ii) compared with other pre-training methods, our method focuses on obtaining reliable prototypes instead of fine-tuning feature extractors for FSL.

Fine-Grained Few-Shot Image Recognition. The results on CUB-200-2011 are shown in Table 2. Similar to Table 1, we observe that MetaNODE significantly outperforms the state-of-the-art methods, achieving $2\% \sim 6\%$ higher accuracy scores. This further verifies the effectiveness of MetaNODE in the fine-grained FSL task, which exhibits smaller class differences than the general FSL task.

4.4 Statistical Analysis

Does MetaNODE obtain more accurate prototypes? In Table 3, we report the cosine similarity between initial (optimal) prototypes, *i.e.*, $p(0)$ ($p(M)$) and real prototypes on 5-way 1-shot tasks of miniImagenet. The real prototypes are obtained by averaging the features of all samples $x_i \in \mathcal{S} \cup \mathcal{Q}$ by following (Liu, Song, and Qin 2020). The results show that MetaNODE obtains more accurate prototypes, which is because MetaNODE regards it as an optimization problem,

Methods	Initial Prototypes	Optimal Prototypes
BD-CSPN	0.64	0.75
SRestoreNet	0.64	0.85
ProtoComNet	0.64	0.91
MetaNODE	0.64	0.93

Table 3: Experiments of prototype bias on miniImagenet.

Methods	Averaged Gradient	Infered Gradient
SIB	0.0441	0.0761
MetaNODE	0.0441	0.1701

Table 4: Experiments of gradient bias on miniImagenet.

and solves it in a continuous dynamics-based manner.

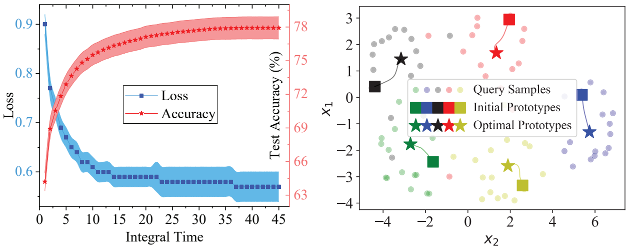
Does MetaNODE alleviate gradient bias? In Table 4, we randomly select 1000 episodes from miniImageNet, and then calculate the cosine similarity between averaged (inferred) and real gradient. Here, the averaged and infered gradients are obtained by Eq. 6 and GradNet, respectively. The real gradients are obtained in Eq. 6 by using all samples. We select SIB as the baseline, which improves GDA by inferring the loss value of unlabeled samples. It can be observed that MetaNODE obtains more accurate gradients than SIB. This is because we model and train a meta-learner from abundant FSL tasks to directly estimate the continuous gradient flows.

Can our meta-optimizer converge? In Figure 4(a), we randomly select 1000 episodes (5-way 1-shot) from miniImageNet, and then report their test accuracy and loss from integral time $t = 1$ to 45. It can be observed that our meta-optimizer can converge to a stable result when $t = 40$. Hence, $M = 40$ is a default setting in our approach.

How our meta-optimizer works? We visualize the prototype dynamics of a 5-way 1-shot task of miniImagenet, in Figure 4(b). Note that (1) since there is only one support sample in each class, its feature is used as the initial prototypes; (2) the curve from the square to the star denotes the trajectory of prototype dynamics and its tangent vector represents the gradient predicted by our GradNet. We find that the initial prototypes marked by squares flow to optimal prototypes marked by stars along prototype dynamics, much closer to the class centers. This indicates that our meta-optimizer effectively learns the prototype dynamics.

4.5 Ablation Study

Is our meta-optimizer effective? In Table 5, we analyzed the effectiveness of our meta-optimizer. Specifically, in inductive FSL setting, (i) we remove our meta-optimizer as a baseline; (ii) we add the MetaLSTM meta-optimizer (Ravi and Larochelle 2017) on (i) to optimize prototypes; (iii) we replace MetaLSTM by the ALFA (Baik et al. 2020) on (ii); (iv) different from (iii), we replace MetaLSTM by our MetaNODE; and (v) we further explore unlabeled samples on (iv). From the results of (i) $\sim$ (iv), we observe that: 1) the performance of (ii) and (iii) exceeds (i) around $1\% \sim 3\%$, which means that it is helpful to leverage the existing meta-optimizer to polish prototypes and validates the effectiveness of the proposed framework. 2) the performance of (iv) exceeds (ii) $\sim$ (iii) around $2\% \sim 3\%$, which shows the superiority of our MetaNODE meta-optimizer. This is because MetaNODE regards the gradient flow as meta-knowledge, instead of hyperparameters like weight decay. Finally, comparing the results of (iv) with (v), we find that using unlabeled samples can significantly enhance MetaNODE.



(a) Performance

(b) Feature space

Figure 4: Visualization of Neural ODE on miniImagenet.

	Method	5-way 1-shot	5-way 5-shot
(i)	Baseline	$61.22 \pm 0.84\%$	$78.72 \pm 0.60\%$
(ii)	+ MetaLSTM	$63.85 \pm 0.81\%$	$79.49 \pm 0.65\%$
(iii)	+ ALFA	$64.37 \pm 0.79\%$	$80.75 \pm 0.57\%$
(iv)	+ Neural ODE	$66.07 \pm 0.79\%$	$81.93 \pm 0.55\%$
(v)	+ Neural ODE + QS	$77.92 \pm 0.99\%$	$85.13 \pm 0.62\%$

Table 5: Analysis of meta-optimizer on miniImagenet. QS denotes unlabeled (query) samples from query set Q.

	Method	5-way 1-shot	5-way 5-shot
(i)	MetaNODE	$77.92 \pm 0.99\%$	$85.13 \pm 0.62\%$
(ii)	w/o ensemble	$75.34 \pm 1.10\%$	$84.00 \pm 0.53\%$
(ii)	w/o attention	$75.10 \pm 0.98\%$	$83.90 \pm 0.56\%$
(ii)	w/o exponential decay	$76.02 \pm 1.17\%$	$84.16 \pm 0.64\%$

Table 6: Analysis of GradNet components on miniImagenet.

Are these key components (*i.e.*, ensemble, attention, and exponential decay) effective in GradNet? In Table 6, we evaluate the effect of these three components. Specifically, (i) we evaluate MetaNODE on miniImagenet; (ii) we remove ensemble mechanism on (i), *i.e.*, $H = 1$; (iii) we remove attention mechanism on (i); (iii) we remove exponential decay mechanism on (i). We find that the performance decreases by around $1\% \sim 3\%$ when removing these three components, respectively. This result implies that employing these three key components is beneficial for our MetaNODE.

5 Conclusion

In this paper, we propose a novel meta-learning based prototype optimization framework to obtain more accurate prototypes for few-shot learning. In particular, we design a novel Neural ODE-based meta-optimizer to capture the continuous prototype dynamics. Experiments on three datasets show that our model significantly obtains superior performance over state-of-the-art methods. We also conduct extensive statistical experiments and ablation studies, which further verify the effectiveness of our method.

Acknowledgments

This work was supported by the Shenzhen Science and Technology Program under Grant No. JCYJ20180507183823045, JCYJ20200109113014456, and JCYJ20210324120208022, and NSFC under Grant No. 61972111.

References

Alexander, R. 1990. Solving Ordinary Differential Equations I: Nonstiff Problems (E. Hairer, S. P. Norsett, and G. Wanner). *SIAM Rev.*, 32(3): 485–486.

Baik, S.; Choi, M.; Choi, J.; et al. 2020. Meta-Learning with Adaptive Hyperparameters. In *NeurIPS*.

Boudiaf, M.; Ziko, I. M.; Rony, J.; Dolz, J.; Piantanida, P.; and Ayed, I. B. 2020. Transductive Information Maximization For Few-Shot Learning. In *NeurIPS*.

Bu, Z.; Xu, S.; and Chen, K. 2020. A Dynamical View on Optimization Algorithms of Overparameterized Neural Networks. *CoRR*, abs/2010.13165.

Chen, C.; Yang, X.; Xu, C.; Huang, X.; and Ma, Z. 2021. ECKPN: Explicit Class Knowledge Propagation Network for Transductive Few-Shot Learning. In *CVPR*, 6596–6605.

Chen, E. Z.; Chen, T.; and Sun, S. 2020. MRI Image Reconstruction via Learning Optimization Using Neural ODEs. In *MICCAI 2020*, volume 12262, 83–93.

Chen, T. Q.; Rubanova, Y.; Bettencourt, J.; and Duvenaud, D. 2018. Neural Ordinary Differential Equations. In *NeurIPS*, 6572–6583.

Chen, W.; Liu, Y.; Kira, Z.; Wang, Y. F.; and Huang, J. 2019. A Closer Look at Few-shot Classification. In *ICLR*.

Chen, Y.; Wang, X.; Liu, Z.; Xu, H.; and Darrell, T. 2020. A New Meta-Baseline for Few-Shot Learning. In *ICML*.

Clevert, D.; Unterthiner, T.; and Hochreiter, S. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). In *ICLR*.

Ding, Z.; Han, Z.; Ma, Y.; and Tresp, V. 2021. Temporal Knowledge Graph Forecasting with Neural ODE. *CoRR*, abs/2101.05151.

Flennerhag, S.; Rusu, A. A.; Pascanu, R.; Visin, F.; Yin, H.; and Hadsell, R. 2020. Meta-Learning with Warped Gradient Descent. In *ICLR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *CVPR*, 770–778.

Hu, S. X.; Moreno, P. G.; Xiao, Y.; Shen, X.; et al. 2020. Empirical Bayes Transductive Meta-Learning with Synthetic Gradients. In *ICLR*.

Li, A.; Luo, T.; Xiang, T.; Huang, W.; and Wang, L. 2019a. Few-Shot Learning With Global Class Representations. In *ICCV*, 9714–9723.

Li, H.; Eigen, D.; Dodge, S.; Zeiler, M.; and Wang, X. 2019b. Finding task-relevant features for few-shot learning by category traversal. In *CVPR*, 1–10.

Li, P.; Fu, Y.; and Gong, S. 2021. Regularising Knowledge Transfer by Meta Functional Learning. In *IJCAI*.

Liu, B.; Cao, Y.; Lin, Y.; Li, Q.; Zhang, Z.; Long, M.; and Hu, H. 2020. Negative Margin Matters: Understanding Margin in Few-Shot Classification. In *ECCV*, 438–455.

Liu, J.; Song, L.; and Qin, Y. 2020. Prototype Rectification for Few-Shot Learning. In *ECCV*.

Nguyen, V. N.; Løkse, S.; Wickstrøm, K.; Kampffmeyer, M.; Roverso, D.; and Jenssen, R. 2020. SEN: A Novel Feature Normalization Dissimilarity Measure for Prototypical Few-Shot Learning Networks. In *ECCV*, 118–134.

Prabhu, V.; Kannan, A.; Ravuri, M.; et al. 2019. Few-Shot Learning for Dermatological Disease Diagnosis. In *MLHC*, volume 106, 532–552.

Raghu, A.; Raghu, M.; Bengio, S.; and Vinyals, O. 2020. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. In *ICLR*.

Ravi, S.; and Larochelle, H. 2017. Optimization as a Model for Few-Shot Learning. In *ICLR*.

Rodríguez, P.; Laradji, I. H.; Drouin, A.; and Lacoste, A. 2020. Embedding Propagation: Smoother Manifold for Few-Shot Classification. In *ECCV*, 121–138.

Rubanova, Y.; Chen, T. Q.; and Duvenaud, D. 2019. Latent Ordinary Differential Equations for Irregularly-Sampled Time Series. In *NeurIPS*, 5321–5331.

Shen, J.; Li, Z.; Yu, L.; Xia, G.; and Yang, W. 2020. Implicit Euler ODE Networks for Single-Image Dehazing. In *CVPRW 2020*, 877–886. IEEE.

Shen, Z.; Liu, Z.; Qin, J.; Savvides, M.; and Cheng, K. 2021. Partial Is Better Than All: Revisiting Fine-tuning Strategy for Few-shot Learning. In *AAAI*, 9594–9602.

Snell, J.; Swersky, K.; and Zemel, R. S. 2017. Prototypical Networks for Few-shot Learning. In *NeurIPS*, 4077–4087.

Tang, S.; Chen, D.; Bai, L.; Liu, K.; Ge, Y.; and Ouyang, W. 2021. Mutual CRF-GNN for Few-Shot Learning. In *CVPR*, 2329–2339.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In *NeurIPS*, 5998–6008.

Vinyals, O.; Blundell, C.; Lillicrap, T.; Kavukcuoglu, K.; and Wierstra, D. 2016. Matching Networks for One Shot Learning. In *NeurIPS*, 3630–3638.

von Oswald, J.; Zhao, D.; Kobayashi, S.; Schu, S.; Caccia, M.; et al. 2021. Learning where to learn: Gradient sparsity in meta and continual learning. In *NeurIPS*.

Wang, Y.; Xu, C.; Liu, C.; Zhang, L.; and Fu, Y. 2020. Instance Credibility Inference for Few-Shot Learning. In *CVPR*, 12833–12842.

Xue, W.; and Wang, W. 2020. One-Shot Image Classification by Learning to Restore Prototypes. In *AAAI*, 6558–6565.

Yang, L.; Li, L.; Zhang, Z.; Zhou, X.; Zhou, E.; and Liu, Y. 2020. DPGN: Distribution Propagation Graph Network for Few-Shot Learning. In *CVPR*, 13387–13396.

Zhang, B.; Li, X.; Ye, Y.; Huang, Z.; and Zhang, L. 2021a. Prototype Completion With Primitive Knowledge for Few-Shot Learning. In *CVPR*, 3754–3762.

Zhang, Z.; Lan, C.; Zeng, W.; Chen, Z.; and Chang, S. 2021b. Uncertainty-Aware Few-Shot Image Classification. In *IJCAI*.

Zheng, Y.; Liu, S.; Li, Z.; and Wu, S. 2021. Cold-start Sequential Recommendation via Meta Learner. In *AAAI*.

Ziko, I. M.; Dolz, J.; Granger, E.; and Ayed, I. B. 2020. Laplacian Regularized Few-Shot Learning. In *ICML*, volume 119, 11660–11670.