Meta Navigator: Search for a Good Adaptation Policy for Few-shot Learning

Chi Zhang[1] Henghui Ding[1,2] Guosheng Lin[1*] Ruibo Li[1] Changhu Wang[2] Chunhua Shen[3]

[1] Nanyang Technological University [2] ByteDance [3] Monash University

chi007@e.ntu.edu.sg gslin@ntu.edu.sg

Abstract

Few-shot learning aims to adapt knowledge learned from previous tasks to novel tasks with only a limited amount of labeled data. Research literature on few-shot learning exhibits great diversity, while different algorithms often excel at different few-shot learning scenarios. It is therefore tricky to decide which learning strategies to use under different task conditions. Inspired by the recent success in Automated Machine Learning literature (AutoML), in this paper, we present Meta Navigator, a framework that attempts to solve the aforementioned limitation in few-shot learning by seeking a higher-level strategy and proffer to automate the selection from various few-shot learning designs. The goal of our work is to search for good parameter adaptation policies that are applied to different stages in the network for few-shot classification. We present a search space that covers many popular few-shot learning algorithms in the literature, and develop a differentiable searching and decoding algorithm based on meta-learning that supports gradient-based optimization. We demonstrate the effectiveness of our searching-based method on multiple benchmark datasets. Extensive experiments show that our approach significantly outperforms baselines and demonstrates performance advantages over many state-of-the-art methods.

1. Introduction

Convolutional Neural Networks (CNNs) have become indispensable in a variety of computer vision tasks [27, 39–41, 55, 58, 59]. A crucial reason is that the knowledge learned by CNNs can be transferred across different vision tasks in the form of hierarchical feature representations. Nevertheless, a sufficiently large amount of annotated data is still necessary to achieve good generalization accuracy due to CNNs' data-hungry properties, which inevitably hinders the application of CNNs in real-world scenarios.

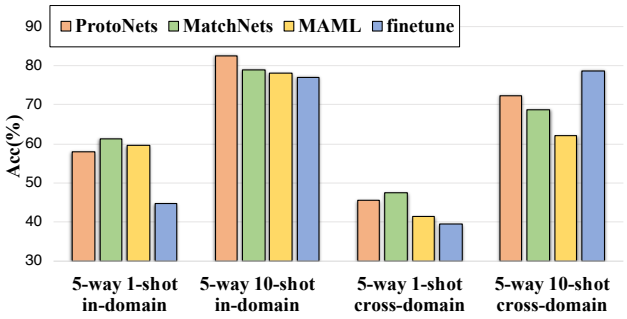

Figure 1 – Comparison of some popular few-shot learning algorithms under various few-shot learning task settings. All models are based on the same network architecture, with weights pre-trained on *mini*ImageNet dataset. Cross-domain experiments are evaluated on the CUB dataset. Existing few-shot learning algorithms are highly task-specific and are outperformed by a simple fine-tune baseline when the domain difference is large and more support data are available.

Few-shot learning is proposed as a promising direction to alleviate the need for exhaustively labeled data by exploring an extreme case where only a few labeled data is available to undertake a novel task based on prior knowledge learned on previous tasks. A typical application scenario is few-shot image classification [11,37,44]. Literature on few-shot learning exhibits great diversity, while different algorithms often excel at different few-shot learning scenarios. Fig. 1 compares some popular few-shot learning algorithms on different few-shot learning tasks. Here we consider three test cases, including 1) the extreme low-shot case, *i.e.*, 1-shot; 2) a medium-shot case, where the size of support set is relatively larger than the common benchmarks, *e.g.*, 10-shot; 3) a cross-domain case where the training and testing tasks are sampled from different domains. As is shown, existing few-shot learning algorithms are highly task-specific and no single algorithm can show superiority over others across all tasks. In particular, when the domain difference is large, the compared few-shot learning algorithms can not sufficiently utilize the increasing number of support data to accommodate the domain difference, while the simple fine-

*Corresponding author: G. Lin (e-mail: gslin@ntu.edu.sg)

tuning strategy can beat all other few-shot learning methods, although it is significantly outperformed by others in the 1-shot case due to over-fitting. Therefore, it is almost impossible to find one single optimal few-shot learner that works well for all tasks. This makes many few-shot learning algorithms difficult to be applied as a general tool to solve the data scarcity issue in machine learning, even though they can perform very well on some specific benchmarks.

In recent years, there is a surging interest in automating the design of machine learning algorithms (AutoML), instead of relying too much on heuristic manual designs. In particular, the idea of AutoML has been successfully applied to Neural Architecture Search (NAS) [26, 45, 52], where the model learns to identify high-performance architectures by exploring a large candidate architecture space. With the same intuition, in this work, we attempt to solve the aforementioned limitation in few-shot learning by seeking a higher-level strategy and take initiatives to automate the selection of few-shot learning designs. The goal of our work is to search for good parameter adaptation policies that are applied to different stages in the network for few-shot learning. The search space in our network include two parts: the policy to adapt convolutional layers in different stages of the backbone and the policy to obtain class prototypes in the classifier, which together construct a hierarchical policy search space. At each network stage, various candidates policies are available for adapting the parameters, and the whole search space covers many popular meta-learning algorithms in the literature, such as Prototypical Networks [37], Matching Networks [44], baseline++ [2], MAML [11], *etc*.

In order to search from a pool of discrete adaptation policies, we develop a differentiable searching algorithm based on meta-learning that allows efficient gradient-based optimization. Inspired by the differentiable designs in NAS literature [26], our searching system is built upon a continuous relaxation of the discrete meta-learning policy, where each candidate policy is associated with a learnable policy selection indicator. However, as each adaptation policy is an optimization process rather than a differentiable operation, directly porting the formulation in DARTS [26] would not suffice. To tackle this issue, we further associate each policy with a group of policy-specific model parameters. Then, the decision of choosing the optimal policy becomes jointly learning the policy selection indicators as well as the policy parameters. The searching is conducted via a bi-level optimization paradigm based on meta-learning. Specifically, the optimization goal in the inner loop is to adapt the parameters in each candidate policy using the support data in sampled tasks, while the optimization objective in the outer loop alternates between learning the policy-specific parameters and learning the policy selection indicators. During searching, we progressively decode the supernet from front to back stages, with fine-tune in between, based on a perturbation-based policy selection scheme [45] that measures each policy's influence on the supernet. At the end of the training, each network stage is associated with an adaptation policy with parameters learned.

To validate the effectiveness of our design, we conduct various experiments on multiple benchmark datasets, including the challenging cross-domain experiments. Our experiment results show that our searched-based model not only outperforms the random search baseline but also demonstrates significant performance advantages over many previous methods covered in our search space. Our main contributions are summarized as follows:

- Our work is the first attempt to search meta-learning designs for few-shot learning tasks.
- We propose a hierarchical policy search space that covers many previous meta-learning algorithms.
- We develop a differentiable meta-learning policy searching algorithm that can conduct policy searching efficiently by meta-learning.
- Experiments on five popular datasets show that our method significantly outperforms the baselines and achieves new state-of-the-art results on many benchmarks.

Next we review some related work.

2. Related work

Few-shot classification. Various few-shot learning paradigms were proposed in the literature [7, 15, 23, 25, 29, 36, 46–48, 50, 51, 53]. Metric-based approaches and optimization-based approaches are two dominating lines of efforts. Metric-based methods [9, 14, 22, 32, 37, 42, 44, 49, 53, 54] aim to learn a deep metric to inference data relations for predictions. Usually, once the model is learned, the parameters are fixed when it is deployed to inference in new tasks. Therefore, metric based approaches have advantages in inference speed and often perform well in the extremely low-shot case. Optimization-based methods [1, 11–13, 15, 18–20, 33, 38, 60] aim to design effective learning paradigms for few-shot learning. For example, MAML [11] aims to learn a good model initialization that can enable fast adaptations of network parameters in novel tasks. Chen *et al*. [2] find that by simply pre-training the model weights with all training classes, many early works, such as ProtoNet [37], MatchNet [44], and MAML [11] can be rejuvenated and reach state-of-the-art performance. In our experiments, we demonstrate the advantages of our model over these baselines that adopt a fixed policy across all network stages. Besides image recognition, few-shot learning is also investigated in segmentation tasks [5, 28, 56, 57].

Network Architecture Search (NAS). Our work draws connections with the NAS literature [26,45], which aims for identifying effective building elements in the CNN structures. The most related work to ours is DARTS [26], which applies continuous relaxation that transforms the discrete choice of architectures into architecture weights. In DARTS, different candidate operations together constitute a supernet that is optimized in a bi-level scheme, where a training set is used to learn the operation-specific parameters and a validation set is used to optimize the architecture weights. After training, the optimal operation is chosen by selecting the one with the largest architecture weights. Despite its simplicity, many recent works question the effectiveness of DARTS [3,4,45,52]. For example, a simple random search baseline can outperform the architecture searched by DARTS [26], and the searching favors parameter-free operations, *e.g.*, skip connections [52]. Recently, many improved designs are proposed to solve the issues in DARTS [3, 4, 6, 21, 45, 52]. For example, Wang *et al*. [45] find that the architecture weights may not be a good indicator for decoding the supernet, and propose a perturbation-based architecture selection approach. Specifically, after the training of supernet, the best operation is chosen based on how much each operation perturbs the supernet performance when it is removed. In our work, we also adopt such model discretization scheme to obtain the final policies.

Recently, NAS for few-shot learning is explored in [8,10, 24,30], *which aims to identify high-performing task-specific network structures in the few-shot learning tasks*. The difference of our work with them is that the searching goal in our framework is to find good parameter adaptation policies rather than architectures, and different candidate policy share the same architecture.

3. Policy Search Space

This section describes our meta-learning policy search space. A standard CNN structure mainly include two components: the feature backbone that encodes the input image into representations and a classifier that classifies the data embedding. Therefore, we divide the search space in our framework into two parts: policies at the representation encoding (RE) stages (Sec. 3.1) and the policies at the prototype learning (PL) stage (Sec. 3.2).

3.1. Search Space for Representation Encoding

In a standard convolutional neural network, denoted by $\mathcal{G}(;)$, the backbone is used to encode input images into high-dimensional representations for classification by a sequence of convolutional layers. Due to the hierarchical design in CNNs, we can divide the sequential convolutional layers into several groups $\{g^1, g^2, ...\}$, *e.g.* 4 layer blocks in ResNet. Our goal is to search an adaptation policy for each layer group in the context of few-shot classification tasks. Specifically, given a layer group $g^l(;\theta)$, where l is the index of the group and θ is the parameters in it, an adaptation policy is defined as the approach to adapt the parameters θ based on the support set $\mathcal{S}$ in a task $\mathcal{T}$, such that θ becomes $\hat{\theta}$. For each layer group in the backbone of a CNN, three kinds of candidate policies are involved:

I. Fixed Parameters (RE-FIX). In RE-FIX, the parameters learned by training tasks are kept fixed without any adaptation in a novel task, *i.e.*, $\hat{\theta} \leftarrow \theta$. Such design is widely seen in the metric-based approaches, where the learned data encoder on training tasks is directly reused to encode data in novel tasks. Similarly, baseline++ [2] also freezes the pre-trained backbone in novel tasks and only fine-tunes the classifier.

II. Fine-Tuning the Weights (RE-FT). In this case, the learned parameters of RE-FIX in the group can be fine-tuned with the support set $\mathcal{S}$ in novel tasks by stochastic gradient descent, *i.e.*,

$$\hat{\theta} = \theta - \beta \nabla_\theta \mathcal{L}_\mathcal{T}(\theta), \tag{1}$$

where β is the learning rate and $\mathcal{L}$ is the loss function. By varying the hyper-parameters during fine-tuning, *e.g.*, the learning rate or the number of iterations, we can obtain a collection of sub-candidates, such as a strong RE-FT policy with a large learning rate or a weak RE-FT policy that slightly fine-tunes the parameters with a small learning rate.

III. Fast Adaptation (RE-FA). Different from RE-FT that fine-tunes from the weights of RE-FIX, RE-FA fine-tunes the model from the model initialization that is meta-learned, as done in MAML [11]. In other words, the fine-tuning behavior in RE-FA influences the learning of model initialization, while RE-FT does not. Similar to RE-FT, we can further obtain sub-candidates of RE-FA based on the adaptation hyper-parameters.

We search for an adaptation policy for each layer group $\mathcal{G}_l$ in the backbone. Therefore, dividing the backbone into M stages results in a search space with $[1 + S_{\text{RE-FT}} + S_{\text{RE-FA}}]^M$ candidates for representation encoding, where S denotes the number of sub-candidates in the policies.

3.2. Search Space for Prototype Learning

After encoding the input image into a vector representation $\mathbf{v} \in \mathbb{R}^C$ with the backbone, the classifier linearly projects the representation into the scores of each class s_i with a weight matrix $\mathbf{W} \in \mathbb{R}^{N \times C}$, where N is the number of classes and C is the feature dimension. From a prototype view of such operations, the weight matrix $\mathbf{W}$ essentially stores a collection of prototype vectors $[\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_N]^T$ for all classes, where $\mathbf{w}_i \in \mathbb{R}^C$, and the class score s_i of a specific class i is computed by the inner product between the data representation $\mathbf{v}$ and the class prototype $\mathbf{w}_i$, *i.e.*,

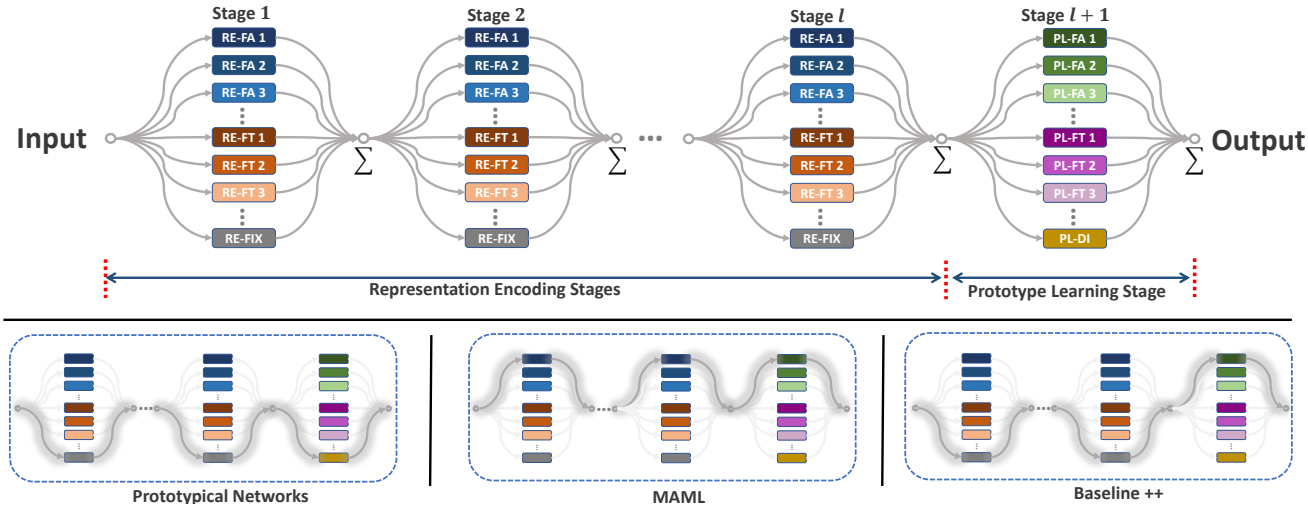

Figure 2 – Our search-based framework for few-shot classification. We aim to search for a good adaptation policy at each network stage. Our search space mainly include two parts: the policies for representation encoding and the policies for prototype learning. Please refer to Sec. 3 for the explanation of different policies. Based on the continuous relaxation of different candidate policies at each stage, we construct a differentiable supernet that can be optimized end-to-end (top). Our search space covers many well-known few-shot learning algorithms, such as Prototypical Networks [37], MAML [11] and Baseline++ [2] (bottom).

$s_i = \mathbf{w}_i \mathbf{v}^T$. The inner product operation can also be replaced by other similarity metrics, such as negative L2 distance [37], cosine similarity [44] and deep Earth Mover's Distance [53]. Based on such interpretation, we summarize the following policies as searching candidates to obtain the class prototypes for computing class scores:

I. Data Initialization (PL-DI). Metric-based meta-learning algorithms, such as Prototypical Networks [37], can be seen as directly parameterizing the class prototypes with the data embeddings of support images. Typically, in the 1-shot case where each class has one support image, the encoded support data are directly used as the prototypes, while in the k-shot case, the averaged data embedding in each class is set as the class prototype [37]. Therefore, our first candidate is directly parameterizing the classifier with data embeddings without any adaptation.

II. Fine-Tuning from Data Embeddings (PL-FT). In this case, the data initialized prototypes in PL-DI are only served as the starting point for fine-tuning.

III. Fine-Tuning by Fast Adaptation (PL-FA). Similar to the RE-FA policy for representation learning, the PL-FA fine-tunes the classifier from a meta-learned initialization of prototypes.

Likewise, we can obtain the strong or weak sub-candidates for PL-FT and PL-FA. We use cosine similarity to compute the class cores for all policies. After involving the search space for prototypes, the whole search space $\mathcal{O}$ include $[1 + S_{\mathrm{RE-FT}} + S_{\mathrm{RE-FA}}]^M \times [1 + S_{\mathrm{PL-FT}} + S_{\mathrm{PL-FA}}]$ candidates, and the searching goal is to find a high-performing combination of policy candidates in the whole search space.

Our search space covers many well-known meta-learning algorithms, as shown in Fig. 2. For example, if all parameters are kept frozen for representation encoding and the classifier is directly parameterized by data embeddings (RE-FIX + PL-DI), the model becomes metric-based methods, such as Prototypical Networks [37] and Matching Networks [44]; if all stages choose fast adaptation (RE-FA + PL-FA), the model becomes MAML; if the parameters in the backbone are kept frozen and the classifier is fine-tuned by fast adaptation (RE-FIX + PL-FA), it is close to baseline++ [2].

4. Method

In this part, we present our method for searching the policies in the aforementioned search space. We begin by introducing a continuous relaxation of different candidate policies that involve all policies into a differentiable supernet (Sec. 4.1). Then, we discuss how to optimize the model for searching (Sec. 4.2), and how to progressively decode the supernet (Sec. 4.3).

4.1. Continuous Relaxation of Policies

In order to search over a pool of discrete choice of meta-learning policies $\mathcal{O}^l$ on a specific stage l, we reuse the idea of continuous relaxation of individual choices, proposed in DARTS [26]. At the beginning, each candidate policy o_i^l in the search space $\mathcal{O}^l$ is associated with a normalized policy selection weight α_i^l as well as a copy of parameters θ_i^l at

stage l, where,

$$\sum_{i \in |\mathcal{O}^l|} \alpha_i^l = 1, \alpha_i^l > 0, \tag{2}$$

implemented by softmax. To obtain a continuous relaxation of discrete policies, we take the weighted sum of the outputs generated by different policies. Specifically, given the output of the previous stage O^{l-1}, the output of the current stage is computed by:

$$O^l = \sum_{i \in |\mathcal{O}^l|} \alpha_i^l g^l(O^{l-1}, \hat{\theta}_i^l). \tag{3}$$

Therefore, all candidate policies in different stages of the network together construct a differentiable supernet [26]. In particular, $\hat{\theta}_i^l$ is the parameters after adaptation in individual policies, while the policy selection weight α specifies the contribution of different candidate policies, which amounts to the architectures weights in DARTS [26].

4.2. Optimization

After constructing the supernet of the model, the next goal is to learn the parameters θ in the individual policies, as well as the policy selection weight α. Recall that DARTS [26] alternatively optimizes the architecture weights and the operation parameters with two disjoint sets, which approximates a bi-level optimization process. With similar formulation, the optimization in our framework also alternates between two meta-learning objectiveness:

1. Update the parameter θ in different candidate policies with $\nabla_\theta . \mathcal{L}_{\mathcal{T}_t}(\hat{\theta}, \alpha)$
2. Update the policy weights α with $\nabla_\alpha \mathcal{L}_{\mathcal{T}_v}(\hat{\theta}, \alpha)$.

Alg. 1 presents the pipeline of the optimization process in the form of pseudo-codes. In comparison with the bi-level optimization scheme for NAS, there are two main difference: 1) Different from DARTS where the weights are learned on a specific task, the optimization in our framework is based on meta-learning, which samples tasks from two disjoint task domains $p(\mathcal{T}_\text{A})$ and $p(\mathcal{T}_\text{B})$, based on the training set and the validation set, respectively. 2) Each of the optimization objective above has a nested optimization problem. Specifically, in the inner loop of both optimization objectives, the goal is to adapt the policy weights θ to obtain task specific policy weights $\hat{\theta}$, while the outer loop alternatively optimizes the parameters in different policies θ and the policy selection weight α, with $\nabla_\theta . \mathcal{L}_{\mathcal{T}_\text{A}}(\hat{\theta}, \alpha)$ and $\nabla_\alpha \mathcal{L}_{\mathcal{T}_\text{B}}(\hat{\theta}, \alpha)$, respectively. Moreover, the way to obtain the gradient with respect to the policies weights, *i.e.*, $\nabla_\theta . \mathcal{L}_{\mathcal{T}_\text{A}}(\hat{\theta}, \alpha)$, varies in different policies. For example, since no adaptation operation is applied in RE-FIX, a closed-form expression of the gradients with respect to the parameters in RE-FIX can be obtained, while RE-FA and PL-FA must differentiate through the nested optimization trajectory that requires the computation of gradients of gradients [11]. Noted that as all sub-candidates of RE-FT and PL-FT fine-tune the parameters from the pre-trained weights (RE-FIX) or data embeddings, the parameters θ for fine-tuning in these policies are generated online in each task, and hence there is no learnable parameter of these policies in the outer loop of Step 1.

4.3. Decoding Discrete Policies

During the supernet training, we progressively decode the supernet such that only one candidate policy is left in each stage. Following [45], we adopt a perturbation-based decoding strategy that the strength of each poly is defined as how much it contributes to the performance of the supernet, which is implemented by masking out the path of each policy and observing the performance drop. The policy that leads to the largest accuracy drop on the validation set after being masked out is considered as the optimal policy in this stage. We decode each stage one-by-one from front to back layers. After the decoding of each stage, we fine-tune the supernet with Alg. 1 for a few episodes to recover the accuracy drop caused by discretization.

It is important to notice that the direct discretization will change the behaviors of fine-tuning based policies. Concretely, before decoding of the supernet, an SGD step in a policy i at layer l is

$$\begin{aligned} \hat{\theta}_i^l &= \theta_i^l - \beta \frac{\partial \mathcal{L}_\text{S}}{\partial \theta_i^l} \\ &= \theta_i^l - \beta \frac{\partial \mathcal{L}_\text{S}}{\partial O^l} \frac{\partial O^l}{\partial g^l} \frac{\partial g^l}{\partial \theta_i^l} \\ &= \theta_i^l - \beta \frac{\partial \mathcal{L}_\text{S}}{\partial O^l} \alpha_i^l \frac{\partial g^l}{\partial \theta_i^l} \end{aligned} \tag{4}$$

where g^l is the output of the policy and O_l is the output at the stage. The issue here is that, since the continuous relaxation in Eq. (3) is discretized after decoding, the partial derivatives $\frac{\partial O^l}{\partial g^l}$ changes from α_i^l to 1. Hence, given the same gradients $\frac{\partial \mathcal{L}_\text{S}}{\partial O^l}$ back-propagated to this stage, the discretization scales the adaptation strength of the policy by $1/\alpha_i^l$. To resolve such discrepancy, we fuse the policy selection weight α into the learning rate after decoding, *i.e.*, $\beta \leftarrow \beta \alpha_i^l$ and the adaptation step after decoding becomes

$$\begin{aligned} \hat{\theta}_i^l &= \theta_i^l - (\beta \alpha_i^l) \frac{\partial \mathcal{L}_\text{S}}{\partial O^l} \frac{\partial O^l}{\partial g^l} \frac{\partial g^l}{\partial \theta_i^l} \\ &= \theta_i^l - (\beta \alpha_i^l) \frac{\partial \mathcal{L}_\text{S}}{\partial O^l} \frac{\partial g^l}{\partial \theta_i^l} \end{aligned} \tag{5}$$

As a result, the actual adaptation steps in the policy before and after decoding are identical, and the initial assigned

Algorithm 1: Optimization of the supernet to search for good adaptation policies.

Input: $p(\mathcal{T}_A)$, $p(\mathcal{T}_B)$: two disjoint task distributions
Input: $\mathcal{G}(;\Theta,\alpha)$: a supernet with pre-trained initialized weights in each policy
while *not done* **do**
  # Step 1: Optimize parameters Θ in the policies
  Sample a task $\mathcal{T}_A = \{\mathcal{S},\mathcal{Q}\}$ from $p(\mathcal{T}_A)$;
  $\hat{\Theta} \leftarrow \mathcal{G}(\mathcal{S};\Theta,\alpha)$, Adapt weights in each policy with the support set $\mathcal{S}$ (# inner loop);
  Make predictions for the query set $\mathcal{Q}$ with $\mathcal{G}(\mathcal{S};\hat{\Theta},\alpha)$;
  Calculate loss $\nabla_\Theta.\mathcal{L}_{\mathcal{T}_A}(\hat{\Theta},\alpha)$ and optimize Θ;
  # Step 2: Optimize policy selection weights α
  Sample a task $\mathcal{T}_B = \{\mathcal{S},\mathcal{Q}\}$ from $p(\mathcal{T}_B)$;
  $\hat{\Theta} \leftarrow \mathcal{G}(\mathcal{S};\Theta,\alpha)$, Adapt weights in each policy with the support set $\mathcal{S}$ (# inner loop) ;
  Make predictions for the query set $\mathcal{Q}$ with $\mathcal{G}(\mathcal{S};\hat{\Theta},\alpha)$;
  Calculate loss $\nabla_\alpha.\mathcal{L}_{\mathcal{T}_B}(\hat{\Theta},\alpha)$ and optimize α;
end
Decoding the supernet

learning rate only serves as the upper bound during searching.

5. Experiments

5.1. Dataset Statistics

To validate the effectiveness of our framework, we conduct experiments on five benchmark datasets, including miniImageNet, tieredImageNet, Fewshot-CIFAR100 (FC100), CIFAR-FewShot (CIFAR-FS) and Caltech-UCSD Birds-200-2011 (CUB).

***mini*ImageNet.** *mini*ImageNet is the most popular few-shot classification dataset, proposed in [44]. The dataset is built upon the ImageNet dataset [34], and contains 100 classes with 600 images in each class. The numbers of classes for training, validation and testing, are 64 ,16 and 20, respectively.

***tiered*ImageNet.** *tiered*ImageNet is also a few-shot classification dataset build upon ImageNet, which includes 608 classes. The splits of training(20), validation(6) and testing(8) classes are set according to the super-classes to enlarge domain gaps between training and testing time.

Fewshot-CIFAR100. FC100 is a few-shot classification dataset build on CIFAR100 [17]. Following the split division in [31], the training ,validation and testing sets include 60, 20, and 20 classes respectively.

CIFAR-FewShot. CIFAR-FS is also a few-shot classification dataset built upon CIFAR100, proposed in [7]. It contains 64, 16, 20 classes for training, validation and testing, respectively.

Caltech-UCSD Birds-200-2011. CUB is a fine-grained bird classification dataset. Following [2], we divide 200 classes into 100, 50 and 50 for training, validation and testing, respectively.

Model	1-**shot**	5-**shot**	10-**shot**
ProtoNets [37]	57.89	78.75	82.66
MatchNets [44]	61.47	75.41	78.87
MAML [11]	59.58	75.80	78.20
Baseline++ [2]	61.50	79.47	83.63
finetune	44.81	68.77	76.94
Random search	64.10	77.97	80.94
Ours	**65.91**	**82.66**	**85.46**

Table 1 – Comparison with baseline models for 5-way few-shot classification on *mini*ImageNet dataset. The searched model for 5-shot tasks is re-used to undertake 10-shot tasks in this experiment. Our searched policy outperforms baselines consistently on various tasks.

5.2. Implementation Details

We employ ResNet-12 as our network backbone to conduct all experiments. As there are four layer blocks in the ResNet backbone, we can naturally divide the network parameters into five stages, including four stages in the backbone and 1 stage in the classifier. We set two sub-candidates for RE-FT, RE-FA, PL-FT, PL-FA policies, including a strong version that fine-tunes parameters with the learning rate of 0.1 and a weak version with the learning rate of 0.01, and all the sub-candidates adapt the parameters for 10 epochs. Therefore, our search space covers $(1+2+2)\times(1+2+2)^4 = 3125$ candidates totally. Before the optimization of the supernet, we pre-train a backbone with all data in the training set, and use the pre-trained weights to initialize all policies at the representation learning stage. We train the supernet with Alg. 1 for 1000 episodes and then start decoding from front to back layers. After decoding each stage, we fine-tune the supernet for 100 episodes to recover the accuracy drop caused by discretization. After all stages are decoded, we further fine-tune the networks for 2000 episodes. Random scale, random crop and random horizontal flip are employed for data augmentation at training time. All models in our experiment are evaluated with 600 testing episodes, and we report the average accuracy.

5.3. Results and Analysis

Visualization of searched models. We first present the visualization of the searched models under different few-shot learning tasks. We plot the final selection of policies at each stage as well as the policy selection weights of different stages before the initial decoding in Fig. 3. Noted that the policy selection weights do not necessarily indicate the

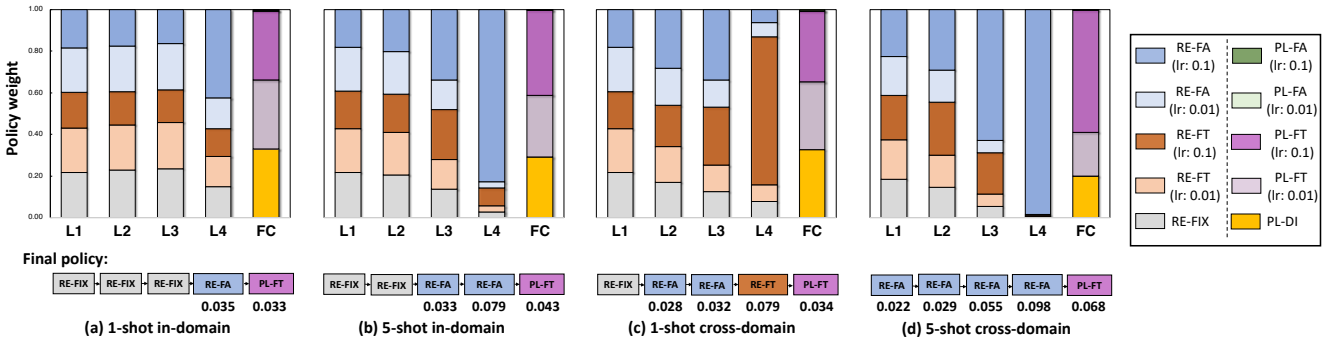

Figure 3 – Visualization of the searched policies under different 5-way few-shot learning tasks. We plot the distributions of the policy selection weights before the initial decoding as well as the final policy at each network stage. Learning rates are noted below all fine-tune based policies. Please refer to Sec. 5.3 for our analysis.

Model	1-shot	5-shot	10-shot
ProtoNets [37]	45.52	66.80	72.29
MatchNets [44]	47.41	63.63	68.83
MAML [11]	41.29	58.10	62.18
Baseline++ [2]	47.79	70.01	76.13
finetune	39.49	67.88	78.60
Random Search	49.22	67.47	71.64
Ours	**53.80**	**72.43**	**81.05**

Table 2 – Cross-domain experiments on CUB dataset. Models trained with *mini*ImageNet dataset are evaluated on multiple 5-way task settings. Our model performs well across all task settings and demonstrates performance advantages over all baselines.

decoding selections, but we can observe the behavior of the searching model by comparing the selection distributions on different tasks or layers. We have the following findings based on the visualizations:

1. What is shared across all tasks is that the prototypes that are fine-tuned with a relatively large learning rate based on data initialization (PL-FT) are always favored, while the prototypes that are fine-tuned from meta-learned initialization (PL-FA) are completely ignored. This emphasizes the preference of the data initialization for prototypes learning in our searching model. Meanwhile, the last representation encoding layer, *i.e.*, layer 4, in all tasks also chooses adaptation with a relatively large learning rate (RE-FA or RE-FT).
2. We find that as the data-initialization-based policies dominate the prototype learning stage in the supernet, the gradients propagated to the front layers are weak in the one-shot case. As a result, the outputs of different policies at the front layers, *e.g.*, layer 1, are very similar, and the distribution is therefore close to uniform. Nevertheless, after the perturbation-based decoding, all models except the one for the k-shot cross-domain task, choose to fix the parameters in the first layer block.
3. By comparing the distributions and the final policies of different tasks, we can find that when the domain difference is large or the number of support data grows, more layers choose to be fine-tuned and the learning rate also increases.

Comparison with baselines. In order to demonstrate the advantages of our design, we compare our searched policies with the following baseline models that relate to our proposed search space:

I. Prototypical Networks and Matching Networks, the two representative metric-based methods in literature. Their difference is that Prototypical Networks [37] use negative L2 distance to compute class scores and the averaged embeddings as prototypes in the classifier, while Matching Networks [44] use cosine similarity to compute class scores, and use individual data embeddings as prototypes and then fuse the scores generated by different prototypes from the same class.

II. MAML [11]. Based on our search space, all representation encoding stages adopt RE-FA policies and the prototype learning stage adopts the PL-FA policy. We choose the learning rates for the backbone and the classifier from $\{0.01, 0.1\}$ and report the optimal result.

III. Baseline++ [2]. Baseline++ freezes the pre-trained backbone and fine-tunes a cosine classifier. We fine-tune the classifier for 100 iterations for all experiments. The learning rates are chosen from $\{0.01, 0.1\}$, and we report the optimal result.

IV. Fine-tune. We simply fine-tune the pre-trained model for 100 iterations with different learning rates for the backbone and the classifier. The learning rates are chosen from $\{0.01, 0.1\}$, and we report the optimal result.

V. Random Search. We randomly sample the policies at each stage in the proposed search space to construct a model, and report the averaged performance of 10 sampled

Method	Backbone	*mini*ImageNet		*tiered*ImageNet		FC100		CIFAR-FS	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
TADAM [31]	ResNet-12	58.50 ± 0.30	76.70 ± 0.30	–	–	40.1 ± 0.4	56.1 ± 0.4	–	–
MTL [38]	ResNet-12	61.2 ± 1.8	75.5 ± 0.8	65.6 ± 1.8	80.8 ± 0.8	45.1 ± 1.9	57.6 ± 1.0	–	–
FEAT [49]	ResNet-25‡	62.96 ± 0.20	78.49 ± 0.15	–	–	–	–	–	–
LEO [35]	WRN-28-10‡	61.76 ± 0.08	77.59 ± 0.12	66.33 ± 0.05	81.44 ± 0.09	–	–	–	–
Dhillon *et al*. [7]	WRN-28-10‡	57.73 ± 0.62	78.17 ± 0.49	66.58 ± 0.70	85.55 ± 0.48	38.25 ± 0.52	57.19 ± 0.57	68.72 ± 0.67	86.11 ± 0.47
MetaOptNet [18]	ResNet-12	62.64 ± 0.82	78.63 ± 0.46	65.99 ± 0.72	81.56 ± 0.53	41.1 ± 0.6	55.5 ± 0.6	72.0 ± 0.7	84.2 ± 0.5
CAN [14]	ResNet-12	63.85 ± 0.48	79.44 ± 0.34	69.89 ± 0.51	84.23 ± 0.37	–	–	–	–
CTM [22]	ResNet-18‡	64.12 ± 0.82	80.51 ± 0.13	68.41 ± 0.39	84.28 ± 1.73	–	–	–	–
DSN-MR [36]	ResNet-12	64.60 ± 0.72	79.51 ± 0.50	67.39 ± 0.82	82.85 ± 0.56	–	–	75.6 ± 0.9	86.2 ± 0.6
Tian *et al*. [43]	ResNet-12	64.82 ± 0.60	82.14 ± 0.43	71.52 ± 0.69	86.03 ± 0.49	44.6 ± 0.7	60.9 ± 0.6	73.9 ± 0.8	86.9 ± 0.5
Kim *et al*. [16]	ResNet-12	65.08 ± 0.86	82.70 ± 0.54	–	–	42.31 ± 0.75	57.56 ± 0.78	73.51 ± 0.92	85.49 ± 0.68
DeepEMD [53]	ResNet-12	65.91 ± 0.82	82.41 ± 0.56	71.16 ± 0.87	**86.03** ± 0.58	**46.47** ± 0.78	**63.22** ± 0.71	–	–
Ours	ResNet-12	**65.91** ± 0.83	**82.66** ± 0.55	**73.52** ± 0.88	**85.34** ± 0.62	45.60 ± 0.81	59.93 ± 0.76	74.01 ± 0.96	**86.03** ± 0.62
Ours + MC	ResNet-12	**67.14** ± 0.80	**83.82** ± 0.51	**74.58** ± 0.88	**86.73** ± 0.61	**46.40** ± 0.81	61.33 ± 0.71	**74.63** ± 0.91	**86.45** ± 0.59

‡ Different backbone with ours.

Table 3 – Comparison with the state-of-the-art 5-way few-shot classification results on *mini*ImageNet, *tiered*ImageNet, FC100, and CIFAR-FS datasets. **MC** denotes multi-crop testing. Our approach outperforms state-of-the-art performance on multiple datasets.

models. We omit the policy PL-FA ($\mathrm{lr} = 0.01$), as we find it always generates poor results, no matter what policies are adopted at the prototype encoding stages.

All the baseline models are pre-trained, and the results are presented in Table 1. As we can see, given the similar architectures shared by all compared models, our searched adaptation policy achieve optimal results on all task settings. In particular, our model outperforms the random search baseline consistently, which validates the effectiveness of our searching algorithm.

Cross domain experiments. We next perform cross-domain experiments to further evaluate the effectiveness of our design, where the training data and testing data are sampled from different datasets. We evaluate the model trained with *mini*Imagenet data on CUB dataset. Since CUB is a fine-grained classification dataset, there exists a domain gap between the training and testing tasks, which can better evaluate how well a few-shot learning algorithm adapt a model for novel tasks. In the cross-domain experiments, we use the validation set from the target domain as the set B in Alg. 1 to conduct searching. As we can see from the results in Table 2, our model outperform baseline models consistently on different datasets with remarkable performance advantages. In particular, on the 10-shot tasks, we outperform the random search baseline by 9.41% and baseline++ by 4.92%.

5.4. Comparison with State-of-the-Art Methods

To better position our method among the few-shot learning literature, we compare the results of our network with the state-of-the-art performance. We report the Top 1 accuracy with the 95% confidence intervals on four benchmark datasets: *mini*Imagenet, *tiered*Imagenet, CIFAR-FS, and FC100. As we find the predictions of the few-shot learners are often sensitive to the scales and shifts in input data, we also report the result that employs multi-crop testing, denoted by **MC**. Specifically, we simply re-use the data augmentation operations at training time to scale and crop the query images for 10 times and average the predicted logits of them as the final predictions. The result is shown in Table 3. *Though our method aims to solve a more general few-shot learning problems, we still obtain remarkable performance on the popular benchmarks.* In particular, on popular *tiered*Imagenet dataset, we obtain 74.58% 1-shot accuracy, which outperforms previous state-of-the-art by 3.42%.

6. Conclusion

In this paper, we have presented a search-based framework for few-shot learning classification that aims to find a good parameter adaptation policy at each network stage. With a continuous relaxation of the discrete meta-learning policy, our searching model is differentiable and end-to-end trainable. We further develop a decoding algorithm that progressively select the optimal choice at each stage. Our designed search space covers many popular few-shot learning designs in literature. Extensive experiments validate the effectiveness of our design, and we obtain new state-of-the-art performance on multiple benchmarks.

Acknowledgement

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2018-003), and the MOE Tier-1 research grants: RG28/18 (S), RG22/19 (S) and RG95/20.

References

[1] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *Proc. Int. Conf. Learn. Representations*, 2019.

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *Proc. Int. Conf. Learn. Representations*, 2019.

[3] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization. In *Proc. Int. Conf. Mach. Learn.*, 2020.

[4] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. DrNAS: Dirichlet neural architecture search. In *Proc. Int. Conf. Learn. Representations*, 2021.

[5] Xiaoyu Chen, Chi Zhang, Guosheng Lin, and Jing Han. Compositional prototype network with multi-view comparision for few-shot point cloud semantic segmentation. *arXiv preprint arXiv:2012.14255*, 2020.

[6] Xiangxiang Chu, Tianbao Zhou, Bo Zhang, and Jixiang Li. Fair darts: Eliminating unfair advantages in differentiable architecture search. In *Proc. Eur. Conf. Comp. Vis.*, pages 465–480. Springer, 2020.

[7] Guneet Singh Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *Proc. Int. Conf. Learn. Representations*, 2020.

[8] Sivan Doveh, Eli Schwartz, Chao Xue, Rogerio Feris, Alex Bronstein, Raja Giryes, and Leonid Karlinsky. Metadapt: Meta-learned task-adaptive architecture for few-shot classification. *arXiv preprint arXiv:1912.00412*, 2019.

[9] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Diversity with cooperation: Ensemble methods for few-shot classification. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 3723–3731, 2019.

[10] Thomas Elsken, Benedikt Staffler, Jan Hendrik Metzen, and Frank Hutter. Meta-learning of neural architectures for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 12365–12375, 2020.

[11] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. Int. Conf. Mach. Learn.*, pages 1126–1135, 2017.

[12] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 9537–9548, 2018.

[13] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas L. Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *Proc. Int. Conf. Learn. Representations*, 2018.

[14] Ruibing Hou, Hong Chang, MA Bingpeng, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 4005–4016, 2019.

[15] Shell Xu Hu, Pablo G Moreno, Xi Shen1 Yang Xiao, Neil D Lawrence, Guillaume Obozinski, Andreas Damianou, and France Champs-sur Marne. Empirical bayes meta-learning with synthetic gradients. In *Proc. Int. Conf. Learn. Representations*, 2020.

[16] Jaekyeom Kim, Hyoungseok Kim, and Gunhee Kim. Model-agnostic boundary-adversarial sampling for test-time generalization in few-shot learning. *Proc. Eur. Conf. Comp. Vis.*, pages 599–617, 2020.

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

[18] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 10657–10665, 2019.

[19] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *Proc. Int. Conf. Mach. Learn.*, pages 2933–2942, 2018.

[20] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *Proc. Int. Conf. Mach. Learn.*, pages 2933–2942, 2018.

[21] Guohao Li, Guocheng Qian, Itzel Delgadillo, Matthias Muller, Ali Thabet, and Bernard Ghanem. Sgas: Sequential greedy architecture search. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1620–1630, 2020.

[22] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1–10, 2019.

[23] Kai Li, Yulun Zhang, Kunpeng Li, and Yun Fu. Adversarial feature hallucination networks for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 13470–13479, 2020.

[24] Dongze Lian, Yin Zheng, Yintao Xu, Yanxiong Lu, Leyu Lin, Peilin Zhao, Junzhou Huang, and Shenghua Gao. Towards fast adaptation of neural architectures with meta learning. In *Proc. Int. Conf. Learn. Representations*, 2020.

[25] Bin Liu, Yue Cao, Yutong Lin, Qi Li, Zheng Zhang, Mingsheng Long, and Han Hu. Negative margin matters: Understanding margin in few-shot classification. In *Proc. Eur. Conf. Comp. Vis.*, pages 438–455. Springer, 2020.

[26] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proc. Int. Conf. Learn. Representations*, 2019.

[27] Weide Liu, Chi Zhang, Guosheng Lin, Tzu-Yi HUNG, and Chunyan Miao. Weakly supervised segmentation with maximum bipartite graph matching. In *Proc. ACM Int. Conf. Multimedia*, pages 2085–2094, 2020.

[28] Weide Liu, Chi Zhang, Guosheng Lin, and Fayao Liu. Crnet: Cross-reference networks for few-shot segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4165–4173, 2020.

[29] Yaoyao Liu, Bernt Schiele, and Qianru Sun. An ensemble of epoch-wise empirical bayes for few-shot learning. In *Proc. Eur. Conf. Comp. Vis.*, pages 404–421. Springer, 2020.

[30] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Neural architecture transfer. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.

[31] Boris N. Oreshkin, Pau Rodríguez, and Alexandre Lacoste. TADAM: task dependent adaptive metric for improved few-shot learning. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 719–729, 2018.

[32] Hugo Prol, Vincent Dumoulin, and Luis Herranz. Cross-modulation networks for few-shot learning. *Proc. 2nd Workshop on Meta-Learning, at Advances in Neural Inf. Process. Syst.*, 2018.

[33] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *Proc. Int. Conf. Learn. Representations*, 2017.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vision*, 115(3):211–252, 2015.

[35] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *Proc. Int. Conf. Learn. Representations*, 2019.

[36] Christian Simon, Piotr Koniusz, Richard Nock, and Mehrtash Harandi. Adaptive subspaces for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4136–4145, 2020.

[37] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 4077–4087, 2017.

[38] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 403–412, 2019.

[39] Xin Sun, Henghui Ding, Chi Zhang, Guosheng Lin, and Keck-Voon Ling. M2iosr: Maximal mutual information open set recognition. *arXiv preprint arXiv:2108.02373*, 2021.

[40] Xin Sun, Zhenning Yang, Chi Zhang, Keck-Voon Ling, and Guohao Peng. Conditional gaussian distribution learning for open set recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 13480–13489, 2020.

[41] Xin Sun, Chi Zhang, Guosheng Lin, and Keck-Voon Ling. Open set recognition with conditional probabilistic generative models. *arXiv preprint arXiv:2008.05129*, 2020.

[42] 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 *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1199–1208, 2018.

[43] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *Proc. Eur. Conf. Comp. Vis.*, 2020.

[44] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 3630–3638, 2016.

[45] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Cho-Jui Hsieh. Rethinking architecture selection in differentiable NAS. In *Proc. Int. Conf. Learn. Representations*, 2021.

[46] Yikai Wang, Chengming Xu, Chen Liu, Li Zhang, and Yanwei Fu. Instance credibility inference for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 12836–12845, 2020.

[47] Yikai Wang, Li Zhang, Yuan Yao, and Yanwei Fu. How to trust unlabeled data? instance credibility inference for few-shot learning. *arXiv preprint arXiv:2007.08461*, 2020.

[48] Ling Yang, Liangliang Li, Zilun Zhang, Xinyu Zhou, Erjin Zhou, and Yu Liu. Dpgn: Distribution propagation graph network for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 13390–13399, 2020.

[49] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Learning embedding adaptation for few-shot learning. *arXiv*, 1812.03664, 2018.

[50] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 8808–8817, 2020.

[51] Zhongjie Yu, Lin Chen, Zhongwei Cheng, and Jiebo Luo. Transmatch: A transfer-learning scheme for semi-supervised few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 12856–12864, 2020.

[52] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *Proc. Int. Conf. Learn. Representations*, 2020.

[53] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019.

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

[55] Chi Zhang, Guankai Li, Guosheng Lin, Qingyao Wu, and Rui Yao. Cyclesegnet: Object co-segmentation with cycle refinement and region correspondence. *IEEE Trans. Image Process.*, 2021.

[56] Chi Zhang, Guosheng Lin, Fayao Liu, Jiushuang Guo, Qingyao Wu, and Rui Yao. Pyramid graph networks with connection attentions for region-based one-shot semantic segmentation. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 9587–9595, 2019.

[57] Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, and Chunhua Shen. Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5217–5226, 2019.

[58] Chi Zhang, Nan Song, Guosheng Lin, Yun Zheng, Pan Pan, and Yinghui Xu. Few-shot incremental learning with continually evolved classifiers. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 12455–12464, June 2021.

[59] Chi Zhang, Rui Yao, and Jinpeng Cai. Efficient eye typing with 9-direction gaze estimation. *Multimedia Tools and Applications*, 77(15):19679–19696, 2018.

[60] Ruixiang Zhang, Tong Che, Zoubin Grahahramani, Yoshua Bengio, and Yangqiu Song. Metagan: An adversarial approach to few-shot learning. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 2371–2380, 2018.