Few-shot visual understanding with deep neural networks

Zhang, Chi

2021

Zhang, C. (2021). Few-shot visual understanding with deep neural networks. Doctoral thesis, Nanyang Technological University, Singapore. <https://hdl.handle.net/10356/154696>

<https://hdl.handle.net/10356/154696>

<https://doi.org/10.32657/10356/154696>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 19:50:18 SGT

Few-Shot Visual Understanding with Deep Neural Networks

Zhang Chi

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

02/09/2021

.....

Date

TU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU

.....

Zhang Chi

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

02/09/2021

.....

Date

NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU

.....

Asst. Prof. Guosheng Lin

Authorship Attribution Statement

This thesis contains material from 4 papers published in the following peer-reviewed conferences in which I am listed as the first author for all 4 papers.

Chapter 1 is published as [Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, Chunhua Shen](#). “CANet: Class-Agnostic Segmentation Networks with Iterative Refinement and Attentive Few-Shot Learning.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (**CVPR**). Long Beach, United States. 2019.

The contributions of the co-authors are as follows:

- Prof. Guosheng Lin pointed out the overall research direction and supervised me during the research.
- I designed and implemented the algorithm, conducted the experiments, and prepared the manuscript drafts.
- Dr. Fayao Liu, Prof. Rui Yao and Prof. Chunhua Shen joined the early discussion and provided valuable comments on the manuscript.

Chapter 2 is published as [Chi Zhang, Guosheng Lin, Fayao Liu, Jiushuang Guo, Qingyao Wu, Rui Yao](#). “Pyramid Graph Networks with Connection Attentions for Region-Based One-Shot Semantic Segmentation.” Proceedings of the IEEE International Conference on Computer Vision (**ICCV**). Seoul, Korea. 2019.

The contributions of the co-authors are as follows:

- Prof. Guosheng Lin pointed out the overall research direction and supervised me during the research.
- I designed and implemented the algorithm, conducted the experiments, and prepared the manuscript drafts.
- Dr. Fayao Liu, Prof. Rui Yao, Prof. Qingyao Wu and Ms. Jiushuang Guo, joined the early discussion and provided valuable comments on the manuscript.

Chapter 3 is published as [Chi Zhang, Yujun Cai, Guosheng Lin, Chunhua Shen](#). “DeepEMD: Few-Shot Image Classification with Differentiable Earth Mover’s Distance and Structured Classifiers.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (**CVPR, Oral Presentation**). Virtual Conference. 2020.

The contributions of the co-authors are as follows:

- Prof. Guosheng Lin pointed out the overall research direction and supervised me during the research.
- I designed and implemented the algorithm, conducted the experiments, and prepared the manuscript drafts.
- Prof. Chunhua Shen and Dr. Yujun Cai, joined the early discussion and provided valuable comments on the manuscript.

Chapter 4 is published as [Chi Zhang, Henghui Ding, Guosheng Lin, Ruibo Li, Changhu Wang, Chunhua Shen](#). “Meta Navigator: Search for a Good Adaptation Policy for Few-shot Learning.” *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Virtual Conference. 2021.

The contributions of the co-authors are as follows:

- Prof. Guosheng Lin pointed out the overall research direction and supervised me during the research.
- I designed and implemented the algorithm, conducted the experiments, and prepared the manuscript drafts.
- Prof. Chunhua Shen, Dr. Henghui Ding, Dr. Changhu Wang, and Mr. Ruibo Li joined the early discussion and provided valuable comments on the manuscript.

02/09/2021

.....
Date

TU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
NTU NTU NTU NTU NTU NTU NTU NTU
TU NTU NTU NTU NTU NTU NTU NTU

.....
Zhang Chi

Acknowledgements

First, I want to express my sincere gratitude to my supervisor, Prof. Guosheng Lin. I highly appreciate the valuable opportunity he has provided to me to study at NTU, which always encourages me to keep fighting during the days at NTU. His generous guidance on both academics and lives and his trust in me make my Ph.D. study happier and easier. I would like to thank Prof. Chunhua Shen, Prof. Rui Yao, Prof. Tianyi Zhou, and Prof. Chunyan Miao for the fruitful discussions with them on life choices and academics. I would also like to thank my teacher, Mr. Taifeng Liu, who inspires me in spirit, integrity, and persistence.

I would like to thank my dear fellows at MICL lab, Xu Yang, Xiaofeng Yang, Sheng Yang, Ruibo Li, Kaihua Tang, Yiming Wang, Weide Liu and those not listed here. I really enjoy the atmosphere in the lab when all of them are around.

I would like to thank all the relatives and friends in my life. Their support and company give me courage, optimism, and intelligence.

Finally, I want to express my deepest gratitude to my parents for their infinite love for me. Their happiness and pride are always the greatest motivations for me to pursue excellence and overcome difficulties.

Abstract

Deep Neural Networks (DNNs) have become indispensable for a variety of computer vision tasks, such as image recognition, image segmentation, and object detection. The availability of large-scale labeled datasets and the powerful fitting capability of deep models are two crucial factors that contribute to its success. However, models trained under fully supervised learning have many limitations that hinder their applications in real-world scenarios. For example, a trained CNN model can only apply to a set of pre-defined classes and it needs a large amount of labeled data to fine-tune a model for tasks on new categories. Moreover, data labelling can be very expensive for some vision tasks, such as image segmentation. Few-shot learning is proposed as a promising direction to alleviate the need for exhaustively labeled data by exploring a learning case where only a few labeled data is available to undertake a novel task based on prior knowledge learned on previous tasks. Typical application scenarios are few-shot image classification and few-shot image segmentation.

In this thesis, we propose some novel algorithms to address the few-shot learning problems on image recognition and image segmentation tasks. In detail, for the task of few-shot image segmentation, we solve it as a message-passing problem that aims to extract useful information from the limited training data for the predictions on test images. We present two frameworks to achieve this goal. One employs the idea of prototype learning by representing the message in a category as a class-specific prototype, and the other method employs graph models to conduct message passing between local regions. For few-shot classification, we propose an algorithm that utilizes local representations in the images and structured distance to determine the image similarity for classification. To further solve the limitation in current few-shot learning methods that different few-shot learning algorithms often excel at different few-shot learning scenarios, we proffer to automate the selection from various few-shot learning designs and present a searching-based framework, which is inspired by the recent success in Automated Machine Learning literature (AutoML).

Our main contributions in this thesis are summarized as follows:

- We present **CANet**, a class-agnostic segmentation network that performs few-shot segmentation on new classes with only a few annotated images available. Our network consists of a two-branch dense comparison module which performs multi-level feature comparison between the support image and the query image, and an iterative optimization module which iteratively refines the predicted results. Furthermore, we introduce an attention mechanism to effectively fuse information from multiple support examples under the setting of k -shot learning.
- We propose to **model structured segmentation data with graphs** and apply attentive graph reasoning to propagate label information from support data to query data for few-shot segmentation tasks. The graph attention mechanism could establish the element-to-element correspondence across structured data by learning attention weights between connected graph nodes. To capture correspondence at different semantic levels, we further propose a pyramid-like structure that models different sizes of image regions as graph nodes and undertakes graph reasoning at different levels.
- We develop methods for few-shot image classification from a new perspective of optimal matching between image regions. We employ the Earth Mover’s Distance (EMD) as a metric to compute a structural distance between dense image representations to determine image relevance. The EMD generates the optimal matching flows between structural elements that have the minimum matching cost, which is used to calculate the image distance for classification. To generate the important weights of elements in the EMD formulation, we design a cross-reference mechanism, which can effectively alleviate the adverse impact caused by the cluttered background and large intra-class appearance variations. To implement k -shot classification, we propose to learn a structured fully connected layer that can directly classify dense image representations with the EMD. Based on the implicit function theorem, the EMD can be inserted as a layer into the network for end-to-end training.
- We present Meta Navigator, a framework that attempts to solve the limitation of current few-shot learning methods by seeking a higher-level strategy and proffer to automate the selection from various few-shot learning designs. The goal of this method 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.

Contents

Acknowledgements	ix
Abstract	xi
List of Figures	xix
List of Tables	xxiii
1 Introduction	1
1.1 Few-Shot Visual Understanding	1
1.2 Few-Shot Learning: from Recognition to Segmentation	2
1.3 Challenges and Contributions	2
1.4 Outline of the Thesis	4
2 Literature Review	7
2.1 Overview	7
2.2 Few-Shot Learning	7
2.3 Semantic Segmentation	8
2.4 Few-Shot Segmentation	9
2.5 Earth Mover’s Distance.	10
2.6 Parameterized optimization.	10
2.7 Network Architecture Search (NAS).	11
3 CANet: Class-Agnostic Segmentation Networks with Iterative Refinement and Attentive Few-Shot Learning	13
3.1 Introduction	13
3.2 Task Description	16
3.3 Method	17
3.3.1 Dense Comparison Module	17
3.3.2 Iterative Optimization Module	19
3.3.3 Attention Mechanism for k-shot Segmentation	21
3.3.4 Bounding Box Annotations	22
3.4 Experiments	23
3.4.1 PASCAL-5i	24

3.4.1.1	Comparison with the State-of-the-art Methods . . .	24
3.4.1.2	Experiments on Bounding Box Annotations	24
3.4.1.3	Ablation Study	25
3.4.2	COCO	28
3.5	Conclusion	29
4	Pyramid Graph Networks with Connection Attentions for Region-Based One-Shot Semantic Segmentation	31
4.1	Introduction	31
4.2	Method	34
4.2.1	Graph Attention Unit	34
4.2.2	Pyramid Graph Reasoning	37
4.3	Experiment	38
4.3.1	Implementation details.	39
4.3.2	Dataset and Evaluation Metric	39
4.3.3	Ablation Analysis	40
4.3.4	Qualitative Results	43
4.3.5	Comparison with the State-of-the-art Results	43
4.4	Conclusions	45
5	DeepEMD: Few-Shot Image Classification with Differentiable Earth Mover’s Distance and Structured Classifiers	47
5.1	Introduction	47
5.2	Preliminary	51
5.3	Our Method	51
5.3.1	Revisiting the Earth Mover’s Distance	52
5.3.2	EMD for Few-Shot Classification	53
5.3.3	End-to-End Training	55
5.3.4	Weight Generation	57
5.3.5	Structured Fully Connected Layer	60
5.4	Experiments	61
5.4.1	Implementation Details	62
5.4.2	Dataset Description	63
5.4.3	Analysis	64
5.4.4	Cross-Domain Experiments	72
5.4.5	Visualization of matching flows and weights	73
5.4.6	Time Complexity	73
5.4.7	Comparison with State-of-the-art Methods	74
5.5	Conclusion	75
6	Meta Navigator: Search for a Good Adaptation Policy for Few-shot Learning	77
6.1	Introduction	77
6.2	Policy Searching Space	80

6.2.1	Search Space for Representation Encoding	80
6.2.2	Search Space for Prototype Learning	82
6.3	Method	83
6.3.1	Continuous Relaxation of Policies	83
6.3.2	Optimization	84
6.3.3	Decoding Discrete Policies	85
6.4	Experiments	86
6.4.1	Implementation Details	86
6.4.2	Results and Analysis	87
6.4.3	Comparison with State-of-the-Art Methods	91
6.5	Conclusion	91
7	Summary	93
7.1	Conclusion	93
7.1.1	Few-Shot Image Segmentation	93
7.1.2	Few-Shot Image Classification	94
7.2	Future Works	95
7.2.1	Few-Shot Image Segmentation	95
7.2.2	Few-Shot Image Classification	96
7.2.3	Unified Few-Shot Learning Algorithms	96
	List of Publications	97
	Bibliography	99

List of Figures

3.1	Overview of our proposed network for 1-shot segmentation. Our framework consists of a dense comparison module (DCM) and an iterative optimization module (IOM). Given only one annotated training image, our network is able to segment test images with new classes and iteratively optimize the results.	14
3.2	CANet for 1-shot semantic segmentation. (a) Overview of our network structure. (b) Dense Comparison Module (DCM). The DCM exploits multiple levels of feature representations from CNNs to make dense feature comparison. (c) Iterative Optimization Module (IOM). The IOM is used to refine predicted results in an iterative manner. ASPP denotes Atrous Spatial Pyramid Pooling module.	18
3.3	Attention mechanism for k -shot semantic segmentation. We use the softmax function to normalize the outputs of the attention module from different support examples.	20
3.4	(a) CANet with pixel-wise annotated support set. (b) CANet with bounding box annotated support set.	22
3.5	Qualitative examples of 1-shot segmentation on the PASCAL-5 ⁱ dataset. The first row is query images and support images (right bottom) with ground-truth annotations. The second row is our predicted results. Note that the 5th and the 6th examples have the same query images and our model is able to segment different classes when different support examples are presented.	25
3.6	Visualization of the iterative optimization process. The first column shows the query and support images with ground-truth masks annotated. The rest columns show our iterative optimization results.	27
4.1	Illustration of the proposed Pyramid Graph Networks for solving one-shot image segmentation. Given only one training image, our model could perform segmentation on new test images. The proposed Graph Attention mechanism could find the correspondence between image regions.	32

4.2	Illustration of our network on 1-shot image segmentation task. Given a sampled image pair, we first use a shared CNN to extract their features, then we model the query feature maps and the foreground region in support feature maps with graphs. After that, a set of paralleled adaptive pooling layers are applied to the query features to acquire different sub-region representations, which are sent to different Graph Attention Units for graph reasoning. Finally, the outputs of different branches are fused by addition and further processed by the rest convolutions. ASPP denotes the Atrous Spatial Pyramid Pooling module. The GAU branch with equal attention weights has the same effect with global pooling.	35
4.3	Illustration of our proposed Graph Attention Unit with one example node in the query graph.	36
4.4	Qualitative results of our network. The first row is the query images. The second row is our network predictions of the query images. The third row is support images with ground-truth annotations. The fourth row is the corresponding attention maps of selected regions (marked with red rectangles in the query images). The grids in the query image indicate which branch we extract the attention maps from. The attention values are normalized to $[0,1]$ to highlight the salient region in the support images.	38
5.1	Illustration of using the Earth Mover's Distance for one-shot image classification. Our method uses an optimal matching cost between image regions to represent the image distance more faithfully. . . .	48
5.2	Our framework for 1-shot image classification. Given a pair of images, we first extract their local embeddings which are two sets of feature vectors. Then we use the Earth Mover's Distance to generate the optimal matching flows between two sets, which have the minimum overall matching cost. Finally, based on the optimal matching flows and matching costs, we can compute the distance between two images, which is used for classification. We explore three methods to extract local embeddings: (a) fully convolutional networks, (b) cropping image patches based on grids, and (c) random sampling of image patches. The details of the three methods are provided in Section 5.3.2.	52
5.3	Pyramid structures applied on DeepEMD-FCN and DeepEMD-Grid to extract local embeddings. The feature pyramid structure (a) adopts RoI poolings with different output sizes on the feature maps to generate local embeddings at multiple scales while the image pyramid structure (b) crops the input image into patches according to different grid sizes and all patches are sent to the CNN to generate local embeddings.	58

5.4	Comparison of the standard fully connected layer (a) and our proposed structured fully connected layer (SFC) (b) . The SFC learns a group of vectors as the prototype for each class such that we can use the EMD to generate category scores.	59
5.5	Experiment on 5-way k -shot classification. The proposed structured FC layer significantly outperforms baselines and previous k -shot solutions.	62
5.6	Visual reconstruction experiment by DeepEMD-FCN. Given two images (left and right), we plot the best matched patch (the one with the highest flow value) of each local region in the left image (middle). It can be seen as the reconstruction of the left image using patches from the right one. The weight controls the brightness of the corresponding region. Our algorithm can effectively establish semantic correspondence between local regions and assign small weights to the background regions.	69
5.7	Visualization of full matching flows in DeepEMD-FCN. Each grid cell in the middle contains the matched patches (from the right image) of the corresponding region in the left image. The brightness is controlled by the flow values and the weights. <i>Please zoom in for details.</i>	70
6.1	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 <i>mini</i> ImageNet dataset. Cross-domain experiments are evaluated on CUB dataset. Existing few-shot learning algorithms are highly task-specific and are outperformed by a simple finetune baseline when the domain difference is large and more support data are available.	78
6.2	Our search-based framework for few-shot classification. We aim to search for a good adaptation policy at each network stage. Our searching space mainly include two parts: the policies for representation encoding and the policies for prototype learning. Please refer to Sec. 6.2 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 [1], MAML [2] and Baseline++ [3] (bottom).	83
6.3	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. 6.4.2 for our analysis.	88

List of Tables

3.1	Results on the PASCAL-5 ⁱ dataset. Our proposed method outperforms all previous methods under both evaluation metrics and sets a new state-of-the-art performance (bold).	22
3.2	Evaluation with different support set annotations. Our model with bounding box annotated support set can achieve comparable performance to the result with pixel-wise annotations	25
3.3	Ablation experiments on the choice of features in ResNet for comparison. The combination of features after <i>block2</i> and <i>block3</i> achieves the best result.	26
3.4	Ablation experiments on the iterative optimization module. CANet-Init denotes the initial prediction from CANet without additional optimization. Our iterative optimization scheme outperforms the baseline models by 2.8% and is more effective in refining the segmentation maps than DenseCRF.	27
3.5	Comparison of different 5-shot solutions. Our attention method performs best and brings the most increment in the meanIoU score over the 1-shot baseline.	28
3.6	MeanIoU results on COCO dataset. MS denotes multi-scale evaluation.	29
4.1	Our model with different backbone networks. PGNet-Dot denotes our model with dot-product as the pair-wise function $f(\cdot)$. Our network with ResNet-50 backbone achieves better results.	40
4.2	Our network with dot-product and linear transformation as the pair-wise function $f(\cdot)$. The dot-product version achieves higher mean IoU score.	40
4.3	Comparison of using dilated convolution or adaptive pooling to undertake multi-range graph reasoning. Both two methods could boost performance over the baseline model. Adopting adaptive average pooling on the query feature map yields the best result.	41
4.4	Comparison of our graph-based network with model variants based on global vector guidance. Our proposed method achieves better results than the baseline methods and their ensemble versions.	43
4.5	Influence of multi-scale input test. All predictions are rescaled to the original image size and fused by average.	43

4.6	Comparison with the state-of-the-art 1-shot segmentation performance on PASCAL VOC 2012 dataset.	44
4.7	Comparison with the state-of-the-art 1-shot segmentation results on PASCAL VOC 2012 dataset, regarding the evaluation metric proposed in [4]	45
4.8	Comparison with the state-of-the-art 5-shot segmentation performance on PASCAL VOC 2012 dataset. Our network outperforms previous methods under both evaluation metrics.	45
5.1	Comparison with different baseline methods for 1-shot classification. Our model with EMD as the distance metric significantly outperforms baseline models based on image-level representations and local representations.	64
5.2	Different methods for setting the weights in the EMD. We report the 1-shot performance with only the feature pre-training step. EMD with our cross-reference (CR) mechanism yields the best result. The model variant that is solely based on the cross-reference mechanism as attention and removes the EMD can cause a significant performance drop. The combination of the EMD and the cross-reference mechanism generates the best result.	65
5.3	Comparison of different local embedding extractors described in Section 5.3.2 on 1-shot tasks. $\mathbf{P}_{\text{feat}}$ denotes the feature pyramid structure applied on DeepEMD-FCN and $\mathbf{P}_{\text{grid}}$ denotes the image pyramid structure applied on DeepEMD-Grid. The parameters in $\mathbf{P}_{\text{feat}}$ and $\mathbf{P}_{\text{grid}}$ are the RoI pooling size and the grid size, respectively. DeepEMD-Sampling outperforms the plain version of other two methods and both pyramid structures in feature level and image level can effectively boost the performance.	66
5.4	Model pre-training with self-supervised auxiliary tasks. The self-supervised auxiliary task at the pre-training stage can effectively boost the 1-shot performance of our models, while applying the random rotation as data augmentation alone degrades the performance. Please refer to Section 5.4.3 for analysis.	67
5.5	Cross-domain experiments (<i>mini</i> Imagenet $\rightarrow$ CUB). We report the performance with 95% confidence intervals on 1-shot 5-way and 5-shot 5-way tasks. We use FCN to extract local features for KNN and our method. Our proposed algorithm outperforms baseline methods with a large margin.	67
5.6	Computation time of the EMD layer in DeepEMD-FCN. In a 5-way 1-shot task with 10 query images, we vary the spatial size and the feature dimension of two input feature maps and record their computation time.	68

5.7	Comparison with the state-of-the-art 1-shot 5-way and 5-shot 5-way performance (%) with 95% confidence intervals on <i>miniImageNet</i> (a), <i>tieredImageNet</i> (a), CIFAR-FewShot (a) Fewshot-CIFAR100 (b) , and Caltech-UCSD Birds-200-2011 (c) datasets. Our model achieves new state-of-the-art performance on all datasets, and even outperforms methods with deeper backbones [†]	76
6.1	Comparison with baseline models for 5-way few-shot classification on <i>miniImageNet</i> 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.	88
6.2	Cross-domain experiments on CUB dataset. Models trained with <i>miniImageNet</i> dataset are evaluated on multiple 5-way task settings. Our model performs well across all task settings and demonstrates performance advantages over all baselines.	88
6.3	Comparison with the state-of-the-art 5-way few-shot classification results on <i>miniImageNet</i> , <i>tieredImageNet</i> , FC100, and CIFAR-FS dataset. MC denotes multi-crop testing. Our approach outperforms state-of-the-art performance on multiple datasets.	89

Chapter 1

Introduction

1.1 Few-Shot Visual Understanding

Deep neural networks have made significant breakthroughs in a variety of visual understanding tasks, such as image recognition, image segmentation, and image detection. A successful learning paradigm for neural networks is fully supervised learning where sufficient labeled training data in a given vision task are prepared to optimize the models in a differentiable manner. Usually, by increasing the number of training images in a task, the generalization capability of the models can be continually improved. Such data-hungry properties transform various highly complex visual understanding tasks into data-driven learning problems, which demonstrates great advantages over conventional hand-designed algorithms. However, this is different from how humans undertake a visual understanding task that humans are able to quickly understand a novel visual concept given only a few samples based on priors and accurately undertake tasks, such as classifying the categories of an image or locating the regions of the concept in the images. Such a gap between humans and deep learning algorithms motivates the study of few-shot learning where an algorithm is expected to generalize across tasks with only a few data in each task. Unlike the goal of fully supervised learning where a model is trained to perform well on a specific task, few-shot learning aims to learn a model that can perform well on various tasks with minor adaptations or no adaptations.

1.2 Few-Shot Learning: from Recognition to Segmentation

Image recognition, as the most fundamental visual understanding task, naturally becomes the most well-studied testbed for few-shot learning algorithms. As a few-shot classification task can be easily constructed by randomly sampling some classes and data from a large database, we can sufficiently evaluate the generalization capability of a few-shot learning algorithm by repeatedly sampling tasks from current large-scale datasets. Usually, a large training dataset is provided to learn a few-shot algorithm, while the model is evaluated on another dataset where no overlapped classes exist in the two datasets. The research literature on few-shot classification demonstrates great diversities, such as meta-learning, also called learning to learn, which focuses on designing effective learning and optimization algorithms for neural networks; metric-based methods, also called learning to compare, which aims to learn a deep metric to inference image relevance for classification.

Few-shot segmentation naturally extends the idea of few-shot learning in image recognition to image segmentation, where each pixel must be classified in an image. Given a few labeled example images of a visual concept, such as images with binary masks, a few-shot segmentation model should locate all the pixels that belong to the specified class in test images. Similar to the few-shot classification literature, methods on few-shot segmentation also have meta-learning based approaches and non-learning based approaches. In particular, most recent works prefer modeling the few-shot segmentation task as a conditional prediction model where the labelled data serve as the condition for predictions on the test images.

1.3 Challenges and Contributions

Both few-shot classification and few-shot segmentation tasks can be readily undertaken by humans, but they can be challenging to the deep learning algorithms due to their data-hungry natures. Directly learning with standard iterative gradient descents may lead to significant overfitting issues, which results in poor generalization in novel classes. This thesis solves the few-shot learning problems from many

aspects. including the 1) techniques for few-shot classification, such as data representations that can better capture semantics and data structures, distance metrics that can better reflect the relations between data, and flexible learning paradigm that can apply to multiple few-shot scenarios, and 2) the techniques for few-shot segmentation, such as better image representations that are more transferable, and better message passing schemes that effectively extract information from support images.

More specifically, in the few-shot segmentation task, the key is to effectively extract useful and discriminative information from the limited labeled training set as conditions for the predictions on the test images. We explore two strategies to achieve this goal. The first work (Chapter 3) borrows the idea of prototype learning from previous few-shot classification literature that aims to represent the information of a specific class with a vector. Then the prototype is fused into the main branch of a segmentation model as the step of message passing. To generate useful prototypes, we explore the utilization of multi-level representations in the CNN models. We further explore using graphs to achieve the goal of message passing in our second work on few-shot segmentation (Chapter 4). The graph model can avoid the need to pool the representations of training data and thus maintains the structures and locality of data representations. It essentially solves the message passing by many-to-many correspondences, which demonstrates advantages over prototype-based methods which establish many-to-one correspondences. For the fundamental few-shot classification task, we first revisit the popular metric-based method that predicts the labels with a data encoder and distance metrics. We emphasize the importance of structures and locality in data representations and propose a novel structured differentiable distance metric, Deep Earth Mover’s Distance (Chapter 5). Current few-shot learning algorithms often focus on a specific task setting, such as 1-shot and 5-shot tasks, and different algorithms often excel at different few-shot learning scenarios. To design a few-shot learning algorithm that can perform well on various few-shot learning tasks, we design a searching-based framework that can find good few-shot learning algorithms on different few-shot learning tasks (Chapter 6). All the methods in this thesis are validated on multiple benchmark datasets and they achieve superior performance over previous state-of-the-art methods.

1.4 Outline of the Thesis

This thesis is organized as follows:

- Chapter 1 introduces the background of few-shot learning in computer vision tasks, and specifically reviews the few-shot learning history in image recognition and segmentation tasks. This chapter points out the challenges in the few-shot classification and segmentation tasks and the limitations in current methods. Finally, we briefly summarize the aspects we solve these issues and conclude the main contributions in our methods.
- Chapter 2 reviews the related research topics, including the previous works on the few-shot image recognition and segmentation tasks, and the related techniques in our methods.
- Chapter 3 presents class-agnostic segmentation networks for few-shot image segmentation. The network consists of a two-branch dense comparison module which performs multi-level feature comparison between the support image and the query image, and an iterative optimization module which iteratively refines the predicted results. Furthermore, we introduce an attention mechanism to effectively fuse information from multiple support examples under the setting of k -shot learning.
- Chapter 4 presents our method that addresses the few-shot segmentation problem with graph techniques. Previous methods often simplify the many-to-many message passing issues in the few-shot segmentation to a one-to-many problem by squeezing support data to a global descriptor. However, a mixed global representation drops the data structure and information of individual elements. We propose to model structured segmentation data with graphs and apply attentive graph reasoning to propagate label information from support data to query data. The graph attention mechanism could establish the element-to-element correspondence across structured data by learning attention weights between connected graph nodes. To capture correspondence at different semantic levels, we further propose a pyramid-like structure that models different sizes of image regions as graph nodes and undertakes graph reasoning at different levels.

- Chapter 5 presents our methods for few-shot image classification from a new perspective of optimal matching between image regions. We employ the Earth Mover’s Distance (EMD) as a metric to compute a structural distance between dense image representations to determine image relevance. The EMD generates the optimal matching flows between structural elements that have the minimum matching cost, which is used to calculate the image distance for classification. To generate the important weights of elements in the EMD formulation, we design a cross-reference mechanism, which can effectively alleviate the adverse impact caused by the cluttered background and large intra-class appearance variations. To implement k -shot classification, we propose to learn a structured fully connected layer that can directly classify dense image representations with the EMD. Based on the implicit function theorem, the EMD can be inserted as a layer into the network for end-to-end training.
- Chapter 6 presents Meta Navigator, a framework that attempts to solve the 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 this 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.
- Chapter 7 summarizes the contributions of each of our proposed methods in this thesis. I then discuss the potential research directions in few-shot learning that relate to our works.

Chapter 2

Literature Review

2.1 Overview

This section provides the literature review of related topics, including the prior arts on few-shot image recognition and few-shot image segmentation tasks, and the related techniques in our methods.

2.2 Few-Shot Learning

The research literature on few-shot learning shows great diversity [5–15]. There are two main streams in the few-shot classification literature, metric-based approaches and optimization-based approaches. Optimization-based methods, *e.g.*, [3, 16–41], target at effectively adapting model parameters to new tasks in the low-shot regime. For example, MAML [2] and many of its variants [42–45] aim to learn a good model initialization that can rapidly adapt to novel tasks with limited optimization steps. Our design in Chapter 5 is more related to the metric-based methods [1, 46–56], which aim to represent samples in an appropriate feature space where data from different categories can be distinguished with distance metrics. To achieve this goal, most previous methods represent the whole image as a data point in the feature space. There are also some works utilizing local features to make predictions. For example, Lifchitz *et al.* [27] directly make predictions with each local feature and fuse their results. Li *et al.* [50] adopt k -NN to fuse local distances. Cross

Attention Networks [51] use attention mechanisms to highlight the target object regions and generate discriminative features for few-shot classification. CrossTransformers [57] compute the distances between spatially-corresponding features in the query and labeled samples for few-shot classification. Our solution to the k -shot problem in DeepEMD also draws connections to optimization-based methods since we learn a classifier that can directly classify structured representations with the Earth Mover’s Distance, which can benefit from the increasing number of support samples.

Besides the two popular approaches, many other promising methods have also been proposed to tackle the few-shot classification problem, such as works based on graph theories [58–61], reinforcement learning [62], differentiable SVM [63], generative models [64–72], transductive learning [73–77], recurrent models [78, 79], self-supervised learning [80, 81], the recent capsule network [82], and temporal convolutions [83].

2.3 Semantic Segmentation

Semantic segmentation is the task of classifying each pixel in an image to a set of pre-defined categories [84–90]. State-of-the-art methods are based on Fully Convolutional Networks (FCNs), which often employ a convolutional neural network (CNN) pre-trained for classification as the backbone architecture. To fit the task of dense prediction, fully connected layers are replaced by a convolutional layer that predicts the label of each pixel. In order to capture abstract feature representations, CNNs adopt consecutive pooling operations or convolution striding to decrease the spatial resolution of feature maps. However, this conflicts with dense prediction tasks where the output should be of high resolution. In order to balance the output resolution and receptive field of the network, dilated convolutions [85] are often used in dense prediction tasks. Dilation removes downsampling operations in the last few layers and inserts holes to convolutional filters to enlarge the receptive field. In our model, we also adopt dilated convolutions to maintain spatial resolution. In fully supervised segmentation, training an FCN model requires a large number of expensive pixel-level annotated images, and once a model is trained, it can not perform segmentation on new categories. *Our few-shot learning*

model, on the other hand, can be generalized to any new categories with only a few annotated examples.

Pyramid Structure in Segmentation. There has existed several pyramid-like structures in the segmentation literature. Early works employ Image Pyramid that different scales of input images are fed into the CNN to handle multi-scale objects in the images [91]. Input with small scales allows the features to capture long-range representations, while large-scale inputs could generate fine-grained details. Pyramid Scene Parsing network [92] performs paralleled adaptive pooling to downsample the feature maps to a set of grid scales in order to incorporate different sizes of context. DeepLab [85] shares the same spirit and adopts several paralleled dilated convolutions to capture multi-size context information.

2.4 Few-Shot Segmentation

Few-shot segmentation extends few-shot classification to pixel levels. Previous works follow a two-branch network design [4, 93–97]. The support branch extracts information from labeled support data to guide segmentation in the query branch. In [93], the query branch is an FCN and the support branch directly predicts a vector as the parameters of a layer in the query branch to guide segmentation. In [4, 95], the generated global vector from the support branch is upsampled and concatenated with the query feature for dense predictions. Zhang *et al.* [94] re-weight the query feature map by calculating a similarity score between the global vector and each query position to segment the target category.

The intrinsic spirits behind the above methods are to simplify the many-to-many correspondence problem to a one-to-many problem. To generate the global descriptor, early works [4, 93, 96] hide the background in the support RGB image and perform global average pooling at the end of the support branch to generate a global descriptor. Works in [94, 95] improve it by performing average pooling over the foreground mask regions in intermediate features to extract the global vector. Our method in Chapter 4, on the other hand, establishes element-to-element correspondence that each location in the query image could selectively extract useful information from the support set.

2.5 Earth Mover's Distance.

Earth Mover's Distance (EMD) was originally proposed in [98] as a metric for color and texture based image retrieval. EMD has the formulation of the well studied transportation problem in linear programming, and thus the global optimal matching can be found by solving a linear program. EMD has several desirable properties that make it a popular method to compare structured representations. First, EMD can generate a structural similarity without explicit alignment information. It extends the distance between single elements to the distance between sets or distributions. Second, the number of elements in the sets can vary and EMD allows for partial matching when the total weights of two sets are not equal. EMD has been widely applied to many areas. For example, Kusner *et al.* [99] use EMD to measure the similarity between two documents, which calculates the minimal cost to transfer the word embeddings in a document to the other for document classification. Wang and Chan [100] propose to represent the hand shapes and textures with superpixels and employ EMD to measure the dissimilarity between the hand gestures for gesture recognition. In [101], Nikolentzos *et al.* represent graph data as a set of vectors corresponding to the vertices and use EMD to determine the similarity of two graphs for graph comparison. Schuler *et al.* [102] solve the multi-object tracking problem with a network flow formulation that learns features for network-flow-based data association. Zhao *et al.* [103] propose to use the differential EMD to tackle the visual tracking problem based on the sensitivity analysis of the simplex method. Li [104] uses a tensor-SIFT based EMD to tackle the contour tracking problem.

2.6 Parameterized optimization.

Parameterized optimization problems have a parameterized object function and constraints that depend on input data. Many previous works have investigated differentiation through the argmin operators. In [105], Gould *et al.* present the methods for differentiation through optimization problems with only equality constraints. Agrawal *et al.* [106] propose a method that can compute the gradient of the solution with respect to the coefficients in the convex cone program, which can

scale to large problems. Barratt [107] describes the general case of using the implicit function theorem and interior point methods to compute the Jacobian of the solution with respect to the problem parameters. With the same theory, Amos and Kolter [108] design a batched Quadratic Programming solver as a layer that can be integrated into a neural network for end-to-end training. In [109], Agrawal *et al.* propose a differentiable convex optimization layer that can differentiate through disciplined convex programs and allow users to define problems in a natural syntax without converting problems to canonical forms. Vlastelica *et al.* [110] introduce the combinatorial building blocks into neural networks and the end-to-end trainable network can generate informative backward gradients through any black-box implementations of combinatorial solvers. Based on such building blocks, Rolínek *et al.* [111] design an end-to-end network that incorporates a combinatorial solver to solve the graph matching problem.

2.7 Network Architecture Search (NAS).

Our work draws connections with the NAS literature [112, 113], which aims for identifying effective building elements in the CNN structures. The most related work to ours is DARTS [112], 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 [113–116]. For example, a simple random search baseline can outperform the architecture searched by DARTS [112], and the searching favors parameter-free operations, *e.g.*, skip connections [114]. Recently, many improved designs are proposed to solve the issues in DARTS [113–118]. For example, Wang *et al.* [113] 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 it perturbs the supernet performance when an operation 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 [119–122], 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.

Chapter 3

CANet: Class-Agnostic Segmentation Networks with Iterative Refinement and Attentive Few-Shot Learning¹

3.1 Introduction

Deep Convolutional Neural Networks have made significant breakthroughs in many visual understanding tasks including image classification [123–125], object detection [126–128], and semantic segmentation [84–86]. One crucial reason is the availability of large-scale datasets such as ImageNet [129] that enable the training of deep models. However, data labeling is expensive, particularly for dense prediction tasks, *e.g.*, semantic segmentation and instance segmentation. In contrast to machine learning algorithms, humans are able to segment a new concept from the image easily when only seeing a few examples. The gap between humans and machine learning algorithms motivates the study of few-shot learning that aims to learn a model which can be generalized well to new classes with scarce labeled training data.

¹The work in this chapter has been published in the paper : Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, Chunhua Shen. “CANet: Class-Agnostic Segmentation Networks with Iterative Refinement and Attentive Few-Shot Learning.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (**CVPR**). Long Beach, United States. 2019. [95]

FIGURE 3.1: Overview of our proposed network for 1-shot segmentation. Our framework consists of a dense comparison module (DCM) and an iterative optimization module (IOM). Given only one annotated training image, our network is able to segment test images with new classes and iteratively optimize the results.

In this chapter, we undertake the task of few-shot semantic segmentation that only uses a few annotated training images to perform segmentation on new classes. Previous work [4, 93, 96] on this task follows the design of two-branch structure which includes a support branch and a query branch. The support branch aims to extract information from the support set to guide segmentation in the query branch. We also adopt the two-branch design in our framework to solve the few-shot segmentation problem.

Our network includes a two-branch dense comparison module, in which a shared feature extractor extracts representations from the query set and the support set for comparison. The design of the dense comparison module takes inspiration from metric learning [1, 49] on image classification tasks where a distance function evaluates the similarity between images. However, different from image classification where each image has a label, image segmentation needs to make predictions on data with structured representation. It is difficult to directly apply metric learning to dense prediction problems. To solve this, one straightforward approach is to make comparisons between all pairs of pixels. However, there are millions of pixels in an image and comparison of all pixel pairs takes enormous computational cost. Instead, we aim to acquire a global representation from the support image for comparison. Here, to only focus on the assigned category, we use global average pooling over the foreground area to filter out irrelevant information. Then the

global feature is compared with each location in the query branch, which can be seen as a dense form of the metric learning approach.

Under the few-shot setting, the network should be able to handle new classes that are never seen during training. Thus we aim to mine transferable representations from CNNs for comparison. As is observed in feature visualization literature [130, 131], features in lower layers relate to low-level cues, *e.g.*, edges and colors while features in higher layers relate to object-level concepts such as categories. We focus on middle-level features that may constitute object parts shared by unseen classes. For example, if the CNN learns a feature that relates to *wheel* when the model is trained on the class *car*, such feature may also be useful for feature comparison on new vehicle classes, *e.g.*, *truck* and *bus*. We extract multiple levels of representations in CNNs for dense comparison.

As there exist variances in appearance within the same category, objects from the same class may only share a few similar features. Dense feature comparison is not enough to guide segmentation of the whole object area. Nevertheless, this gives an important clue of where the object is. In semi-automatic segmentation literature, weak annotations are given for class-agnostic segmentation, *e.g.*, interactive segmentation with click or scribble annotations [132, 133] and instance segmentation with bounding box or extreme point priors [134, 135]. Transferable knowledge to locate the object region is learned in the training process. Inspired by semi-automatic segmentation tasks, we hope to gradually differentiate the objects from the background given the dense comparison results as priors. We propose an iterative optimization module (IOM) that learns to iteratively refine the predicted results. The refinement is performed in a recurrent form that the dense comparison result and the predicted masks are sent to an IOM for optimization, and the output is sent to the next IOM recurrently. After a few iterations of refinement, our dense comparison module is able to generate fine-grained segmentation maps. Inside each IOM, we adopt residual connections to efficiently incorporate the predicted masks in the last iteration step. Fig. 3.1 shows an overview of our network for one-shot segmentation.

Previous methods for k -shot segmentation is based on the 1-shot model. They use non-learnable fusion methods to fuse individual 1-shot results, *e.g.*, averaging 1-shot predictions or intermediate features. Instead, we adopt an attention mechanism to effectively fuse information from multiple support examples.

To further reduce the labeling efforts for few-shot segmentation, we explore a new test setting: our model uses the bounding box annotated support set to perform segmentation in the query image. We conduct comprehensive experiments on the PASCAL VOC 2012 dataset and COCO dataset to validate the effectiveness of our network. Main contributions in this chapter are summarized as follows.

- We develop a novel two-branch dense comparison module which effectively exploits multiple levels of feature representations from CNNs to make dense feature comparison.
- We propose an iterative optimization module to refine predicted results in an iterative manner. The ability of iterative refinement can be generalized to unseen classes with few-shot learning for generating fine-grained maps.
- We adopt an attention mechanism to effectively fuse information from multiple support examples in the k -shot setting, which outperforms non-learnable fusion methods of 1-shot results.
- We demonstrate that given support set with weak annotations, *i.e.*, bounding boxes, our model can still achieve comparable performance to the result with expensive pixel-level annotated support set, which further reduces the labeling efforts of new classes for few-shot segmentation significantly.
- Experiments on the PASCAL VOC 2012 dataset show that our method achieves a mean Intersection-over-Union score of 55.4% for 1-shot segmentation and 57.1% for 5-shot segmentation, which significantly outperform state-of-the-art results by 14.6% and 13.2%, respectively.

3.2 Task Description

Suppose that our model is trained on a dataset with the class set $\mathcal{C}_{train}$, our goal is to use the trained model to make the prediction on a different dataset with new classes $\mathcal{C}_{test}$ where only a few annotated examples are available. Intuitively, we train the model to have the ability that for a new class $c \notin \mathcal{C}_{train}$, our model is able to segment the class from the images when only sees a few pictures of this class. Once the model is trained, the parameters are fixed and require no optimization when tested on a new dataset.

We align training and testing with the episodic paradigm [46] to handle the few-shot scenario. Specifically, given a k -shot learning task, each episode is constructed by sampling 1) a support (training) set $\mathcal{S} = \{(x_s^i, y_s^i(c))\}_{i=1}^k$, where $x_s^i \in \mathbb{R}^{H_i \times W_i \times 3}$ is an RGB image and $y_s^i(c) \in \mathbb{R}^{H_i \times W_i}$ is a binary mask for class c in the support image; and 2) a query (test) set $\mathcal{Q} = \{x_q, y_q(c)\}$ where x_q is the query image and $y_q(c)$ is the ground-truth mask for class c in the query image. The input to the model is the support set $\mathcal{S}$ and the query image x_q , and the output is the predicted mask $\hat{y}_q(c)$ for class c in the query image. As there may be multiple classes in one query image x_q , the ground truth query mask is different when a different label c is assigned. Fig. 3.1 shows an illustration of the task when $k = 1$.

3.3 Method

We propose a new framework that solves the few-shot semantic segmentation problem. We begin with the illustration of our model in the 1-shot setting first without loss of generality. Our network consists of two modules: the dense comparison module (DCM) and the iterative optimization module (IOM). The DCM performs dense feature comparison between the support example and the query example, while IOM performs iterative refinement of predicted results. Fig. 3.2 (a) shows an overview of our framework. To generalize our network from 1-shot learning to k -shot learning, we adopt an attention mechanism to fuse information from different support examples. Moreover, we propose a new test setting that uses support images with bounding box annotations for few-shot segmentation, which is described subsequently.

3.3.1 Dense Comparison Module

We develop a two-branch dense comparison module that densely compares each position in the query image with the support example, as shown in Fig. 3.2 (b). The module consists of two sub-modules: a feature extractor that extracts representations and a comparison module that performs feature comparison.

Feature extractor. The feature extractor aims to harvest different levels of representations from CNNs for feature matching. We use a ResNet-50 [124] as the

FIGURE 3.2: CANet for 1-shot semantic segmentation. (a) Overview of our network structure. (b) Dense Comparison Module (DCM). The DCM exploits multiple levels of feature representations from CNNs to make dense feature comparison. (c) Iterative Optimization Module (IOM). The IOM is used to refine predicted results in an iterative manner. ASPP denotes Atrous Spatial Pyramid Pooling module.

backbone of the feature extractor. As done in previous few-shot segmentation work, the backbone model is pre-trained on Imagenet [129]. As is observed in CNN feature visualization literature [130, 131], features in lower layers often relate to low-level cues, *e.g.*, edges and colors while features in higher layers relate to object-level concepts such as object categories. In the few-shot scenario, our model should adapt to any unseen classes. Thus we can not assume that a feature corresponding to an unseen category is learned during training. Instead, we focus on middle-level features that may constitute object parts shared by unseen classes. The layers in ResNet are divided into 4 blocks based on the spatial resolution which naturally correspond to 4 different levels of representation. We choose features generated by *block2* and *block3* for feature comparison and abandon layers after *block3*. We use dilated convolutions [85] in layers after *block2* to maintain the spatial resolution of feature maps. All feature maps after *block2* have a fixed size of $1/8$ of the input image. Features after *block2* and *block3* are concatenated and encoded to 256 dimensions by 3×3 convolutions. We investigate the choice of

features for comparison in Section 3.4.1.3. Both the support branch and the query branch use the same feature extractor. We keep the weights in ResNet fixed during training.

Dense Comparison. As there may be multiple object categories and cluttered backgrounds in the support image, we want to acquire an embedding that only corresponds to the target category for comparison. Here, we use global average pooling over the foreground area to squeeze the feature maps to a feature vector. Global image features turn out to be useful in segmentation tasks [92, 136, 137], which can be easily achieved by global average pooling. In our network, we only average features over the foreground area to filter out irrelevant areas. After we obtain the global feature vector from the support set, we concatenate the vector with all spatial locations in the feature map generated by the query branch. This operation aims to compare all the spatial locations in the query branch to the global feature vector from the support branch. Then, the concatenated feature maps go through another convolutional block with $256\ 3 \times 3$ convolutional filters for comparison.

For efficient implementation, we first bilinearly downsample the binary support mask to the same spatial size of the feature maps and then apply element-wise multiplication with the feature maps. As a result, features belonging to the background area become zero. Then we adopt global sum pooling and divide the resulting vector by the foreground area to obtain the average feature vector. We upsample the vector to the same spatial size of query features and concatenate them for dense comparison.

3.3.2 Iterative Optimization Module

As there exist variances in appearance within the same category, dense comparison can only match a part of the object, which may not be sufficiently powerful to accurately segment the whole object in the image. We observe that the initial prediction is an important clue about the rough position of the objects. We propose an iterative optimization module to optimize the predicted results iteratively. The structure is shown in Fig. 3.2 (c). The module’s input is the feature maps generated by the dense comparison module and predicted masks from the last iteration. Directly concatenating feature maps with predicted masks as additional channels

FIGURE 3.3: Attention mechanism for k -shot semantic segmentation. We use the softmax function to normalize the outputs of the attention module from different support examples.

causes mismatch to the feature distribution as there is no predicted mask for the first forward pass. Instead, we propose to incorporate the predicted masks in a residual form:

$$M_t = x + F(x, y_{t-1}), \quad (3.1)$$

where x is the output feature of the dense comparison module; y_{t-1} is the predicted masks from the last iteration step, and M_t is the output of the residual block. Function $F(\cdot)$ is the concatenation of feature x and predicted masks y_{t-1} , followed by two 3×3 convolution blocks with 256 filters. Then we add two vanilla residual blocks with the same number of convolutional filters. On top of that, we use Atrous Spatial Pyramid Pooling module (ASPP) proposed in Deeplab V3 [137] to capture multi-scale information. The module consists of four parallel branches that include three 3×3 convolutions with atrous rates of 6, 12, and 18 respectively and a 1×1 convolution. The 1×1 convolution is operated on the image-level feature which is achieved by global average pooling. Then the resulting vector is bilinearly upsampled to the original spatial size. The output features from 4 branches are

concatenated and fused by another 1×1 convolution with 256 filters. Finally, we use 1×1 convolution to generate the final masks which include a background mask and a foreground mask. We use a softmax function to normalize the score in each location, which outputs the confidence maps of the foreground and the background. The confidence maps are then fed to the next IOM for optimization. Our final result is achieved by bilinearly upsampling the confidence map to the same spatial size of the query image and classifying each location according to the confidence maps. At the training time, to avoid the iterative optimization module over-fitting the predicted masks, we alternatively use predicted masks in the last epoch and empty masks as the input to IOM. The predicted masks y_{t-1} is reset to empty masks with a probability of p_r . This can be seen as dropout of the whole mask, an extension of the standard dropout [138]. In comparison to previous iterative refinement methods in segmentation literature [133, 139, 140], our method integrates the refinement scheme into the model with residual connection so that the whole model could run in a feed-forward manner and is trained end-to-end.

3.3.3 Attention Mechanism for k-shot Segmentation

In order to efficiently merge information in the k -shot setting, we use an attention mechanism to fuse the comparison results generated by different support examples. Specifically, we add an attention module parallel to the dense comparison convolution in DCM (see Fig. 3.3). The attention branch consists of two convolutional blocks. The first one has 256 3×3 filters, followed by 3×3 max pooling. The second one has one 3×3 convolution followed by a global average pooling. The result from the attention branch serves as the weight λ . Then, the weights from all support examples are normalized by a softmax function:

$$\hat{\lambda}_i = \frac{e^{\lambda_i}}{\sum_{j=1}^k e^{\lambda_j}}. \quad (3.2)$$

The final output is the weighted sum of features generated by different support samples.

FIGURE 3.4: (a) CANet with pixel-wise annotated support set. (b) CANet with bounding box annotated support set.

Method	1-shot					5-shot				
	split-0	split-1	split-2	split-3	mean	split-0	split-1	split-2	split-3	mean
OSLSM[93]	33.6	55.3	40.9	33.5	40.8	35.9	58.1	42.7	39.1	43.9
CANet	52.5	65.9	51.3	51.9	55.4	55.5	67.8	51.9	53.2	57.1

(A) 1-shot and 5-shot results under the meanIoU evaluation metric.

Method	1-shot					5-shot				
	split-0	split-1	split-2	split-3	mean	split-0	split-1	split-2	split-3	mean
OSLSM [93]	-	-	-	-	61.3	-	-	-	-	61.5
co-FCN [4]	-	-	-	-	60.1	-	-	-	-	60.2
PL [96]	-	-	-	-	61.2	-	-	-	-	62.3
CANet	71.0	76.7	54.0	67.2	66.2	74.2	80.3	57.0	66.8	69.6

(B) 1-shot and 5-shot results under the FB-IoU evaluation metric.

TABLE 3.1: Results on the PASCAL-5ⁱ dataset. Our proposed method outperforms all previous methods under both evaluation metrics and sets a new state-of-the-art performance (bold).

3.3.4 Bounding Box Annotations

As the essence of our dense comparison module is to densely compare each location in the query image to the global representation provided by the support example, we explore a new form of support set annotation that uses bounding boxes. Compared with pixel-wise annotations, the bounding box annotation uses a rectangular

box to denote the object area, which is often used in object detection tasks. Labeling bounding box annotations is much cheaper than pixel-wise labeling. We relax the support set by treating the whole bounding box area as the foreground. We test our model under this setting to evaluate the capability of our framework. The comparison of the two test settings is shown in Fig. 3.4.

3.4 Experiments

To evaluate the performance of our proposed method, we conduct extensive experiments on the PASCAL VOC 2012 dataset and COCO dataset. Our network is trained end-to-end. The loss function is the mean of cross-entropy loss over all spatial locations in the output map. Our network is trained using SGD for 200 epochs with the PyTorch library on Nvidia Tesla P100 GPUs. We set the learning rate to 0.0025 and set probability p_r to 0.7. We use a mini-batch of 4 episodes for training on PASCAL-5ⁱ and 8 on COCO. At inference time, we iteratively optimize the predicted results for 4 times after the initial prediction.

Evaluation metric. There is a minor difference of evaluation metrics in previous work. Shaban *et al.* [93] measure the per-class foreground Intersection-over-Union (IoU) and use the average IoU over all classes (meanIoU) to report the results. While in [4, 96], they ignore the image categories and calculate the mean of foreground IoU and background IoU over all test images (FB-IoU). We choose the meanIoU evaluation metric for our analysis experiments due to the following reasons: 1) The numbers of test samples in different classes are not balanced (*e.g.*, 49 of class *sheep* vs. 378 of class *person*). Ignoring the image categories may lead to a biased result towards the class with more images. Also, we can observe the effectiveness of our model in different classes with the meanIoU evaluation metric. 2) As most objects are small relative to the whole image, even though the model fails to segment any objects, the background IoU can still be very high, thus failing to reflect the capability of the model. 3) Foreground IoU is more often used in binary segmentation literature (*e.g.*, video segmentation and interactive segmentation). Nevertheless, we still compare our results with previous work under both evaluation metrics.

3.4.1 PASCAL-5i

PASCAL-5ⁱ is a dataset for few-shot semantic segmentation proposed in [93]. It is built on images from PASCAL VOC 2012 and extra annotations from SDS [141]. 20 object categories from PASCAL VOC are evenly divided into 4 splits with three splits for training and one split for testing. At test time, 1000 support-query pairs are randomly sampled in the test split. More details of PASCAL-5ⁱ can be found in [93].

3.4.1.1 Comparison with the State-of-the-art Methods

We compare our model with the state-of-the-art methods in Table 3.1. Table 3.1 (a) shows the results evaluated under the meanIoU evaluation metric and Table 3.1 (b) shows the results under the FB-IoU metric. For the performance of [93] under the FB-IoU metric, we quote the result reproduced in [4]. *Our model significantly outperforms the state-of-the-art methods under both evaluation metrics.* Particularly, our meanIoU score outperforms the state-of-the-art results by 14.6% for the 1-shot task and 13.2% for the 5-shot task.

Qualitative Results. Fig. 3.5 shows some qualitative examples of our segmentation results. Note that given the same query image, our model is able to segment different classes when different support examples are presented (See the 5th and the 6th examples in Fig. 3.5).

3.4.1.2 Experiments on Bounding Box Annotations

We evaluate CANet with the bounding box annotated support set at test time. We acquire bounding box annotations from the PASCAL VOC 2012 dataset and SDS [141]. The support mask is the region inside the bounding box of one instance instead of all instances in a support image. The instance is chosen randomly. As is shown in Table 3.2, *the performance with bounding box annotated support set is comparable to the result with expensive pixel-level annotated support set*, which means our dense comparison module is able to withstand noise introduced by the background area within the bounding box.

Annotation	Result (meanIoU %)
Pixel-wise labels	54.0
Bounding box	52.0

TABLE 3.2: Evaluation with different support set annotations. Our model with bounding box annotated support set can achieve comparable performance to the result with pixel-wise annotations

FIGURE 3.5: Qualitative examples of 1-shot segmentation on the PASCAL-5ⁱ dataset. The first row is query images and support images (right bottom) with ground-truth annotations. The second row is our predicted results. Note that the 5th and the 6th examples have the same query images and our model is able to segment different classes when different support examples are presented.

3.4.1.3 Ablation Study

We implement extensive ablation experiments on the PASCAL-5ⁱ dataset to inspect the effectiveness of different components in our network. All results are average meanIoU over 4 splits on the PASCAL-5ⁱ dataset.

Features for Comparison. In Table 3.3, we compare our model variants that use different levels of feature in ResNet-50 for feature comparison. In all cases, we encode the features to 256 dimensions before comparison and we do not adopt iterative optimization. We experiment feature comparison with single block and multiple blocks. When single block is used for comparison, *block3* performs best. When multiple blocks are used for comparison, the combination of *block2* and *block3* achieves the best result. The reason is that *block2* corresponds to relatively low-level cues, which alone is not enough to match object parts. While *block4* corresponds to high-level features, *e.g.*, categories, and incorporates a great number of parameters (2048 channels), which makes it hard to optimize under the few-shot setting. The combination of *block2* and *block3* is the best for matching class-agnostic object parts. We also implement experiments with VGG16 as the feature extractor. We choose features of stage 2, 3, and 4 (out of 5). The final

<i>block2</i>	<i>block3</i>	<i>block4</i>	meanIoU
✓			46.6
	✓		50.8
		✓	48.4
✓	✓		51.2
	✓	✓	49.2
✓		✓	49.6
✓	✓	✓	49.5

TABLE 3.3: Ablation experiments on the choice of features in ResNet for comparison. The combination of features after *block2* and *block3* achieves the best result.

multi-scale test result with VGG as the backbone is 54.3%. Compared with the ResNet50 version (55.4%), the performance only drops by 1.1% and still significantly outperforms the state-of-the-art results.

Iterative Optimization Module. To validate the effectiveness of our proposed iterative optimization module, we compare our network with a baseline model that does not employ additional IOM for optimization, *i.e.*, the initial prediction from CANet(CANet-Init). We also compare our iterative optimization scheme with DenseCRF [142], which is a post-processing method widely used in segmentation literature to refine segmentation maps. Table 3.4 shows the results of different model variants. As is shown, the iterative optimization yields 2.8% improvement over the initial prediction. DenseCRF does not significantly improve the few-shot segmentation prediction. We visualize the results and find that for the predicted masks which successfully locate most of the object region, DenseCRF can effectively improve segmentation results, particularly in the region of object boundaries. However, for failure masks, *e.g.*, false localization of objects, DenseCRF expands false positive regions, which deteriorates the IoU score. Our IOM, on the other hand, can effectively fill the object region and remove irrelevant areas in a learnable way. We visualize the intermediate results of our iterative optimization process in Fig. 3.6.

Attention *vs.* Feature Fusion *vs.* Mask Fusion. In the k -shot setting, we compare our attention mechanism to several solutions in previous work: 1) Feature-level average fusion. We experiment the method in [4], which is to average

Method	Result (meanIoU %)
CANet-Init	51.2
CANet-Init + DenseCRF	51.9
CANet	54.0

TABLE 3.4: Ablation experiments on the iterative optimization module. CANet-Init denotes the initial prediction from CANet without additional optimization. Our iterative optimization scheme outperforms the baseline models by 2.8% and is more effective in refining the segmentation maps than DenseCRF.

FIGURE 3.6: Visualization of the iterative optimization process. The first column shows the query and support images with ground-truth masks annotated. The rest columns show our iterative optimization results.

the features generated by different support examples. 2) Logic OR fusion for masks. Shaban *et al.* [93] use 1-shot model to make predictions with each support example and use logic OR operation to fuse individual predicted masks. Logic OR operation means that a position is predicted as foreground if any support example predicts it as foreground. 3) Average fusion for masks. Moreover, we also experiment with average operation to fuse individual 1-shot predicted confidence maps. We report the results of CANet with different fusion solutions in Table 3.5. Our attention mechanism performs best and brings the most increment over 1-shot

Method	Result (meanIoU %)	Increment
1-shot baseline	54.0	0
Feature-Avg	55.0	1.0
Mask-Avg	54.5	0.5
Mask-OR	53.4	-0.6
Attention	55.8	1.8

TABLE 3.5: Comparison of different 5-shot solutions. Our attention method performs best and brings the most increment in the meanIoU score over the 1-shot baseline.

baseline. This indicates that a learned attention module can be more effective in fusing information from different support examples than non-learnable fusion methods in feature level or image level. Using logic OR operation to fuse predicted masks does not show improvement over the 1-shot result.

Multi-scale evaluation. We also experiment multi-scale evaluation as is commonly done in segmentation literature. Specifically, we re-scale the query image by $[0.7, 1, 1.3]$ and average their predicted results. Multi-scale evaluation brings 1.4% and 1.3% meanIoU improvement in 1-shot and 5-shot settings, respectively.

3.4.2 COCO

COCO 2014 [143] is a challenging large-scale dataset, which contains 80 object categories. The original dataset contains 82,783 and 40,504 images for training and validation respectively. Directly experimenting on the original dataset is very demanding on time and computation. Instead, we choose a subset of the original dataset to evaluate our model and for further research on this topic. We choose 40 classes for training, 20 for validation and 20 for test, which contain 39,107 (train), 5,895 (validation) and 9,673 (test) samples, respectively. Training images are chosen from the COCO training set, while validation and test images are chosen from the COCO validation set.

For the 1-shot task, we compare our network with the baseline model that does not employ additional iterative optimization (CANet-Init), and for the 5-shot task, we compare our attention mechanism with three non-learnable fusion methods described in Section 3.4.1.3. The result is shown in Table 3.6. In the 1-shot setting, our iterative optimization scheme brings 4.1% meanIoU improvement. Multi-scale

Method	MS	Result (meanIoU %)
CANet-Init		42.2
CANet		46.3
CANet	✓	49.9

(A) 1-shot results on COCO dataset.

Method	MS	Result (meanIoU %)
Feature-Avg		48.9
Mask-Avg		49.2
Mask-OR		46.2
Attention		49.7
Attention	✓	51.6

(B) 5-shot results on COCO dataset.

TABLE 3.6: MeanIoU results on COCO dataset. MS denotes multi-scale evaluation.

evaluation shows extra 3.3% increase. In the 5-shot setting, our attention mechanism outperforms all non-learnable methods. Multi-scale evaluation obtains another 1.9% gain.

3.5 Conclusion

We have presented CANet, a novel class-agnostic segmentation network with few-shot learning. The dense comparison module exploits multiple levels of feature in CNNs to perform dense feature comparison and the iterative optimization module learns to iteratively refine the predicted results. Our attention mechanism for solving the k -shot problem turns out to be more effective than non-learnable methods. Comprehensive experiments show the effectiveness of our framework, and the performance significantly outperforms all previous work.

Chapter 4

Pyramid Graph Networks with Connection Attentions for Region-Based One-Shot Semantic Segmentation¹

4.1 Introduction

Fully supervised learning has shown great success in many computer vision tasks, thanks to large-scale datasets [129, 143] and deep neural networks [123, 124]. However, standard supervised learning tasks, such as image classification and semantic segmentation, have their intrinsic limitations that both training and testing have to be applied on a specific task, *i.e.*, the categories to be classified. More recently, meta-learning along with the power of deep neural networks has received growing interests in the machine learning community. Meta-learning, also called learning to learn, aims to train a meta learner that is able to produce good generalization performance on unseen but similar tasks with scarce training data. The training of a meta-learner is not performed on a specific task, and instead, on a distribution of similar tasks.

¹The work in this chapter has been published in the paper :Chi Zhang, Guosheng Lin, Fayao Liu, Jiushuang Guo, Qingyao Wu, Rui Yao. “Pyramid Graph Networks with Connection Attentions for Region-Based One-Shot Semantic Segmentation.” Proceedings of the IEEE International Conference on Computer Vision (ICCV). Seoul, Korea. 2019. [144]

FIGURE 4.1: Illustration of the proposed Pyramid Graph Networks for solving one-shot image segmentation. Given only one training image, our model could perform segmentation on new test images. The proposed Graph Attention mechanism could find the correspondence between image regions.

A well-studied area of meta-learning is one-shot image classification which aims to classify a test image given only one training image in each class. One-shot segmentation further extends this task to pixel levels. With only one pixel-wise labeled training image, the goal is to predict a binary mask in the testing images. Unlike one-shot image classification tasks where each data point has a label, in segmentation, the data is represented with a group of structured pixels. This yields a many-to-many message passing problem that we do not know which pixels in two images are related.

To solve the problems mentioned above, previous methods transform it to a one-to-many problem, using global pooling operations to squeeze structured support data to a global descriptor and exploit it to guide dense predictions [4, 93–96]. In this case, the prediction of each pixel location references the same guidance vector. For example, in [94–96], the network structure encourages the foreground pixels in the query image to have a close distance to the global descriptor, via cosine distance metric or a learnable distance metric. However, two grouped pixels may only show connections in a small fraction of the whole representations. It is common that a part of the objects shown in the support set may not appear in the query set. In this case, the global average descriptor may introduce noise, as we only want the shared components to provide guidance information.

In this work, we argue that the query set and the support set can be modeled with graphs and the process of propagating label information from a structured representation to another one can be viewed from the graph reasoning standpoint. We combine the query graph and support graph into a big bipartite graph that we assume each node in the query graph is fully connected with the nodes in the support graph. To reason the underlying correlations between the unlabeled query nodes and labeled support nodes, we apply graph attention mechanism [145] to weight the connections of each neighbouring node. With the attention weights, each query node could selectively aggregate label information from the support graph and thus help the downstream node classification task. This could also provide some interpretability by inspecting the attention distribution. An illustration of our attention mechanism is shown in Fig. 4.1.

The next question is how we should model the images with graph representations. Given two locations in the query image and the support image, they may show connections at different levels. For example, the connection of two human images can be an elementary feature, *e.g.*, eyes, which corresponds to a small region, but it can also be a more abstract feature, *e.g.*, faces, constituted by a group of elementary features. With the intuition above, we propose a pyramid-like branched graph reasoning structure to establish such multi-level connections. The graph nodes in different branches capture features of sub-regions at different scales. At the bottom level, each pixel in the feature map is modeled as the node in the graph. The rest branches model different sizes of sub-regions in the image as the graph nodes via adaptive pooling. Adaptive pooling could downsample arbitrary sized feature maps to a set of fixed-sized grids. Each of the branches performs graph reasoning independently, and their results are projected back to the original size. Finally, we fuse all branches for the final prediction. The pyramid structure enables graph reasoning at different scales to help the node classification.

In our experiment, We also explore various model variants that adopt other techniques for capturing multi-level relations, *e.g.*, dilated convolutions. We empirically demonstrate that our graph-based method achieves better results than the baseline methods based solely on global guidance. Our contributions are summarized as follows:

- We propose Pyramid Graph Networks (PGNet) for one-shot image segmentation which establishes correspondence between object parts by attention

mechanism. Compared with previous work based on global information, the proposed graphical model allows each pixel prediction to reference more related area in the support set.

- We propose a pyramid architecture that constructs multi-level region-based connections between two images to effectively propagate label information from the support set to the query set.
- The proposed attention mechanism also provides interpretability of the relations by visualizing the node connection weights.
- Experiments on PASCAL VOC 2012 dataset show that our method significantly outperforms the baseline model and achieves new state-of-the-art 1-shot and 5-shot segmentation results.

4.2 Method

In this section, we present our PGNet with the motivation of establishing the element-to-element correspondence between structured data. The network is constructed with a few primary building blocks, *i.e.*, the Graph Attention Unit (GAU). In what follows, we begin with the illustration of our Graph Attention Unit. Then we introduce our pyramid graph reasoning module that models different sizes of sub-regions as the graph nodes. The overview of our network is shown in Fig. 4.2.

4.2.1 Graph Attention Unit

Given a query image x_q and a labeled support image x_s , we first employ a shared convolutional neural network to convert them into feature maps where each pixel location is represented by a vector. A feature map can be modeled with a graph that each node corresponds to a feature vector in the original feature maps. Our goal is to establish relations between two graph domains to propagate label information from the support images to unlabeled query images for classification.

Our Graph Attention Unit (GAU) takes inspiration from Graph Attention Network [145], where the model learns a weight coefficient for each neighbour and reconstructs each node as the weighted sum of neighbouring nodes for the downstream graph mining tasks. Our Graph Attention Unit shares the same spirit and

FIGURE 4.2: Illustration of our network on 1-shot image segmentation task. Given a sampled image pair, we first use a shared CNN to extract their features, then we model the query feature maps and the foreground region in support feature maps with graphs. After that, a set of paralleled adaptive pooling layers are applied to the query features to acquire different sub-region representations, which are sent to different Graph Attention Units for graph reasoning. Finally, the outputs of different branches are fused by addition and further processed by the rest convolutions. ASPP denotes the Atrous Spatial Pyramid Pooling module. The GAU branch with equal attention weights has the same effect with global pooling.

combines the query graph and the support graph into a big bipartite graph that each query node is fully connected to all support nodes. After that, we reconstruct each query node by fusing all neighbouring support nodes with attention mechanism. Then fused nodes, along with their original node representations, are used for node classification. The architecture of the Graph Attention Unit is shown in Fig. 4.3. We first elaborate the node updating process with one example query node and such operations can be applied to all query nodes in parallel.

Assume we have a query node representation $\vec{h}_q$ and all support node representations $\{\vec{h}_s^1, \vec{h}_s^2, \dots, \vec{h}_s^N\}$, where $\vec{h} \in \mathbb{R}^C$, N is number of all nodes in the support graph and C is the feature dimension, the Graph Attention Unit updates the query node by selectively accumulating information from all support nodes. To that end, we need to find a pairwise function $f(\cdot)$ that generates a scalar e_j as the correlation score between the query node $\vec{h}_q$ and a support node $\vec{h}_s^j$

$$e_j = f(\theta(\vec{h}_q), \phi(\vec{h}_s^j)) \quad (4.1)$$

where ϕ and θ are linear transformation functions, which project the node features to a new space. Here, we experiment two choices of function $f(\cdot)$ that are commonly used for vector comparison.

FIGURE 4.3: Illustration of our proposed Graph Attention Unit with one example node in the query graph.

Linear Transformation. The original formulation in Graph Attention Network [145] for computing the correlation score is to concatenate two vectors and apply linear transformation with the weight vector $\vec{w}_f$:

$$f(\phi(\vec{h}_q), \theta(\vec{h}_s^j)) = \vec{w}_f^T(\phi(\vec{h}_q) || \theta(\vec{h}_s^j)) \quad (4.2)$$

where $||$ denotes the concatenation operation.

Inner product. We also attempt using the dot products of two vectors to compute the scalar:

$$f(\phi(\vec{h}_q), \theta(\vec{h}_s^j)) = \phi(\vec{h}_q)^T \theta(\vec{h}_s^j) \quad (4.3)$$

Once we obtain the correlation factor e_j of all neighbouring support nodes, we normalize them with the softmax function and generate the weights a_j . Based on that, we fuse all support node representations with the weighted sum. Finally, the fused node is concatenated with the original input query node vector, and they are

fused with another linear transformation function:

$$a_j = \frac{\exp(e_j)}{\sum_{k=1}^N \exp(e_k)}, \quad (4.4)$$

$$\vec{v}_q = \sum_{j=1}^N a_j g(\vec{h}_s^j), \quad (4.5)$$

$$\vec{h}'_q = \psi(\vec{v}_q || g(\vec{h}_q)) \quad (4.6)$$

where $g(\cdot)$ and $\psi(\cdot)$ are linear projection functions, followed by ReLU.

Parallel Computation. At implement time, all linear transformation operations, *i.e.*, $g(\cdot)$, $\psi(\cdot)$, $\theta(\cdot)$ and $\phi(\cdot)$ can be operated on all nodes concurrently with 1×1 convolutions due to its grid arrangement property. For the pair-wise operation $f(\cdot)$, as we only model the foreground region in the support image as the support graph, we could apply $f(\cdot)$ to all positions in the support set in the first place and then mask the attention values that correspond to the background with $-\infty$ before softmax normalization. When dot product is used in $f(\cdot)$, we could use matrix multiplication to compute attention value in parallel, and when linear transformation is used, we can apply tensor broadcasting and 1×1 convolution in the channel dimension for efficient computation.

4.2.2 Pyramid Graph Reasoning

We have presented the GAU by modeling each pixel as the node vector to perform graph reasoning. However, such connections built on pixel-wise elementary features may not be sufficient to discover the ideal relations. For example, an eye-like feature may falsely establish relations between humans' eyes and dogs' eyes. We may want an abstract feature, *e.g.*, faces being modeled into the graph for better reasoning.

We find that a complex object can be decomposed to a set of basic elements. For instance, faces can roughly be composed of eyes, noses, and mouths. To acquire an unseen high-level representation, we could simply mix the elementary representations. With this intuition, we propose a multi-level graph reasoning scheme that models different sizes of sub-regions as the graph nodes to undertake graph reasoning at different levels. We have demonstrated the bottom level, where each pixel is modeled as the node in the graph. The other two branches in the pyramid structure

have similar structures but adopt different sizes of adaptive pooling to the query feature maps before sending them to the GAU. Then, the fused graphs of each Graph Attention Unit are upsampled back to the original feature size with bilinear interpolation, as shown in Fig. 4.2. Adaptive pooling can extract representations of assigned size given an arbitrary sized feature map. All pixel locations inside a sub-region together constitute a feature representation which is then modeled as a graph node.

We additionally add one GAU that sets equal attention values e_j . In this case, each query node is fused with the average support nodes, which aims to incorporate the global statistics from the support set. The output maps of different branches are fused by addition and processed by three residual convolution blocks [124]. Finally, we add the Atrous Spatial Pyramid Pooling Module (ASPP) [85] at the end to generate the final results.

4.3 Experiment

FIGURE 4.4: Qualitative results of our network. The first row is the query images. The second row is our network predictions of the query images. The third row is support images with ground-truth annotations. The fourth row is the corresponding attention maps of selected regions (marked with red rectangles in the query images). The grids in the query image indicate which branch we extract the attention maps from. The attention values are normalized to $[0,1]$ to highlight the salient region in the support images.

4.3.1 Implementation details.

Our network structure is modified from DeepLab V3 with ResNet-50 as the backbone. Specifically, we decompose the original DeepLab V3 network to a fully convolutional ResNet part and an Atrous Spatial Pyramid Pooling(ASPP) module as the post-processing part. The Resnet backbone, pre-trained on ImageNet, is used as the feature extractor in our network and the ASPP is added at the end. We remove layers in *block-4* and concatenate features of *block-3* and *block-2* as the extracted features, which are sent to different branches. All the convolutional operations inside the GAU, residual blocks and ASPP generate features of 256 channels. The linear projection functions $g(\cdot)$ in different branches share the same parameter. Finally, the model outputs two-channel masks as the predicted foreground and background scores of each location.

At training time, we optimize the network parameter by minimizing the two-class Cross-Entropy loss over all pixel locations with momentum SGD. The network is trained for 600,000 iterations with the learning rate of 0.0025. We adopt random crop, random scale and random horizontal flip on the support images during training for data augmentation.

4.3.2 Dataset and Evaluation Metric

We evaluate the performance of our algorithm on the PASCAL VOC 2012 dataset with extended annotations in [141]. We follow the dataset division in [93] that 20 object classes in their official resealed order are evenly divided into 4 folds and report cross-validation results. Namely, 15 object categories are used as training tasks with the rest as the testing tasks. At test time, we random sample 1,000 tasks in each test fold. For more details about the dataset, please refer to [93].

We align the evaluation metric with previous works. Given predicted masks from 1000 test episodes, we first calculate a standard foreground Intersection over Union (IoU) score for each object class, then we average the class-wise IoU of all 5 classes as the mean IoU for this fold. When compared with the state-of-the-art results, we report the mean IoU in each test fold and the mean results over 4 folds.

Model	Backbone	Mean IoU (%)
PGNet-Dot	VGG-16	57.2
PGNet-Dot	ResNet-50	59.4

TABLE 4.1: Our model with different backbone networks. PGNet-Dot denotes our model with dot-product as the pair-wise function $f(\cdot)$. Our network with ResNet-50 backbone achieves better results.

Model	Mean IoU (%)
PGNet-Linear	58.1
PGNet-Dot	59.4

TABLE 4.2: Our network with dot-product and linear transformation as the pair-wise function $f(\cdot)$. The dot-product version achieves higher mean IoU score.

4.3.3 Ablation Analysis

We conduct extensive ablation experiments to validate our network design. Each analysis experiment is performed twice with fold 0 and fold 1 as the test fold, respectively. We report the average performance of the two test folds for all the experiments in this section.

Backbone network. We first evaluate the importance of the backbone model in our architecture. We experiment with two backbone models that are used in previous work: VGG-16 and ResNet-50. We fix the pair-wise function $f(\cdot)$ with dot product. The results of our network with the two backbone models are presented in Table.4.1. As is shown, our model with ResNet-50 yields slightly better performance than the VGG version. We adopt ResNet-50 as the backbone model in all following experiments.

The Pair-wise Function. Table. 4.2 compares our model variants with dot-product and linear transformation as the pair-wise function $f(\cdot)$, which are denoted with **PGNet-Dot** and **PGNet-Linear**, respectively. As is shown, our model with dot-product as the function $f(\cdot)$ yields better results.

Adaptive Pooling vs. Dilated Convolution. Dilated convolution [85] is another common operation to extract information from larger ranges without introducing extra parameters. The receptive field of a filter can be explicitly controlled by varying the dilation rates of convolutional kernels. We experiment a series of

Method	Operated On	Mean IoU (%)
Baseline-Ensemble	-	57.5
Dilated Conv	Support	58.1
Dilated Conv	Query	57.7
Adaptive Max Pool	Support	58.6
Adaptive Max Pool	Query	57.6
Adaptive Avg Pool	Support	57.6
Adaptive Avg Pool	Query	59.4

TABLE 4.3: Comparison of using dilated convolution or adaptive pooling to undertake multi-range graph reasoning. Both two methods could boost performance over the baseline model. Adopting adaptive average pooling on the query feature map yields the best result.

model variant by replacing the adaptive pooling operations with dilated convolutions. To build a dilated-convolution version of our network, we make the following modifications to the original structure: 1) We remove the adaptive pooling operations to the input query feature maps such that all GAUs have the same query input. 2) The query node encoding function $\theta(\cdot)$ in the Graph Attention Unit is replaced with 3×3 dilated convolutions to incorporate information from different ranges. With the aim to capture information from different sized regions, the dilation rates in different Graph Attention Unit are set with 2, 4 and 8 separately, while remaining other network components still.

We also experiment adopting these operations on the support feature maps. Specifically, if adaptive pooling is employed in our network, the query feature maps to different GAUs are kept same and apply adaptive pooling operations on the support feature maps, and if dilated convolutions are used, we simply move the changes mentioned above to the support node encoding function $\phi(\cdot)$. The comparison of adaptive max pooling and adaptive average pooling is also investigated in this part. Our baseline method in this experiment is a branch-ensemble model that neither dilated convolution nor adaptive pooling is used, such that all the branches share the same structure and are all applied on the original query and support feature maps. The comparison of the model variants is shown in Table.4.3. We could find from the result that both the dilated convolution and the adaptive pooling could introduce multi-range information and boost the performance over the baseline result. The optimal result is achieved when adaptive average pooling is employed on query feature maps.

Compared with Global Guidance. As discussed earlier, previous works transform the many-to-many problem to a one-to-many problem that a global description vector from the support set guides the pixel-wise predictions in the query image. Here, we implement several baseline models that adopt the designs in previous works. Our first baseline model, **PGNet-Mask-RGB**, adopts the solutions in [4, 93, 96] that we mask the background region in the support images with zero and perform global average pooling over the support features to generate a global vector. Then, this vector is upsampled to the same spatial size of the query feature maps and they are fused by concatenation. The second baseline model, shares the same spirit with minor differences. In [94, 95] they maintain the original RGB support image as the input and extract the global vector by averaging the support features over the foreground mask region. We denote this baseline method with **PGNet-Mask-Feature**. We replace our pyramid graph reasoning module with the two solutions above while maintaining other network components still to validate our design. The above methods can be seen as a special case of our GAU when the attention values are set equal, such that each query node is fused with an averaging support node.

Moreover, we also experiment branch ensembles of the baseline methods to investigate whether our network performance is driven by introducing more parameters. Similarly, We also construct 4 parallel branches, all of which have identical structures with different parameters, and their results are fused by addition, as done in our pyramid structure. The results are shown in Table.4.4. As is shown, although branch ensembles of baseline methods could slightly boost the baseline performance, our network still outperforms all the baseline methods with a large margin. The attentive graph reasoning method turns out to be more effective in extracting guidance information from the support set than methods solely based on the global vector.

Multi-scale Input Test. As is commonly done in the segmentation literature, we test our network performance with multi-scale query and support image inputs and average their predictions. Specifically, the images are rescaled with the ratio of [0.7, 1, 1.3] and their corresponding predictions are rescaled back to the original size with bilinear interpolation. The effect of multi-scale input test is shown in Table. 4.5.

Model	Ensemble	Mean IoU (%)
PGNet-Mask-RGB		54.7
PGNet-Mask-RGB	✓	54.8
PGNet-Mask-Feature		56.7
PGNet-Mask-Feature	✓	57.1
PGNet-Dot		59.4

TABLE 4.4: Comparison of our graph-based network with model variants based on global vector guidance. Our proposed method achieves better results than the baseline methods and their ensemble versions.

Model	Query	Support	Mean IoU (%)
PGNet-Dot			59.4
PGNet-Dot	✓		61.2
PGNet-Dot		✓	59.5
PGNet-Dot	✓	✓	61.5

TABLE 4.5: Influence of multi-scale input test. All predictions are rescaled to the original image size and fused by average.

4.3.4 Qualitative Results

Fig.4.4 shows some qualitative results of our models. We can see that our network could accurately make predictions of the query images with only one labeled training image. Given sub-regions of different sizes in the query images, we visualize the attention weights with respect to all support locations. Our graph attention mechanism could find the most related areas in the support image to help prediction.

4.3.5 Comparison with the State-of-the-art Results

We compare our final model with the state-of-the-art methods on PASCAL VOC 2012 dataset. We report our results under two different experimental set-ups where the difference is the evaluation metric adopted: The first evaluation metric is the one we have explained at the beginning of this section. The second evaluation metric is the one proposed in [4]. They ignore the object classes and report the mean of foreground IoU and background IoU over all test images in the fold. We denote them with Mean IoU and IoU, respectively, to differentiate. The 1-shot results under two evaluation metrics are shown in Table. 4.6 and Table. 4.7, respectively.

Model	fold-0	fold-1	fold-2	fold-3	mean
Reduced-DFCN8s [146]	39.2	48.0	39.3	34.2	40.2
OSLSM [93]	33.6	55.3	40.9	33.5	40.8
co-FCN [4]	36.7	50.6	44.9	32.4	41.1
SG-One [94]	40.2	58.4	48.4	38.4	46.3
CANet [95]	52.5	65.9	51.3	51.9	55.4
Ours	56.0	66.9	50.6	50.4	56.0

TABLE 4.6: Comparison with the state-of-the-art 1-shot segmentation performance on PASCAL VOC 2012 dataset.

We can see from the tables that under both experiment set-ups, our network outperforms previous methods and achieves a new state-of-the-art performance.

5-shot Experiments. As the proposed Graph Attention Unit dynamically generates the weights between the query nodes and all support nodes, we can extend our model to solve the k -shot learning task easily by modeling all support images into the support graph. Specifically, the foreground regions from different support images together constitute the support graph so that the attention distribution is over all foreground locations in the support set. Thus, our algorithm is able to handle 1-shot and k -shot cases with the same model. In comparison, most previous works can only handle one-shot cases and adopt naive fusion methods to fuse individual 1-shot results. For example, [4, 94, 96] average the global descriptors generated by different support images. Zhang *et al.* [95] trains an additional branch to weight the k global descriptors before averaging them. Shaban *et al.* [93] adopts logic OR operation to fuse individual predicted binary masks.

We report the 5-shot segmentation results for a fair comparison with previous works. The results are shown in Table. 4.8. **PGNet-Fusion** denotes the baseline method that we use the one-shot model to make predictions with each support image and average the 5 predicted masks. **PGNet-Graph** denotes the proposed method that models all support images into the graph. **PGNet-Graph-MS** denotes our proposed model with multi-scale input test operated on the query images. We can see from the results that our graph-based 5-shot learning method is more effective than naive fusion solutions and our final 5-shot result significantly outperforms the state-of-the-art performance under two evaluation metrics.

Model	IoU
co-FCN [4]	60.1
Reduced-DFCN8s [146]	60.9
PL [96]	61.2
A-MCG-Conv-LSTM [97]	61.2
OSLSM [93]	61.3
SG-One [94]	63.1
CANet [95]	66.2
Ours	69.9

TABLE 4.7: Comparison with the state-of-the-art 1-shot segmentation results on PASCAL VOC 2012 dataset, regarding the evaluation metric proposed in [4]

Model (5-shot)	fold-0	fold-1	fold-2	fold-3	mean
co-FCN [4]	37.5	50.0	44.1	33.9	41.4
OSLSM [93]	35.9	58.1	42.7	39.1	43.9
Reduced-DFCN8s [146]	45.3	51.4	44.9	39.5	45.3
SG-One [94]	41.9	58.6	48.6	39.4	47.1
CANet [95]	55.5	67.8	51.9	53.2	57.1
PGNet-Fusion	53.6	65.6	49.9	49.9	54.8
PGNet-Graph	54.9	67.4	51.8	53.0	56.8
PGNet-Graph-MS	57.7	68.7	52.9	54.6	58.5

(A) Evaluation metric of mean class-wise IoU adopted in [93]

Model (5-shot)	IoU
co-FCN [4]	60.2
OSLSM [93]	61.5
A-MCG-Conv-LSTM [97]	62.2
PL [96]	62.3
SG-One [94]	65.9
Reduced-DFCN8s [146]	66.0
CANet [95]	69.6
PGNet-Graph-MS	70.5

(B) Evaluation metric of IoU adopted in [4]

TABLE 4.8: Comparison with the state-of-the-art 5-shot segmentation performance on PASCAL VOC 2012 dataset. Our network outperforms previous methods under both evaluation metrics.

4.4 Conclusions

We have presented Pyramid Graph Networks for one-shot image segmentation. Compared with previous methods solely based on a global supporting vector, our attentive graphical models establish connections between elements across structure represented data that allow each unlabeled pixel to selectively aggregate guidance

information from support image regions. Our pyramid structure models various sizes of regions as the graph nodes to enable graph reasoning at different scale and semantic levels. Experiments on PASVAL VOC 2012 datasets under two evaluation metrics show that our proposed method significantly outperforms the baseline methods and achieves a new state-of-the-art performance.

Chapter 5

DeepEMD: Few-Shot Image Classification with Differentiable Earth Mover’s Distance and Structured Classifiers¹

5.1 Introduction

Deep neural networks have achieved great success in many vision tasks, typically requiring a large amount of labeled data. A notorious drawback of deep learning methods is that they suffer from poor sample efficiency. This is in sharp contrast to how we humans learn. In machine learning, few-shot learning is the task that addresses this issue, which is often solved as a special case of the broader meta learning. Meta learning attempts to learn a model that can generalize to new tasks with minimum adaption effort. One of the most well-studied test-beds for meta-learning algorithms is few-shot image classification, which aims to perform classification on new image categories with only a limited amount of labeled training data. This is the focus of our work here.

¹The work in this chapter has been published in the paper : Chi Zhang, Yujun Cai, Guosheng Lin, Chunhua Shen. “DeepEMD: Few-Shot Image Classification with Differentiable Earth Mover’s Distance and Structured Classifiers.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (**CVPR, Oral Presentation**). Virtual Conference. 2020. [147]

FIGURE 5.1: Illustration of using the Earth Mover’s Distance for one-shot image classification. Our method uses an optimal matching cost between image regions to represent the image distance more faithfully.

To tackle this problem, a line of previous work in literature adopts metric-based methods [1, 46–50] that learn to represent image data in an appropriate feature space and use a distance function to predict image labels. Following the formulation of the standard image classification networks [123, 124], metric-based methods often employ a convolution neural network to learn image feature representations and replace the fully connected layer with a distance function, *e.g.*, the *cosine* distance and Euclidean distance. Such distance functions directly compute the distances between the embeddings of the test images and training images for classification, which bypasses the difficult optimization problem in learning a classifier for the few-shot setting. The network is usually trained by sampling from a distribution of tasks, in the hope of acquiring a good generalization ability to unseen but similar tasks.

Although these methods have achieved some degree of success, we observe that the cluttered background and large intra-class appearance variations may drive the image-level embeddings from the same category far apart in a given metric space. This issue can be largely alleviated by deep neural networks under the setting of

fully supervised learning, thanks to the large capacity of deep models and sufficient training images. However, it is almost inevitably amplified in low-data regimes and thus adversely impacts the image classification accuracy. Moreover, a mixed global representation would struggle to well capture image structures and is likely to lose useful local feature characteristics. Local features can provide discriminative and transferable information across categories, which can be important cues for image classification in the few-shot scenario. Therefore, a desirable metric-based algorithm should have the ability to exploit the local discriminative information and minimize the distraction caused by irrelevant regions.

A natural approach to determine the similarity of two complex structured representations is to compare their building blocks. The difficulty lies in that we do not have their correspondence supervision for training and not all building elements can always find their counterparts in the other structures. To solve the problems above, in this chapter, we formalize the few-shot classification as an instance of optimal matching, and we propose to use the optimal matching cost between two structures to represent their dissimilarity. Given the local feature representation sets generated by two images, we use the Earth Mover’s Distance (EMD) [98] to compute their structural similarity. The EMD is a metric for computing distance between structured representations, which was originally proposed for image retrieval. Given the distance between all element pairs, the EMD can acquire the optimal matching flows between two structures that have the minimum overall distance. It can also be interpreted as the minimum cost to reconstruct a structured representation against the other one. An illustration of our motivation is shown in Fig. 5.1. The EMD has the formulation of the transportation problem [148] and the global minimum can be attained by solving a Linear Programming problem. To embed the optimization problem into the model for end-to-end training, we apply the implicit function theorem [107, 149, 150] to form the Jacobian matrix of the optimal optimization variable with respect to the problem parameters [107]. We explore multiple ways to extract local representations from an image, including fully convolutional networks, image grids, and image region sampling. We also investigate pyramid structures at both the feature level and the image level to capture local representations at different scales.

An important problem-specific parameter in the EMD formulation is the weight of each element. Elements with large weights generate more matching flows and thus

contribute more to the overall distance. Ideally, the algorithm should accommodate the flexibility to assign less weight to irrelevant regions such that they contribute less to the overall distance no matter which elements they match with. To achieve this goal, we propose a cross-reference mechanism to determine the importance of the elements. In the proposed cross-reference mechanism, we determine the weight of each node by comparing it with the global statistics of the other structure. Intuitively, the image region that shows greater relevance to the other image is more likely to be the object region and should be assigned with a larger weight, while the weights of high-variance background regions and the object parts that are not co-occurrent in two images should be eliminated as much as possible when computing the matching cost.

In the k -shot setting where multiple support images are presented, we propose to learn a structured fully connected (FC) layer as the classifier for classification to make use of the increasing number of training images. The structured FC layer includes a group of learnable vectors for each class. At inference time, we use the EMD to compute the distance between the image embeddings and the learnable vector set in each class for classification. The structured FC is an extension of the standard fully connected layer in that it replaces dot product operations between vectors with the EMD function between vector sets such that the structured FC layer can directly classify feature maps. The structured FC layer can also be interpreted as learning the prototype embeddings generated by a dummy image for each category such that the test images can be matched with each of them for classification.

To validate our algorithm, we conduct extensive experiments on multiple datasets to demonstrate the effectiveness of our algorithm. Our main contributions are summarized as follows:

- We propose to formalize the few-shot image classification as an optimal matching problem and employ the Earth Mover’s Distance as the distance metric between structured representations. The EMD layer can be embedded into the network for end-to-end training.
- We propose a cross-reference mechanism to generate the weights of elements in the EMD formulation, which can effectively reduce the noise introduced by the irrelevant background regions in images.

- We propose to learn a structured fully connected layer in the k -shot settings, which is able to directly classify the structured representations of an image using the Earth Mover’s Distance.
- Experiments on five popular few-shot classification benchmark datasets—miniImagnet, tieredImagnet, FC100, CUB, and CIFAR-FS show that our algorithm on both 1-shot and 5-shot classification tasks significantly outperforms the baseline methods and achieves new state-of-the-art performance on all of them.

5.2 Preliminary

Before presenting our algorithm in detail, we first introduce some preliminary concepts in the few-shot classification literature. The general meta-learning algorithm aims to learn transferable knowledge across tasks, where knowledge learned on training tasks can be used to solve novel tasks with only a small amount of training data. In the few-shot classification scenario, a task $\mathcal{T}_i$ is to undertake classification over a set of sampled classes, characterized by scarce training images. Specifically, an N -way K -shot task denotes classification over N classes with K training samples in each class. To acquire generalization ability across tasks, the training and testing of the model are often aligned with the episodic paradigm [46] where batched tasks are sampled for training or evaluation. For each sampled task, the training set $\mathcal{S} = \{(x_1^s, y_1^s), \dots, (x_{NK}^s, y_{NK}^s)\}$ is called the *support set* and the testing set $\mathcal{Q} = \{(x_1^q, y_1^q), \dots, (x_{NK_Q}^q, y_{NK_Q}^q)\}$ is called the *query set*, where x_i is an image, y_i is its corresponding label, $y_i \in \{1, \dots, N\}$, and K_Q is the number of testing images per class. At training time, the ground truth label of the query sets provides learning supervision, and at inference time, we repeatedly sample tasks for evaluation and record their mean accuracy.

5.3 Our Method

In this section, we first present a brief review of the Earth Mover’s Distance and describe how we formulate the one-shot classification as an optimal matching problem that can be trained end-to-end. Then, we describe our cross-reference mechanism

FIGURE 5.2: Our framework for 1-shot image classification. Given a pair of images, we first extract their local embeddings which are two sets of feature vectors. Then we use the Earth Mover’s Distance to generate the optimal matching flows between two sets, which have the minimum overall matching cost. Finally, based on the optimal matching flows and matching costs, we can compute the distance between two images, which is used for classification. We explore three methods to extract local embeddings: **(a)** fully convolutional networks, **(b)** cropping image patches based on grids, and **(c)** random sampling of image patches. The details of the three methods are provided in Section 5.3.2.

to generate the weight of each node, which is an important parameter in the EMD formulation. Finally, we demonstrate how to use the EMD to tackle k -shot learning with our proposed structured fully connected layer. The overview of our framework for one-shot classification is shown in Fig. 5.2.

5.3.1 Revisiting the Earth Mover’s Distance

The Earth Mover’s Distance is a distance measure between two sets of weighted objects or distributions, which is built upon the basic distance between individual objects and the weights of each element. It has the form of the well-studied transportation problem from Linear Programming. Specifically, suppose that a set of sources or suppliers $\mathcal{S} = \{s_i \mid i = 1, 2, \dots, m\}$ are required to transport goods to a set of destinations or demanders $\mathcal{D} = \{d_j \mid j = 1, 2, \dots, k\}$, where s_i denotes the supply units of supplier i and d_j represents the demand of j -th demander. The cost per

unit transported from supplier i to demander j is denoted by c_{ij} , and the number of units transported is denoted by x_{ij} . The goal of the transportation problem is then to find a least-expensive flow of goods $\tilde{\mathcal{X}} = \{\tilde{x}_{ij} \mid i = 1, \dots, m, j = 1, \dots, k\}$ from the suppliers to the demanders:

$$\begin{aligned}
 & \underset{x_{ij}}{\text{minimize}} && \sum_{i=1}^m \sum_{j=1}^k c_{ij} x_{ij} \\
 & \text{subject to} && x_{ij} \geq 0, i = 1, \dots, m, j = 1, \dots, k \\
 & && \sum_{j=1}^k x_{ij} = s_i, \quad i = 1, \dots, m \\
 & && \sum_{i=1}^m x_{ij} = d_j, \quad j = 1, \dots, k
 \end{aligned} \tag{5.1}$$

Note that roles of suppliers and demanders can be switched without affecting the total transportation cost. Here s_i and d_j are also called the weights of the nodes, which controls the total matching flows generated by each node. EMD seeks an optimal matching $\tilde{\mathcal{X}}$ between suppliers and demanders such that the overall matching cost can be minimized. The global optimal matching flows $\tilde{\mathcal{X}}$ can be achieved by solving a Linear Programming problem.

5.3.2 EMD for Few-Shot Classification

In the few-shot classification task, metric-based methods aim to find a good distance metric and data representations to compute the distance between images, which are used to compare images for classification. Different from the previous methods [1, 46] that perform distance computation between the image-level embeddings, our approach advocates the use of discriminative local information. The intuition is that as the goal of few-shot learning is to undertake the classification task on novel categories, directly generating a category-level embedding that corresponds to a new class is difficult. On the other hand, we can decompose an object into a set of object parts that may have been seen in the training process. For example, *wheel* can be a shared building element across vehicle categories, and if such representation is learned during training, it can be useful to classify unseen vehicle categories. Therefore, local discriminative representations are likely to provide more transferable information across categories. In our framework, we decompose images into a set of local representations, and by assigning appropriate weights to local embeddings in two images, we can use the optimal matching cost between

them to represent their dissimilarity. We explore three strategies to generate local representations from an image, as illustrated in Fig. 5.2:

1) **Fully Convolutional Networks.** We can deploy a fully convolutional network (FCN) [86] to generate the dense representation $\mathbf{U} \in \mathbb{R}^{H \times W \times C}$ of an image, where H and W denote the spatial size of the feature map and C is the feature dimension. Each image representation contains a collection of local feature vectors $\{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_{HW}\}$, and each vector $\mathbf{u}_i$ can be seen as a node in the set. Thus, the dissimilarity of two images can be represented as the optimal matching cost between two sets of vectors.

2) **Dividing the input image into grids.** We crop the image evenly into an $H \times W$ grid before sending it to the CNN, and each image patch in the grid cell is encoded by the CNN individually and generates a feature vector. The feature vectors generated by all the patches constitute the embedding set of an image.

3) **Random sampling of image patches.** Instead of generating the image patches by grids, here we randomly sample M patches in the images with different sizes and aspect ratios. The randomly sampled patches are then re-scaled to the same input size and are encoded by CNNs. The embeddings of these sampled patches make up the embedding set of an image.

We denote our networks adopting the three strategies above by *DeepEMD-FCN*, *DeepEMD-Grid*, and *DeepEMD-Sampling*, respectively. As the size of the grid in *DeepEMD-Grid* and the number of patches in *DeepEMD-Sampling* are hyper-parameters in our network, we conduct various experiments to investigate the influence of these parameters in Section 5.4.3. We also investigate pyramid structures on the image level and the feature level to capture local representations at multiple scales, illustrated in Fig. 5.3. Specifically, we add a feature pyramid structure to *DeepEMD-FCN* and the image pyramid structure to *DeepEMD-Grid*. The feature pyramid applies the RoI pooling to the feature maps generated by the FCN, and the resulting feature vectors together with the raw feature vectors constitute the embedding set. For the image pyramid, we simply crop the patches according to different grid sizes and send all patches to the CNN to generate the embedding set.

After acquiring the embedding sets of two images, we can follow the original EMD formulation in Equation (5.1) to compute the distance. Concretely, assuming there are $H \times W$ vectors in each set, the cost per unit is obtained by computing the

pairwise distance between embedding nodes $\mathbf{u}_i, \mathbf{v}_j$ from two image features:

$$c_{ij} = 1 - \frac{\mathbf{u}_i^T \mathbf{v}_j}{\|\mathbf{u}_i\| \|\mathbf{v}_j\|}, \quad (5.2)$$

where nodes with similar representations tend to generate small matching cost between each other. As to the generation of weights s_i and d_j , we leave the detailed elaborations in Section 5.3.4. Once acquiring the optimal matching flows $\tilde{\mathcal{X}}$, we can compute the similarity score s between image representations with:

$$s(\mathbf{U}, \mathbf{V}) = \sum_{i=1}^{HW} \sum_{j=1}^{HW} (1 - c_{ij}) \tilde{x}_{ij}. \quad (5.3)$$

5.3.3 End-to-End Training

In order to embed the optimal matching problem into a neural network for end-to-end training, it is important to make the solution of the optimal matching $\tilde{\mathcal{X}}$ differentiable with respect to the problem parameters θ . As is indicated by [107], we can apply the implicit function theorem [107, 149, 150] on the optimality (KKT) conditions to obtain the Jacobian. For the sake of completeness, we transform the optimization in Equation (5.1) to a compact matrix form:

$$\begin{aligned} & \underset{x}{\text{minimize}} && c(\theta)^T x \\ & \text{subject to} && G(\theta)x \leq h(\theta), \\ & && A(\theta)x = b(\theta). \end{aligned} \quad (5.4)$$

Here $x \in \mathbb{R}^n$ is our optimization variable, with $n = m \times k$ representing the total number of matching flows in $\mathcal{X}$. θ is the problem parameter that relates to the earlier layers in a differentiable way. $Ax = b$ represents the equality constraints and $Gx \leq h$ denotes the inequality constraint in Equation (5.1). Specifically, to construct the compact matrix form of the original optimization, we can build up

the sparse matrix below for the equality constraints:

$$\begin{bmatrix} 1 & \cdots & 1 & & & & \\ & & & 1 & \cdots & 1 & \\ & & & & & & 1 & \cdots & 1 \\ 1 & & & 1 & & & 1 & & \\ & \ddots & & & \ddots & & & \ddots & \\ & & 1 & & 1 & & & & 1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1k} \\ x_{21} \\ x_{22} \\ \vdots \\ x_{2k} \\ \vdots \\ x_{m1} \\ x_{m2} \\ x_{mk} \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \\ d_1 \\ d_2 \\ \vdots \\ d_k \end{bmatrix}, \quad (5.5)$$

and the inequality constraint can be written as:

$$\begin{bmatrix} -1 & & & & & \\ & -1 & & & & \\ & & \ddots & & & \\ & & & -1 & & \\ & & & & -1 & \\ & & & & & -1 \end{bmatrix} \begin{bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{1k} \\ x_{21} \\ x_{22} \\ \vdots \\ x_{2k} \\ \vdots \\ x_{m1} \\ x_{m2} \\ x_{mk} \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}. \quad (5.6)$$

Accordingly, the Lagrangian of the LP problem in Equation (5.4) is given by:

$$L(\theta, x, \nu, \lambda) = c^T x + \lambda^T (Gx - h) + \nu^T (Ax - b), \quad (5.7)$$

where ν denotes the dual variables on the equality constraints and $\lambda \geq 0$ denotes the dual variables on the inequality constraints.

Following the KKT conditions with notational convenience, we can obtain the optimum $(\tilde{x}, \tilde{\nu}, \tilde{\lambda})$ of the objective function by solving $g(\theta, \tilde{x}, \tilde{\nu}, \tilde{\lambda}) = 0$ with primal-dual interior point methods, where

$$g(\theta, x, \nu, \lambda) = \begin{bmatrix} \nabla_{\theta} L(\theta, x, \nu, \lambda) \\ \mathbf{diag}(\lambda)(G(\theta)x - h(\theta)) \\ A(\theta)x - b(\theta) \end{bmatrix}. \quad (5.8)$$

Then, the following theorem holds to help us derive the gradients of the LP parameters.

Theorem 1 (From Barratt [107]) Suppose $g(\theta, \tilde{\lambda}, \tilde{\nu}, \tilde{x}) = 0$. Then, when all derivatives exist, the partial Jacobian of $\tilde{x}$ with respect to θ at the optimal solution $(\tilde{\lambda}, \tilde{\nu}, \tilde{x})$, namely $J_{\theta}\tilde{x}$, can be obtained by satisfying:

$$J_{\theta}\tilde{x} = -J_x g(\theta, \tilde{\lambda}, \tilde{\nu}, \tilde{x})^{-1} J_{\theta} g(\theta, \tilde{x}, \tilde{\nu}, \tilde{\lambda}). \quad (5.9)$$

Here the formula for the Jacobian of the solution mapping is obtained by applying the implicit function theorem to the KKT conditions. For instance, the (partial) Jacobian with respect to θ can be defined as

$$J_{\theta} g(\theta, \tilde{\lambda}, \tilde{\nu}, \tilde{x}) = \begin{bmatrix} J_{\theta} \nabla_x L(\theta, \tilde{x}, \tilde{\nu}, \tilde{\lambda}) \\ \mathbf{diag}(\tilde{\lambda}) J_{\theta} (G(\theta)\tilde{x} - h(\theta)) \\ J_{\theta} (A(\theta)\tilde{x} - b(\theta)) \end{bmatrix}. \quad (5.10)$$

Therefore, once getting the optimal solution $\tilde{x}$ for the LP problem, we can obtain a closed-form expression for the gradient of $\tilde{x}$ with respect to the input LP parameters θ . This helps us achieve an efficient backpropagation through the entire optimization process without perturbation of the initialization and optimization trajectory.

5.3.4 Weight Generation

As can be observed in the EMD formulation, an important problem parameter is the weight of each node, *e.g.*, s_i , which controls the total matching flows $\sum_{j=1}^k x_{ij}$ from it. Intuitively, the node with a larger weight plays a more important role in the comparison of two sets, while a node with a very small weight can hardly influence the overall distance no matter which nodes it connects with. In the

FIGURE 5.3: Pyramid structures applied on DeepEMD-FCN and DeepEMD-Grid to extract local embeddings. The feature pyramid structure (a) adopts RoI poolings with different output sizes on the feature maps to generate local embeddings at multiple scales while the image pyramid structure (b) crops the input image into patches according to different grid sizes and all patches are sent to the CNN to generate local embeddings.

pioneering work that adopts EMD for color-based image retrieval [98], they use the histogram as the elementary feature and perform feature clustering over all pixels to generate the nodes. The weight of each node is set as the size of the corresponding cluster. It makes sense because, for color-based image retrieval, large weights should be assigned to the dominant colors with more pixels, such that the retrieved images can be visually close to the query images. However, for few-shot image classification tasks where features for classification often contain high-level semantic information, the number of pixels does not necessarily reflect the importance. It is common to find image data with greater background regions

FIGURE 5.4: Comparison of the standard fully connected layer (a) and our proposed structured fully connected layer (SFC) (b). The SFC learns a group of vectors as the prototype for each class such that we can use the EMD to generate category scores.

than the target objects in classification datasets, *e.g.*, ImageNet. Therefore, large weights should be given to the foreground object region in a matching algorithm. However, it may be difficult to define what is the foreground region, particularly when there exist multiple object categories in a single image. An object can be both foreground and background in different cases. Instead of determining the weights by inspecting individual images alone, we argue that for the few-shot classification

task, the co-occurrent regions in two images are more likely to be the foreground and the weights of node features should be generated by comparing the nodes on both sides. To achieve this goal, we propose a cross-reference mechanism that uses dot product between a node feature and the average node feature in the other structure to generate a relevance score as the weight value:

$$s_i = \max\left\{\mathbf{u}_i^T \cdot \frac{\sum_{j=1}^{HW} \mathbf{v}_j}{HW}, 0\right\}, \quad (5.11)$$

where $\mathbf{u}_i$ and $\mathbf{v}_j$ denote the vectors from two feature maps, and function $\max(\cdot)$ ensures the weights are always non-negative. Finally, we normalize all the weights in the structure to make both sides have the same total weights for matching:

$$\hat{s}_i = s_i \frac{HW}{\sum_{j=1}^{HW} s_j}. \quad (5.12)$$

For simplicity, here we take s_i as an example and d_i can be obtained in the same manner. The cross-reference mechanism aims to give less weight to the high-variance background regions and more to the co-occurrent object regions in two images. This can also put less weight on the object parts that do not co-occur in two images and thus allows partial matching to some extent. As a result, the proposed distance metric is based only on confident regions and confident features that have high responses.

5.3.5 Structured Fully Connected Layer

Thus far we have discussed using the Earth Mover’s Distance as the metric to generate the distance value between paired images, *i.e.*, the one-shot case. A question is then raised—how do we tackle the k -shot setting where multiple support images are available? Before presenting our design in detail, let us revisit how the standard fully connected layer classifies an image embedding extracted by CNNs. An FC layer, parameterized by $[\Phi_1, \dots, \Phi_N] \in \mathbb{R}^{C \times N}$ contains a set of learnable vectors $\Phi_i \in \mathbb{R}^C$ corresponding to each category. Given an image embedding $\mathbf{u} \in \mathbb{R}^C$ generated by the convolutional layer, the FC layer generates the score of class i by computing the dot product between the image vector $\mathbf{u}$ and the parameter vector Φ_i , and this process is applied to all the categories in parallel by matrix multiplication. There are also some previous works replacing the dot

product operation in the FC layer with the *cosine* function for computing the category scores [3, 151]. The learning of the FC layer can be seen as finding a prototype vector for each class such that we can use distance metrics to classify an image. An illustration of the standard FC layer is shown in Fig. 5.4 (a).

With the same formulation, we can learn a structured fully connected layer that adopts EMD as the distance function to directly classify a structured feature representation. The learnable parameter for each class becomes a group of vectors, rather than one vector, such that we can use the structural distance function EMD to undertake image classification. This can also be interpreted as learning a prototype feature map generated by a dummy image for each class. The comparison of the structured FC and the standard FC can be found in Fig. 5.4. Algorithm 1 provides the pseudo-code of a testing episode at inference time. We fix the network backbone and use SGD to learn the parameters in the structured fully connected layer by sampling data from the support set. After several iterations of optimization, we can generate the category scores by computing the EMD between the query images and each of the prototypes in the SFC.

Algorithm 1: A testing episode for an N -way K -shot task. N_{iter} is the number of optimization iterations.

Input: a trained feature extractor Θ , the support set $\mathcal{S}$, and the query set $\mathcal{Q}$

Output: Testing accuracy Acc of query set.

Initialize SFC layer: $\Phi = \Phi'$;

for i **from** 1 **to** N_{iter} **do**

Sample a mini-batch $\mathcal{B}_i$ from $\mathcal{S}$;
 make predictions for $\mathcal{B}_i$ with our model $[\Theta, \Phi]$;
 calculate cross entropy loss $\mathcal{L}_{\mathcal{T}}$;
 optimize Φ with SGD;

end

make predictions for images in the query set $\mathcal{Q}$ with the model $[\Theta, \Phi]$;

calculate accuracy Acc ;

return Acc .

5.4 Experiments

To evaluate the performance of our proposed algorithm for few-shot classification, we conduct extensive experiments on multiple datasets. In this section, we first present dataset information and some important implementation details in our

FIGURE 5.5: Experiment on 5-way k -shot classification. The proposed structured FC layer significantly outperforms baselines and previous k -shot solutions.

network design. Then, we conduct various ablative experiments to validate the effectiveness of each component in our network. Finally, we compare our model with the state-of-the-art methods on popular benchmark datasets.

5.4.1 Implementation Details

Network. For a fair comparison with previous works, we employ a 12-layer ResNet (ResNet12) as our model backbone, which is widely used in the few-shot classification literature. For DeepEMD-Grid and DeepEMD-Sampling, we remove the fully connected layer in ResNet, such that the network generates a vector for each input image patch. For DeepEMD-FCN, we further remove the global average pooling layer, such that the network is transformed into a fully convolutional network. Specifically, given an image of size 84×84 , the model generates a feature map of size $5 \times 5 \times 512$, *i.e.*, 25 feature vectors. For DeepEMD-Grid, we slightly enlarge

the region of the local patches in the grid by a factor of 2 to incorporate context information, which is found helpful to generate local representations.

Training. At training time, we use the GPU accelerated convex optimization solver QPTH [108] to solve the Linear Programming problem in our network and compute gradients for back-propagation. As is commonly implemented in the state-of-the-art literature [47], we adopt a feature pre-training step followed by episodic meta-training [46] to learn our network. At the pre-training stage, we train a standard classification model with all training classes and we use DeepEMD-FCN for validation with the validation set in the training process. After pre-training, the model with the highest validation accuracy is further optimized by episodic training for 5,000 episodes. In each training episode, we randomly sample a 5-way 1-shot task with 16 query images, which is aligned with the testing episodes. For the k -shot classification task, we re-use the trained 1-shot model as the network backbone to extract features and fix it during training and testing. We initialize the parameters in the structured FC layer with the average local representations of all support data in each class and sample a mini-batch of 5 images from the support set to finetune the structured FC layer for 100 iterations.

5.4.2 Dataset Description

We conduct few-shot classification experiments on five popular benchmark datasets, namely, *miniImageNet* [46], *tieredImageNet* [53], Fewshot-CIFAR100 (FC100) [48], Caltech-UCSD Birds-200-2011 (CUB) [152], and CIFAR-FewShot (CIFAR-FS) [153].

miniImageNet. *miniImageNet* was first proposed in [46] and becomes the most popular benchmark in the few-shot classification literature. It contains 100 classes with 600 images in each class, which are built upon the ImageNet dataset [129]. The 100 classes are divided into 64, 16, and 20 for meta-training, meta-validation, and meta-testing, respectively.

tieredImageNet. *tieredImageNet* is also a subset of ImageNet, which includes 608 classes from 34 super-classes. Compared with *miniImageNet*, the splits of meta-training(20), meta-validation(6), and meta-testing(8) are set according to the super-classes to enlarge the domain difference between training and testing phases.

Model	Embedding	Metric	5-way	10-way
ProtoNet [1]	global	Euclidean	60.37	44.34
MatchingNet [46]	global	<i>cosine</i>	63.08	47.09
FC [3]	global	<i>dot</i>	59.41	44.08
FC [3]	global	<i>cosine</i>	55.43	40.42
KNN [50]	local	<i>cosine</i>	62.52	47.08
Prediction Fusion [27]	local	<i>cosine</i>	62.38	47.04
DeepEMD-FCN (our)	local	EMD	65.91	49.66

TABLE 5.1: Comparison with different baseline methods for 1-shot classification. Our model with EMD as the distance metric significantly outperforms baseline models based on image-level representations and local representations.

The dataset also includes more images for training and evaluation (779,165 images in total).

Fewshot-CIFAR100. FC100 is a few-shot classification dataset build on CIFAR100 [154]. We follow the split division proposed in [48], where 36 super-classes were divided into 12 (including 60 classes), 4 (including 20 classes), and 4 (including 20 classes), for meta-training, meta-validation, and meta-testing, respectively, and each class contains 100 images.

CIFAR-FewShot. CIFAR-FS [153] is also a few-shot classification dataset built on CIFAR100 [154]. It contains 64, 15, and 20 classes for training, validation, and testing, respectively.

Caltech-UCSD Birds-200-2011. CUB was originally proposed for fine-grained bird classification, which contains 11,788 images from 200 classes. We follow the splits in [47] where 200 classes are divided into 100, 50, and 50 for meta-training, meta-validation, and meta-testing, respectively.

5.4.3 Analysis

In this section, we implement various experiments to evaluate the effectiveness of our algorithm. We also explore multiple design variants of our network and compare them with baseline solutions. All the experiments are conducted on the *miniImageNet* dataset.

Method	Operation	5-way	10-way
Full Connections	Average	55.16	40.88
Full Connections	CR	55.41	41.60
EMD	Equal	56.95	42.89
EMD	K-means ($k = 10$)	56.25	41.85
EMD	K-means ($k = 5$)	55.92	41.57
EMD	K-means ($k = 2$)	56.02	41.75
EMD	K-means ($k = 1$)	58.65	44.13
EMD	Mean-shift	53.56	39.70
EMD	CR	61.13	46.92

TABLE 5.2: Different methods for setting the weights in the EMD. We report the 1-shot performance with only the feature pre-training step. EMD with our cross-reference (**CR**) mechanism yields the best result. The model variant that is solely based on the cross-reference mechanism as attention and removes the EMD can cause a significant performance drop. The combination of the EMD and the cross-reference mechanism generates the best result.

Comparison with methods based on image-level representations. In the beginning, we first compare our method with a set of methods that utilize image-level vector representations on the 1-shot task. These methods maintain the global average pooling operation in ResNet to generate vector representations for images and use various distance metrics for classification. We select the representative methods in the literature for comparison: 1) Prototypical Network [1] with Euclidean distance. 2) Matching Network [46] with *cosine* distance. 3) Finetuning a FC classifier. In [3], Chen *et al.* propose to fix the pretrained feature extractor and finetune the FC layer with the support images. For fair comparisons, we adopt the same backbones and training schemes for all these baseline methods. We use FCN to extract local features in this experiment such that the only difference in the backbone is the lack of global average pooling operation in our model. The experiment result is shown in Table 5.1. As we can see, our algorithm significantly outperforms baseline methods that rely on image-level vector representations under both 1-shot 5-way and 1-shot 10-way settings, which indicates the effectiveness of the optimal matching based method that relies on local features.

Comparison with methods based on local representations. There are also a few methods in the literature focusing on local representations to solve few-shot classification. They all remove the global average pooling in the CNN to achieve dense representations of images. In [50], Li *et al.* use the top k nearest vectors (**KNN**) between two feature maps to represent the image-level distance.

Method	Embedding	5-way	10-way
DeepEMD-FCN	5×5	65.91	49.66
	$\mathbf{P}_{\text{feat}}(5, 3)$	66.27	49.85
	$\mathbf{P}_{\text{feat}}(5, 2)$	66.42	50.02
	$\mathbf{P}_{\text{feat}}(5, 2, 1)$	66.50	50.09
	6×6	63.61	48.72
DeepEMD-Grid	5×5	65.38	50.08
	4×4	66.57	51.42
	3×3	67.31	52.26
	2×2	66.09	50.94
	$\mathbf{P}_{\text{grid}}(3, 2)$	67.74	52.76
	$\mathbf{P}_{\text{grid}}(5, 3)$	67.01	51.89
	$\mathbf{P}_{\text{grid}}(5, 2)$	67.39	52.20
	$\mathbf{P}_{\text{grid}}(5, 3, 2)$	67.83	52.85
	9 patches	68.09	53.04
DeepEMD-Sampling	16 patches	68.54	53.61
	25 patches	68.77	53.83

TABLE 5.3: Comparison of different local embedding extractors described in Section 5.3.2 on 1-shot tasks. $\mathbf{P}_{\text{feat}}$ denotes the feature pyramid structure applied on DeepEMD-FCN and $\mathbf{P}_{\text{grid}}$ denotes the image pyramid structure applied on DeepEMD-Grid. The parameters in $\mathbf{P}_{\text{feat}}$ and $\mathbf{P}_{\text{grid}}$ are the RoI pooling size and the grid size, respectively. DeepEMD-Sampling outperforms the plain version of other two methods and both pyramid structures in feature level and image level can effectively boost the performance.

Lifchitz *et al.* [27] propose to make predictions with each local representation and average their output probabilities. We replace our EMD head with their methods for comparison. The result is shown in Table 5.1. Our optimal matching based algorithm outperforms all other model variants. Compared with other methods based on local features, the advantage of our method is that, although the basic ground distance in the EMD is based on local features, our algorithm compares the two structures in a global way. Predictions solely based on nearest local features in two images may not extract sufficient information to differentiate images. For example, *eyes* can be the nearest feature between animal images, but such a feature can hardly be used to differentiate animal species.

Weights in the EMD. We next investigate the influence of weights in the EMD. In the earlier work [98] using the EMD for image retrieval, they use the pixel color as the feature and cluster pixels to generate nodes. The weight of the node is set with the portion of pixels in this cluster. We examine two clustering algorithms to generate weights as the baseline models: K-means [158] and mean-shift [159]

Model	Rotation	Self-Supervision	5-way	10-way
DeepEMD-FCN	✓		65.91	49.66
			65.45	49.70
		✓	66.64	50.99
DeepEMD-Grid	✓		67.31	52.26
			66.96	52.09
		✓	67.87	53.08
DeepEMD-Sampling	✓		68.77	53.83
			67.98	53.42
		✓	69.17	54.52

TABLE 5.4: Model pre-training with self-supervised auxiliary tasks. The self-supervised auxiliary task at the pre-training stage can effectively boost the 1-shot performance of our models, while applying the random rotation as data augmentation alone degrades the performance. Please refer to Section 5.4.3 for analysis.

Model	1-shot	5-shot
ProtoNet [1]	50.01 $\pm$ 0.82	72.02 $\pm$ 0.67
MatchingNet [46]	51.65 $\pm$ 0.84	69.14 $\pm$ 0.72
<i>cosine</i> classifier [3]	44.17 $\pm$ 0.78	69.01 $\pm$ 0.74
<i>linear</i> classifier [3]	50.37 $\pm$ 0.79	73.30 $\pm$ 0.69
KNN [50]	50.84 $\pm$ 0.81	71.25 $\pm$ 0.69
DeepEMD-FCN	54.24 $\pm$ 0.86	78.86 $\pm$ 0.65

TABLE 5.5: Cross-domain experiments (*mini*Imagenet $\rightarrow$ CUB). We report the performance with 95% confidence intervals on 1-shot 5-way and 5-shot 5-way tasks. We use FCN to extract local features for KNN and our method. Our proposed algorithm outperforms baseline methods with a large margin.

which are implemented by the Scikit-learn [160] library. For K-means, we choose the number of clusters from $\{10, 5, 2, 1\}$. For mean-shift, as the bandwidth used in the RBF kernel can be automatically estimated in the Scikit-learn [160] library, the number of clusters is determined automatically. For both clustering algorithms, we use the cluster mean as the local embedding and the cluster size, *i.e.*, the number of feature vectors in each cluster, as the weight. As the clustering process of the aforementioned algorithms are non-differentiable, for a fair comparison, we fix the parameters in the backbone after pre-training to evaluate all methods. We also incorporate a baseline model with equal weights into comparison. To further test whether our performance is solely brought by the cross-reference mechanism, we also compare our network with a model variant that is solely based on the cross-reference mechanism without EMD. Concretely, we compute the *cosine* distance between all vector pairs, and compute a weighted sum of these distances with the node weights generated by the cross-reference mechanism. This baseline draws

Solver	Spatial Size	Dimension	Time
QPTH [108]	3×3	256	0.294 s
	3×3	512	0.294 s
	3×3	1024	0.295 s
	3×3	2048	0.295 s
	4×4	256	1.489 s
	4×4	512	1.493 s
	4×4	1024	1.498 s
	4×4	2048	1.499 s
	5×5	256	8.595 s
	5×5	512	8.638 s
	5×5	1024	8.638 s
	5×5	2048	8.650 s
OpenCV [155]	5×5	256	0.018 s
	5×5	512	0.019 s
	5×5	1024	0.021 s
	5×5	2048	0.025 s

TABLE 5.6: Computation time of the EMD layer in DeepEMD-FCN. In a 5-way 1-shot task with 10 query images, we vary the spatial size and the feature dimension of two input feature maps and record their computation time.

connections with CrossTransformers [57] and Cross Attention Networks [51], where spatial attentions based on local correspondence between samples are used to refine the data embeddings for better classification. Here we use the cross-reference mechanism to weight the distances between local embeddings, which plays a similar role with attentions. We use the FCN to extract local embeddings for all the methods in this experiment.

As we can see from the results in Table 5.2, our cross-reference mechanism can bring an improvement of 4.2% over the baseline with equal weights, while using the clustering-based methods to set the weights can hardly improve the performance, which indicates that the number of pixels does not necessarily correspond to the importance in few-shot classification. Noted that the optimal result of K-means is obtained when the number of clusters is 1. However, this means each image is represented by the global representation, *i.e.*, the mean of all local representations, and the EMD distance becomes the cosine distance and is no longer the structural distance. For the model variant solely based on the cross-reference mechanism as an attention, it can only slightly improve the result of simple average operation, while a combination of the cross-reference mechanism and the EMD can yield a

FIGURE 5.6: Visual reconstruction experiment by DeepEMD-FCN. Given two images (left and right), we plot the best matched patch (the one with the highest flow value) of each local region in the left image (middle). It can be seen as the reconstruction of the left image using patches from the right one. The weight controls the brightness of the corresponding region. Our algorithm can effectively establish semantic correspondence between local regions and assign small weights to the background regions.

significant performance improvement, which again validates the advantages of using the EMD as the metric and the effectiveness of the cross-reference mechanism.

Local embedding extractor. In Table 5.3, we compare the three methods to extract local embeddings of the input image described in Section 5.3.2. We also investigate some key parameters in respective methods, *e.g.* the size of grids in DeepEMD-Grid and the number of patches in DeepEMD-Sampling. From the experiment result, we have the following findings:

- 1) DeepEMD-Sampling shows distinct advantages over the plain versions of the other two methods. For DeepEMD-Sampling, increasing the number of sampled patches can consistently boost the performance, even if more low-quality patches, *e.g.* patches from the background region, are likely to be sampled. We observe the weights of these patches and find that they are usually assigned with small weights, thanks to our proposed cross-reference mechanism. Therefore, these noisy patches contribute less to the overall distance and increasing the number of sampled patches does not negatively influence the prediction.
- 2) Increasing the number of cells in DeepEMD-Grid results in small input image patches and the performance degrades. A possible explanation is that small image

FIGURE 5.7: Visualization of full matching flows in DeepEMD-FCN. Each grid cell in the middle contains the matched patches (from the right image) of the corresponding region in the left image. The brightness is controlled by the flow values and the weights. *Please zoom in for details.*

patches decompose the object in the image into small pieces, which may lose context information and raises the difficulty in generating high-level representations. 3) we speculate that the advantage of DeepEMD-Sampling in the performance over other two methods is gained by multi-scale information implicitly captured by random sampling, as the embedding set contains randomly sampled local descriptors that cover different sizes of local regions. Moreover, as all sampled patches are resized to the input size, an object may appear at an appropriate size in the input image, which can have a stronger feature response in the CNN. To validate our hypothesis, we add pyramid structures to the other two methods to incorporate multi-scale information, as illustrated in Fig. 5.3. Then the new embeddings sets generated by the pyramid structures are used for computing the EMD. As is shown in Table 5.3, the pyramid structures can effectively boost the performance of DeepEMD-FCN and DeepEMD-Grid, which indicates that multi-scale information is useful for reasoning the relations between images in the low-data regime, and pyramid structures can work at both the image level and the feature level.

Comparison with other k -shot methods. We next compare our proposed SFC layer (denoted by **EMD + SFC**) with some baseline methods and other k -shot

solutions in the literature.

- **EMD + NN.** As the EMD distance metric is a paired function for two structures, the first baseline model for k -shot experiment is the nearest neighbor (NN) method. We compute the Earth Mover’s distance between the query image and all support images, and then classify the query images as the category of the nearest support sample.
- **EMD + Fusion.** Instead of taking the label of the nearest sample, we can fuse the distances of all support images belonging to the same class and classify the query image to the class with the minimum overall distance.
- **EMD + Merge.** As the number of elements in the compared sets can differ, we can also merge the local embeddings of all support samples in each class into a big embedding set to compute EMD distance, and there will be $k \times 5 \times 5$ embedding vectors in the set. The problem in this baseline is that as the size of the set grows linearly with respect to the number of shots, the time complexity of solving the transportation problem can increase cubically with the QPTH solver, which makes the inference of many-shot tasks very slow. Please refer to Section 5.4.6 for the analysis of time complexity.
- **Prototype Learning.** In [1], they average the feature embeddings of support images belonging to the same class as the category prototype and apply the nearest neighbor method for classification.
- **Finetuning a cosine classifier [3].** Similar to our proposed SFC layer, the network backbone is fixed and the support images are used to learn a fully connected layer as the classifier. The difference is the parameters of the prototypes and the distance metrics for classification, illustrated earlier in Fig. 5.4.

We compare these models on the k -shot 5-way tasks with multiple k values, and the results are shown in Fig. 5.5. We can find that the performance of non-optimization based methods often gets saturated quickly and the performance increases slowly with more support samples available. Our structured FC layer can consistently outperform baseline models, and with the number of support images increasing, our network shows even more advantages. The comparison between our method and the

finetuned *cosine* classifier further shows the advantages of structured prototypes and EMD as the metric in the proposed SFC layer.

Pre-training with self-supervised auxiliary tasks. As the weight generation and the computation of optimal matching flows rely on the representations encoded by the backbone, the feature pre-training plays an important role in our framework. We next investigate the use of self-supervision at the pre-training stage. Self-supervised learning [161, 162] has been recently employed as a pretext task to learn generic representations, which benefit many downstream vision tasks, such as object detection and instance segmentation. Following [162], we add a self-supervised auxiliary learning task during backbone pre-training, and observe its influence on our method. Specifically, we randomly rotate the input images by r degrees and $r \in \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}$. The self-supervised learning task is to predict which rotation angle is applied to each image. More concretely, we add a 4-way linear classifier after the network backbone to predict the selections, as is done in [162]. The auxiliary cross-entropy loss is weighted by a factor of $\gamma = 0.5$, which is searched based on the validation accuracy. As can be seen from the result in Table 5.4, the self-supervised auxiliary task at the feature pre-training stage can effectively boost the performance of our proposed method by up to 1.3%, while applying the random rotation as data augmentation alone degrades the performance in many tasks. This indicates that class-agnostic self-supervised learning task is beneficial to the representation learning in our framework.

5.4.4 Cross-Domain Experiments

Following the experimental setups in [3], we perform a cross-domain experiment where models are trained on *mini*Imagenet and evaluated on the CUB dataset. Due to the large domain gap, we can better evaluate the models' ability to tackle large domain difference between tasks. We compare our proposed method with baseline models in the Table 5.5. As we can see, our algorithm *outperforms the baseline models with a large margin*. This shows that local features can provide more transferable and discriminative information across domains. Moreover, due to our cross-reference mechanism, the optimal matching can be restricted within the co-occurrent object regions that have high feature response, such that the final

distance is based on confident regions and representations, and thus has the ability to filter noise when there is a huge domain shift.

5.4.5 Visualization of matching flows and weights

It is interesting to visualize the optimal matching flows and node weights in the network inference process. We conduct two visualization experiments: First, since we have the correspondence information between the regions in two images, we can reconstruct the one image with the local patches from the other image. In Fig. 5.6, we paste the best matched patches from the right image to the corresponding position in the left image and the weights of patches control the brightness of the regions. As we can see, our algorithm can effectively establish semantic correspondence between local regions, and the background regions in the left image are assigned with small weights, thus contributing less to the overall distance. Second, as the correspondence between regions is not strictly 1-to-1, we then plot the full optimal matching flows in Fig. 5.7. As is shown, one patch can be related to multiple regions in the other image with different weights, which is a useful property when the sizes of the same object are different in two images.

5.4.6 Time Complexity

Compared with the methods with a closed-form distance metric, the training and inference of DeepEMD come with more computation cost, as an LP problem must be solved for each forward process. As is discussed in [108], the main computation lies in the factorization of the KKT matrix as well as back-substitution when using the interior point method to solve the LP problem, which have cubic and quadratic time complexity respectively with respect to the number of optimization variables. The depth of the backbone network has negligible influence on the computation time of the EMD layer, as only the weight of each vector and ground distance matrix are needed for computing EMD and they can be efficiently computed in parallel in modern tensor processing libraries, *e.g.* PyTorch, TensorFlow, and NumPy. Therefore, for DeepEMD-FCN, poolings are helpful in reducing computation time when a feature map with a large spatial size is generated by the backbone. In Table 5.6, we compare the computation time of the EMD layer in

a forward pass of a 5-way 1-shot task, given different sizes of feature maps and dimensions. At training time, we use the QPTH library which adopts the interior point method to solve the LP problem. As we can see, the feature dimension has little influence on the computation time, which indicates that we can replace the backbone with deeper ones, *e.g.*, ResNet-101 (2048 dimensions), without significantly increasing the inference time. As the interior point method is only necessary for computing gradients at training time, after the model is trained, we can replace the solver with other solvers that can make faster inference. At inference time, we deploy the model with the OpenCV [155] library which adopts a modified Simplex algorithm to solve the LP problem. As is shown in Table 5.6, the Simplex solver is much faster than interior point method. Therefore, we use QPTH to train the network and use OpenCV to deploy a trained model for inference.

5.4.7 Comparison with State-of-the-art Methods

Finally, we compare our algorithm with the state-of-the-art methods. We report 1-shot 5-way and 5-shot 5-way performance on 5 popular benchmarks: *miniImageNet*, *tieredImageNet*, FC100, CUB and CIFAR-FS. For the 1-shot experiment, we repeatedly sample 5000 testing episodes and record their average accuracy, and for 5-shot experiments we sample 600 episodes. We reproduce the methods in some earlier works [1, 3, 46] with our network backbone and training strategies, and report the higher performance between our results and their reported ones. The results are shown in Table 5.7. *Our algorithm achieves new state-of-the-art performance on all datasets without seeking any extra data. In particular, our results outperform the state-of-the-art performance by a significant margin on multiple tasks, e.g., 1-shot (3.69%) and 5-shot (1.43%) on the miniImageNet dataset; 1-shot (2.77%) and 5-shot (1.05%) on the tieredImageNet dataset; 1-shot (7.40%) and 5-shot (4.72%) on the CUB dataset; 5-shot (5.55%) on the FC100 dataset. We observe that on the FC100 and CIFAR-FS datasets, DeepEMD-FCN outperforms DeepEMD-Grid and DeepEMD-Sampling, which is different from the observations on other datasets. The possible reason is that these two datasets are built upon CIFAR100, where the size of images is 32×32 . Further cropping patches from the image results in very small input images, which makes the CNNs difficult to generate useful representations.*

5.5 Conclusion

We have proposed a few-shot classification framework that employs the Earth Mover’s Distance as the distance metric. The implicit theorem allows our network to be end-to-end trainable. Our proposed cross-reference mechanism for setting the weights of nodes turns out crucial in the EMD formulation and can effectively minimize the negative impact caused by irrelevant regions. The learnable structured fully connected layer can directly classify dense representations of images in the k -shot settings. Our algorithm achieves new state-of-the-art performance on multiple datasets.

Method	Backbone	<i>miniImageNet</i>		<i>tieredImageNet</i>		CIFAR-FS	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
<i>cosine</i> classifier [3]	<i>ResNet12</i>	55.43 $\pm$ 0.81	77.18 $\pm$ 0.61	61.49 $\pm$ 0.91	82.37 $\pm$ 0.67	-	-
TADAM [48]	<i>ResNet12</i>	58.50 $\pm$ 0.30	76.70 $\pm$ 0.30	-	-	-	-
ECM [156]	<i>ResNet12</i>	59.00 $\pm$ -	77.46 $\pm$ -	63.99 $\pm$ -	81.97 $\pm$ -	-	-
TPN [157]	<i>ResNet12</i>	59.46 $\pm$ -	75.65 $\pm$ -	59.91 $\pm$ 0.94	73.30 $\pm$ 0.75	-	-
PPA [20]	<i>WRN-28-10[†]</i>	59.60 $\pm$ 0.41	73.74 $\pm$ 0.19	65.65 $\pm$ 0.92	83.40 $\pm$ 0.65	-	-
Dhillon <i>et al.</i> [74]	<i>WRN-28-10[†]</i>	57.73 $\pm$ 0.62	78.17 $\pm$ 0.49	66.58 $\pm$ 0.70	85.55 $\pm$ 0.48	68.72 $\pm$ 0.67	86.11 $\pm$ 0.47
ProtoNet [1]	<i>ResNet12</i>	60.37 $\pm$ 0.83	78.02 $\pm$ 0.57	65.65 $\pm$ 0.92	83.40 $\pm$ 0.65	-	-
wDAE-GNN [59]	<i>WRN-28-10[†]</i>	61.07 $\pm$ 0.15	76.75 $\pm$ 0.11	68.18 $\pm$ 0.16	83.09 $\pm$ 0.12	-	-
MTL [44]	<i>ResNet12</i>	61.20 $\pm$ 1.80	75.50 $\pm$ 0.80	-	-	-	-
LEO [43]	<i>WRN-28-10[†]</i>	61.76 $\pm$ 0.08	77.59 $\pm$ 0.12	66.33 $\pm$ 0.05	81.44 $\pm$ 0.09	-	-
DC [27]	<i>ResNet12</i>	62.53 $\pm$ 0.19	79.77 $\pm$ 0.19	-	-	-	-
MetaOptNet [63]	<i>ResNet12</i>	62.64 $\pm$ 0.82	78.63 $\pm$ 0.46	65.99 $\pm$ 0.72	81.56 $\pm$ 0.53	72.00 $\pm$ 0.70	84.20 $\pm$ 0.50
FEAT [47]	<i>ResNet24[†]</i>	62.96 $\pm$ 0.20	78.49 $\pm$ 0.15	-	-	-	-
MatchNet [46]	<i>ResNet12</i>	63.08 $\pm$ 0.80	75.99 $\pm$ 0.60	68.50 $\pm$ 0.92	80.60 $\pm$ 0.71	-	-
CAN [51]	<i>ResNet12</i>	63.85 $\pm$ 0.48	79.44 $\pm$ 0.34	69.89 $\pm$ 0.51	84.23 $\pm$ 0.37	-	-
DSN [35]	<i>ResNet12</i>	62.64 $\pm$ 0.66	78.83 $\pm$ 0.45	66.22 $\pm$ 0.75	82.79 $\pm$ 0.48	72.30 $\pm$ 0.80	85.10 $\pm$ 0.60
Tian <i>et al.</i> [81]	<i>ResNet12</i>	64.82 $\pm$ 0.60	82.14 $\pm$ 0.43	71.52 $\pm$ 0.69	86.03 $\pm$ 0.49	73.90 $\pm$ 0.80	86.90 $\pm$ 0.50
CTM [52]	<i>ResNet18[†]</i>	64.12 $\pm$ 0.82	80.51 $\pm$ 0.13	68.41 $\pm$ 0.39	84.28 $\pm$ 1.73	-	-
Kim <i>et al.</i> [36]	<i>ResNet12</i>	65.08 $\pm$ 0.86	82.70 $\pm$ 0.54	-	-	73.51 $\pm$ 0.92	85.49 $\pm$ 0.68
DeepEMD-FCN	<i>ResNet12</i>	66.50 $\pm$ 0.80	82.41 $\pm$ 0.56	72.65 $\pm$ 0.31	86.03 $\pm$ 0.58	74.58 $\pm$ 0.29	86.92 $\pm$ 0.41
DeepEMD-Grid	<i>ResNet12</i>	67.83 $\pm$ 0.29	83.14 $\pm$ 0.57	73.13 $\pm$ 0.32	87.08 $\pm$ 0.60	73.31 $\pm$ 0.29	85.43 $\pm$ 0.37
DeepEMD-Sampling	<i>ResNet12</i>	68.77 $\pm$ 0.29	84.13 $\pm$ 0.53	74.29 $\pm$ 0.32	86.98 $\pm$ 0.60	74.48 $\pm$ 0.29	86.37 $\pm$ 0.36

(A) Results on *miniImageNet*, *tieredImageNet*, and CIFAR-FewShot datasets

Method	Backbone	1-shot	5-shot
<i>cosine</i> classifier [3]	<i>ResNet12</i>	38.47 $\pm$ 0.70	57.67 $\pm$ 0.77
TADAM [48]	<i>ResNet12</i>	40.10 $\pm$ 0.40	56.10 $\pm$ 0.40
MetaOptNet [63]	<i>ResNet12</i>	41.10 $\pm$ 0.60	55.5 $\pm$ 0.60
ProtoNet [1]	<i>ResNet12</i>	41.54 $\pm$ 0.76	57.08 $\pm$ 0.76
DC [27]	<i>ResNet12</i>	42.04 $\pm$ 0.17	57.05 $\pm$ 0.16
MatchNet [46]	<i>ResNet12</i>	43.88 $\pm$ 0.75	57.05 $\pm$ 0.71
MTL [44]	<i>ResNet12</i>	45.10 $\pm$ 1.8	57.6 $\pm$ 0.9
DeepEMD-FCN	<i>ResNet12</i>	46.60 $\pm$ 0.26	63.22 $\pm$ 0.71
DeepEMD-Grid	<i>ResNet12</i>	45.23 $\pm$ 0.26	61.39 $\pm$ 0.76
DeepEMD-Sampling	<i>ResNet12</i>	45.37 $\pm$ 0.25	61.51 $\pm$ 0.70

(B) Results on Fewshot-CIFAR100 dataset.

Method	Backbone	1-shot	5-shot
ProtoNet [1]	<i>ResNet12</i>	66.09 $\pm$ 0.92	82.50 $\pm$ 0.58
RelationNet [3, 49]	<i>ResNet34[†]</i>	66.20 $\pm$ 0.99	82.30 $\pm$ 0.58
DEML [21]	<i>ResNet50[†]</i>	66.95 $\pm$ 1.06	77.11 $\pm$ 0.78
MAML [2, 3]	<i>ResNet34[†]</i>	67.28 $\pm$ 1.08	83.47 $\pm$ 0.59
<i>cosine</i> classifier [3]	<i>ResNet12</i>	67.30 $\pm$ 0.86	84.75 $\pm$ 0.60
MatchNet [46]	<i>ResNet12</i>	71.87 $\pm$ 0.85	85.08 $\pm$ 0.57
DeepEMD-FCN	<i>ResNet12</i>	77.14 $\pm$ 0.29	88.98 $\pm$ 0.49
DeepEMD-Grid	<i>ResNet12</i>	77.64 $\pm$ 0.29	89.25 $\pm$ 0.53
DeepEMD-Sampling	<i>ResNet12</i>	79.27 $\pm$ 0.29	89.80 $\pm$ 0.51

(C) Results on the Caltech-UCSD Birds-200-2011 dataset.

TABLE 5.7: Comparison with the state-of-the-art 1-shot 5-way and 5-shot 5-way performance (%) with 95% confidence intervals on *miniImageNet* (a), *tieredImageNet* (a), CIFAR-FewShot (a) Fewshot-CIFAR100 (b), and Caltech-UCSD Birds-200-2011 (c) datasets. Our model achieves new state-of-the-art performance on all datasets, and even outperforms methods with deeper backbones[†].

Chapter 6

Meta Navigator: Search for a Good Adaptation Policy for Few-shot Learning¹

6.1 Introduction

Convolutional Neural Networks (CNNs) have become indispensable in a variety of computer vision tasks. 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.

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 [1, 2, 46]. Literature on few-shot learning exhibits great diversity, while different algorithms often excel at different few-shot learning scenarios. Fig. 6.1 compares some popular few-shot learning algorithms on different few-shot learning tasks.

¹The work in this chapter has been published in the paper :Chi Zhang, Henghui Ding, Guosheng Lin, Ruibo Li, Changhu Wang, Chunhua Shen. “Meta Navigator: Search for a Good Adaptation Policy for Few-shot Learning.” Proceedings of the IEEE International Conference on Computer Vision (ICCV). Virtual Conference. 2021. [163]

FIGURE 6.1: 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 *miniImageNet* dataset. Cross-domain experiments are evaluated on CUB dataset. Existing few-shot learning algorithms are highly task-specific and are outperformed by a simple finetune baseline when the domain difference is large and more support data are available.

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-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. Thus, 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) [112–114], 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 searching 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 constructs a hierarchical policy searching space. At each network stage, various candidates policies are available for adapting the parameters, and the whole searching space covers many popular meta-learning algorithms in the literature, such as Prototypical Networks [1], Matching Networks [46], baseline++ [3], MAML [2], *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 [112], 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 [112] 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 [113] 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 achieve new state-of-the-art results on many benchmarks.

6.2 Policy Searching Space

This section describes our meta-learning policy search space. A standard CNN structure mainly include two components: the feature backbone that encode 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. 6.2.1) and the policies at the prototype learning (PL) stage (Sec. 6.2.2).

6.2.1 Search Space for Representation Encoding

In a standard convolutional neural network, denoted by $\mathcal{G}(\cdot)$, 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 $\{g^1, g^2, \dots\}$, *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 $g^l(\cdot; \theta)$, 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++ [3] also freezes the pre-trained backbone in novel tasks and only finetunes the classifier.

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

$$\hat{\theta} = \theta - \beta \nabla_{\theta} \mathcal{L}_{\mathcal{T}}(\theta), \quad (6.1)$$

where β is the learning rate and $\mathcal{L}$ is the loss function. By varying the hyper-parameters during finetuning, *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 finetunes the parameters with a small learning rate.

III. Fast Adaptation (RE-FA). Different from RE-FT that finetunes from the weights of RE-FIX, RE-FA finetunes the model from the model initialization that is meta-learned, as done in MAML [2]. In other words, the finetuning 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 $(1 + S_{RE-FT} + S_{RE-FA})^M$ candidates for representation encoding, where S denotes the number of sub-candidates in the policies.

6.2.2 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 $[\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_N]^T$ 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.*, $s_i = \mathbf{w}_i \mathbf{v}^T$. The inner product operation can also be replaced by other similarity metrics, such as negative L2 distance [1] and cosine similarity [46]. 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 [1], 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 [1]. Therefore, our first candidate is directly parameterizing the classifier with data embeddings without any adaptation.

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

III. Finetuning by Fast Adaptation (PL-FA). Similar to the RE-FA policy for representation learning, the PL-FA finetunes 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 searching space for prototypes, the whole search space $\mathcal{O}$ include $(1 + S_{RE-FT} + S_{RE-FA})^M \times (1 + S_{PL-FT} + S_{PL-FA})$ candidates, and the searching goal is to find a high-performing combination of policy candidates in the whole search space. Our searching space covers many well-known meta-learning algorithms, as shown in Fig.6.2. 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 [1] and Matching Networks [46]; if all stages choose fast adaptation (RE-FA + PL-FA), the model becomes MAML; if the parameters in

FIGURE 6.2: Our search-based framework for few-shot classification. We aim to search for a good adaptation policy at each network stage. Our searching space mainly include two parts: the policies for representation encoding and the policies for prototype learning. Please refer to Sec. 6.2 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 [1], MAML [2] and Baseline++ [3] (bottom).

the backbone are kept frozen and the classifier is finetuned by fast adaptation (RE-FIX + PL-FA), it is close to baseline++ [3].

6.3 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. 6.3.1). Then, we discuss how to optimize the model for searching (Sec. 6.3.2), and how to progressively decode the supernet (Sec. 6.3.3).

6.3.1 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 [112]. 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,

$$\sum_{i \in |\mathcal{O}^l|} \alpha_i^l = 1, \alpha_i^l > 0, \quad (6.2)$$

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:

$$O^l = \sum_{i \in |\mathcal{O}^l|} \alpha_i^l g^l(O^{l-1}, \hat{\theta}_i^l). \quad (6.3)$$

Therefore, all candidate policies in different stages of the network together construct a differentiable supernet [112]. 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 [112].

6.3.2 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 [112] 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. 2 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 $p(\mathcal{T}_A)$ and $p(\mathcal{T}_B)$, 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 $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_A}(\hat{\theta}, \alpha)$ and $\nabla_{\alpha} \mathcal{L}_{\mathcal{T}_B}(\hat{\theta}, \alpha)$, respectively. Moreover, the way to obtain the gradient with respect to the policies weights, *i.e.*, $\nabla_{\theta} \mathcal{L}_{\mathcal{T}_A}(\hat{\theta}, \alpha)$, 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 second-order gradients [2] or implicit gradients [32]. Noted that as all sub-candidates of RE-FT and PL-FT finetune the parameters from the pre-trained weights (RE-FIX) or data embeddings, the parameters θ for finetuning in these policies are generated online in each task, and hence there is no learnable parameter of these policies in the outer loop of the step 1.

6.3.3 Decoding Discrete Policies

During the supernet training, we progressively decode the supernet such that only one candidate policy is left in each stage. Following [113], we adopt a perturbation-based decoding strategy that the strength of each policy 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 finetune the supernet with Alg. 2 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 finetuning-based policies. Concretely, before decoding of the supernet, an SGD

step in a policy i at layer l is

$$\begin{aligned}
\hat{\theta}_i^l &= \theta_i^l - \beta \frac{\partial \mathcal{L}_s}{\partial \theta_i^l} \\
&= \theta_i^l - \beta \frac{\partial \mathcal{L}_s}{\partial O^l} \frac{\partial O^l}{\partial g^l} \frac{\partial g^l}{\partial \theta_i^l} \\
&= \theta_i^l - \beta \frac{\partial \mathcal{L}_s}{\partial O^l} \alpha_i^l \frac{\partial g^l}{\partial \theta_i^l}
\end{aligned} \tag{6.4}$$

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. 6.3 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 $\frac{\partial \mathcal{L}_s}{\partial O^l}$ 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

$$\begin{aligned}
\hat{\theta}_i^l &= \theta_i^l - (\beta \alpha_i^l) \frac{\partial \mathcal{L}_s}{\partial O^l} \frac{\partial O^l}{\partial g^l} \frac{\partial g^l}{\partial \theta_i^l} \\
&= \theta_i^l - (\beta \alpha_i^l) \frac{\partial \mathcal{L}_s}{\partial O^l} \frac{\partial g^l}{\partial \theta_i^l}
\end{aligned} \tag{6.5}$$

As a result, the actual adaptation steps in the policy before and after decoding are identical, and the initial assigned learning rate only serves as the upper bound during searching.

6.4 Experiments

6.4.1 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 searching space

Algorithm 2: Optimization of the supernet to search for good adaptation policies.

Input: $p(\mathcal{T}_A), p(\mathcal{T}_B)$: two disjoint task distributions

Input: $\mathcal{G}(\cdot; \Theta, \alpha)$: a supernet with 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

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 directly use the backbone of the supernet without adaptation to encode the support data to obtain the prototypes in PL-DI. We train the supernet with Alg. 2 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. For more analysis and experiment results, please refer to our supplementary material.

6.4.2 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. 6.3. Noted that the policy selection weights do

Model	1-shot	5-shot	10-shot
ProtoNets [1]	57.89	78.75	82.66
MatchNets [46]	61.47	75.41	78.87
MAML [2]	59.58	75.80	78.20
Baseline++ [3]	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 6.1: 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.

Model	1-shot	5-shot	10-shot
ProtoNets [1]	45.52	66.80	72.29
MatchNets [46]	47.41	63.63	68.83
MAML [2]	41.29	58.10	62.18
Baseline++ [3]	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 6.2: 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.

FIGURE 6.3: 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. 6.4.2 for our analysis.

not necessarily indicate the 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

Method	Backbone	<i>miniImageNet</i>		<i>tieredImageNet</i>		FC100		CIFAR-FS	
		1-shot	5-shot	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
TADAM [48]	ResNet-12	58.50 $\pm$ 0.30	76.70 $\pm$ 0.30	—	—	40.1 $\pm$ 0.4	56.1 $\pm$ 0.4	—	—
MTL [44]	ResNet-12	61.2 $\pm$ 1.8	75.5 $\pm$ 0.8	65.6 $\pm$ 1.8	80.8 $\pm$ 0.8	45.1 $\pm$ 1.9	57.6 $\pm$ 1.0	—	—
FEAT [47]	ResNet-25 [‡]	62.96 $\pm$ 0.20	78.49 $\pm$ 0.15	—	—	—	—	—	—
LEO [43]	WRN-28-10 [‡]	61.76 $\pm$ 0.08	77.59 $\pm$ 0.12	66.33 $\pm$ 0.05	81.44 $\pm$ 0.09	—	—	—	—
Dhillon <i>et al.</i> [74]	WRN-28-10 [‡]	57.73 $\pm$ 0.62	78.17 $\pm$ 0.49	66.58 $\pm$ 0.70	85.55 $\pm$ 0.48	38.25 $\pm$ 0.52	57.19 $\pm$ 0.57	68.72 $\pm$ 0.67	86.11 $\pm$ 0.47
MetaOptNet [63]	ResNet-12	62.64 $\pm$ 0.82	78.63 $\pm$ 0.46	65.99 $\pm$ 0.72	81.56 $\pm$ 0.53	41.1 $\pm$ 0.6	55.5 $\pm$ 0.6	72.0 $\pm$ 0.7	84.2 $\pm$ 0.5
CAN [51]	ResNet-12	63.85 $\pm$ 0.48	79.44 $\pm$ 0.34	69.89 $\pm$ 0.51	84.23 $\pm$ 0.37	—	—	—	—
DSN-MR [35]	ResNet-12	64.60 $\pm$ 0.72	79.51 $\pm$ 0.50	67.39 $\pm$ 0.82	82.85 $\pm$ 0.56	—	—	75.6 $\pm$ 0.9	86.2 $\pm$ 0.6
Tian <i>et al.</i> [81]	ResNet-12	64.82 $\pm$ 0.60	82.14 $\pm$ 0.43	71.52 $\pm$ 0.69	86.03 $\pm$ 0.49	44.6 $\pm$ 0.7	60.9 $\pm$ 0.6	73.9 $\pm$ 0.8	86.9 $\pm$ 0.5
Kim <i>et al.</i> [36]	ResNet-12	65.08 $\pm$ 0.86	82.70 $\pm$ 0.54	—	—	42.31 $\pm$ 0.75	57.56 $\pm$ 0.78	73.51 $\pm$ 0.92	85.49 $\pm$ 0.68
DeepEMD [147]	ResNet-12	65.91 $\pm$ 0.82	82.41 $\pm$ 0.56	71.16 $\pm$ 0.87	86.03 $\pm$ 0.58	46.47 $\pm$ 0.78	63.22 $\pm$ 0.71	—	—
Ours	ResNet-12	65.91 $\pm$ 0.83	82.66 $\pm$ 0.55	73.52 $\pm$ 0.88	85.34 $\pm$ 0.62	45.60 $\pm$ 0.81	59.93 $\pm$ 0.76	74.01 $\pm$ 0.96	86.03 $\pm$ 0.62
Ours + MC	ResNet-12	67.14 $\pm$ 0.80	83.82 $\pm$ 0.51	74.58 $\pm$ 0.88	86.73 $\pm$ 0.61	46.40 $\pm$ 0.81	61.33 $\pm$ 0.71	74.63 $\pm$ 0.91	86.45 $\pm$ 0.59

[‡] Different backbone with ours.

TABLE 6.3: Comparison with the state-of-the-art 5-way few-shot classification results on *miniImageNet*, *tieredImageNet*, FC100, and CIFAR-FS dataset. **MC** denotes multi-crop testing. Our approach outperforms state-of-the-art performance on multiple datasets.

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 [1] use negative L2 distance to compute class scores and the averaged embeddings as prototypes in the classifier, while Matching Networks [46] 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 [2]. 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++ [3]. 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 models. We omit the policy PL-FA ($lr = 0.01$), 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 6.1. 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. 2 to conduct searching. As we can see from the results in Table 6.2, 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%.

6.4.3 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 6.3. *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%.

6.5 Conclusion

In this thesis, 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.

Chapter 7

Summary

7.1 Conclusion

In this thesis, I have presented four novel methods to solve the few-shot learning problems in image classification and image segmentation tasks. Before presenting our methods, we discussed the challenges of the few-shot learning tasks and the limitations in current methods, which are the motivations of our respective methods.

7.1.1 Few-Shot Image Segmentation

Specifically, in the few-shot segmentation task, to bypass a difficult learning issue that often leads to overfitting in low-shot cases, many previous and current methods model the task as a conditional prediction task where information from the labeled support set is extracted as the condition to predict the label of query images. Then, the challenge here is how to effectively represent data and how to effectively transfer information. In our first work (**Chapter 3**), we design CANets, inspired by the idea of prototype learning in few-shot classification. The networks represent a novel category with a class-specific prototype vector using middle-level features. Then, the prototype is fused into the query branch by upsampling and vector concatenation. To further improve the predicted result, we design an iterative refinement module, which iteratively refines the initial predictions. To solve the k-shot learning issues, we design a module to fuse the prototypes generated by

different data points by attentions. PGNet in **Chapter 4** further improves the ways to represent data and transfer information from the support set to the query set. Instead of pooling the representations of the images into a prototype vector, it maintains the structured representations of images generated by the CNNs and uses graph methods to transfer information. Compared with the CANet, it can selectively transfer information to each pixel location in the query images. As the graph models do not require a fixed number of inputs, the framework can naturally handle both the 1-shot case and the k -shot case.

7.1.2 Few-Shot Image Classification

For the few-shot classification task, we first present an improved metric-based few-shot learning framework (**Chapter 5**). Compared with previous metric-based methods that utilize metrics between vectors, such as cosine distance and L2 distance, our method computes the structured distance between data points. Concretely, compared with previous data encoding ways that use a CNN to encode each data point into a point in feature space, we represent a data point as a distribution where each point in the distribution can be seen as a local representation. We investigate three ways for data encoding, including, fully convolutional networks, image patches based on grids, and image patches based on random sampling. Then, to reflect the semantic relevance between data points, we use Earth Mover’s Distance to compute the similarity values between distributions. To compute the important weights in EMD, we design a cross-reference mechanism to appropriately assign the weights to local representations. To handle the k -shot learning issues, we design a structured fully connected layer, which takes inspiration from the design in the standard linear classifiers. The structured distance metric demonstrates great advantages over previous metric-based methods.

We further re-think the few-shot classification in **Chapter 6** and show that different few-shot learning algorithms often excel at different few-shot learning tasks. Inspired by the AutoML literature, we design a searching-based framework for few-shot learning, which aims to find good adaptation policies for the network parameters. We first design a search space that covers many classic designs in the few-shot learning literature, then we design a differentiable searching algorithm

and decoding algorithm to find the optimal policies. Compared with many previous works that focus on a specific task setting, our model can handle various task settings, which is an early attempt to bridge the gap between low-shot learning and many-shot learning.

7.2 Future Works

Based on the concluded challenges and limitations, and our proposed methods, we summarize the related potential research directions in few-shot learning.

7.2.1 Few-Shot Image Segmentation

Better Image Representations. As we have done in our works, effectively encoding the data is a crucial step in a few-shot segmentation model. As the model is trained on training classes while it is expected to work on novel classes, there might be a gap between two domains. In Chapter 3, we have demonstrated that using middle-level features is very helpful to few-shot learners as they correspond to object parts, which can be shared across tasks. More advanced techniques are likely to be applied to solve the domain gaps in this few-shot segmentation tasks, such as the methods in domain adaptation literature. Moreover, advanced few-shot learning paradigms, such as those from meta learning literature, may also help the representation learning in few-shot segmentation. Self-supervised learning is also a promising direction for few-shot segmentation, as it encodes class-agnostic representations, which has a similar learning objective in our learning paradigm.

Message Passing. In few-shot segmentation model based on conditional predictions, how to effectively represent and transfer the information for conditional predictions is important. Our methods present two methods to achieve this goal, including the prototype based method with global average pooling and the graph based method that applies to each pixel location. There can be many other ways to achieve this. For example, we may use clustering methods or mixture models to better represent the information from the support set. To establish meaningful correspondence between the support set and the query set, we can use more

advanced graph reasoning techniques, such as the literature about Graph Neural Networks.

7.2.2 Few-Shot Image Classification

Representation Learning. In metric-based methods, as the learnable parameters are usually only in the data encoders, how to encode useful representations for classification is crucial. To achieve this goal, we may make improvements from many aspects, including 1) self-supervised learning that also learns class-agnostic features; 2) Better learning paradigm, such as the state-of-the-art meta-learning algorithms or the state-of-the-art regularization techniques; 3) addressing the domain gap between training and testing classes. 4) involving task contexts to improve representation encoding. As the model is optimized to learn discriminative representations for the current tasks, they may not still be discriminative in other tasks. It is therefore useful to involve task contexts as additional knowledge to improve the feature encoding.

Generalized Few-Shot Learning As we have pointed out in Chapter 6, many existing few-shot learning algorithms focus on designing effective models on specific task settings, such as 1-shot and 5-shot tasks, rather than solving a general problem of data scarcity. It is therefore meaningful and useful to design learning algorithms that can apply to different task settings or tasks. Meta Navigator is an early attempt to solve this issue. More advanced searching algorithms and more broad searching spaces are expected to address the issues better.

7.2.3 Unified Few-Shot Learning Algorithms

Current few-shot learning algorithms only focus on specific task types, such as image classification, image segmentation, and image detection. It is expected that a unified few-shot learning algorithm that can apply to various computer vision tasks can be designed in the future. More recently, Transformer shows a promising direction in computer vision where the similar architecture can be used solve multiple vision tasks. The success of the Transformer on NLP may provide inspirations on how various few-shot learning tasks can be unified and solved.

List of Publications¹

Conference Proceedings

- **Chi Zhang**, Guosheng Lin, Fayao Liu, Rui Yao, Chunhua Shen, “CANet: Class-Agnostic Segmentation Networks with Iterative Refinement and Attentive Few-Shot Learning,” in *IEEE Conference on Computer Vision and Pattern Recognition, 2019*.
- **Chi Zhang**, Guosheng Lin, Fayao Liu, Jiushuang Guo, Qingyao Wu, Rui Yao, “Pyramid Graph Networks with Connection Attentions for Region-Based One-Shot Semantic Segmentation,” in *IEEE International Conference on Computer Vision, 2019*.
- **Chi Zhang**, Yujun Cai, Guosheng Lin, Chunhua Shen, “DeepEMD: Few-Shot Image Classification with Differentiable Earth Mover’s Distance and Structured Classifiers,” in *IEEE Conference on Computer Vision and Pattern Recognition, 2020*.
- Weide Liu, **Chi Zhang**, Guosheng Lin, Fayao Liu, “CRNet: Cross-Reference Networks for Few-Shot Segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition, 2020*.
- Xin Sun, Zhenning Yang, **Chi Zhang**, Guohao Peng, Keck-Voon Ling, “Conditional Gaussian Distribution Learning for Open Set Recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition, 2020*.
- Weide Liu*, **Chi Zhang***, Guosheng Lin, Tzu-Yi HUNG, Chunyan Miao , “Weakly Supervised Segmentation with Maximum Bipartite Graph Matching,” in *Proceedings of ACM International Conference on Multimedia, 2020*.

¹The superscript * indicates joint first authors

- **Chi Zhang***, Nan Song*, Guosheng Lin, Yun Zheng, Pan Pan, Yinghui Xu, “Few-Shot Incremental Learning with Continually Evolved Classifiers,” in *IEEE Conference on Computer Vision and Pattern Recognition, 2021*.
- **Chi Zhang**, Henghui Ding, Guosheng Lin, Ruibo Li, Changhu Wang, Chunhua, “Meta Navigator: Search for a Good Adaptation Policy for Few-shot Learning,” in *IEEE International Conference on Computer Vision, 2021*.

Journal Articles

- **Chi Zhang**, Guankai Li, Guosheng Lin, Qingyao Wu, Rui Yao, ‘CycleSegNet: Object Co-segmentation with Cycle Refinement and Region Correspondence,’ in *IEEE Transactions on Image Processing, 2021*.
- **Chi Zhang**, Yujun Cai, Guosheng Lin, Chunhua Shen, “DeepEMD: Differentiable Earth Mover’s Distance for Few-Shot Learning,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence (Under review; Minor revision)*.

Bibliography

- [1] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *Proc. Advances in Neural Inf. Process. Syst.*, 2017. [xxi](#), [7](#), [14](#), [48](#), [53](#), [64](#), [65](#), [67](#), [71](#), [74](#), [76](#), [77](#), [79](#), [82](#), [83](#), [88](#), [90](#)
- [2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. Int. Conf. Mach. Learn.*, 2017. [xxi](#), [7](#), [76](#), [77](#), [79](#), [81](#), [83](#), [85](#), [88](#), [90](#)
- [3] 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. [xxi](#), [7](#), [61](#), [64](#), [65](#), [67](#), [71](#), [72](#), [74](#), [76](#), [79](#), [81](#), [83](#), [88](#), [90](#)
- [4] Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alyosha Efros, and Sergey Levine. Conditional networks for few-shot semantic segmentation. In *ICLR Workshop*, 2018. [xxiv](#), [9](#), [14](#), [22](#), [23](#), [24](#), [26](#), [32](#), [42](#), [43](#), [44](#), [45](#)
- [5] Rohit Keshari, Mayank Vatsa, Richa Singh, and Afzel Noore. Learning structure and strength of CNN filters for small sample size training. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. [7](#)
- [6] Fei-Fei Li, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(4):594–611, 2006.
- [7] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.
- [8] Shipeng Yan, Songyang Zhang, and Xuming He. A dual attention network with semantic embedding for few-shot learning. In *Proc. AAAI Conf. Artificial Intell.*, 2019.
- [9] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019.
- [10] 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.

- [11] Arman Afrasiyabi, Jean-François Lalonde, and Christian Gagné. Associative alignment for few-shot image classification. In *Proc. Eur. Conf. Comp. Vis.*, pages 18–35. Springer, 2020.
- [12] 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.
- [13] Moshe Lichtenstein, Prasanna Sattigeri, Rogerio Feris, Raja Giryes, and Leonid Karlinsky. Tafssl: Task-adaptive feature sub-space learning for few-shot classification. In *Proc. Eur. Conf. Comp. Vis.*, pages 522–539. Springer, 2020.
- [14] 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.
- [15] Weijian Xu, Yifan Xu, Huaijin Wang, and Zhuowen Tu. Attentional constellation nets for few-shot learning. 2021. [7](#)
- [16] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, and Adam Trischler. Rapid adaptation with conditionally shifted neurons. In *Proc. Int. Conf. Mach. Learn.*, 2018. [7](#)
- [17] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy P. Lillicrap. Meta-learning with memory-augmented neural networks. In *Proc. Int. Conf. Mach. Learn.*, 2016.
- [18] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *Proc. Int. Conf. Mach. Learn.*, 2017.
- [19] Tyler R. Scott, Karl Ridgeway, and Michael C. Mozer. Adapted deep embeddings: A synthesis of methods for k-shot inductive transfer learning. In *Proc. Advances in Neural Inf. Process. Syst.*, 2018.
- [20] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L. Yuille. Few-shot image recognition by predicting parameters from activations. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. [76](#)
- [21] Fengwei Zhou, Bin Wu, and Zhenguo Li. Deep meta-learning: Learning to learn in the concept space. *arXiv*, 1802.03596, 2018. [76](#)
- [22] Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. Meta-learning update rules for unsupervised representation learning. In *Proc. Int. Conf. Learn. Representations*, 2019.
- [23] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *Proc. Int. Conf. Mach. Learn.*, 2018.
- [24] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. In *Proc. Int. Conf. Mach. Learn.*, 2018.

- [25] Jelena Luketina, Tapani Raiko, Mathias Berglund, and Klaus Greff. Scalable gradient-based tuning of continuous regularization hyperparameters. In *Proc. Int. Conf. Mach. Learn.*, 2016.
- [26] Devang K Naik and RJ Mammone. Meta-neural networks that learn by learning. In *Proc. Int. Joint Conf. Neural Networks*, 1992.
- [27] Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. Dense classification and implanting for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019. [7](#), [64](#), [66](#), [76](#)
- [28] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero- and few-shot learning via aligned variational autoencoders. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019.
- [29] Sebastian Flennerhag, Andrei A Rusu, Razvan Pascanu, Hujun Yin, and Raia Hadsell. Meta-learning with warped gradient descent. *arXiv preprint arXiv:1909.00025*, 2019.
- [30] Xinzhe Li, Qianru Sun, Yaoyao Liu, Qin Zhou, Shibao Zheng, Tat-Seng Chua, and Bernt Schiele. Learning to self-train for semi-supervised few-shot classification. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 10276–10286, 2019.
- [31] Eunbyung Park and Junier B Oliva. Meta-curvature. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 3309–3319, 2019.
- [32] Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. Meta-learning with implicit gradients. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 113–124, 2019. [85](#)
- [33] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *Proc. Int. Conf. Mach. Learn.*, 2018.
- [34] Hang Qi, Matthew Brown, and David G Lowe. Low-shot learning with imprinted weights. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 5822–5830, 2018.
- [35] 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. [76](#), [89](#)
- [36] Jaekyeom Kim, Hyoungeok 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. [76](#), [89](#)
- [37] Manli Zhang, Jianhong Zhang, Zhiwu Lu, Tao Xiang, Mingyu Ding, and Songfang Huang. IEPT: Instance-level and episode-level pre-text tasks for few-shot learning. 2021.

- [38] Nanyi Fei, Zhiwu Lu, Tao Xiang, and Songfang Huang. Melr: Meta-learning via modeling episode-level relationships for few-shot learning. In *Proc. Int. Conf. Learn. Representations*, 2021.
- [39] Jaehoon Oh, Hyungjun Yoo, ChangHwan Kim, and Se-Young Yun. BOIL: Towards representation change for few-shot learning. In *Proc. Int. Conf. Learn. Representations*, 2021.
- [40] Jake Snell and Richard Zemel. Bayesian few-shot classification with one-vs-each polya-gamma augmented Gaussian processes. In *Proc. Int. Conf. Learn. Representations*, 2021.
- [41] Massimiliano Patacchiola, Jack Turner, Elliot J Crowley, Michael O’Boyle, and Amos Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. 2020. 7
- [42] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. In *Proc. Int. Conf. Learn. Representations*, 2019. 7
- [43] 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. 76, 89
- [44] 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.*, 2019. 76, 89
- [45] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019. 7
- [46] 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.*, 2016. 7, 17, 48, 51, 53, 63, 64, 65, 67, 74, 76, 77, 79, 82, 88, 90
- [47] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Learning embedding adaptation for few-shot learning. *arXiv*, 1812.03664, 2018. 63, 64, 76, 89
- [48] 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.*, 2018. 63, 64, 76, 89
- [49] 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.*, 2018. 14, 76

- [50] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Re-visiting local descriptor based image-to-class measure for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019. [7](#), [48](#), [64](#), [65](#), [67](#)
- [51] Ruibing Hou, Hong Chang, Bingpeng Ma, Shiguang Shan, and Xilin Chen. Cross attention network for few-shot classification. In *Proc. Advances in Neural Inf. Process. Syst.*, 2019. [8](#), [68](#), [76](#), [89](#)
- [52] 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.*, 2019. [76](#)
- [53] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *Proc. Int. Conf. Learn. Representations*, 2018. [63](#)
- [54] Jinlu Liu, Liang Song, and Yongqiang Qin. Prototype rectification for few-shot learning. *arXiv preprint arXiv:1911.10713*, 2019.
- [55] Chen Xing, Negar Rostamzadeh, Boris N Oreshkin, and Pedro O Pinheiro. Adaptive cross-modal few-shot learning. *arXiv preprint arXiv:1902.07104*, 2019.
- [56] 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.*, June 2021. [7](#)
- [57] Carl Doersch, Ankush Gupta, and Andrew Zisserman. CrossTransformers: spatially-aware few-shot transfer. In *Proc. Advances in Neural Inf. Process. Syst.*, 2020. [8](#), [68](#)
- [58] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. Edge-labeling graph neural network for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019. [8](#)
- [59] Spyros Gidaris and Nikos Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019. [76](#)
- [60] Victor Garcia and Joan Bruna. Few-shot learning with graph neural networks. *arXiv preprint arXiv:1711.04043*, 2017.
- [61] 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. [8](#)
- [62] Wen-Hsuan Chu, Yu-Jhe Li, Jing-Cheng Chang, and Yu-Chiang Frank Wang. Spot and learn: A maximum-entropy patch sampler for few-shot image classification. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019. [8](#)

- [63] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2019. [8](#), [76](#), [89](#)
- [64] Sergey Bartunov and Dmitry P. Vetrov. Few-shot generative modelling with generative matching networks. In *Proc. Int. Conf. Artificial Intell. & Stat.*, 2018. [8](#)
- [65] Yu-Xiong Wang, Ross B. Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018.
- [66] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Rogério Schmidt Feris, Abhishek Kumar, Raja Giryes, and Alexander M. Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *Proc. Advances in Neural Inf. Process. Syst.*, 2018.
- [67] Akshay Mehrotra and Ambedkar Dukkipati. Generative adversarial residual pairwise networks for one shot learning. *arXiv*, 1703.08033, 2017.
- [68] 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.*, 2018.
- [69] Hongguang Zhang, Jing Zhang, and Piotr Koniusz. Few-shot learning via saliency-guided hallucination of samples. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, June 2019.
- [70] Wei Shen, Ziqiang Shi, and Jun Sun. Learning from adversarial features for few-shot classification. *arXiv preprint arXiv:1903.10225*, 2019.
- [71] 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.
- [72] Shuo Yang, Lu Liu, and Min Xu. Free lunch for few-shot learning: Distribution calibration. *arXiv: Comp. Res. Repository*, 2021. [8](#)
- [73] Pau Rodríguez, Issam Laradji, Alexandre Drouin, and Alexandre Lacoste. Embedding propagation: Smoother manifold for few-shot classification. In *Proc. Eur. Conf. Comp. Vis.*, pages 121–138. Springer, 2020. [8](#)
- [74] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv: Comp. Res. Repository*, 2019. [76](#), [89](#)
- [75] Shell Xu Hu, Pablo G Moreno, Yang Xiao, Xi Shen, Guillaume Obozinski, Neil D. Lawrence, and Andreas Damianou. Empirical Bayes transductive meta-learning with synthetic gradients. *arXiv: Comp. Res. Repository*, 2020.

- [76] Zhongqi Yue, Hanwang Zhang, Qianru Sun, and Xian-Sheng Hua. Interventional few-shot learning. *arXiv preprint arXiv:2009.13000*, 2020.
- [77] Malik Boudiaf, Ziko Imtiaz Masud, Jérôme Rony, José Dolz, Pablo Piantanida, and Ismail Ben Ayed. Transductive information maximization for few-shot learning. *arXiv preprint arXiv:2008.11297*, 2020. [8](#)
- [78] Pranav Shyam, Shubham Gupta, and Ambedkar Dukkipati. Attentive recurrent comparators. In *Proc. Int. Conf. Mach. Learn.*, pages 3173–3181. JMLR. org, 2017. [8](#)
- [79] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *Proc. Int. Conf. Learn. Representations*, 2017. [8](#)
- [80] Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. When does self-supervision improve few-shot learning? In *Proc. Eur. Conf. Comp. Vis.*, pages 645–666. Springer, 2020. [8](#)
- [81] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020. [8](#), [76](#), [89](#)
- [82] Fangyu Wu, Jeremy S Smith, Wenjin Lu, Chaoyi Pang, and Bailing Zhang. Attentive prototype few-shot learning with capsule network-based embedding. In *Proc. Eur. Conf. Comp. Vis.*, pages 237–253. Springer, 2020. [8](#)
- [83] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. Snail: A simple neural attentive meta-learner. In *Proc. Int. Conf. Learn. Representations*, 2018. [8](#)
- [84] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian D Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2017. [8](#), [13](#)
- [85] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018. [8](#), [9](#), [18](#), [38](#), [40](#)
- [86] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):640–651, 2017. [13](#), [54](#)
- [87] Guosheng Lin, Fayao Liu, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for dense prediction. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.
- [88] Guosheng Lin, Chunhua Shen, Anton Van Den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3194–3203, 2016.

- [89] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Amrith Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 7151–7160, 2018.
- [90] Yizhou Zhou, Xiaoyan Sun, Zheng-Jun Zha, and Wenjun Zeng. Context-reinforced semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 4046–4055, 2019. [8](#)
- [91] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3640–3649, 2016. [9](#)
- [92] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2881–2890, 2017. [9](#), [19](#)
- [93] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. In *Proc. British Machine Vis. Conf.*, 2017. [9](#), [14](#), [22](#), [23](#), [24](#), [27](#), [32](#), [39](#), [42](#), [44](#), [45](#)
- [94] Xiaolin Zhang, Yunchao Wei, Yi Yang, and Thomas Huang. Sg-one: Similarity guidance network for one-shot semantic segmentation. *arXiv preprint arXiv:1810.09091*, 2018. [9](#), [32](#), [42](#), [44](#), [45](#)
- [95] 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.*, 2019. [9](#), [13](#), [42](#), [44](#), [45](#)
- [96] Nanqing Dong and Eric Xing. Few-shot semantic segmentation with prototype learning. In *Proc. British Machine Vis. Conf.*, 2018. [9](#), [14](#), [22](#), [23](#), [32](#), [42](#), [44](#), [45](#)
- [97] Tao Hu, Pengwan Yang, Chiliang Zhang, Gang Yu, Yadong Mu, and Cees GM Snoek. Attention-based multi-context guiding for few-shot semantic segmentation. In *Proc. AAAI Conf. Artificial Intell.*, 2019. [9](#), [45](#)
- [98] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover’s distance as a metric for image retrieval. *Int. J. Comput. Vision*, 40(2):99–121, 2000. [10](#), [49](#), [58](#), [66](#)
- [99] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *Proc. Int. Conf. Mach. Learn.*, pages 957–966, 2015. [10](#)
- [100] C Wang and SC Chan. A new hand gesture recognition algorithm based on joint color-depth superpixel earth mover’s distance. In *Proc. Int. Workshop Cognitive Information Processing*, pages 1–6. IEEE, 2014. [10](#)

- [101] Giannis Nikolentzos, Polykarpos Meladianos, and Michalis Vazirgiannis. Matching node embeddings for graph similarity. In *Proc. AAAI Conf. Artificial Intell.*, 2017. [10](#)
- [102] Samuel Schuster, Paul Vernaza, Wongun Choi, and Manmohan Chandraker. Deep network flow for multi-object tracking. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 6951–6960, 2017. [10](#)
- [103] Qi Zhao, Zhi Yang, and Hai Tao. Differential earth mover’s distance with its applications to visual tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(2):274–287, 2008. [10](#)
- [104] Peihua Li. Tensor-sift based earth mover’s distance for contour tracking. *J. Mathematical Imaging & Vision*, 46(1):44–65, 2013. [10](#)
- [105] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016. [10](#)
- [106] Akshay Agrawal, Shane Barratt, Stephen Boyd, Enzo Busseti, and Walaa M Moursi. Differentiating through a conic program. *arXiv preprint arXiv:1904.09043*, 2019. [10](#)
- [107] Shane Barratt. On the differentiability of the solution to convex optimization problems. *arXiv preprint arXiv:1804.05098*, 2018. [11](#), [49](#), [55](#), [57](#)
- [108] Brandon Amos and J. Zico Kolter. OptNet: Differentiable optimization as a layer in neural networks. In *Proc. Int. Conf. Mach. Learn.*, volume 70, pages 136–145, 2017. [11](#), [63](#), [68](#), [73](#)
- [109] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 9558–9570, 2019. [11](#)
- [110] Marin Vlastelica, Anselm Paulus, Vít Musil, Georg Martius, and Michal Rolínek. Differentiation of blackbox combinatorial solvers. *arXiv preprint arXiv:1912.02175*, 2019. [11](#)
- [111] Michal Rolínek, Paul Swoboda, Dominik Zietlow, Anselm Paulus, Vít Musil, and Georg Martius. Deep graph matching via blackbox differentiation of combinatorial solvers. *arXiv preprint arXiv:2003.11657*, 2020. [11](#)
- [112] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proc. Int. Conf. Learn. Representations*, 2019. [11](#), [79](#), [83](#), [84](#)
- [113] Ruochen Wang, Minhao Cheng, Xiangning Chen, Xiaocheng Tang, and Chojui Hsieh. Rethinking architecture selection in differentiable {nas}. In *Proc. Int. Conf. Learn. Representations*, 2021. [11](#), [79](#), [85](#)

- [114] 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. [11](#), [79](#)
- [115] Xiangning Chen and Cho-Jui Hsieh. Stabilizing differentiable architecture search via perturbation-based regularization, 2020.
- [116] Xiangning Chen, Ruochen Wang, Minhao Cheng, Xiaocheng Tang, and Cho-Jui Hsieh. Dr{nas}: Dirichlet neural architecture search. In *Proc. Int. Conf. Learn. Representations*, 2021. [11](#)
- [117] Guohao Li, Guocheng Qian, Itzel C 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.
- [118] 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. [11](#)
- [119] 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. [12](#)
- [120] Sivan Doherty, 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.
- [121] Zhichao Lu, Gautam Sreeksan, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Neural architecture transfer. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2021.
- [122] 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. [12](#)
- [123] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 1097–1105, 2012. [13](#), [31](#), [48](#)
- [124] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 770–778, 2016. [17](#), [31](#), [38](#), [48](#)
- [125] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. [13](#)
- [126] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 91–99, 2015. [13](#)

- [127] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2980–2988, 2017.
- [128] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 779–788, 2016. [13](#)
- [129] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, 2015. [13](#), [18](#), [31](#), [63](#)
- [130] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proc. Eur. Conf. Comp. Vis.*, pages 818–833, 2014. [15](#), [18](#)
- [131] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015. [15](#), [18](#)
- [132] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 373–381, 2016. [15](#)
- [133] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3159–3167, 2016. [15](#), [21](#)
- [134] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. Learning to segment every thing. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. [15](#)
- [135] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2018. [15](#)
- [136] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015. [19](#)
- [137] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. [19](#), [20](#)
- [138] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. [21](#)

- [139] Xiang Wang, Shaodi You, Xi Li, and Huimin Ma. Weakly-supervised semantic segmentation by iteratively mining common object features. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1354–1362, 2018. [21](#)
- [140] Lane McIntosh, Niru Maheswaranathan, David Sussillo, and Jonathon Shlens. Recurrent segmentation for variable computational budgets. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1648–1657, 2018. [21](#)
- [141] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. In *Proc. Eur. Conf. Comp. Vis.*, pages 297–312. Springer, 2014. [24](#), [39](#)
- [142] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 109–117, 2011. [26](#)
- [143] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proc. Eur. Conf. Comp. Vis.*, pages 740–755, 2014. [28](#), [31](#)
- [144] 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. [31](#)
- [145] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017. [33](#), [34](#), [36](#)
- [146] Mennatullah Siam and Boris Oreshkin. Adaptive masked weight imprinting for few-shot segmentation. *arXiv preprint arXiv:1902.11123*, 2019. [44](#), [45](#)
- [147] 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.*, 2020. [47](#), [89](#)
- [148] Frank L. Hitchcock. The distribution of a product from several sources to numerous localities. *J. Mathematics and Physics*, 20(1-4):224–230, 1941. [49](#)
- [149] Steven G Krantz and Harold R Parks. *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2012. [49](#), [55](#)
- [150] Asen L Dontchev and R Tyrrell Rockafellar. Implicit functions and solution mappings. *Springer Monographs in Mathematics*. Springer, 208, 2009. [49](#), [55](#)
- [151] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Proc. Advances in Neural Inf. Process. Syst.*, pages 901–909, 2016. [61](#)

- [152] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. [63](#)
- [153] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018. [63](#), [64](#)
- [154] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 2009. [64](#)
- [155] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008. [68](#), [74](#)
- [156] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. *arXiv preprint arXiv:1905.04398*, 2019. [76](#)
- [157] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018. [76](#)
- [158] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA, 2010. [66](#)
- [159] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, May 2002. [66](#)
- [160] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Machine Learning Research*, 12:2825–2830, 2011. [67](#)
- [161] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2020. [72](#)
- [162] Puneet Mangla, Nupur Kumari, Abhishek Sinha, Mayank Singh, Balaji Krishnamurthy, and Vineeth N Balasubramanian. Charting the right manifold: Manifold mixup for few-shot learning. In *The IEEE Winter Conference on Applications of Computer Vision*, pages 2218–2227, 2020. [72](#)
- [163] Chi Zhang, Henghui Ding, Guosheng Lin, Ruibo Li, Changhu Wang, and Chunhua Shen. Meta navigator: Search for a good adaptation policy for few-shot learning. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2021. [77](#)