공학박사 학위논문

Fast Adaptation of Deep Learning Vision Applications with Limited Data for Edge Device

엣지 장비를 위한 한정된 데이터를 가지는 딥러닝 비전 어플리케이션의 빠른 적응

2022 년 2 월

서울대학교 대학원

컴퓨터공학부

이 태 민

Fast Adaptation of Deep Learning Vision Applications with Limited Data for Edge Device

엣지 장비를 위한 한정된 데이터를 가지는 딥러닝 비전 어플리케이션의 빠른 적응

지도교수 유 승 주

이 논문을 공학박사 학위논문으로 제출함

2021 년 11 월

서울대학교 대학원

컴퓨터공학부

이 태 민

이태민의 공학박사 학위논문을 인준함

2021 년 12 월

위 원 장 이 재 욱 (인)

부위원장 유 승 주 (인)

위 원 이 영 기 (인)

위 원 이 영 주 (인)

위 원 김 용 덕 (인)

Abstract

Fast Adaptation of Deep Learning Vision Applications with Limited Data for Edge Device

Taemin Lee

Department of Computer Science and Engineering

The Graduate School

Seoul National University

The remarkable success of deep learning-based methods are mainly accomplished by a large amount of labeled data. Compared to conventional machine learning methods, deep learning-based methods are able to learn high quality model with a large dataset size. However, high-quality labeled data is expensive to obtain and sometimes preparing a large dataset is impossible due to privacy concern. Furthermore, human shows outstanding generalization performance without a huge amount of labeled data.

Edge devices have a limited capability in computation compared to servers. Especially, it is challenging to implement training on edge devices. However, training on edge device is desirable when considering

domain-shift problem and privacy concern. In this dissertation, I consider adaptation process as a conventional training counterpart for low computation capability edge device.

Conventional classification assumes that training data and test data are drawn from the same distribution and training dataset is large. Unsupervised domain adaptation addresses the problem when training data and test data are drawn from different distribution and it is a problem to label target domain data using already existing labeled data and models. Few-shot learning assumes small training dataset and it is a task to predict new data based on only a few labeled data. I present 1) co-optimization of backbone network and parameter selection in unsupervised domain adaptation for edge device and 2) augmenting few-shot learning with supervised contrastive learning. Both methods are targeting low labeled data regime but different scenarios.

The first method is to boost unsupervised domain adaptation by co-optimization of backbone network and parameter selection for edge device. Pre-trained ImageNet models are crucial when dealing with small dataset such as Office datasets. By using unsupervised domain adaptation algorithm that does not update feature extractor, large and powerful pre-trained ImageNet models can be used to boost the accuracy. We report state-of-the-art accuracy result with the method. Moreover, we conduct an experiment to use small and lightweight pre-trained ImageNet models for edge device. Co-optimization is performed to reduce the to-

tal latency by using predictor-guided evolutionary search. We also consider pre-extraction of source feature. We conduct more realistic scenario for edge device such as smaller target domain data and object detection. Lastly, We conduct an experiment to utilize intermediate domain data to reduce the algorithm latency further. We achieve 5.99× and 9.06× latency reduction on Office31 and Office-Home dataset, respectively.

The second method is to augment few-shot learning with supervised contrastive learning. We cannot use pre-trained ImageNet model in the few-shot learning benchmark scenario as they provide base dataset to train the feature extractor from scratch. Instead, we augment the feature extractor with supervised contrastive learning method. Combining supervised contrastive learning with information maximization and prototype estimation technique, we report state-of-the-art accuracy result with the method. Then, we translate the accuracy gain to total runtime reduction by changing the feature extractor and early stopping. We achieve 3.87× latency reduction for transductive 5-way 5-shot learning scenarios.

Our approach can be summarized as boosting the accuracy followed by latency reduction. We first upgrade the feature extractor by using more advanced pre-trained ImageNet model or by supervised contrastive learning to achieve state-of-the-art accuracy. Then, we optimize the method end-to-end with evolutionary search or early stopping to reduce the latency. Our two stage approach which consists of accuracy boosting and latency reduction is sufficient to achieve fast adaptation of deep learning

vision applications with limited data for edge device.

Keywords: Neural network, edge device, unsupervised domain adaptation, few-shot learning, pseudo labeling, contrastive learning, information maximization

Student Number: 2015-31052

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The exceptional success of deep learning-based methods are mainly achieved by increasing size of labeled data. However, large labeled data by human annotation is hard to obtain and sometimes it is not possible because of privacy concern. On the other hand, human shows excellent generalization performance without a large labeled dataset, bringing motivation to the field of unsupervised domain adaptation and few-shot learning. Typical classification tasks assume that training data and test data are drawn from the same distribution and the training dataset is large. Unsupervised domain adaptation task relieves the same distribution assumption and deals with the situation that training data and test data are drawn from the different distribution. Few-shot learning also relieves the large training dataset assumption and classifies the new data with a few labeled examples per class. Therefore, both tasks aim to address non-ideal cases of low labeled data count but with different scenarios.

Unsupervised domain adaptation aims to address the problem by reusing already existing labeled data and models to label target domain data. The

source domain has a labeled dataset while the target domain has unlabeled dataset and the goal of the methods is to label the target domain dataset. The problem is that when the model trained by source domain dataset is tested using target domain dataset, domain shift problem occurs and degrades the models' performance. Therefore, it is required to reduce domain shift problem by aligning distributions of source and target domain.

The aim of few-shot learning is to classify unlabeled data based on the observation of a few labeled data. Given a labeled base dataset and a novel dataset, the objective of a few-shot learning task is to build a visual model using the base dataset and to generalize to the novel dataset which has only a few training images per class. A transductive few-shot learning is in introduced to address the low data count. The setting allows that the model can access all the test data at once instead of one by one in the inference stage.

The speed of the adaptation algorithm is important in some working scenarios. For example, we consider driving a car into the tunnel. Characteristics of environment change drastically and they are maintained for several minutes. The car cannot rely on server assistance due to high communication cost. In such scenario, edge devices in the car need to adapt to new environment by itself in the order of multiple seconds. Another example is about personalization which aims to provide customized service to individuals based on private data. Each user has its own con-

text or domain which is based on its own environment to which algorithm needs to perform domain adaptation to adapt. The local data cannot be transferred to the centralized server due to the privacy concern. Indeed, we suppose each edge device has its own private data and perform adaptation. The adaptation algorithm needs to be optimized to provide interactive service. Note that both car environment and user response are not ultra fast (e.g., in the order of microseconds). Still, fast adaptation is important in such scenarios and it is critical to minimize computation cost of adaptation as edge devices have a low computation capability compared to typical servers.

Note that small data has not always adverse effect. We assume data in each edge device is small and private because each edge device tries to solve meaningful problem without transferring data to server due to privacy concern and communication cost. If data is small, performing adaptation to new data is relatively fast as typical execution time of algorithm is proportional to data size. We keep the positive effect of limited data in mind and develop approaches for the scenarios when the dataset is not large.

In this dissertation, I propose two approaches to address low data count. I present 1) co-optimization of backbone network and parameter selection in unsupervised domain adaptation for edge device and 2) augmenting few-shot learning with supervised contrastive learning. Both methods have similar two stage approach that consists of accuracy boost-

ing and latency reduction to support fast adaptation of deep learning applications with limited data for edge device.

The first method is to boost unsupervised domain adaptation by co-optimization of backbone network and parameter selection for edge device. The first stage of the first method is to boost the accuracy as much as possible. The quality of feature extractor is crucial thereby we use pre-trained ImageNet models to handle small datasets like Office datasets. However, some of the unsupervised domain adaptation methods are not resilient with advanced feature extractors due to complex and unreproducible feature extractor training protocols. Inspired by the existing work [1], we use large and heavyweight feature extractors combined with an algorithm without updating the feature extractor part. We report the accuracy surpassing the state-of-the-art with EfficientNet [2] and Vision Transformer [3] feature extractor combined with selective peseudo labeling (SPL) [4] algorithm.

The second stage of the first method is to reduce the latency while maintaining the baseline accuracy. We experiment that uses small and lightweight pre-trained ImageNet models named once-for-all (OFA) [5] network and validate the fact that there is a few accuracy margin which can be translated to latency reduction. We conduct co-optimization to control both the feature extractor and SPL algorithm parameter to reduce the total latency of the algorithm. Because of the large search space, a predictor-guided evolutionary search is implemented to efficiently find

the appropriate feature extractor and SPL algorithm parameter settings. We also experiment transferability between Office31 dataset and Office-Home dataset due to the cost of data collection is high. We assume pre-extraction of source feature to relieve privacy concern and propose a method to store source feature along to each OFA networks instead of specialized OFA network. We also conduct more realistic scenario for edge device such as smaller target domain data and object detection. Lastly, We conduct an experiment to utilize intermediate domain data to reduce the algorithm latency further. We report 5.99$\times$ and 9.06$\times$ latency reduction on Office31 and Office-Home dataset, respectively.

The second method is to augment few-shot learning with supervised contrastive learning. Again, the first stage of the second method is to boost the accuracy of the few-shot learning scenarios. In the few-shot learning scenarios, base dataset is provided to train the feature extractor from scratch thereby we cannot use pre-trained ImageNet models. Instead of training the feature extractor using cross-entropy loss, we propose to use supervised contrastive learning [6] technique to augment the quality of the feature extractor. We observe that supervised contrastive loss instead of cross-entropy loss is especially effective in the low data regime. We combine prototype estimation [7, 8] and transductive information maximization (TIM) [9] with supervised contrastive learning to achive state-of-the-art accuracy results. Furthermore, we show that a large dataset is needed to address domain shift degradation and if our method

is applied, the need for a large dataset is removed.

the second stage of the second method is to reduce the latency of the few-shot learning scenarios. We observe that MobileNet backbone network with our method surpasses the baseline ResNet backbone model and we use the accuracy margin to be translated to runtime reduction. By applying early stopping with MobileNet backbone network, We achieve $3.87\times$ latency reduction for transductive 5-way 5-shot learning scenarios.

In our method, characteristics of edge device (e.g., memory bandwidth and small neural processing unit) are not considered. Instead, we report latency measured on working edge system (i.e., NVIDIA Jetson TX2). More sophisticated method which considers edge device characteristics is left for future work. This dissertation is organized as follows. Chapter 2 introduces background. Chapter 3 presents problem definitions and solutions overview. Chapter 4 explains the proposed method to boost unsupervised domain adaptation with pre-trained ImageNet models for edge device. Chapter 5 explains the proposed method to augment few-shot learning with supervised contrastive learning. Chapter 6 concludes the dissertation.

Chapter 2

Background

2.1 Dataset Size for Vision Applications

Dataset size is important for machine learning. Especially deep learning methods are known for learning better classifier compared to classical machine learning methods when the dataset size is large. Table 2.1 shows image count of famous datasets from several vision applications namely classification, unsupervised domain adaptation and few-shot learning.

There are diverse datasets with diverse image count. For example, the image count of datasets for unsupervised domain adaptation varies from 4,110 (i.e., Office31) to 569,010 (i.e., DomainNet). The image count of datasets for few-shot learning varies from 11,788 (CUB) to 779.165 (Tiered-ImageNet). In this dissertation we concentrate on low data count regime thereby Office31, Office-Home, CUB, and Mini-ImageNet are our main focus.

Table 2.1 Dataset size for vision applications

Applications	Dataset	Image count
Classification	ILSVRC2012 [10] (ImageNet)	1,281,167
Unsupervised domain adaptation	Office31 [11]	4,110
	Office-Home [12]	15,588
	Syn2Real [13]	280,157
	DomainNet [14]	569,010
Few-shot learning	CUB [15]	11,788
	Mini-ImageNet [16]	60,000
	Tiered-ImageNet [17]	779,165

2.2 ImageNet Pre-trained Models

The performance and the size of model are also important for implementing deep learning based method. Especially when the dataset size is small, pre-training the model with ImageNet is widely used. We report the number of parameters for feature extraction, the number of multiply-accumulate operations for feature extraction, the size of input image and accuracy on ImageNet validation dataset of the models in Table 2.2 and Table 2.3. We measure the cost of feature extraction instead of end-to-end cost because we use other methods for classifier in the subsequent chapters. Note that the results are reproduced and may differ from the result reported by papers.

Table 2.2 lists various ImageNet pre-trained models including ResNet [18], MobileNet [19, 20], NASNet [21], Inception [22], Xception [23], EfficientNet [2]. The accuracy of models ranges from 71.9 (i.e., Mobilenet_v2) to 88.4 (i.e., Efficientnet_l2_ns). Also the number of parameters of models ranges from 2.2 millions (i.e., Mobilenet_v2) to 474.4 millions. The number of multiply-accumulate operations ranges from 152.2 millions (i.e., MobileNetv3_large_075) to 478,888.5 millions (i.e., Efficientnet_l2_ns) The input size ranges from typical 224 to 800 (i.e., Efficientnet_l2_ns). Note that the accuracy of EfficientNet surpasses ResNet baseline by a large margin.

Table 2.3 lists various OFA networks [5] and specialized OFA net-

Table 2.2 ImageNet pre-trained models for feature extraction

Models	Parameters (M)	MAC (M)	Input size	Accuracy
Resnet50	23.5	4,109.5	224	76.13
Mobilenetv3_large_100	4.2	214.9	224	75.5
Mobilenetv3_large_075	2.7	152.2	224	73.4
Nasnetalarge	84.7	23,922.5	331	82.6
Inceptionresnetv2	54.3	13,193.1	299	80.4
Inceptionv4	41.1	12,287	299	80.2
Xception	20.8	8,397.8	299	79
Nasnetamobile	4.2	577.6	224	74.1
Mobilenet_v2	2.2	312.9	224	71.9
Efficientnet_l2_ns	474.7	478,888.5	800	88.4
Efficientnet_l2_ns_475	474.7	172,028.7	475	88.2
Efficientnet_b7_ns	63.7	38,191.9	600	86.8
Efficientnet_b6_ns	40.7	19,294	528	86.5
Efficientnet_b5_ns	28.3	10,389.6	456	86.1
Efficientnet_b8_ap	84.5	63,316.8	672	85.4
Efficientnet_b4_ns	17.5	4,435.5	380	85.2
Vit_large_patch16_384	304	174,701.7	384	87.1
Vit_large_patch16_224	304	59,646.3	224	85.8
Vit_large_patch32_384	306.4	44,241.7	384	81.5

Table 2.3 Specialized OFA models for feature extraction

Models	Parameters (M)	MAC (M)	Input size	Accuracy
ofa_mbv3_w1.0	6.4	583.7	224	77.4
ofa_mbv3_w1.2	9.2	866.4	224	79.0
ofa_proxyless_w1.3	6.3	1,030	224	77.7
note10@80.2	7.6	765	260	80.2
note10@79.7	7.5	570.9	220	79.7
note10@79.3	7.5	469.5	220	79.3
note10@78.4	5.9	349.3	224	78.4
note10@76.6	4.6	244.6	224	76.6
note10@75.5	3.7	167.6	192	75.5
note10@73.6	3	113	176	73.6
note10@71.4	2.5	78.1	160	71.4
pixel1@76.9	4.5	237	220	76.9
LG-G8@76.4	4.5	237	220	76.4
1080ti@76.4	4.8	406	188	76.4
s7edge@76.3	5.1	225.7	192	76.3
note8@76.1	4	226.4	204	76.1
v100_gpu64@76.1	4.5	359.6	192	76.1
pixel2@75.8	4.5	214.2	208	75.8
tx2_gpu16@75.8	4.5	357	172	75.8

works. The first row group shows three OFA networks which are used to derive specialized OFA networks for target hardware platform. The second row group presents specialized OFA networks for Samsung Note10 cell phone with various accuracy. The third row group lists the specialized OFA networks that have similar accuracy with ResNet for various hardware platforms. The accuracy of models ranges from 71.4 (i.e., note10@71.4) to 80.2 (note10@80.2). The number of parameters of models ranges from 2.5 millions (i.e., note10@71.4) to 9.2 millions (i.e., ofa_mbv3_w1.2). The number of multiply-accumulate operations ranges from 78.1 millions (i.e., note10@71.4) to 1,030 millions (i.e., ofa_proxyless_w1.3). The input size ranges from 160 (i.e., note10@71.4) to 260 (note10@80.2). Note that the accuracy of specialized OFA networks surpasses ResNet baseline while maintaining smaller parameters and MAC.

2.3 Augmentation Methods for ImageNet

We introduce several augmentation methods on ImageNet large scale visual recognition challenge. RandAugment [24] shows a high level of accuracy with a reduced search space thus it is practical to use. The algorithm has two parameters N and M. The parameter N denotes the number of data augmentation functions used and the parameter M denotes the strength of the data augmentation. Both parameters have similar intention that the larger parameters incurs the more regularization strength.

The code follows.

```
transforms = [
'Identity', 'AutoContrast', 'Equalize',
'Rotate', 'Solarize', 'Color', 'Posterize',
'Contrast', 'Brightness', 'Sharpness',
'ShearX', 'ShearY', 'TranslateX', 'TranslateY']

def randaugmnet(N, M):
    sampled_ops = np.random.choice(transforms, N)
    return [(op, M) for op in sampled_ops]
```

The parameters N and M can be determined by simple grid search. RandAugment shows great success on various datasets achieving similar accuracy with heavyweight data augmentation methods.

CutMix [25] is another very effective method to augment ImageNet accuracy. It combines two training data to generate new training sample for better generalizability of the model. CutMix algorithm is defined as follows.

$$\tilde{x} = \mathbf{M} \odot x_A + (\mathbf{1} - \mathbf{M}) \odot x_B$$

$$\tilde{y} = \lambda y_A + (1 - \lambda) y_B$$

where $\mathbf{M}$ denotes a binary mask that specifies where to remove and copy from two images, $\mathbf{1}$ is a binary mask filled with one, $\odot$ denotes element-

wise multiplications, and λ denotes a combination ratio between two data which.

The algorithm first samples the bounding box coordinates (r_x, r_y, r_w, r_h) by using uniform sampling where the combination ratio is sampled from the beta distribution.

$$r_x = Unif(0, W), \quad r_w = W\sqrt{1-\lambda}$$
$$r_y = Unif(0, H), \quad r_h = H\sqrt{1-\lambda}$$

CutMix shows great success on various datasets including ImageNet showing 2.28% accuracy improvements on ResNet.

In our experiment for few-shot learning, however, the accuracy gain is marginal indicating that the accuracy boosting methods for ImageNet is not always adaptable for few-shot learning scenarios.

2.4 Contrastive Learning

Contrastive learning is a method to pull similar samples (positives) and push dissimilar samples (negatives) apart in the embedding space. We introduce SimCLR [26] as a representative for contrastive learning. SimCLR consists of data augmentation module, base encoder, projection head and contrastive loss. A data augmentation module is used to generate two views of the same sample, which is to create positives in the embedding space. Random cropping, color distortions and random Gaussian

blur are included in the data augmentation module. Note that the combination of random crop and color distortion is important to boost the accuracy. A base encoder is a neural network that extracts features from augmented data. SimCLR uses ResNet for simplicity. A projection head is an MLP with one hidden layer that converts features to space where contrastive loss is applied. A contrastive loss is a measure to pull similar samples and push dissimilar samples apart in the embedding space. The self-supervised contrastive loss is defined as follows.

$$L^{self} = -\sum_{i \in I} \log \frac{\exp\left(z_i \cdot z_{j(i)} / \tau\right)}{\sum_{a \in A(i)} \exp\left(z_i \cdot z_a / \tau\right)}$$

$z_l = Proj(Enc(\tilde{x}_l))$ is a representation of the input in the embedding space, the $\cdot$ symbol denotes the inner product, τ is a scalar temperature parameter, and $A(i) \equiv I \setminus i$ where the index i denotes anchor, the index $j(i)$ denotes the positive, and the other indices denote negatives. Note that the inner product operation measures the similarity between two operands as the result value is larger if two operands are more similar. The numerator of the loss is larger if an anchor and the positive is similar. On the other hand, the denominator of the loss is larger if an anchor and remaining negatives are dissimilar. As gradients flow in the direction which loss becomes smaller, it encourages to pull the positive and push the negative apart.

SimCLR shows great success on ImageNet especially when the smaller

fraction of label is used. For example, SimCLR achieves 85.8% top-5 accuracy on ImageNet with only 1% of labels while it achieves 92.6% top-5 accuracy on ImageNet with 10% of labels.

Chapter 3

Problem Definitions and Solutions Overview

3.1 Problem Definitions

This section presents the formulation of unsupervised domain adaptation task and few-shot learning task in detail. Unsupervised domain adaptation task aims to address the problem when training data and test data are drawn from different distribution. Few-shot learning task addresses the problem when there is only a few labeled data per class. Note that an ideal classification tasks have the same distribution between training dataset and test dataset and there are plenty of labeled dataset for training. Both methods try to deal with non-ideal cases but different scenarios.

3.1.1 Unsupervised Domain Adaptation

The source domain S has a labeled dataset $D^S = (x_i^S, y_i^S), i = 1, 2, ..., n_S$, where $x_i^S \in \mathbb{R}^d$ denotes the feature vector of i-th labeled sample in the

source domain, d is the dimension of the feature vector and $y_i^S \in \mathscr{Y}^S$ represents the source domain label. Unsupervised domain adaptation is designed to label an unlabeled dataset $D^T = x_i^T, i = 1,2,...,n_T$ from the target domain T, $x_i^T \in \mathbb{R}^d$ denotes the feature vector of i-th unlabeled sample in the target domain. Note that the space of the target label $\mathscr{Y}^T$ is the same with the space of the source domain $\mathscr{Y}^S$. Both the labeled source domain data and the unlabeled target domain data are accessible in the model training phase unless the setting considers pre-extraction of source feature. If the setting considers pre-extraction of source feature, then the model training is two-phase. Only the source domain image is available in the first phase while only the target domain image is available in the second phase.

3.1.2 Few-shot learning

Given a labeled base dataset $\mathbf{D}_{\text{base}} := \{(\mathbf{x}_\text{i}, \mathbf{y}_\text{i}), \mathbf{y}_\text{i} \in \mathbf{C}_{\text{base}}\}$ and a novel dataset $\mathbf{D}_{\text{novel}} := \{(\mathbf{x}_\text{i}, \mathbf{y}_\text{i}), \mathbf{y}_\text{i} \in \mathbf{C}_{\text{novel}}\}$ where $\mathbf{C}_{\text{base}} \cap \mathbf{C}_{\text{novel}} = \emptyset$, the goal of a few-shot learning task is to train a visual model using the base dataset $\mathbf{D}_{\text{base}}$ and to generalize to the novel dataset $\mathbf{D}_{\text{novel}}$ which has a few training images per class. At inference, each few-shot learning task episode consists of a support set and a query set sampled from the novel dataset. The support set (S) is labeled and includes K samples per class with N classes (N-way K-shot setting), whereas the query set (Q) includes T

samples per class with the same N classes without data labels. The goal is to map the samples in the query set to the desired label using the information gained from the support set. In the transductive setting, the model can access the entire dataset including the query set (i.e., $N \times K + N \times T$ samples) at once instead of one by one (i.e., $N \times K + 1$ samples each) in the traditional inductive setting.

3.2 Solutions overview

This section presents the proposed idea for unsupervised domain adaptation task and few-shot learning task briefly. Both solutions have a similar approach. Because the quality and latency of feature extractor is critical in both unsupervised domain adaptation and few-shot learning, we upgrade the feature extractor using ImageNet pre-trained models or supervised contrastive learning to boost the accuracy. Then, we perform co-optimization by evolutionary search or early stopping to reduce the latency while maintaining the baseline accuracy. The solution for unsupervised domain adaptation is applied in situations which we can use a large (i.e., ImageNet) pre-trained model while the solution for few-shot learning is applied in situations which we train the feature extractor from scratch instead of ImageNet pre-trained models. Especially, both solutions are effective in limited data regime.

3.2.1 Co-optimization of Backbone Network and Parameter Selection in Unsupervised Domain Adaptation for Edge Device

We use highly optimized ImageNet pre-trained feature extractor, namely EfficientNet [2] and Vision Transformer [3] for accuracy boosting and once-for-all (OFA) [5] network which is designed for edge device. With EfficientNet feature extractor, we report state-of-the-art 95.8% and 91.5% accuracy on Office31 and Office-Home dataset, respectively. With OFA feature extractor, we report 2.6% and 2% higher accuracy than baseline ResNet on Office31 and Office-Home dataset, respectively. We use the accuracy margin to reduce the latency of the total algorithm afterward. There are two representative ways to reduce the latency. One is to change the feature extractor with less heavyweight network. The other is to tune the SPL algorithm with smaller parameters. We conduct a co-optimization problem using predictor-guided evolutionary search algorithm to set the feature extractor and the SPL parameters. Considering the fact that training predictors are expensive, we also experiment transferability between Office31 and Office-Home dataset where predictors are trained on one dataset and testing is performed on the other dataset. Lastly, pre-extraction of source feature is considered and we propose to store features only for OFA networks instead of all the OFA specialized networks. We report 5.99$\times$ and 9.06$\times$ latency reduction on Office31 and

Office-Home dataset, respectively.

3.2.2 Augmenting Few-Shot Learning with Supervised Contrastive Learning

We use supervised contrastive learning [6] to augment the feature extractor following the few-shot learning base dataset training protocol. After the feature extractor is trained on supervised contrastive learning, 5-epochs of fine-tuning process is followed. At the beginning of TIM [9] algorithms, prototype estimation method [7, 8] determines initialization points by combining support set examples and query set examples instead of the simple mean of support set examples. We report state-of-the-art results 78.83% (87.76%) for 1-shot (5-shot) few-shot learning tasks on Mini-ImageNet dataset and 88.81% (93.11%) for 1-shot (5-shot) few-shot learning tasks on CUB dataset. Note that the accuracy we obtain on CUB dataset is higher than the accuracy that is domain-shifted by a large dataset (i.e., Tiered-ImageNet) thereby removing the need for a large base dataset. We observe that our methods with MobileNet surpasses the baseline ResNet by 2.6% (0.8%) for 1-shot (5-shot) classification on Mini-ImageNet. We exploit the accuracy gain to be translated to latency reduction by using early-stopping for TIM algorithm. By doing so, we report 3.87$\times$ latency reduction for transductive 5-way 5-shot learning scenarios.

Chapter 4

Co-optimization of Backbone Network and Parameter Selection in Unsupervised Domain Adaptation for Edge Device

Unsupervised domain adaptation deals with the absence of target domain data label which incurs performance degradation with a simple transfer method. Especially for edge device, little work was done to address how various pre-trained ImageNet models influence the performance of algorithms targeted to solve unsupervised domain adaptation. We propose a model selection approach for unsupervised domain adaptation scenarios. With the fact that the speed and accuracy of the feature extractor is a critical factor in unsupervised domain adaptation, we select the feature extractor with stronger pre-trained ImageNet models which are designed to run efficiently for edge devices. Furthermore, we adjust the parameters of unsupervised domain adaptation algorithm by co-optimization. At first, we demonstrate that gradient-based approaches are insufficient for unsupervised domain adaptation with small dataset because the feature

extractor scaling is limited. Instead, we use a pseudo-labeling algorithm with a large model to boost the performance and report the new state-of-the-art accuracy for Office datasets. We also investigate the impact of neural architecture search-based models with the pseudo-labeling algorithm for higher runtime efficiency. In this case, NVIDIA Jetson TX2 platform is used to report the execution time of the algorithm.

4.1 Introduction

The recent success of deep neural networks is mainly supported by a large labeled dataset such as ImageNet [10]. However, the labeled dataset to train the deep model is often hard to archive because human labeling is expensive and it raises a privacy concern. Unsupervised domain adaptation tries to relive the problem by reusing already existing labeled data and models to label new data. The challenge is that when using the model trained on one dataset to test another, domain shift problem occurs and degrades the model's accuracy. Therefore, the methods for unsupervised domain adaptation are required to reduce the domain shift problem by typically aligning marginal or conditional distribution of source and target domains.

The quality of feature extractor is crucial when building a better model for classification. The recent improvement on ImageNet classification provides better feature extractors off-the-shelf. Therefore, it is

natural to apply the state-of-the-art image classification model to unsupervised domain adaptation context. However, some of the methods are not adaptable because of the complex and unreproducible feature extractor training protocol. It is recommended to use unsupervised domain adapatation methods without the need to fine-tune the feature extractor part when applying the heavyweight feature extractors. Inspired by the existing work [1], which summarized the effect of various ImageNet models on domain adaptation scenarios, we report the accuracy surpassing the state-of-the-art by using the heavyweight EfficientNet [2] and Vision Transformer [3] feature extractor and Selective Pseudo Labeling (SPL) [4] algorithm.

Although the heavyweight feature extractor is sufficient to achieve new state-of-the-art accuracy, the problem is raised when it is applied to edge devices. Considering the fact that Office dataset, which is commonly used to evaluate the performance of unsupervised domain adaptation, is small compared to ImageNet, it is reasonable to implement the entire pipeline of the algorithm on edge devices. We show that the cost of feature extraction is dominant with the heavyweight feature extractors. If an edge device is connected to server with high speed bandwidth, then all the new data can be delivered to server for annotation and the edge device merely performs a terminal. However, when an edge device is not connected to server or data is not delivered due to the privacy concern, then the entire pipeline of the algorithm is contained in the edge device and

the efficiency of both the feature extractor and the back-end algorithm becomes crucial.

For efficient feature extractor, neural architecture search is proposed to build neural network models for edge devices. Conventional approach is to design a neural network with hand-crafted sub-modules designed by experts. The process of neural network design is indeed labor-intensive and expensive. Neural architecture search automates the selection of deign choices by mathematical formulation (e.g., minimizing loss function). Therefore, it is possible to apply neural architecture search algorithm on each of diverse hardware with small additional cost. Specifically, we consider Once-for-All (OFA) [5] neural architecture search algorithm because the final models designed by OFA algorithm is highly efficient on diverse hardware. Furthermore, OFA releases off-the-shelf neural network models in various hardware and various latency target.

For efficient implementation of SPL algorithm, we consider adjusting the parameters of the algorithm. There are main parameters that affect model runtime significantly. The problem is to select which and how to maximize the efficiency of the entire algorithm. Note that the feature extractor selection is also a critical factor for the accuracy and runtime of the entire algorithm pipeline. We examine that applying OFA neural architecture search to find a network model based on SPL accuracy instead of ImageNet accuracy is based on fallacy. Rather, the use of pre-trained ImageNet model from OFA is a reasonable design choice. Based on the

observation, we propose an co-optimization method to jointly select the feature extractor and the SPL parameters. The co-optimization algorithm is based on the predictor-assisted evolutionary search to minimize the search cost.

In recent years, unsupervised domain adaptation with a lack of access to source data is proposed [27]. The algorithm trains the target-specific feature extractor by information maximization and self-supervised pseudo-labeling. Such source-free regime is also adaptable for the unsupervised domain adaptation algorithms without fine-tuning the feature extractor. Using the fact that the algorithms require source domain data features not the source domain data itself, storing the source data features just beside the feature extractor is sufficient. We name such scheme pre-extraction of source feature. However, pre-extraction of source feature combined with the neural architecture search is not straightforward because the neural architecture search produces diverse set of specialized feature extractors to support diverse hardware. Storing all the features from all the diverse set of feature extractor is inefficient. We propose a simple practical approach to support the diverse set of specialized feature extractor (i.e., storing only one representative feature).

We also conduct more realistic scenario for edge device such as smaller target domain data and object detection. Lastly, We conduct an experiment to utilize intermediate domain data to reduce the algorithm latency further.

In our method, characteristics of edge device (e.g., memory bandwidth and small neural processing unit) are not considered. Instead, we report latency measured on working edge system (i.e., NVIDIA Jetson TX2). More sophisticated method which considers edge device characteristics is left for future work.

In summary, the contributions of our study are as follows:

- We propose the use of a heavyweight feature extractor with an unsupervised domain adaptation algorithm without fine-tuning feature extractor to boost the classification accuracy on Office dataset.

- We show the implementation of neural architecture search algorithm on unsupervised domain adaptation is not straightforward and we propose evolutionary search based co-optimization approach to jointly select the feature extractor and the algorithm parameters on Office dataset.

- We propose to store only one representative feature from the OFA network instead of storing all the features from diverse set of specialized networks to support diverse set of hardware.

- We show that our method is effective on more realistic scenario which includes smaller target domain and object detection.

- We propose that our method can be faster by fitting a classifier using intermediate domain data.

4.2 Related Works

Unsupervised domain adaptation: To alleviate domain shift problem, unsupervised domain adaptation methods are proposed. There are traditional approaches [28–31] and deep learning-based approaches [32–35] in unsupervised domain adaptation.

Traditional approaches mainly utilize extracted features from data images. Previously lower-level SURF features are widely used in domain adaptation [28]. After the emergence of deep neural networks, pretrained ImageNet models are used to extract features from data images (Alexnet [36], Decaf [30], Resnet50 [37], Xception [31], Nasnetlarge [1]). Note that the traditional approaches don't fine-tune the deep neural networks (i.e., feature extractor part). Performance of the features extracted from deep neural networks typically exceed that from SURF features. Furthermore, it is reasonable to conclude that the ImageNet accuracy of the deep neural network has some positive correlation with the accuracy of unsupervised domain adaptation [1]. [38] proposes transfer component analysis for domain adaptation and it learns transfer components across domains in a reproducing kernel Hilbert space using maximum mean discrepancy. [39] presents kernel distribution embedding and Hilbert-Schmidt independence criterion based method to reduce the dimensionality of the data while it preserves the structural information. [40] introduces transfer joint matching approach to formulate a joint op-

timization problem of feature matching and instance reweighting. [29] proposes a kernelized local-global approach to consider the domain adaptation problem as a bi-object optimization problem via the kernel method. [41] proposes a method to extract conditional transferable components whose conditional distribution is invariant after proper location-scale transformations. [42] presents a general cross-domain learning framework which uses the intra-affinity of classes to perform intra-class knowledge transfer. [43] introduces balanced distribution adaptation method that can adaptively leverage the importance of the marginal and conditional distribution discrepancies and also proposes weighted balanced distribution adaptation which tackles the class imbalance problem. [30] proposes a manifold embedded distribution alignment approach to learn a domain-invariant classifier in Grassmann manifold with structural risk minimization. [44] presents structural correspondence learning to automatically induce correspondences among features from different domains [40] introduces a novel transfer joint matching approach to model both feature matching and instance reweighting in a unified optimization problem to reduce the domain difference by a principled dimensionality reduction procedure. SPL [4] algorithm is proposed to tackle inaccurate pseudo-labeling problem by selective pseudo-labeling strategy based on structured prediction.

In recent years, deep learning-based methods are used to build a more effective feature representation by gradient updates. [33] proposes a Con-

volutional Neural Network (CNN) architecture that introduces an adaptation layer and an additional domain confusion loss. [34] presents a deep adaptation network architecture that generalizes deep convolutional neural network to the domain adaptation scenarios by embedding hidden representations of all task-specific layers. [45] introduces multi-task autoencoder which extends the standard denoising autoencoder framework by substituting artificially induced corruption. [35] proposes a neural network architectures which are trained on labeled data from the source domain and unlabeled data from the target domain not to discriminate between the source and target domains. [46] presents adversarial discriminative domain adaptation which combines discriminative modeling, untied weight sharing, and a generative adversarial network [47] loss. [48] introduces domain separation network that learns to extract image representations which are private to each domain and shared across domains inspired by private-shared component analysis. [49] proposes Batch Spectral Penalization (BSP) which is a general approach for penalizing the largest singular values to enhance transferability.

ImageNet models: Many deep nerual network architectures are trained for ImageNet classification task. [50] presents AlexNet that consists of five convolutional layers followed by max-pooling layers and three fully connected layers. [51] introduces a network architecture with very small convolution filters and pushes the depth to 16-19 weight layers. [52] proposes SqueezeNet, which is a small deep neural network architecture

to achieve AlexNet [50] accuracy with 50 times fewer parameters. [53] introduces GoogLeNet that consists of 22 layers and carefully crafted design with increasing the depth and width of the network while keeping the computational budget constant. [54] presents ShuffleNet, which adapts pointwise group convolution and channel shuffle to reduce computation cost while maintaining the accuracy for mobile devices. [18] proposes residual learning framework to ease the training of deep networks with a depth of up to 152 layers by reformulating the layers as learning residual functions. [19] introduces MobileNetV2 architecture that consists of inverted residual structure and linear bottleneck layers for mobile devices. [20] presents MobileNetV3 structure using a combination of complementary search techniques based on hardware-aware network architecture search complemented by the NetAdapt algorithm. [21] proposes NASNet architecture using the design of a new search space that enables transferability from CIFAR-10 dataset to ImageNet dataset. [55] introduces DenseNet, which connects each layer to every other layer in a feed-forward manner. [23] presents Xception architecture that Inception modules are replaced with depthwise separable convolutions. [22] proposes Inception-v4, which combines Inception architectures with residual connections. [2] introduces EfficientNet using neural architecture search to design a baseline network and adapting a novel scaling method that scales the model with carefully balancing network depth, width, and resolution. [56] presents RegNet using a network de-

sign space that parametrizes populations of networks instead of focusing on designing individual network instances. [3] proposes Vision Transformer, which deploys a pure transformer architecture applied directly to sequences of image patches.

Neural architecture search: Neural architecture search automates the deep neural network architecture design process [21, 57–60]. Initial neural architecture search algorithms [21, 58, 61] find the network architecture with a goal of maximum accuracy. Recent neural architecture search algorithms [62–64] take hardware efficiency into consideration. Furthermore, OFA [5] removes model retraining process to reduce GPU hours, dollars, and CO_2 emission. [57] proposes a method using a recurrent neural network (RNN) to generate the model descriptions of neural networks and train this RNN with reinforcement learning. [58] presents AmoebaNet using tournament selection evolutionary algorithm by introducing an age property to favor the younger genotypes. [59] introduces a framework toward efficient architecture search using reinforcement learning by exploring the architecture space based on the current network and reusing its weights. [60] proposes the continuous relaxation of the architecture representation allowing efficient search of the architecture using gradient descent. [61] presents a function-preserving transformation for efficient neural architecture search, which allows reusing previously trained networks and existing architectures. [62] introduces ProxylessNAS that can directly learn the architectures for large-scale tar-

get tasks and target hardware platforms by addressing the high memory consumption of differentiable neural architecture search. [63] proposes MnasNet for mobile neural network search, which explicitly incorporate model latency into the main objective. [64] presents FBNets family that optimizes convolutional neural network architecture for mobile devices with differentiable neural architecture search framework.

4.3 Methodology

This section presents the proposed idea in detail.

4.3.1 Examining an Unsupervised Domain Adaptation Method

In this study, we consider the selective pseudo labeling (SPL) unsupervised domain adaptation algorithm [4]. The algorithm aligns the conditional distribution of source domain and target domain using following techniques.

Firstly, the algorithm performs dimensionality reduction using principle component analysis (PCA) because of redundant information contained in the high dimensional feature vector. the redundant information incurs unnecessary computation. PCA is formulated as eigenvalue problem and solved by singular value decomposition (SVD) solver. Feature

vectors from the source domain and target domain are concatenated as a matrix $[x_1^S, ..., x_{n_S}^S, x_1^T, ..., x_{n_T}^T]$ and performs PCA to reduce dimnesionality of the matrix. Additional L2 normalization is performed to encourage samples of different domains to be distributed on the surface of the same hyper-sphere. The L2 normalization has a empirical evidence that it leads to better performance.

Second, the algorithm performs domain alignment using the supervised locality preserving projection (SLPP) [65] to learn a projection matrix $\mathbf{P}$ which aligns samples from different domains to the same latent subspace.

$$\min_{\mathbf{P}} \sum_{i,j} ||\mathbf{P}^T \bar{x}_i - \mathbf{P}^T \bar{x}_j||_2^2 \mathbf{M}_{ij}$$

where $\mathbf{P} \in \mathbb{R}^{d_1 \times d_2}$, $d_! \leq d_2$ is the dimensionality of the learned space, $\bar{x}_i$ is the i-th column of the labeled data matrix and $\mathbf{M_{ij}}$ is the similarity matrix and defined as follows.

$$\mathbf{M_{ij}} = \begin{cases} 1 & y_i = y_j \\ 0 & otherwise. \end{cases}$$

The objective of the projection is that samples from the sample class are projected adjacent to each other in the subspace. The problem is formulates as generalized eigenvalue problem and solved by sparse eigen solver.

Third, the algorithm performs pseudo-labeling. There are two methods of pseudo-labeling based on the distance definition. One is nearest

class prototype (NCP). In NCP, the class prototype for each class is defined as the mean of the samples with the same label. Then, the target samples are classified as the label with minimum Euclidean distance to the class prototype as the name nearest class prototype refers to. The other is structured prediction (SP). SP aims to exploit intrinsic structure information of the target samples. Structured prediction employs K-means clustering algorithm to generate clusters beforehand. The cluster centers constitute class prototype for the target domain. Afterwards, the algorithm assigns one-to-one match between the class prototypes from the source domain and the cluster centers from the target domain. The objective is to minimize the sum of all matched pairs. Therefore, the target domain samples are classified collectively following the clusters they are involved in. The algorithm combines two methods by choosing maximum value of the probabilities which the target sample belongs to class prototype from the source domain.

Fourth, the algorithm performs iterative learning strategy. Instead of using full set of pseudo labels the algorithm generates, only pseudo labels with probability greater than threshold are used. One drawback of the iterative learning strategy is that it only selects labels from the specific class. Therefore, the algorithm implements the class-wise selection strategy to assign each class from the target domain have the same importance.

The time complexity of PCA is $O(dn^2 + d^3)$ approximately and the

time complexity of SLPP is $O(T(2d_1n^2 + d_1^3))$.

4.3.2 Boosting Accuracy with Pre-Trained ImageNet Models

We examine various ImageNet models for boosting unsupervised domain adaptation algorithm. One of them is EfficientNet [2] which brings new state-of-the-art result. EfficientNet proposes compound model scaling which consists of depth, width and resolution scaling. Scaling up any dimension of the neural network improves accuracy, however the accuracy gain saturates soon. Therefore, it is important to scale up three dimensions carefully to maximize the accuracy gain. The compound scaling method is defined as follows.

$$\text{depth: } d = \alpha^{\phi}$$

$$\text{width: } w = \beta^{\phi}$$

$$\text{resolution: } r = \gamma^{\phi}$$

$$\text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$$

$$\alpha \geq 1, \beta \geq 1, \gamma \geq 1$$

where α, β, γ are assigned by a grid search and ϕ is a coefficient user can control to determine the amount of resources the neural network model can use. EfficientNet architecture is a family of networks that first assign a good baseline network and scales up with the compound scaling method. The baseline network is generated by a multi-objective neural

architecture search using the search space of [63] and use $ACC(m) \times [FLOPS(m)/T]^w$ as the optimization objective and called EfficientNet-B0. As the search space is the same with [63], the network is similar to MnasNet while EfficientNet-B0 is slightly bigger. Building block of the network is mobile inverted bottleneck and squeeze-and-excitation is added.

The other ImageNet models that we use is Vision Transformer [3] which outperforms convolutional neural networks with very different network architecture. Transformer architecture is widely used for natural language processing tasks achieving new state-of-the-art accuracy on various tasks. The Vision Transformer is a new adoption of the Transformer architecture for vision application. Sequences of image patches are treated as tokens in natural language processing application and directly fed into the Vision Transformer model which consists of embedding, Transformer encoder and multilayer perceptron head. Transformer encoder is composed of multi-head attention, layer normalization and multilayer perceptron which is same as the standard Transformer architecture. Pure Transformer architecture scores the accuracy nearly the same level of convolutional neural network counterparts.

Note that if latency of the algorithm doesn't matter and server can assist for feature extraction then using heavyweight feature extractor is desirable. It provides high quality features and high accuracy by sacrificing latency. However, we aim the scenario where latency matters and

end-to-end algorithm is executed on the edge device thereby we cannot use heavyweight feature extractor. Instead, we need to use light but efficient feature extractor which is introduced in next subsection.

4.3.3 Boosting Accuracy for Edge Device

We need to use light but efficient feature extractor in the scenario where the total latency of algorithm is important. In this study, we examine once-for-all (OFA) [5] neural architecture search method because it shows great success in many low-power computer vision challenges. Once-for-all network is a combination of many sub-networks that share weights. OFA proposes progressive shrinking (PS) process that supports different depth, width, kernel size and resolution to reduce interference between sub-networks. PS starts with the largest network and then progressively samples smaller sub-network within the largest network and fine-tunes it. After successfully trains the once-for-all network, the algorithm find the proper sub-network for each hardware device. The search space is approximately 2×10^{19} and the search process is guided by predictor-assisted evolutionary search. The latency predictor is built using look-up table (LUT) and the accuracy predictor is built using neural network trained on data from randomly sampled 16K sub-networks. The found sub-network that is optimized for specific hardware is fine-tuned for 25 or 75 epoches with ImageNet training data.

4.3.4 Co-optimization of Backbone Network and Parameter Selection

For neural network deployment, efficient inference is crucial. However, there are many knobs to control the accuracy/latency trade-off. For example, the feature extractor can be downgraded to reduce the feature extraction latency. As noted earlier, OFA network provides a variety of ImageNet pre-trained network for various hardware platforms with various ImageNet accuracy. We use specialized OFA networks for Samsung Note10 because it provides the largest ImageNet accuracy range from 71.4 to 80.2. In fact, there are 8 specialized OFA networks for Samsung Note10 off-the-shelf. Although we use specialized OFA networks for Note10, we measure the latency on NVIDIA Jetson TX2 platform which is a good representative for mobile GPU. Another examples for reducing the algorithm's execution time is to optimize the SPL hyperparameters. We control d, d_1, and the number of iterations to optimize the SPL algorithm's execution time. Note that decreasing d and d_1 parameters corresponds to reducing the execution time of SVD solver and eigen solver because smaller matrix is injected to the solvers. Decreasing the number of iterations incurs the execution time reduction because the algorithm performs pseudo labeling for more instances. Therefore, the execution time of end-to-end algorithm depends on the choice of both the feature extractor and hyperparameters of SPL algorithm.

As search space is about 10^6, enumerating all the possible combinations is prohibitive. Instead, we implemented an accuracy predictor and a latency predictor to perform evolutionary search, which is analogous to OFA [5]. The accuracy predictor has an input of (feature extractor, SPL parameters) pair to output the accuracy of unsupervised domain adaptation accuracy on Office dataset. The latency predictor also has an input of (feature extractor, SPL parameters) pair to output the latency of total algorithm execution. And the evolutionary algorithm we use is [58]. In our experiment for Office31 dataset, we collect 200 data points running on NVIDIA Jetson TX2 by measuring average latency and average accuracy to generate train set and validation set with 0.3 test split ratio (i.e., 140 data points for train and 60 data points for validation). We implemented gradient boosting regressor for both accuracy and latency predictor because it shows excellent performance with manageable cost for prediction. We conduct grid search with 5-fold cross validation to setup hyperparameters of the gradient boosting regressors. We use max depth 3 and the number of estimators 300 for the gradient boosting regressors. Accuracy performance is 0.33% RMSE for the accuracy predictor and 4.3 seconds RMSE on Office31 dataset. We implement evolutionary search algorithm with the following objective.

$$OBJ = ACCURACY - c \cdot LATENCY$$

where c is a coefficient to control accuracy/latency trade-off. If c is zero,

then the algorithm seeks the most accurate model regardless of the algorithm's latency. If c is nonzero, then the algorithm finds a model under latency constraints. Note that [62] uses similar strategy when combining the multiple objectives into one while [64] uses multiplication to combine the multiple objectives. For details about evolutionary search, the number of elements in population is 200 with number of iterations 2000. For each iteration, the algorithm randomly samples 20 elements within the population and choose an element with the highest *OBJ* value. Then, mutation process, which changes one of attribute randomly, is performed to guarantee diversity in the population. Lastly, remove the oldest element in the population and insert the mutated element into the population. As the evolutionary search process is guided by two predictors, total cost of the search algorithm is negligible compared to data collections.

4.4 Experiments

Datasets: We examined two unsupervised domain adaptation datasets, namely Office31 and Office-Home. The **Office31** dataset [11] is composed of three domains (i.e., Amazon, Webcam and DSLR) and 31 common classes. The images within Amazon are from amazon.com and the images within Webcam or DSLR are taken using a webcam and a dslr camera, respectively. It has 2,817, 795 and 498 images for Amazon, Webcam and DSLR, respectively (i.e., 4,110 images in total). The **Office-**

Home dataset [12] is composed of four domains (i.e., Artistic images, Clipart, Product images and Real-World images) and 65 common classes. The images within Art are from paintings, sketches or artistic depictions and the images within Clipart are from clipart images and the images within Product are from images without background and the images within Real-World are from regular images capture with a camera. It has 2,427, 4,365, 4,439 and 4,357 images for Art, Clipart, Product and Real-World, respectively (i.e., 15,588 images in total).

Evaluations: We evaluate the algorithm's score by comparing the predicted label of target domain with the ground truth label. For each unsupervised domain adaptation scenario, one pair of domains are selected iteratively to serve as a source domain and a target domain. Therefore, there are 6 pairs for Office31 dataset and 12 pairs for Office-Home dataset. Once source domain and target domain is determined, samples and labels of the source domain and samples of the target domain are accessible by the algorithm. After accuracy scores of all the pairs are evaluated, the accuracy scores are also averaged to summarize the algorithm's performance.

Implementation details: We examined various backbone network models, namely ResNet [18], MobileNet [19, 20], NASNet [21], Inception [22], Xception [23], EfficientNet [2], Vision Transformer [3] and OFA [5] network models. We rewrite original Matlab code [1] to support Python

[1] https://github.com/hellowangqian/domain-adaptation-capls

environment using NumPy [66], SciPy [67] and Scikit-learn [68]. The functional correctness is verified by comparing a significant digits of Python code with Matlab code when the backbone network is ResNet50. We rewrite the constructW1 function by changing from a for loop to an outer product of prime numbers because profiler indicates that the function is a bottleneck. When extracting the features form the backbone networks using PyTorch [69], we follow the standard image processing pipeline as described with the backbone networks. We manually choose bilinear interpolation for the image size smaller or equal than 250 and choose bicubic interpolation otherwise. The interpolation method marginally affect the accuracy of the model. When changing the backbone network, we also conduct additional hyperparameter tuning using grid search to report the impact of the hyperparameter.

4.4.1 ImageNet and Unsupervised Domain Adaptation Accuracy

We evaluated the accuracy of SPL algorithm with various feature extractors on the Office31 and Office-Home datasets. First of all, the Office31 results are summarized in Table 4.1. The first row group describes the verification of functional correctness. Accuracy numbers from the Python code is almost the same with accuracy numbers from the reproduced Matlab code and the paper. Therefore, we conclude that our Python code is functionally correct. After that, we experiment various

Table 4.1 Backbone networks and SPL on Office31

Model	AW	DW	WD	AD	DA	WA	Avg
Resnet50 (paper)	92.7	98.7	99.8	93.0	76.4	76.8	89.6
Resnet50 (reproduced, Matlab)	92.7	98.7	99.8	93.0	76.4	76.7	89.6
Resnet50 (reproduced, Python)	92.7	98.7	99.8	93.0	76.4	76.7	89.6
Resnet50 (torchvision)	93.3	98.6	100	93.2	75.7	77.1	89.7
Mobilenetv3_large_100	93.8	98	99.8	91	77.4	77.4	89.6
Mobilenetv3_large_075	90.7	97.6	99.6	91.8	75.3	76.4	88.6
Mobilenetv3_large_minimal_100	87.9	97	99.4	82.7	74.2	71.8	85.5
Mobilenetv3_small_100	80.3	97.9	99	84.7	71.1	64.6	82.9
Mobilenetv3_small_075	82	96.1	98.6	79.5	63.8	64.3	80.7
Mobilenetv3_small_minimal_100	75.7	96.5	99.2	75.7	59.7	62.4	78.2
Mobilenetv3_large_100	88.3	99	100	79.1	75.7	75.2	86.2
Mobilenetv3_rw	92.6	98	99.8	93.4	76.4	76.5	89.4
Nasnetalarge	96.2	98.9	99.4	95.4	79.9	80.2	91.7
Inceptionresnetv2	95	99	100	95.2	80	79.3	91.4
Inceptionv4	90.4	98.6	99.4	95.6	76.7	78.9	89.9
Xception	92.2	98.9	100	93.2	76.9	77.8	89.8
Nasnetamobile	86.4	94.8	95.8	90	73.3	74.4	85.8
Mobilenetv2	83.4	97.1	99.6	93.2	73.4	73.2	86.6
Efficientnet_l2_ns	99	100	100	99.4	84.7	87.6	95.1
Efficientnet_l2_ns_475	98.1	99.2	100	99	85.3	85.3	94.5
Efficientnet_b7_ns	98.7	99.1	100	98	84.8	85.6	94.4
Efficientnet_b6_ns	97.7	99.1	100	98	82.6	83.5	93.5
Efficientnet_b5_ns	98.6	99.1	100	97.8	84.6	86.6	94.5
Efficientnet_b8_ap	97.2	99	100	97.8	83.6	83.1	93.5
Efficientnet_b4_ns	95.8	98.9	99.8	96.4	82.1	83	92.7
Efficientnet_b3_ns	96	98.9	100	97.8	81.3	82.5	92.7
Efficientnet_b2_ns	95.2	98.2	100	94.8	81.2	79.6	91.5
Efficientnet_lite4	94.8	98	99.8	92	77.2	76.7	89.8
Efficientnet_b1_ns	95	98.7	99.8	96.6	79.8	80.3	91.7
Efficientnet_lite3	94.8	98.9	99.6	96.6	77.5	77.3	90.8
Efficientnet_lite2	88.9	98	99.4	92.6	78.5	76.4	88.9
Efficientnet_lite1	91.4	98.2	99.2	92.6	74.4	75.6	88.6
Efficientnet_lite0	88.4	97.9	99	82.5	75.4	73.1	86.1
Vit_large_patch16_384	99.9	99.2	100	99	88.3	88.4	95.8
Vit_large_patch16_224	98.4	99.2	100	98.4	87	87.5	95.1
Vit_large_patch32_384	96.1	98.7	100	96.8	85.7	85.2	93.8

feature extractor with our Python code. The second row group indicates the MobileNetV3 model family with SPL algorithm. Accuracy of the MobileNetV3 is almost same with accuracy of the ResNet50 when the large model is deployed. The third row group implements other feature extractors. NASNet performs the best among them exceeding ResNet50 by about 2%. The fourth row group shows the performance of EfficientNet feature extractors. The largest EfficientNet-L2 scores the best among them exceeding ResNet50 by a large margin. Specifically, the performance of EfficientNet-L2 is over 99% for A $\rightarrow$ W, D $\rightarrow$ W, W $\rightarrow$ D and A $\rightarrow$ D which means that the algorithm finds almost correct labels for the target domain samples. The performance of EfficientNet-L2 is slightly lower (i.e., over 80%) for D $\rightarrow$ A and W $\rightarrow$ A. Note that the image counts of domain D and W are comparably smaller than the images counts of domain A. Therefore, we conjecture that domain shift from a small domain to a large domain is harder to solve than domain shift from a large domain to a small domain or domain shift from a similar sized domain. The fifth row group summarizes the accuracy of Vision Transformer feature extractor. Vision Transformer surpasses the accuracy of other feature extractors, it shows 95.8% accuracy on Office31 dataset.

Table 4.2 shows the accuracy of SPL algorithm with various feature extractors on Office31 dataset with hyperparameter tuned using grid search. We focused on d and d_1 parameters as they are main parameters to affect the accuracy of the algorithm. Note that the case of ResNet50

Table 4.2 Backbone networks and SPL on Office31 (hyperparameter tuned)

Model	AW	DW	WD	AD	DA	WA	Avg	diff
Resnet50 (torchvision)	95.1	98.7	100	94.8	76.1	78.6	90.6	+1.0
Mobilenetv3_large_100	92.7	98	99.8	92.6	77.6	77.1	89.6	+0
Mobilenetv3_large_075	90.7	97.6	99.6	91.8	75.3	76.4	88.6	+0
Mobilenetv3_large_minimal_100	89.9	96.9	99.4	91	73.3	73.3	87.3	+1.8
Mobilenetv3_small_100	82.4	97.6	99	88	72.8	64.2	84	+1.1
Mobilenetv3_small_075	84	97.6	98.8	80.7	67.6	65.1	82.3	+1.6
Mobilenetv3_small_minimal_100	76	96.5	99.2	77.7	60.2	61.4	78.5	+0.3
Mobilenetv3_large_100	91.6	98.5	99.4	93.4	75.2	72.7	88.5	+2.3
Mobilenetv3_rw	94.1	98.1	100	94.6	75.9	77.3	90	+0.6
Nasnetalarge	96.7	98.6	99.4	96.2	79.9	80.7	91.9	+0.2
Inceptionresnetv2	94.7	98.9	99.8	95.8	80.9	79.7	91.6	+0.2
Inceptionv4	92.8	98.4	99.6	95.6	76.6	79.5	90.4	+0.5
Xception	94.3	98.5	99.8	96.2	76.9	80.7	91.1	+1.3
Nasnetamobile	87.5	94.7	95.8	91	73.8	74.3	86.2	+0.4
Mobilenetv2	86.7	96.9	99.6	89.2	76.4	73.8	87.1	+0.5
Efficientnet_l2_ns	99	100	100	99.4	85.2	87.5	95.2	+0.1
Efficientnet_l2_ns_475	98.9	99.2	100	99	88.1	87.7	95.5	+1.0
Efficientnet_b7_ns	98.9	99.1	100	99.4	86.2	85.7	94.9	+0.5
Efficientnet_b6_ns	97.9	99.1	100	97.8	83.6	85.6	94	+0.5
Efficientnet_b5_ns	98.5	99.1	100	98.2	86	86.3	94.7	+0.2
Efficientnet_b8_ap	97.1	99	100	98.2	83.8	83.7	93.6	+0.1
Efficientnet_b4_ns	95.8	98.9	99.8	96.4	82.3	83.6	92.8	+0.1
Efficientnet_b3_ns	96.6	98.7	99.8	97	81.8	84	93	+0.3
Efficientnet_b2_ns	95.6	97.9	100	96.4	80.2	80.8	91.8	+0.3
Efficientnet_lite4	94.6	98	99.8	92.2	78.2	76.7	89.9	+0.1
Efficientnet_b1_ns	95	98.7	99.8	96.6	79.9	80.3	91.7	+0
Efficientnet_lite3	95.3	98.9	99.6	97.6	79	77.4	91.3	+0.5
Efficientnet_lite2	93.3	98.1	99.4	94	78	76.1	89.8	+0.9
Efficientnet_lite1	94.6	98.4	99.4	93.2	74.3	74.7	89.1	+0.5
Efficientnet_lite0	91.7	97.7	99	87.3	74.9	72.7	87.2	+1.1
Vit_large_patch16_384	100	99.2	100	99.2	89.6	88.2	96	+0.2
Vit_large_patch16_224	98.7	99.2	100	99.2	88.5	88.7	95.7	+0.6
Vit_large_patch32_384	97.6	98.4	100	97.2	85.8	85.7	94.1	+0.3

the accuracy increases about 1% while the improvement of MobileNetV3 is marginal. In the case of EfficientNet, EfficientNet-L2-475 surpasses EfficientNet-L2 after the hyperparameter tuning. We conjecture that the input image size of the EfficienNet-L2 is too big (i.e., 800×800) for Office31 dataset because the Office31 dataset typically has image size of $250 \times 250 \sim 1001 \times 1001$. Note that we use bicubic interpolation for feature extractor with input image size larger than 250×250. Instead, EfficientNet-L2-475 has the input image size of 475, which is more appropriate for the Office31 dataset. Still, Vision Transformer surpasses all the other feature extractors.

Table 4.3 and Table 4.4 presents the accuracy of SPL algorithm with various feature extractors on Office-Home dataset without hyperparameter tuning. Office-Home datset is more complex than Office31 dataset so the accuracy of the algorithm on Office-Home is lower than that of Office31. The first row group of Table 4.3 shows the functional correctness of our Python code. The accuracy reproduced by Matlab code is the same with the accuracy reproduced by our Python code. The second row group of Table 4.3 presents the accuracy of MobileNetV3 model family with SPL algorithm. Unlike Office31 dataset, MobileNetV3 models underperform on Office-Home dataset compared to ResNet50 about 4%. This indicates careful model selection is needed to guarantee ResNet50-level performance on complex dataset (i.e., Office-Home). The second row group of Table 4.3 shows the performance of other feature extractors. The

Table 4.3 Backbone networks and SPL on Office-Home

Model	AC PA	AP PC	AR PR	CA RA	CP RC	CR RP	Avg
Resnet50 (paper)	54.5	77.8	81.9	65.1	78.0	81.1	
	66.0	53.1	82.8	69.9	55.3	86.0	71.0
Resnet50 (reproduced, Matlab)	54.5	77.8	81.9	65.1	78.0	81.1	
	66.0	53.1	82.8	69.9	55.3	86.0	71.0
Resnet50 (reproduced, Python)	54.1	77.8	81.8	65.1	78.2	81.1	
	66.0	53.1	82.8	69.9	55.5	86.0	71.0
Resnet50 (torchvision)	51.5	78.5	82.2	65.1	78	80.3	
	66.3	48.9	83	71.7	52.3	85.6	70.3
Mobilenetv3_large_100	52.3	76.8	79.1	58.3	72.7	75.6	
	61	49.7	80.4	66.9	52.3	80.2	67.1
Mobilenetv3_large_075	47.4	71.7	76.4	53.4	70.5	74.2	
	56.3	47.8	78.5	64.8	49.3	78.5	64.1
Mobilenetv3_large_minimal_100	42.9	72.3	73.4	49.4	67.6	69.5	
	54.3	41.4	77.8	62.3	45.7	77	61.1
Mobilenetv3_small_100	41.8	67.2	68.6	44.3	62.9	65.3	
	47	40.2	71.6	55.6	45.6	73.7	57
Mobilenetv3_small_075	38.4	59.5	64.9	41.4	57.5	58.8	
	44.7	38	68	52	41.6	71.6	53
Mobilenetv3_small_minimal_100	34.9	55	60.7	34.9	56.5	55.7	
	38.9	35.1	63.9	49	37	69.7	49.3
Mobilenetv3_large_100	53	73.6	78.8	56.7	70.1	74.5	
	60.7	50.7	81.1	69.5	54.8	80.8	67
Mobilenetv3_rw	51.2	75.9	79.4	55.7	72.8	76.2	
	60.9	49.5	80.5	68	54	80.2	67
Nasnetalarge	54.1	83.6	84.3	72.4	83.5	83.8	
	74	53.3	85.7	76.7	55.3	87.5	74.5
Inceptionresnetv2	54.2	81.9	84.7	72.4	82.6	83.8	
	74.2	54.2	85.8	76.8	58.9	87.2	74.7
Inceptionv4	52.5	81.4	82	70.7	79.4	80.6	
	74.4	52	84.2	76.1	52.6	85.6	72.6
Xception	53	81.3	82.5	70.9	77.8	80.8	
	73.4	52.8	83.9	75.7	54.4	85.2	72.6
Nasnetamobile	40.4	72.9	75.4	54.2	71.5	72.5	
	61.4	41	76.4	63.7	44.7	78	62.7
Mobilenetv2	44.4	71.9	75.1	56.1	70.7	74.4	
	56.3	44	78.8	65.2	45.9	81.7	63.7

Table 4.4 Backbone networks and SPL on Office-Home (cont.)

Model	AC PA	AP PC	AR PR	CA RA	CP RC	CR RP	Avg
Efficientnet_l2_ns	74	93.6	92	88.2	93.6	91.7	
	85.5	73.3	92.9	88.4	75.2	95.4	87
Efficientnet_l2_ns_475	78.7	93.1	92.5	89.8	93.8	92.2	
	87.2	78	93.3	90.5	77.9	94.9	88.5
Efficientnet_b7_ns	72.2	90	91.3	84.8	91.6	90	
	83.7	69.7	92	86.2	72.4	94.5	84.9
Efficientnet_b6_ns	71.8	89.9	91	83.5	89.3	89.5	
	81.9	71.3	91	84.9	73.6	92.5	84.2
Efficientnet_b5_ns	71.2	89.7	90.1	83	89.3	89.5	
	80.2	68	91.3	83	69.3	93.1	83.1
Efficientnet_b8_ap	65.8	89.1	89.3	81.6	89.3	88.6	
	81.1	63	89.8	83.5	67.2	91.3	81.6
Efficientnet_b4_ns	67.4	89.1	89.7	80.2	85.8	86.9	
	77.4	64.5	89.9	81.5	66.7	92	80.9
Efficientnet_b3_ns	64.2	88.5	88.5	79.9	87.3	86.8	
	77.5	65.9	89.6	80.2	67.1	90	80.5
Efficientnet_b2_ns	59.9	86	87.4	78.3	83.6	87.9	
	75.4	62.1	88.7	78	62.8	88.7	78.2
Efficientnet_lite4	51.5	82.2	83.6	68.2	80.8	81.3	
	67.2	52.1	84.5	72.7	53	85	71.8
Efficientnet_b1_ns	56.7	84.6	87.1	74.1	84.4	85	
	71.4	59.6	87.8	77.1	60	88.8	76.4
Efficientnet_lite3	51.5	79	83.2	64.9	77	81.7	
	66.8	51.2	83.1	72.6	52.3	83.3	70.5
Efficientnet_lite2	50.4	76	80.2	59.4	77.1	79	
	62.5	50.6	82.5	68.6	53.2	82.3	68.5
Efficientnet_lite1	50.1	77	80	55.8	74.1	75.8	
	63.1	48.6	80.1	65.8	52.7	79.3	66.9
Efficientnet_lite0	48.3	68.6	74.9	54	70.1	74	
	58.1	50.4	76	62.9	52.2	78.9	64
Vit_large_patch16_384	86.2	95.2	94	91.3	95.4	94.3	
	90.1	84.9	94.1	91	85.9	95.2	91.5
Vit_large_patch16_224	83.3	94.7	93.6	89.7	93.2	93	
	88.8	82.4	93.8	89.5	82.9	94.9	90
Vit_large_patch32_384	70.7	91.7	90	83.7	91.8	91.1	
	82.9	69.7	90.8	84.1	69.8	91.9	84

accuracy of InceptionResNetV2 is the best among them. Table 4.4 implements EfficientNet model family and Vision Transformer as a feature extractor. Most of the network models from the EfficientNet outperform ResNet50 except models with lite postfix (i.e., EfficientNet_lite0, EfficientNet_lite1, EfficientNet_lite2 and EfficientNet_lite3). Furthermore, EfficientNet-L2-475 scores the accuracy of 88.5 exceeding ResNet50 more than 17% and Vision Transformer shows the accuracy of 91.5. Indeed, this score is new state-of-the-art accuracy on Office-Home dataset. We also observe that the accuracy of EfficienNet-L2 is inferior to EfficientNet-L2-475 more than 1%. The Office-Home dataset has image sizes from 400×283 to 6500×4900. Still, the model with input image size of 475×475 performs better than the model with input image size of 800×800. We empirically conclude that EfficientNet-L2-475 is more suitable than EfficientNet-L2 on the dataset which includes single object per image like Office datasets.

Although there are some exceptions, we observe that accuracy of feature extractor on ImageNet is highly correlated with accuracy of SPL algorithm on Office datasets. Therefore, we recommend to use better pretrained ImageNet model as a feature extractor for SPL algorithms.

Note that using heavyweight feature extractor is desirable in the working scenario where the latency of the algorithm doesn't matter and server can assist for feature extraction. However, we aim the scenario where the total latency of the algorithm matters.

Table 4.5 OFA backbone networks and SPL on Office31

Model	AW	DW	WD	AD	DA	WA	Avg
Resnet50 (paper)	92.7	98.7	99.8	93.0	76.4	76.8	89.6
ofa_D4_E6_K7	93.6	98.6	100	95	77.3	76.8	90.2
ofa_D4_E6_K357	88.2	98.6	100	92.8	78.5	76.8	89.1
ofa_D34_E6_K357	89.3	98.6	100	93.4	77.6	73.4	88.7
ofa_D234_E6_K357	92.8	99.2	99.8	95	80.2	75.3	90.4
ofa_D234_E46_K357	94.7	99.1	100	93.4	78.1	77	90.4
ofa_mbv3_w1.0	89.7	99.1	99.8	93.4	78.6	77.2	89.6
ofa_mbv3_w1.2	91.3	98.2	100	95.8	79.5	79.4	90.7
ofa_proxyless_w1.3	91.9	98.5	99.6	95.4	77	79.3	90.3
note10@80.2	94.6	98.9	100	97.4	81.2	81.2	92.2
note10@79.7	93.6	98.9	100	97	80.2	80.5	91.7
note10@79.3	92.8	98.9	100	95.6	80.4	81.2	91.5
note10@78.4	89.2	99	99.8	93.8	78.2	75.3	89.2
note10@76.6	92.7	99	100	92.6	77.7	75	89.5
note10@75.5	93.8	98.4	100	86.5	76.3	73.2	88
note10@73.6	87.4	99	100	88.6	72	73.1	86.7
note10@71.4	80.8	97.9	99.4	86.9	69.6	66.8	83.6
pixel1@76.9	93	98.7	100	91.6	76.6	73.6	88.9
note10@76.6	92.7	99	100	92.6	77.7	75	89.5
LG-G8@76.4	91.2	98.2	100	93.2	76	73	88.6
1080ti@76.4	93.6	99.1	99.8	92.4	76.2	75.7	89.5
s7edge@76.3	92.1	98.7	100	90.2	75.7	72.8	88.2
note8@76.1	89.2	98.4	100	91.6	76.4	73	88.1
v100_gpu64@76.1	94	98.5	99.8	96	75.9	76.2	90.1
pixel2@75.8	92.7	98.4	100	88	72.9	77.9	88.3
tx2_gpu16@75.8	91.6	98.5	100	93	73.9	77.9	89.1
cpu@75.7	94.8	99	99.8	92	75.9	75.2	89.5
note10@75.5	93.8	98.4	100	86.5	76.3	73.2	88
tx2_gpu16@75.4	90.7	99.1	100	87.1	76	74.4	87.9
1080ti_gpu64@75.3	90.6	99	100	89.8	75.2	75.4	88.3
v100_gpu64@75.3	89.6	98.9	100	86.3	74.9	75.8	87.6

4.4.2 Accuracy with Once-For-All Network

We evaluated the accuracy of SPL algorithm with various OFA feature extractors for edge device on the Office31 and Office-Home datasets. The Office31 results are summarized in Table 4.5. The first row group of the table indicates ResNet50 baseline. The second row group of the table shows the performance of specialized OFA networks. We also report the performance of OFA network in the course of progressive shrinking algorithm. For example, ofa_D4_E6_K7 is an initial network without progressive shrinking while ofa_D4_E6_K357 is a network which supports elastic kernel. The progress of progressive shrinking affects the SPL accuracy marginally. OFA networks after full progressive shrinking are in the third row group of the table. The ofa_mbv3_w1.2 network already surpasses the ResNet50 baseline more than 1%. The fourth row group of the table shows specialized OFA networks for Samsung Note10 smartphone in descending order of ImageNet accuracy. Again, ImageNet accuracy is highly correlated with unsupervised domain adaptation accuracy. The fifth row group of the table indicates specialized OFA networks for various hardware platforms that have similar ImageNet accuracy with ResNet50 in descending order of ImageNet accuracy. The ImageNet accuracy is correlated with unsupervised domain adaptation accuracy, however, the correlation is lower than the specialized OFA networks for Note10 smartphone. For example, the specialized OFA net-

work for Google Pixel1 smartphone at ImageNet accuracy of 76.9 has unsupervised domain adaptation accuracy of 88.9 while the specialized OFA network for NVIDIA V100 GPU at ImageNet accuracy of 76.1 has unsupervised domain adaptation accuracy of 90.1, that is, 0.8% lower ImageNet accuracy incurs 1.2% higher unsupervised domain adaptation. Therefore, ImageNet accuracy of specialized OFA networks is less correlated with unsupervised domain adaptation accuracy when the difference scale is fine-grained. We first assume that better subnetwork selection can boost the unsupervised domain adaptation accuracy after observing the fact and try to add unsupervised domain adaptation accuracy feedback into subnetwork selection process. However, we find that the try is based on a fallacy. The progressive shrinking algorithm is performed on ImageNet dataset. Therefore, subnetworks derived by OFA network are guaranteed to perform predictably on ImageNet dataset. The accuracy predictor of OFA subnetwork selection process has high prediction accuracy on ImageNet dataset thereby enabling the selection algorithm to find better subnetwork. However, when dealing with Office dataset, the accuracy of the subnetwork depends on not only subnetwork selection but also batch normalization statistics, which leads to unstable prediction. One possible solution to solve the problem is to perform progressive shrinking alogirhtm on Office dataset after ImageNet dataset, which is not straightforward because the SPL algorithm is not implemented in differentiable manner.

Table 4.6 OFA backbone networks and SPL on Office31 (hyperparameter tuned)

Model	AW	DW	WD	AD	DA	WA	Avg	diff
Resnet50 (torchvision)	95.1	98.7	100	94.8	76.1	78.6	90.6	+1.0
ofa_D4_E6_K7	94	98.6	100	96.4	77.4	78.5	90.8	+0.6
ofa_D4_E6_K357	93.8	99	100	95.4	78.8	78	90.8	+1.7
ofa_D34_E6_K357	92.6	98.7	99.8	93.2	77.6	77.4	89.9	+1.2
ofa_D234_E6_K357	92.5	99.2	99.8	95	80.2	76.6	90.6	+0.2
ofa_D234_E46_K357	95	99	100	95.4	78	77.4	90.8	+0.4
ofa_mbv3_w1.0	92.5	98.5	100	95	78	78.4	90.4	+0.8
ofa_mbv3_w1.2	92.6	98.2	99.8	97	79.3	79.9	91.1	+0.4
ofa_proxyless_w1.3	95.3	98.7	99.6	96.2	79.2	79.8	91.5	+1.2
note10@80.2	95.1	99	100	98.4	80.4	81	92.3	+0.1
note10@79.7	94.3	98.9	100	96.6	80.9	80.4	91.9	+0.2
note10@79.3	93.3	99	99.8	96.2	81.3	80.5	91.7	+0.2
note10@78.4	90.7	98.7	99.8	95.2	80.2	78.9	90.6	+1.4
note10@76.6	91.6	99.1	100	93.4	78.1	77.4	89.9	+0.4
note10@75.5	92.7	99	99.8	90.2	77.7	77.8	89.5	+1.5
note10@73.6	89.3	98.4	99.6	88.4	77	76.3	88.2	+1.5
note10@71.4	88.7	97.1	99.2	86.1	74.4	73.2	86.5	+2.9
pixel1@76.9	92.8	99	100	95.8	76.8	76.1	90.1	+1.2
note10@76.6	91.6	99.1	100	93.4	78.1	77.4	89.9	+0.4
LG-G8@76.4	92.6	98.5	100	94.6	75.5	76.5	89.6	+1.0
1080ti@76.4	95.1	99	99.8	95.4	76.6	78.9	90.8	+1.3
s7edge@76.3	91.9	98.9	100	91.8	77.9	75.9	89.4	+1.2
note8@76.1	93.2	98.5	99.6	92.6	77.2	74.4	89.2	+1.1
v100_gpu64@76.1	94.7	98.7	100	96.4	78.1	76.8	90.8	+0.7
pixel2@75.8	93.1	98.7	99.8	90.8	78	75.9	89.4	+1.1
tx2_gpu16@75.8	93.7	98.6	99.8	94	76.2	77	89.9	+0.8
cpu@75.7	96.4	99	99.8	93	76.7	76.3	90.2	+0.7
note10@75.5	92.7	99	99.8	90.2	77.7	77.8	89.5	+1.5
tx2_gpu16@75.4	92.6	98.7	99.6	93.4	76.5	76.2	89.5	+1.6
1080ti_gpu64@75.3	92.7	99	100	90.8	74.5	76.2	88.9	+0.6
v100_gpu64@75.3	91.3	99.1	99.8	93.8	74.1	76.5	89.1	+1.5

Table 4.6 shows the accuracy of SPL algorithm with various OFA feature extractors on Office31 dataset with hyperparameter tuned using grid search. Again, we focused on d and d_1 parameters because the parameters are the main parameters to affect the algorithm's accuracy. We report hyperparameter tuning results to quantify the impact of hyperparameter tuning of SPL algorithm. Because the feature extractor is changed, the SPL algorithm's parameter is also changed. Note that the half of the specialized OFA networks for Samsung Note10 surpass the ResNet accuracy, which leads us to the idea of co-optimization of OFA feature extractor and SPL algorithm. Most of the specialized OFA networks for various hardware platforms that have similar ImageNet accuracy with ResNet50 underperform on Office31 dataset compared to ResNet50. We conjecture that highly optimized networks like the specialized OFA network have lower transferability than ResNet which includes high redundancy.

Table 4.7 and Table 4.8 shows the accuracy of SPL algorithm with various OFA feature extractors on Office-Home dataset without hyperparameter tuning. The progressive shrinking algorithm marginally affect the accuracy of the unsupervised domain adaptation as shown in the second row group of Table 4.7. Again, the half of the specialized OFA networks for Samsung Note10 surpass the accuracy of ResNet50 which leads us to perform co-optimization of OFA feature extractor and SPL algorithm. The wider OFA networks surpass the ResNet accuracy while the accuracy of shallower OFA network, namely ofa_mbv3_w1.0,

Table 4.7 OFA backbone networks and SPL on Office-Home

Model	AC PA	AP PC	AR PR	CA RA	CP RC	CR RP	Avg
Resnet50 (paper)	54.5	77.8	81.9	65.1	78.0	81.1	
	66.0	53.1	82.8	69.9	55.3	86.0	71.0
ofa_D4_E6_K7	50	79.7	82.4	63	80.6	80.4	
	63.2	49.4	83.5	71.9	52.8	84.8	70.1
ofa_D4_E6_K357	50.4	79.6	81.9	62.4	80.4	78.7	
	64.6	50.2	84.5	73.4	53.5	86.3	70.5
ofa_D34_E6_K357	49.9	79.3	82.2	62.6	80.5	79.2	
	67.6	50.4	83.8	70.6	53.2	86.3	70.5
ofa_D234_E6_K357	51.2	79.3	81.8	62.7	77.6	80.2	
	66	47.9	84.1	71.4	53.3	85.7	70.1
ofa_D234_E46_K357	53.7	77.4	83.6	63.6	76.2	80.3	
	63.9	53.5	83.7	70.4	53.4	85.2	70.4
note10@80.2	53.6	80.6	84.3	67.5	82.5	83.4	
	69.6	51	85.9	75.9	54.7	87.4	73
note10@79.7	55.5	79.9	83.8	68.1	77.2	82.7	
	69.3	54.1	85.8	75	55.4	87.8	72.9
note10@79.3	54.2	80.3	83.5	67.7	79.4	83.1	
	68.7	53.6	85.7	73.8	55.5	87.7	72.8
note10@78.4	51.6	79.3	83.1	66.2	79.6	82.2	
	66.1	52.5	84.1	73.1	53.2	87.1	71.5
note10@76.6	50.3	76.7	81.7	56.4	74.7	78.2	
	62.4	48.8	82.6	71	50.7	82.1	68
note10@75.5	49.8	73.8	80.7	57.5	76	74.9	
	60.3	48.9	81.5	67.2	52.7	82.3	67.1
note10@73.6	45.7	72.4	78.5	51.7	67.8	73.6	
	55.4	42.9	79.2	64.1	50	81.2	63.5
note10@71.4	41.5	66.9	74.7	49.5	66.8	70.9	
	51.7	39.9	77	62.7	48.6	80.4	60.9

Table 4.8 OFA backbone networks and SPL on Office-Home (cont.)

Model	AC PA	AP PC	AR PR	CA RA	CP RC	CR RP	Avg
ofa_mbv3_w1.0	52.6	78.2	81.3	59.5	73.7	77.9	
	61.3	48.7	83.1	72	53.8	84.1	68.8
ofa_mbv3_w1.2	52.8	80.5	84	67.1	80.2	81.3	
	68	52.4	85.4	74.9	56	86	72.4
ofa_proxyless_w1.3	51.2	79.5	82.8	64.7	74	82.9	
	68.3	50.9	84.5	73.8	54.5	85.6	71.1
pixel1@76.9	50.1	78	81.5	61	76.2	79.3	
	62.8	47.9	83.6	70.4	52.7	83	68.9
note10@76.6	50.3	76.7	81.7	56.4	74.7	78.2	
	62.4	48.8	82.6	71	50.7	82.1	68
LG-G8@76.4	47.8	75.9	81	57.8	76.7	77.6	
	62.6	46.8	81.7	69.8	52.1	83.9	67.8
1080ti@76.4	45.8	76.7	80.7	60.2	74.3	78.4	
	62.6	46.1	82.6	69.4	49.5	84.2	67.5
s7edge@76.3	50.3	76.7	80.8	61.1	75	75.8	
	63.4	50.1	82.7	69.4	54.4	82.2	68.5
note8@76.1	46.7	74.9	79.5	59.4	74.2	75.2	
	58.7	46.3	81.2	67.9	53.2	82.5	66.6
v100_gpu64@76.1	47.3	76.5	81.4	59.7	75.2	77.7	
	61.9	47.7	82.6	69.9	50.1	84	67.8
pixel2@75.8	50.3	74.7	80.5	59.7	75.5	75.6	
	60.4	47.9	81.9	67.8	51.3	82.4	67.3
tx2_gpu16@75.8	49.5	76.7	80.8	61.4	74.4	77.8	
	62.2	46.8	81.3	70.5	50.4	84	68
cpu@75.7	45.4	76.8	81	55.6	73.1	77.9	
	60.4	44.3	81.7	69.1	46.6	83.4	66.3
note10@75.5	49.8	73.8	80.7	57.5	76	74.9	
	60.3	48.9	81.5	67.2	52.7	82.3	67.1
tx2_gpu16@75.4	45.8	76.1	80.2	57.5	73.1	77.2	
	61.3	44.1	82.4	67.7	47.8	83.8	66.4
1080ti_gpu64@75.3	45.2	75.4	80.1	57.4	73.1	77	
	60.9	45.1	82.1	68.2	49.4	83.7	66.5
v100_gpu64@75.3	45.6	75.9	80.5	56.7	72.2	78.3	
	59.8	45.6	81.9	67.8	46.9	83.8	66.2

underperforms on Office-Home dataset compared to ResNet50 as shown in the first row group of Table 4.8. Indeed, all of the specialized OFA networks for various hardware platforms that have similar ImageNet accuracy with ResNet50 underperform on Office-Home dataset compared to ResNet50. Again, We agree that highly optimized networks like the specialized OFA network have lower transferability than ResNet which includes high redundancy.

Although there are some exceptions, the ImageNet accuracy of feature extractor is highly correlated with the Office accuracy of SPL algorithm. And the latency of the feature extraction depends directly on the the feature extractor. Therefore, the accuracy and the latency of feature extractor is crucial. Note that implementing progressive shrinking algorithm for Office dataset is not straightforward as indicated above. We recommend to use highly optimized ImageNet pre-trained network model as feature extractor to boost the efficiency of algorithm execution.

4.4.3 Comparison with State-of-the-Art Results

We summarize the accuracy of our method along with state-of-the-art methods in Table 4.9 and Table 4.10. Our method with EfficientNet and Vision Transformer combined with SPL algorithm surpasses existing methods by a large margin, more than 2% on Office31 dataset. Considering the fact that the accuracy of existing methods on Office31 dataset is already mature (i.e., higher than 92%), the 2% accuracy gain on Office31

dataset is noticeable. Note that PRPL algorithm uses EfficientNet-B7 feature extractor, which has 7.5 times fewer parameters than EfficientNet-L2 with 1.4% lower ImageNet accuracy. The accuracy gain stems from using better feature extractor. Furthermore, our method with the specialized OFA network and SPL algorithm surpasses the existing methods except PRPL algorithm which uses heavyweight feature extractor. The note10@80.2 specialized OFA network has 8.4 times fewer parameters and almost 50 times fewer MAC counts than EfficientNet-B7. It is not reasonable to include the PRPL algorithm with other methods when we consider mobile setting. For mobile setting, therefore, our method with the specialized OFA network scores the best among existing state-of-the-art methods with 0.8% accuracy gain on Office31 dataset. The accuracy result for Office-Home dataset has similar trends. Our method with EfficienNet and Vision Transformre combined with SPL algorithm outperforms all the existing method by a substantial margin, more than 5% on Office-Home dataset. Except PRPL algorithm, our method with the specialized OFA network and SPL algorithm surpasses existing methods on Office-Home dataset.

4.4.4 Co-optimization for Edge Device

Figure 4.1 shows the example of evolutionary search process with various c coefficient. As shown in the $c = 0$ accuracy plot, the search algorithm converges to better accuracy compared to initial points with the

Table 4.9 Accuracy comparison with state-of-the-art on Office31

Model	AW	DW	WD	AD	DA	WA	Avg
DM-ADA [70]	83.9	99.8	99.9	77.5	64.6	64	81.6
DANN [71]	82	96.9	99.1	79.7	68.2	67.4	82.2
HoMM [72]	90.8	99.3	100	87.9	69.3	69.5	86.1
MSTN* [73]	91.3	98.9	100	90.4	72.7	65.6	86.5
SAFN+ENT* [74]	90.3	98.7	100	92.1	73.4	71.2	87.6
rRevGrad+CAT [75]	94.4	98	100	90.8	72.2	70.2	87.6
CDAN+E [76]	94.1	98.6	100	92.9	71	69.3	87.7
DEV [77]	93.2	98.4	100	92.8	70.9	71.2	87.8
DMRL [78]	90.8	99	100	93.4	73	71.2	87.9
SymNets [79]	90.8	98.8	100	93.9	74.6	72.5	88.4
BSP+CDAN [49]	93.3	98.2	100	93	73.6	72.6	88.5
SHOT [27]	90.1	98.4	99.9	94	74.7	74.3	88.6
ALDA [80]	95.6	97.7	100	94	72.2	72.5	88.7
MDD [81]	94.5	98.4	100	93.5	74.6	72.2	88.9
DADA [82]	92.3	99.2	100	93.9	74.4	74.2	89
MCC [83]	95.4	98.6	100	95.6	72.6	73.9	89.4
SPL [4]	92.7	98.7	99.8	93.0	76.4	76.8	89.6
GSDA [84]	95.7	99.1	100	94.8	73.5	74.9	89.7
MDAIR [85]	94	96.9	99.2	92.6	78.7	77.6	89.8
CAN [86]	94.5	99.1	99.8	95	78	77	90.6
SRDC [87]	95.7	99.2	100	95.8	76.7	77.1	90.8
RSDA-MSTN [88]	96.1	99.3	100	95.8	77.4	78.9	91.1
FixBi [89]	96.1	99.3	100	95	78.7	79.4	91.4
PRPL [90]	95.9	97.1	99.2	97	83	82.4	92.4
Ours (Efficientnet_l2_ns+SPL)	99	100	100	99.4	84.7	87.6	95.1
Ours (Vit_large_patch16_384+SPL)	99.9	99.2	100	99	88.3	88.4	95.8
Ours (note10@80.2+SPL)	94.6	98.9	100	97.4	81.2	81.2	92.2

Table 4.10 Accuracy comparison with state-of-the-art on Office-Home

Model	AC PA	AP PC	AR PR	CA RA	CP RC	CR RP	Avg
DANN [71]	45.6	59.3	70.1	47	58.5	60.9	
	46.1	43.7	68.5	63.2	51.8	76.8	57.6
MSTN* [73]	49.8	70.3	76.3	60.4	68.5	69.6	
	61.4	48.9	75.7	70.9	55	81.1	65.7
CDAN+E [76]	50.7	70.6	76	57.6	70	70	
	57.4	50.9	77.3	70.9	56.7	81.6	65.8
BSP+CDAN [49]	52	68.6	76.1	58	70.3	70.2	
	58.6	50.2	77.6	72.2	59.3	81.9	66.3
ALDA [80]	53.7	70.1	76.4	60.2	72.6	71.5	
	56.8	51.9	77.1	70.2	56.3	82.1	66.6
SymNets [79]	47.7	72.9	78.5	64.2	71.3	74.2	
	64.2	48.8	79.5	74.5	52.6	82.7	67.6
MDD [81]	54.9	73.7	77.8	60	71.4	71.8	
	61.2	53.6	78.1	72.5	60.2	82.3	68.1
GSDA [84]	61.3	76.1	79.4	65.4	73.3	74.3	
	65	53.2	80	72.2	60.6	83.1	70.3
GVB-GD [91]	57	74.7	79.8	64.6	74.1	74.6	
	65.2	55.1	81	74.6	59.7	84.3	70.4
RSDA-MSTN [88]	53.2	77.7	81.3	66.4	74	76.5	
	67.9	53	82	75.8	57.8	85.4	70.9
SPL [4]	54.5	77.8	81.9	65.1	78	81.1	
	66	53.1	82.8	69.9	55.3	86.0	71.0
SRDC [87]	52.3	76.3	81	69.5	76.2	78	
	68.7	53.8	81.7	76.3	57.1	85	71.3
SHOT [27]	57.1	78.1	81.5	68	78.2	78.1	
	67.4	54.9	82.2	73.3	58.8	84.3	71.8
FixBi [89]	58.1	77.3	80.4	67.7	79.5	78.1	
	65.8	57.9	81.7	76.4	62.9	86.7	72.7
MDAIR [85]	55.6	80.4	81.6	70.2	80.7	80.8	
	71	55.6	82.5	73.5	57.7	83.9	72.8
PRPL [90]	67.6	84.5	89.4	79.8	85.7	86.3	
	79.2	69.1	88.7	83.8	68.9	91.5	81.2
Ours (Efficientnet_l2_ns+SPL)	74	93.6	92	88.2	93.6	91.7	
	85.5	73.3	92.9	88.4	75.2	95.4	87
Ours (Vit_large_patch16_384+SPL)	86.2	95.2	94	91.3	95.4	94.3	
	90.1	84.9	94.1	91	85.9	95.2	91.5
Ours (note10@80.2+SPL)	53.6	80.6	84.3	67.5	82.5	83.4	
	69.6	51	85.9	75.9	54.7	87.4	73

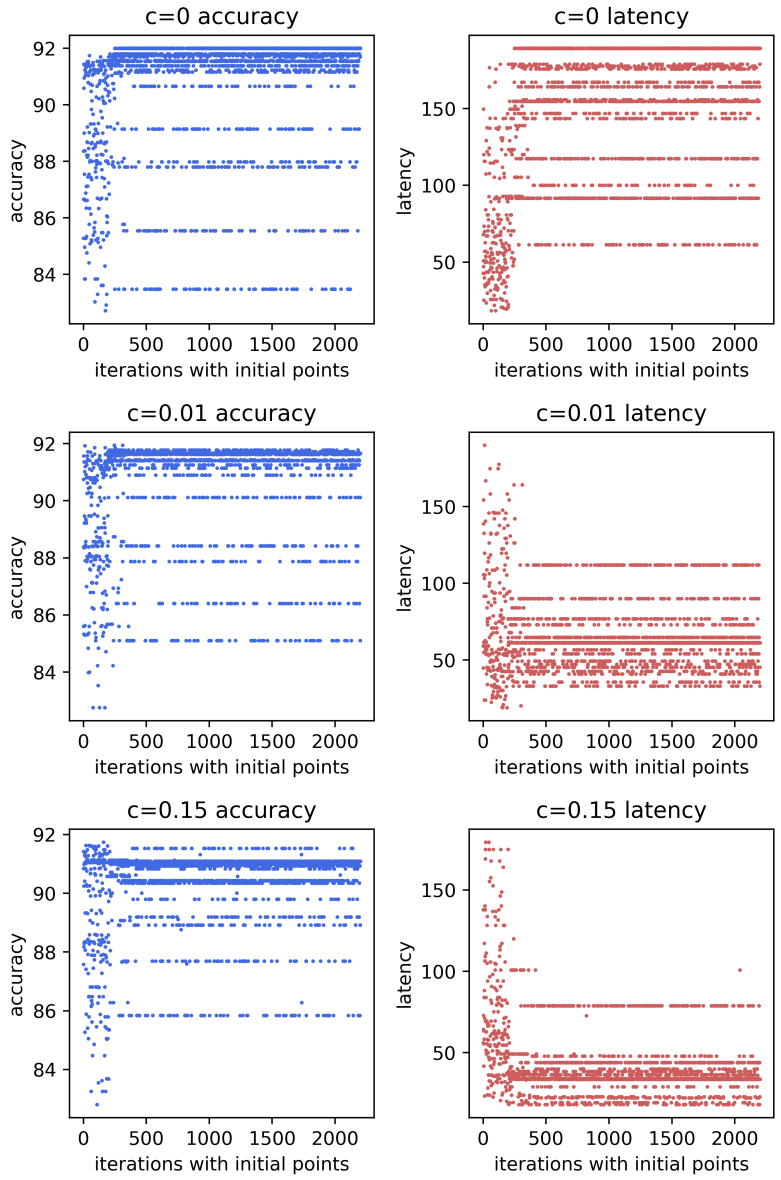

Figure 4.1 Progress of evolutionary search with various c coefficient

diversity incurred by the mutation process. Similarly, the latency of $c = 0$ case also converges higher points without constraints as expected. With $c = 0.01$, the accuracy converges slightly lower points while the latency converges dramatically lower points due to the latency constraints. The trend becomes more obvious with higher c coefficient (i.e., $c = 0.15$). The accuracy of $c = 0.15$ converges lower points due to the latency constraints. We conclude that the evolutionary search algorithm works as intended and it is controlled by the value of c coefficient.

Figure 4.2 presents the impact of various feature extractor and SPL parameter selection on unsupervised domain adaptation accuracy of Office31 and Office-Home dataset. The number of data points are 200 and 280 for Office31 and Office-Home, respectively, and we collect the data points by measuring average accuracy and average latency in grid search manner. As expected, both feature extractor selection and SPL parameter affects accuracy and latency significantly. The points with the same color use the same feature extractor while the variation of SPL parameters incur accuracy/latency difference. Note that the red point in the right indicates the baseline which has ResNet feature extractor with baseline SPL parameters. The other points use specialized OFA network for Samsung Note10 with various latency constraints. There are many points that have better accuracy and latency than the baseline. The problem is to find the near Pareto optimal points without enumerating all the data points and we use evolutionary search. The first characteristic of Pareto optimal

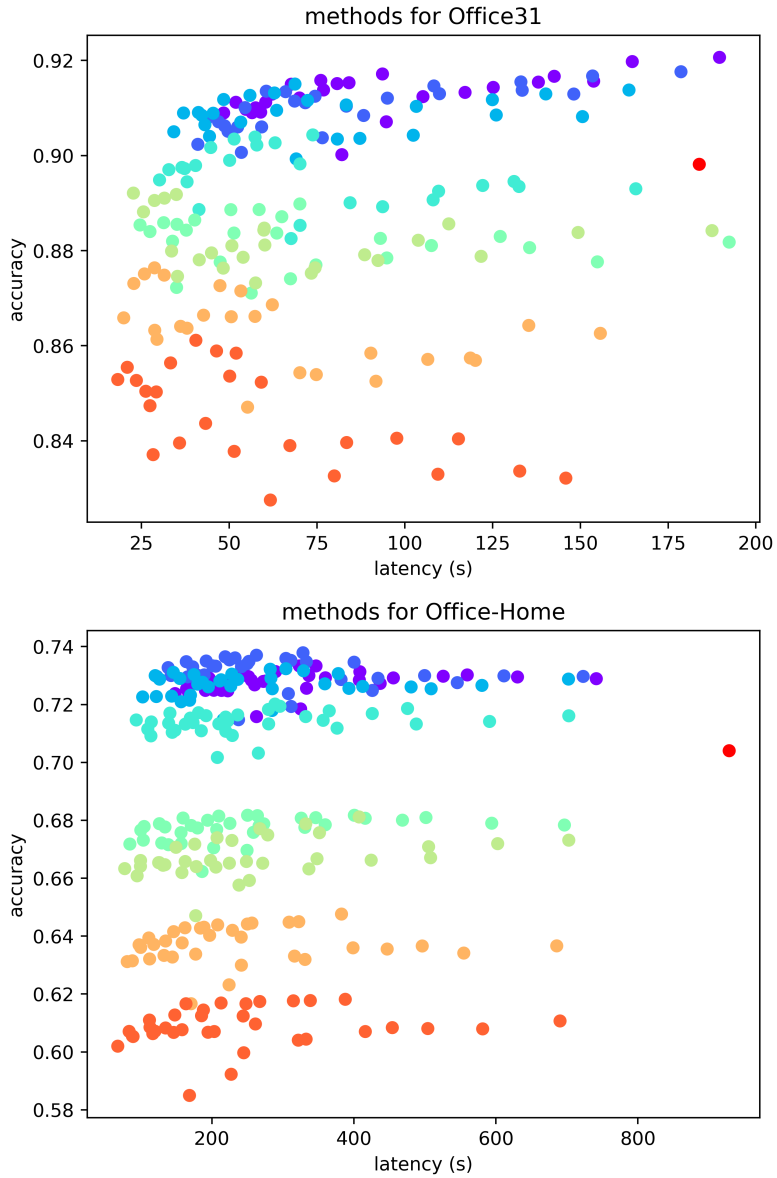

Figure 4.2 Impact of various feature extractor and SPL parameter selection on Office datasets

points is that the feature extractor selection is more important than SPL parameters in the low latency regime. For low latency algorithm execution, specifically, the execution time of SPL algorithm is assigned to near minimal values and the selection of feature extractor can control the algorithm's accuracy/latency. On the other hand, the SPL parameter is more important than the feature extractor selection in the high latency regime as we have no choice but to use the best feature extractor. In the middle latency range, both the feature extractor and SPL parameter selection is important. The evolutionary search algorithm automatically set up the process of feature extractor and SPL parameter selection controlled by c coefficient.

Figure 4.3 shows the result generated by our evolutionary search shown in red with various c coefficient on Office31 dataset. The data points generated by grid search manner is shown in black. We experiment 10 repetitive runs for each c coefficient value. If c value gets larger, the algorithm searches lower latency regime. Although the evolutionary search algorithm has intrinsic randomness, the algorithm outputs near Pareto optimal points without exception. Indeed, some of them surpass Pareto frontier which consist of data points in grid search manner.

Figure 4.4 shows the result generated by our evolutionary search shown in red with various c coefficient on Office-Home dataset. Again, we experiment 10 runs for each c coefficient value and the data points generated by grid search manner is shown in black. Office-Home dataset

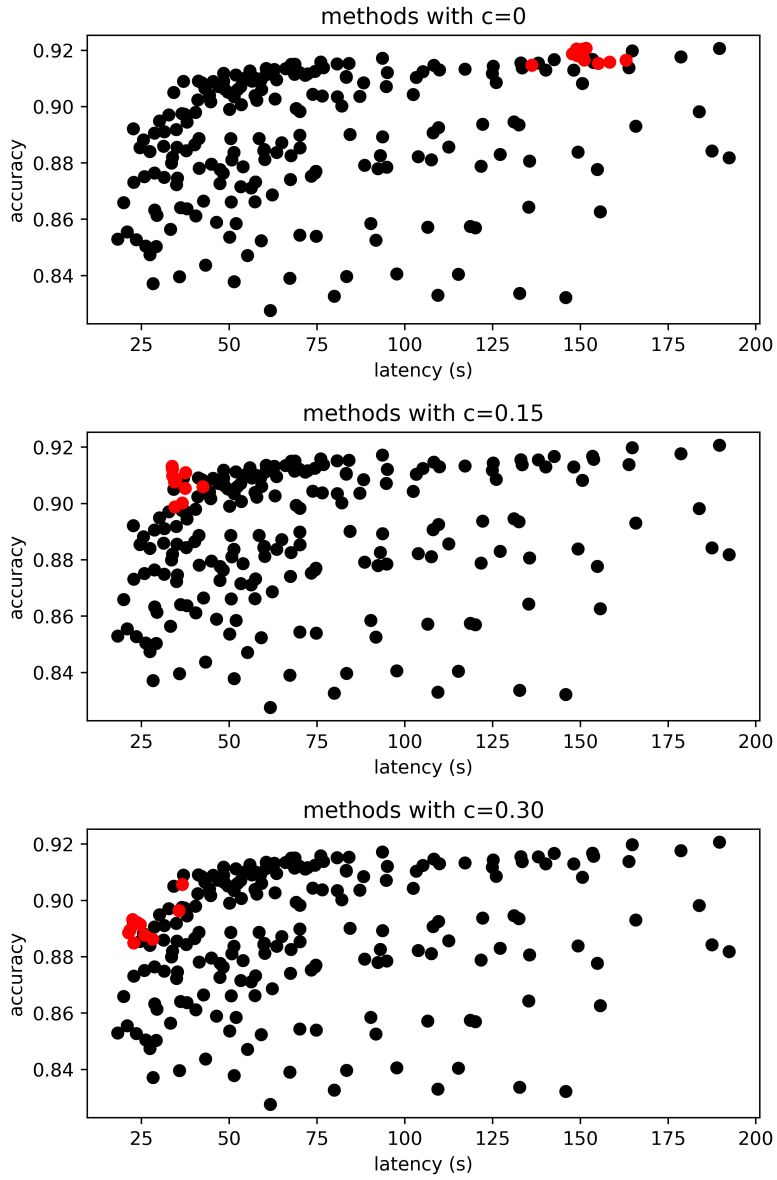

Figure 4.3 The result generated by evolutionary search on Office31 dataset

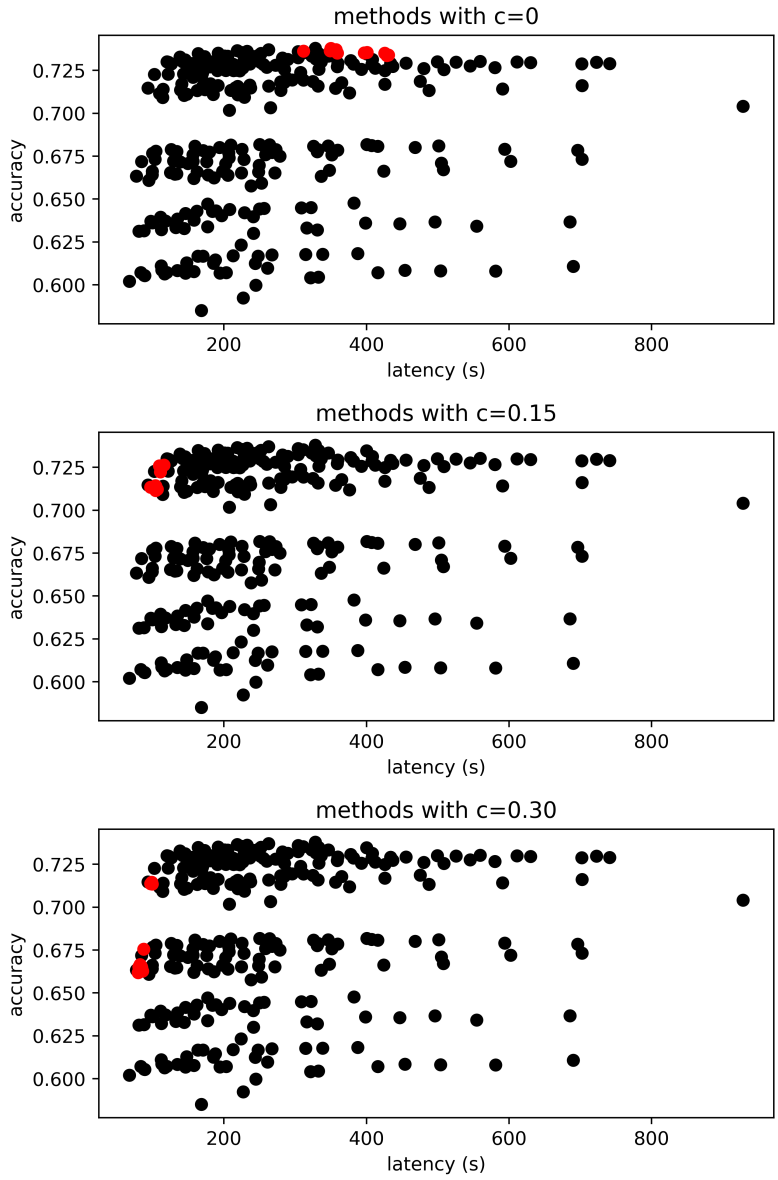

Figure 4.4 The result generated by evolutionary search on Office-Home dataset

has more complex structure than Office31 dataset and the accuracy difference between feature extractor selection on Office-Home dataset is higher than that on Office31 dataset. Still, the evolutionary search algorithm finds near Pareto optimal points without exception according to the c coefficient value.

The last point to mention about co-optimization for edge device is transferability between Office31 dataset and Office-Home dataset. Although the evolutionary search algorithm proves effectiveness to find Pareto optimal setting for Office dataset, the evolutionary search algorithm cannot be implemented on the fly due to the large cost induced by data collection. It is analogous to typical neural architecture search algorithms which consumes a lot of computing resources to find optimal architecture thereby it is assumed to be performed in the server. As we are targeting edge device, all the resource intensive operations are assumed to be implemented in the server and they include the evolutionary search algorithm. As the evolutionary search algorithm requires realistic training, we assume the predictors are trained using Office31 dataset and the test is performed on Office-Home dataset thereby we measure transferability between Office31 dataset and Office-Home dataset. It is reasonable that all the resource intensive operations including evolutionary search are performed in the server and output the specific (feature extractor, SPL parameter) pair on proxy dataset (i.e., Office31) for each hardware platform. Then, an arbitrary edge device loop up off-the-shelf

setting and perform unsupervised domain adaptation task on incoming dataset (i.e., Office-Home). The fundamental assumption of this scenario is that the distribution and characteristic of proxy dataset is similar to that of incoming dataset.

Figure 4.5 shows the transferability result that predictors are trained on Office31 dataset and the found setting is applied on Office-Home dataset. We experiment 10 repetitive runs for each c case. Note that there is large variation in $c = 0$ case, which indicates that the latency predictor for Office31 dataset is not perfectly suited for Office-Home dataset. Although there is dataset discrepancy, the algorithm finds near Pareto optimal methods without exception.

Figure 4.6 shows the transferability result that predictors are trained on Office-Home dataset and the found setting is applied on Office31 dataset. Again, we experiment 10 repetitive runs for each c case. In $c = 0$ case, we observe that the found settings for Office-Home dataset do not fit entirely on Office31 because the settings for Office-Home dataset typically use higher d and d_1 to get better accuracy. On the other hand, the evolutionary search algorithm finds near Pareto optimal methods in $c = 0.15$ and $c = 0.30$ cases without difficulties. We conjecture that accuracy-latency curve is similar between Office31 dataset and Office-Home dataset in the middle and low latency regime.

Note that Office31 dataset has 2,817, 795 and 498 images for Amazon, Webcam and DSLR, respectively while Office-Home dataset has

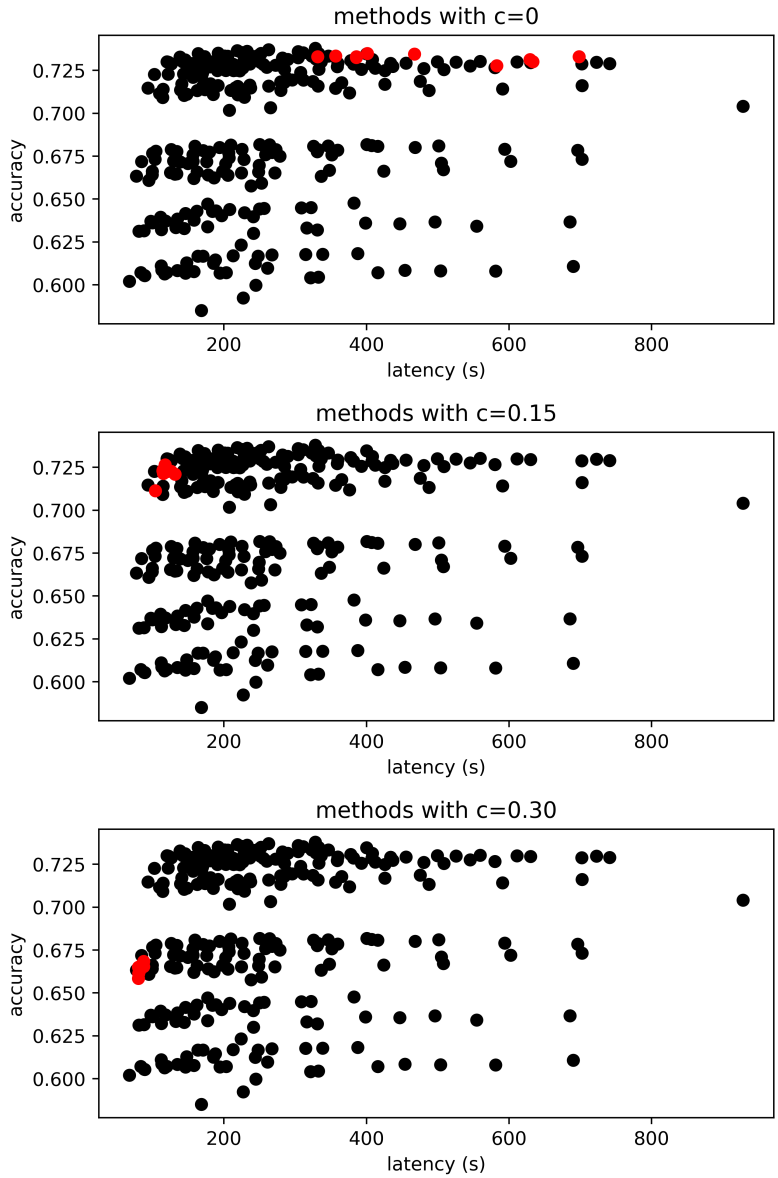

Figure 4.5 Transferability result from Office31 to Office-Home dataset

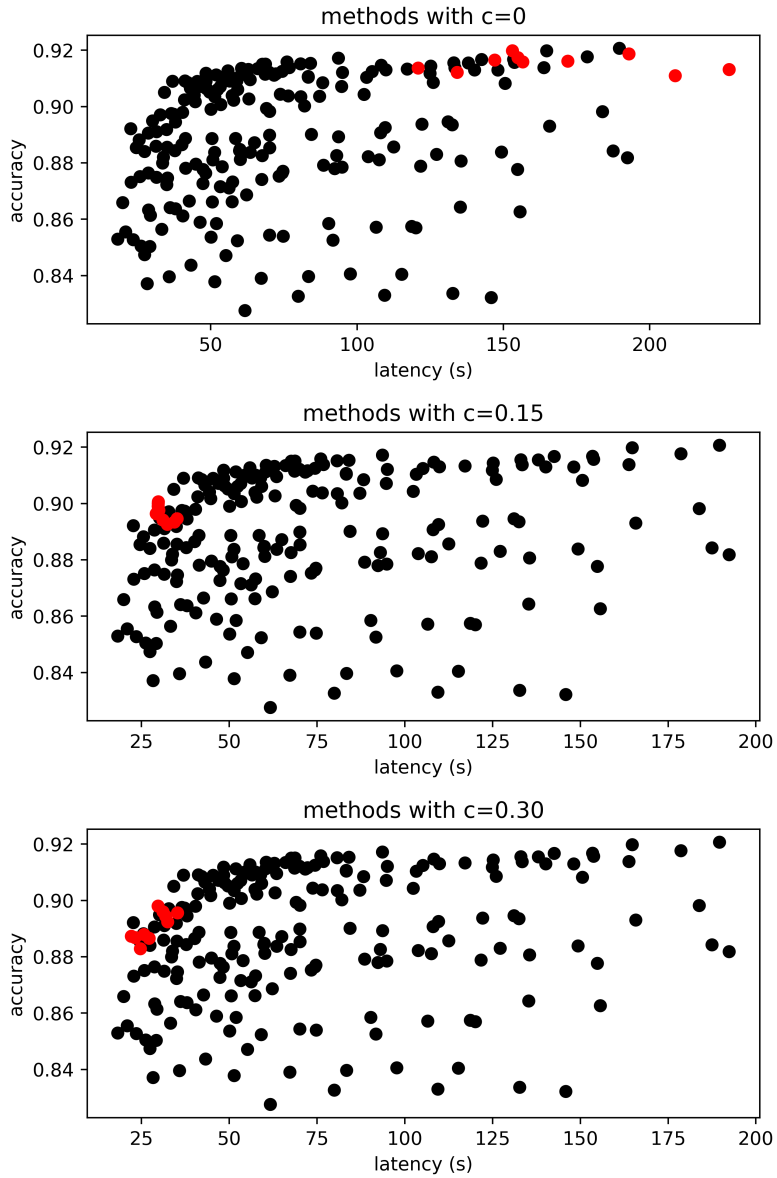

Figure 4.6 Transferability result from Office-Home to Office31 dataset

2,427, 4,365, 4,439 and 4,357 images for Art, Clipart, Product and Real-World, respectively. Although the number of images for Office-Home dataset is about 4 times larger than the number of images for Office31 dataset, both dataset has a similar nature (i.e., classifying office objects and some of domains are more difficult to classify than the others). It is empirical that the average of accuracy for all domain pairs is transferable between Office31 and Office-Home dataset.

4.4.5 Pre-extraction of Source Feature

In the previous work [27], SHOT aims to remove the access to source domain data when adapting the model to target domain data. It is more secure for decentralized private data. Following the setting proposed in [27], we also consider the source-free setting which assumes absent of source data when adapting the model. In this setting, we assume only the source feature is available instead of source data. We assume that revealing the source feature instead of source data is secure because it is not easy to recover source data from the source feature. So, we pre-extract the source feature. It is straightforward to implement pre-extraction of source feature because SPL algorithm requires only source feature instead of source data. In this case, we assume the same network is used to extract source feature and target feature. Storing source feature along to each network which is used to extract source feature is one solution to realize pre-extraction of source feature for unsupervised domain adap-

Table 4.11 Pareto frontier methods on Office31 with pre-extraction of source feature

source network	target network	d	d_1	iterations	avg accuracy
note10@80.2	note10@80.2	512	128	11	92.2
ofa_mbv3_w1.2	note10@80.2	512	128	11	91.2
note10@80.2	note10@80.2	256	64	11	92
ofa_mbv3_w1.2	note10@80.2	256	64	11	91.6
note10@80.2	note10@80.2	512	128	9	91.9
ofa_mbv3_w1.2	note10@80.2	512	128	9	91.3
note10@80.2	note10@80.2	256	128	9	91.5
ofa_mbv3_w1.2	note10@80.2	256	128	9	91.8
note10@80.2	note10@80.2	256	64	5	91.6
ofa_mbv3_w1.2	note10@80.2	256	64	5	90.9
note10@79.7	note10@79.7	256	128	7	91.4
ofa_mbv3_w1.2	note10@79.7	256	128	7	91.2
note10@79.3	note10@79.3	256	128	11	91.4
ofa_mbv3_w1.2	note10@79.3	256	128	11	90.8
note10@79.3	note10@79.3	256	128	7	91.2
ofa_mbv3_w1.2	note10@79.3	256	128	7	90.7
note10@79.3	note10@79.3	128	64	7	90.7
ofa_mbv3_w1.2	note10@79.3	128	64	7	89.4

tation. However, the problem is when we combine specialized OFA network and pre-extraction of source feature for edge device. As there are diverse specialized OFA networks for diverse hardware platforms, we cannot store all the source feature for each specialized OFA networks.

If feature extractors for source and target domain data are different, then the accuracy is near zero. However, we observe that features extracted by OFA networks or specialized OFA networks which share the same OFA network are compatible to each other. We propose to store

Table 4.12 Pareto frontier methods on Office31 with pre-extraction of source feature (cont.)

source network	target network	d	d_1	iterations	avg accuracy
note10@79.3	note10@79.3	256	128	5	90.9
ofa_mbv3_w1.2	note10@79.3	256	128	5	90.4
note10@79.3	note10@79.3	128	64	5	90.8
ofa_mbv3_w1.2	note10@79.3	128	64	5	89.6
note10@79.3	note10@79.3	128	64	3	90
ofa_mbv3_w1.2	note10@79.3	128	64	3	89.1
note10@78.4	note10@78.4	128	64	5	89.4
ofa_mbv3_w1.2	note10@78.4	128	64	5	88.2
note10@78.4	note10@78.4	128	64	3	89.3
ofa_mbv3_w1.2	note10@78.4	128	64	3	88.4
note10@75.5	note10@75.5	128	64	3	89
ofa_mbv3_w1.0	note10@75.5	128	64	3	88.4
note10@73.6	note10@73.6	128	64	3	86.6
ofa_mbv3_w1.0	note10@73.6	128	64	3	88.1
note10@71.4	note10@71.4	128	64	3	85.3
ofa_mbv3_w1.0	note10@71.4	128	64	3	85.2

Table 4.13 Pareto frontier methods on Office-Home with pre-extraction of source feature

source network	target network	d	d_1	iterations	avg accuracy
note10@79.7	note10@79.7	256	64	3	73.3
ofa_mbv3_w1.2	note10@79.7	256	64	3	72.7
note10@79.7	note10@79.7	256	64	5	73.5
ofa_mbv3_w1.2	note10@79.7	256	64	5	73
note10@79.7	note10@79.7	512	64	5	73.4
ofa_mbv3_w1.2	note10@79.7	512	64	5	72.9
note10@79.7	note10@79.7	256	64	7	73.5
ofa_mbv3_w1.2	note10@79.7	256	64	7	72.9
note10@79.7	note10@79.7	512	64	7	73.6
ofa_mbv3_w1.2	note10@79.7	512	64	7	72.9
note10@79.7	note10@79.7	512	64	11	73.6
ofa_mbv3_w1.2	note10@79.7	512	64	11	73
note10@79.3	note10@79.3	128	64	3	72.3
ofa_mbv3_w1.2	note10@79.3	128	64	3	72
note10@79.3	note10@79.3	256	64	3	73
ofa_mbv3_w1.2	note10@79.3	256	64	3	72
note10@78.4	note10@78.4	128	64	3	71.6
ofa_mbv3_w1.2	note10@78.4	128	64	3	70.9
note10@76.6	note10@76.6	128	64	3	67
ofa_mbv3_w1.0	note10@76.6	128	64	3	67.1
note10@75.5	note10@75.5	128	64	3	66.2
ofa_mbv3_w1.0	note10@75.5	128	64	3	66.2
note10@76.6	note10@71.4	128	64	3	60.3
ofa_mbv3_w1.0	note10@71.4	128	64	3	60.2

only one source feature for each OFA network. It is more efficient and scalable because all the specialized OFA networks are derived by several OFA networks. Therefore, we only store several source features.

Table 4.11 and Table 4.12 shows the accuracy of pre-extraction of source feature for Pareto frontier methods on Office31 along with baseline methods. We found Pareto frontier methods in grid search manner. Specifically, we found 17 and 12 Pareto optimal points for Office31 and Office-Home, respectively. It is reasonable to mainly consider these points as working points because the accuracy and latency of the methods are superior than the others. In the table, source network refers to the network which is used to extract source domain data and target network means it is used to extract target domain data. If source network and target network is the same, then it assumes the baseline setting which assumes the source feature is stored along to the network used or we can access to the source domain data. If source network is OFA network and target network is specialized OFA network, then we assume the pre-extraction of source feature. Although there are some accuracy losses, we observe that the accuracy of pre-extraction of source feature for each specialized OFA network is similar to that of baseline setting. Therefore, we can store source feature for several OFA networks instead of all the specialized OFA networks to accomplish pre-extraction of source feature.

Table 4.13 shows the pre-extraction of source feature for Pareto fron-

Table 4.14 Accuracy results on Office31 dataset for small target data scenario

network	d	d_1	iterations	avg accuracy
Resnet50	512	128	11	82.6
note10@78.4	151	90	3	84.6

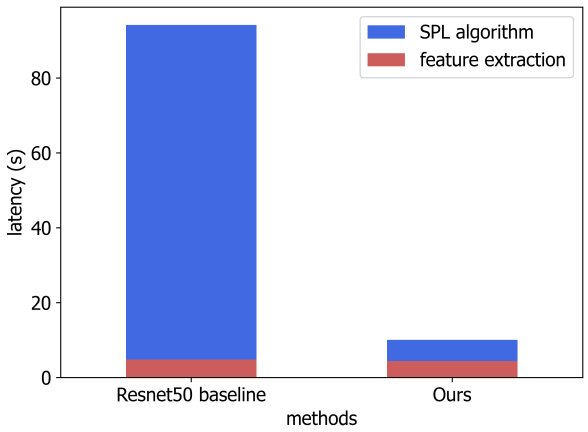

Figure 4.7 Latency results on Office31 dataset for small target data scenario

tier methods on Office-Home dataset along with baseline methods. We present 12 Pareto optimal points. There are only small accuracy losses while the networks are compatible if they are derived from the same OFA network. Therefore, we verify the effectiveness of our method.

4.4.6 Results for Small Target Data Scenario

Although labeling hundreds to thousands of target data images is standard for the Office datasets, it is not realistic to label such amount of data on edge device in real use cases. We assume more realistic scenario

which includes smaller target data. This is the scenario where latency of algorithm is important. We randomly sampled 64 items from original target domain data while we keep the source domain data intact. Therefore, the problem is to label 64 images instead of hundreds or thousands using the information gained from source domain data. The seed for the random number generator is fixed to certain number to make our sampling strategy stable. In our observation, some of the target domain classes are not sampled at all thereby it is not appropriate to use structured prediction which includes clustering algorithm with initial points of the number of target domain classes. Therefore, we exclude the structured prediction in the small target data scenario and achieve about 4% accuracy gain. Table 4.14 shows the accuracy results for the scenario. Accuracy is slightly degraded due to small size of target data information. Our approach scores 2% higher than the baseline method on Office31 dataset. Figure 4.7 shows the latency results for the scenario. Our approach is 9.36 times faster than the baseline method by using lightweight backbone network and smaller SPL algorithm parameter. In real use case, waiting for more than a minute is not tolerable while our approach services a query in about 10 seconds, which is more interactive.

4.4.7 Results for Object Detection

We assume object detection scenario, which is famous application for edge device. We use Faster R-CNN [92] implementation from PyTorch

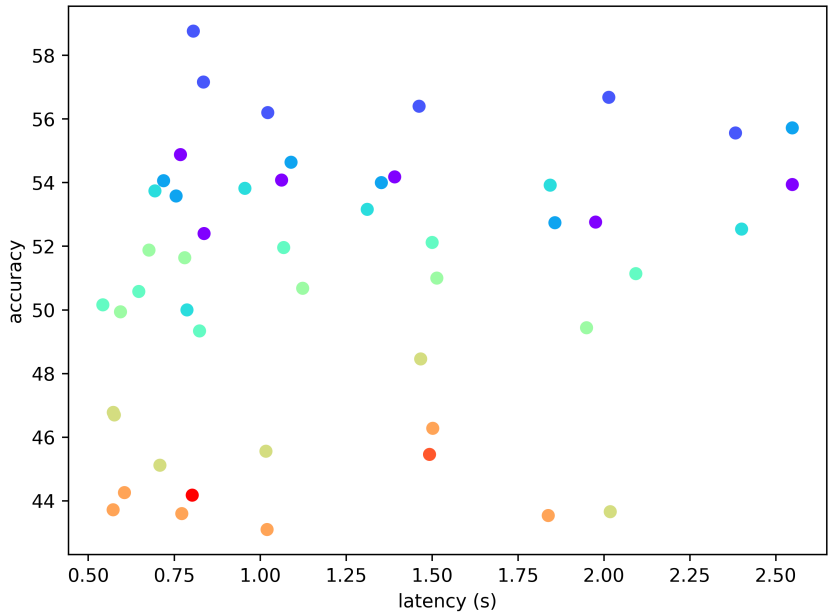

Figure 4.8 Object Detection Results on PennFudan dataset

[69] for experiments. We use PennFudan dataset, which is consisted of 170 images with 375 labeled pedestrians. The goal of object detection algorithm is to find bounding boxes of pedestrians in the input images. The dataset is divided to form 120/25/25 images of training/validation/test set. We conduct co-optimization of backbone network selection and object detection algorithm parameter selection. The search space we consider is 8 OFA backbone networks and an object detection algorithm parameter (i.e., 6 values) namely sampling ratio of multi scale ROI align module. As the size of search space is 48, we conduct grid search based on the accuracy using validation set. Figure 4.8 shows the latency and accuracy of various backbone networks and object detection algorithm

parameters. We averaged 5 runs to get stable latency and accuracy values. The red dot indicates MobileNet v2 baseline and ResNet baseline which is clearly not in pareto optimal frontier. Instead of the baseline method which uses ResNet-50 and the sampleing ratio of two, we use note10@76.6 backbone network and the sampling ratio of one. The total latency is 2.43 times faster (i.e., from 1.40 seconds to 0.58 seconds) while the COCO-style mAP is about 4 points higher (i.e., from 45.3 to 49.8) compared to the baseline method.

4.4.8 Results for Classifier Fitting Using Intermediate Domain

As the latency of SPL algorithm is a large portion in the total runtime, it is desirable to fit a classifier instead of executing SPL algorithm from scratch all the time. We conduct an experiment which has the scenario of source domain, intermediate domain and target domain. The goal of the algorithm is to label target domain data by using the information gained from source domain and intermediate domain when the labels of intermediate domain and target domain doesn't exist. First of all, the algorithm labels intermediate domain data using SPL algorithm with source domain data. Second, the algorithm fits a classifier using the labels of source domain data and intermediate data. After all, the algorithm labels target domain data using a classifier without SPL algorithm. In this scenario, labeling target domain data is fast as the latency of SPL algo-

rithm diminishes. In our experiment, a classifier is consisted of a simple fully-connected layer. If we fit a classifier after note10@78.4 backbone network using Amazon domain data and tested on Webcam domain data, then the accuracy is only 68.8% due to domain shift. If we use SPL algorithm using Amazon domain data to label Webcam domain data, then the accuracy is 90.8%. If we use SPL algorithm using Amazon domain data to label DSLR domain data, then the accuracy is 95%. If we fit a classifier using Amazon domain data and 95% corrected DSLR domain data to test Webcam domain data, then the accuracy is 90.6%. Therefore, we get 90.6% accuracy instead of 90.8% accuracy using intermediate domain information and we reduce the SPL algorithm latency by using classifier fitting.

Note that the process of generating labels by SPL algorithm and perform classifier fitting is analogous to self-training. Implementing more sophisticated algorithm for combining self-training and SPL algorithm can provide both high accuracy and low latency for target domain. Furthermore, choosing hyperparameter via proxy dataset can be altered by self-training criteria. It is left for future work.

4.4.9 Summary

We integrate all the methods previously mentioned and show the summary of the results. For accuracy results on Office31 in Table 4.15, we use the proxy dataset Office-Home to train accuracy and latency predic-

Table 4.15 Summary accuracy results on Office31 dataset

source network	target network	d	d_1	iterations	avg accuracy
Resnet50	Resnet50	512	128	11	89.6
ofa_mbv3_w1.2	note10@78.4	151	90	3	89.6

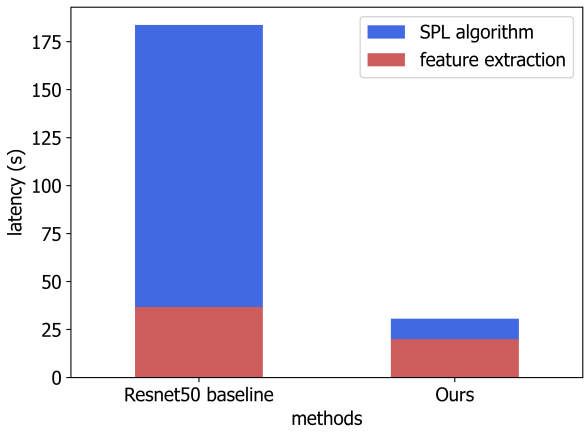

Figure 4.9 Summary latency results on Office31 dataset

tors. Evolutionary search is performed on the proxy dataset and find the suitable feature extractor and SPL setting.

Then we measure the unsupervised domain adaptation accuracy on the Office31 dataset using the found setting and report the accuracy and the latency on NVIDIA Jetson TX2. We also apply pre-extraction of source feature which set source network to be OFA network instead of specialized OFA network. Note that the accuracy is the same between the baseline and the proposed method. The latency result is shown in Figure 4.9. Our approach can reduce the total latency by 5.99× without loss of accuracy on Office31 dataset.

Table 4.16 Summary accuracy results on Office-Home dataset

source network	target network	d	d_1	iterations	avg accuracy
Resnet50	Resnet50	1024	128	11	71.0
ofa_mbv3_w1.2	note10@78.4	171	110	3	71.1

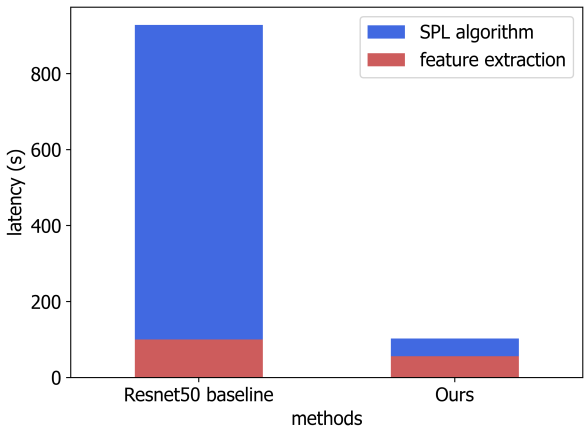

Figure 4.10 Summary latency results on Office-Home dataset

Table 4.16 shows the accuracy results on Office-Home dataset. Differently, we use the proxy dataset Office31 and the test dataset Office-Home. The accuracy is almost the same between the baseline method and our approach. The latency result is shown in Figure 4.10. We can reduce the total latency by 9.06× on Office-Home compared to baseline method.

Note that characteristics of edge device (e.g., memory bandwidth and small neural processing unit) are not considered however we use latency measured on working system for edge device (i.e., NVIDIA Jetson TX2). More sophisticated method which considers edge device characteristics

is left for future work.

4.5 Conclusion

We proposed applying pre-trained ImageNet model to feature extractor of unsupervised domain adaptation for edge device. First, we report that combining large feature extractors and the algorithm which doesn't update the feature extractor is an effective method to achieve state-of-the-art accuracy on Office31 and Office-Home datasets. And we show that highly optimized feature extractor is crucial to achieve state-of-the-art efficiency for edge device. We use a predictor-guided evolutionary search to explore accuracy-latency space of the proposed method. We show that the evolutionary search algorithm is transferable between Office31 and Office-Home datasets especially for middle and low latency regimes. We observe that pre-extraction of source feature for edge device is viable by storing source features along with OFA networks instead of storing source features for all the specialized OFA networks. We report 5.99$\times$ and 9.06$\times$ latency reduction on Office31 and Office-Home datasets, respectively. Lastly, we experiment more realistic scenarios which include small target domain data, object detection, and classifier fitting. We show that our method is still effective on those scenarios.

Chapter 5

Augmenting Few-Shot Learning with Supervised Contrastive Learning

This work was published on IEEE *Access* journal, Apr. 2021 [93]. Few-shot learning deals with a small amount of data which incurs insufficient performance with conventional cross-entropy loss. We propose a pre-training approach for few-shot learning scenarios. That is, considering that the feature extractor quality is a critical factor in few-shot learning, we augment the feature extractor using a contrastive learning technique. It is reported that supervised contrastive learning [6] applied to base class training in transductive few-shot training pipeline leads to improved results, outperforming the state-of-the-art methods on Mini-ImageNet and CUB. Furthermore, our experiment shows that a much larger dataset is needed to retain few-shot classification accuracy when domain-shift degradation exists, and if our method is applied, the need for a large dataset is eliminated. The accuracy gain can be translated to a runtime reduction of $3.87\times$ in a resource-constrained environment.

5.1 Introduction

The impressive results of deep learning-based methods are mainly achieved using a large amount of labeled data [50,94]. However, massive image labeling is labor-intensive, and a balanced dataset is challenging to obtain. By contrast, humans show excellent generalization performance from only one or a few examples, bringing motivation to the field of few-shot learning [16, 95–97]. Likewise, the aim of few-shot learning is to predict unlabeled data based on the observation of a few labeled data (e.g., one or five examples per class). Likewise, the aim of few-shot learning is to predict unlabeled data based on the observation of a few labeled data (e.g., one or five examples per class).

Compared with traditional inductive few-shot learning, two settings are introduced to address the low data count. A semi-supervised few-shot setting [17, 98] assumes that the model can utilize information from additional unlabeled data. Better accuracy can be obtained by increased amount of unlabeled data. A transductive few-shot setting [99, 100] accords that the model can access all the test data at once instead of one by one in the inference procedure. In the scope of this study is confined to the transductive few-shot setting as it is simple, yet effective [101] to achieve state-of-the-art result [9].

Conventional few-shot learning algorithms implement a two-stage training pipeline. *Base classes*, which are used only in the first stage of

training, are large, separate classes for training the feature extractor, usually with conventional cross-entropy loss. In the consecutive stage, *novel classes*, which are a disjoint set of base classes, are learning targets with a few training examples per class. The first training stage attempts to learn general, transferable visual features from the base classes, whereas the main few-shot algorithms are implemented in the second stage to predict images from the novel classes.

As a feature extractor's performance is empirically related to the final classification accuracy, it is reasonable to use various augmentation techniques during the first training stage. These techniques [24, 25, 102, 103] are motivated by large-scale image classification tasks, such as ImageNet. Supervised contrastive learning [6] is proposed to replace cross-entropy loss by applying self-supervised representation learning with label information. It is examined that supervised contrastive loss instead of simple cross-entropy loss in the first training stage improves the final classification accuracy by a large margin, especially when the dataset is not large.

Assume that a few-shot learning task is running on an edge device, considering the scale of the problem. However, as the cost of the step is high—nearly a hundred epochs of training the entire dataset—the base class training step is presumably processed by the server. The cost of supervised contrastive learning is an additional pretraining step at the base class training, which is amortized and processed efficiently by servers.

With the accuracy gain obtained by the supervised contrastive learning, one can optimize the runtime latency of the algorithm with a simple method such as an early stopping.

Few-shot learning is associated with self-supervised representation learning, as noted in [104]. Both approaches have a similar goal: training the model with few or no data labels. Self-supervised representation learning is a method of unsupervised learning, which aims to learn from a dataset with no annotation. Instead, it learns using pretext information, such as the relative location of image patches or the rotation classification of images. Contrastive learning is a form of self-supervised representation learning that trains the model to classify similar (positive) samples and dissimilar (negative) samples in the embedding space. As supervised contrastive learning is an extension of contrastive learning, it implies the gain obtained in our experiment.

We observe that the feature extractor trained on a large, general dataset (i.e., Tiered-ImageNet) performs better than the feature extractor trained on a small, task-specific dataset (i.e., CUB) when evaluating a few-shot learning task. In our experiment, supervised contrastive learning improves the few-shot classification accuracy to the extent that even when trained on a small, task-specific dataset, it performs better than the feature extractor trained on a large, general dataset. Therefore, it is data-efficient and obtains superior performance without resort to a large dataset.

In summary, the contributions of our study are as follows:

- We propose using supervised contrastive learning in the first stage of few-shot learning to boost classification accuracy on the Mini-ImageNet and CUB datasets. Our method is referred as SPTA following the name of combined methods.

- We study the domain-shift setting, in which the feature extractor is trained on a different dataset, and the few-shot algorithm is evaluated on a fine-grained classification dataset, showing that a large dataset (i.e., Tiered-ImageNet) is needed to overcome domain-shift degradation. However, when supervised contrastive learning is applied to the CUB dataset, the case without a large dataset can score higher than the case with a large dataset.

5.2 Related Works

Few-shot learning: There are many approaches to address few-shot learning tasks with less amount of data. Gradient descent-based approaches [105–107] learn how to re-adjust a model with a few gradient descent iterations to deal with a few-shot learning task. The model-agnostic meta-learning (MAML) [105] method trains the model with many tasks to generalize a new task efficiently. Reptile [106] is a first-order gradient-based meta-learning algorithm that trains the initialization of model parameters. [107] proposed a long short-term memory (LSTM)-based meta-learner whose states represent the update of the model parameter.

Metric-learning-based approaches [16, 108–110] learn distance metrics between a support set (training data of the target task) and a query set (test data of the target task) better by reforming feature embedding. [108] introduced Siamese convolutional neural networks that learn generic visual features on the character recognition task. The matching network [16] architecture is inspired by a memory-augmented neural network and generates a weighted nearest neighbor classifier using the distance between samples. Prototypical networks [109] utilize episodic training and assign each class to each prototype in the representation space to predict new data based on the distance metric to each prototype. [110] proposed using an additional data sample generator, which is trained with meta-learning methods, to augment the model training.

Transductive few-shot methods [7–9, 100, 101, 111, 112] assume that the model simultaneously accesses all the query set. A transductive episodic-wise adaptive metric (TEAM) [100] defined the optimization process as a standard semi-definite programming problem to train a generalizable classifier. A distribution propagation graph network (DPGN) [111] proposed utilizing both the distribution-level and instance-level relations by designing a dual complete graph network consisting of a point graph and a distribution graph. [101] proposed transductive fine-tuning, which pursues outputs with a peaked posterior or low Shannon entropy, and a hardness metric to deliver a standardized evaluation protocol. [7] proposed the prototype rectification, which lowers the class prototype's intra-class

bias and cross-class bias and verifies the method theoretically. A synthetic information bottleneck (SIB) [112] introduced an empirical Bayes approach and a two-network architecture consisting of a synthetic gradient network and an initialization network to perform the synthetic gradient descent. LaplacianShot [8] implemented a constrained graph clustering method that attaches the query samples to the nearest prototype, and a pairwise Laplacian term advocates similar samples to output the same label. Transductive information maximization (TIM) [9] maximizes the mutual information between the query features and the predicted query label by minimizing the conditional entropy and maximizing the marginal entropy, and the alternating direction optimizer enables faster convergence than the typical gradient descent optimizer.

Contrastive learning: Contrastive learning [26, 113–117] is a self-supervised learning method inspired by noise contrastive estimation [118, 119] or N-pair losses [120]. [113] proposed the use of a non-parametric softmax classifier to increase the instance-level distance on a 128-dimensional unit sphere after the CNN extracts a feature vector of the image. [114] improved contrastive predictive coding to implement a pretraining stage with a feature extractor and a context network to predict the spatial location of the image patches. Deep InfoMax [115] proposed an approach for training an encoder that maximizes the mutual information between the input data and output features. [116] aimed to maximize the mutual information between different views of the same image by pulling views of the

same scene together and pushing views of different scenes apart. Time-contrastive networks (TCN) [117] proposed learning from multi-view video by pulling the anchor and positive images together while pushing negative images apart. SimCLR [26] implemented two data augmentation paths and a learnable nonlinear transformation to train an encoder with a large batch by pulling the feature embedding from the same image. Supervised contrastive learning [6] is an extension of conventional contrastive learning that has been modified for supervised classification.

5.3 Methodology

This section introduces the proposed idea in detail.

5.3.1 Examining A Few-shot Learning Method

In this study, we examine the transductive information maximization (TIM) few-shot learning algorithm [9]. First, a feature extractor transforms an input image into embedded features. TIM maximizes the modified mutual information between the query image's feature and the query label by updating the soft-classifier's trainable weights. To maximize the information, TIM minimizes the conditional entropy and maximizes the marginal entropy. Minimizing conditional entropy aims to make confident predictions by modeling the cluster assumption, which implies that the classification criterion should not be present in the dense regions of the unlabeled features. Maximizing marginal entropy pushes the

marginal distribution of labels to be uniform, which attempts to avoid the solution of outputting only one class. Together with the conventional cross-entropy loss, the TIM loss is defined as follows:

$$L^{tim} = -\frac{\lambda}{|S|} \sum_{i \in S} \sum_{n=1}^{N} y_{in} \log p_{in} - \mathscr{I}$$

$$\mathscr{I} := -\sum_{n=1}^{N} \hat{p_n} \log \hat{p_n} + \frac{\alpha}{|Q|} \sum_{i \in Q} \sum_{n=1}^{N} p_{in} \log p_{in}$$

where p_{in} is the posterior distribution over the labels given the features and $\hat{p_n}$ is the marginal distribution over the query labels.

Given the loss objective, two optimization methods are presented [9]. One is a conventional gradient descent (TIM-GD) method that minimizes the loss objective through mini-batch sampling. Although TIM-GD shows the best results, it is two orders of magnitude slower than inductive methods, which leads to the second method called the alternating direction method (TIM-ADM), which divides the problem into two more manageable subproblems and optimizes them iteratively. TIM-ADM shows competitive results compared to TIM-GD while being one order of magnitude faster. In both methods, sufficiently large number of iterations were required to converge to the best results. Typical values for the number of iterations for TIM-GD and TIM-ADM were 1,000 and 150, respectively.

5.3.2 Augmenting Few-shot Learning with Supervised Contrastive Learning

The quality of a feature extractor is one of the main challenges in improving a few-shot learning algorithm because it is directly related to the quality of the feature embeddings. Supervised contrastive learning [6] is an extension of self-supervised representation learning; it has a similar two-stage training procedure, as shown in Figure 5.1. The first stage prepares two copies of an input image and preprocesses them. An encoder network then transforms the images into normalized embedding, and an additional projection network transforms the embedding into a low-dimensional embedding. Supervised contrastive loss is computed on the low-dimensional embedding by attracting positive samples, which have the same class label or are from the same copied images, and by repelling the negative samples. The supervised contrastive loss is defined as follows:

$$L^{sup} = -\sum_{i \in I} \frac{1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp\left(z_i \cdot z_p / \tau\right)}{\sum\limits_{a \in A(i)} \exp\left(z_i \cdot z_a / \tau\right)}$$

where z_l is the low-dimensional embedding, τ is a temperature parameter, $A(i) \equiv I \setminus \{i\}$, i is an anchor index, and $P(i) \equiv \{p \in A(i) : \bar{y}_p = \bar{y}_i\}$ is the set of indices of all positives except the anchor. The inner product operation on the embedding space measures the similarity between two feature embeddings. The loss is minimized when an anchor's feature embedding is similar to all the positive's feature embeddings and is different

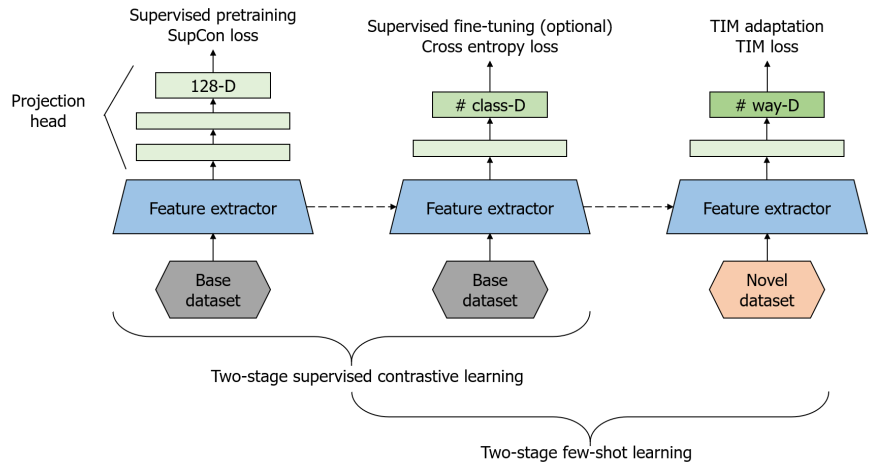

Figure 5.1 The proposed pretraining approach for few-shot learning consists of a multi-stage training process.

from all the negative's feature embeddings. The loss is generalized from the conventional SimCLR [26] self-supervised contrastive loss to support multiple positives in the multiviewed batch.

Notably, performing supervised contrastive learning in the first stage of few-shot learning is proposed instead of performing the conventional training with base classes and cross entropy. The second step of the training procedure is to discard the projection network and fine-tune the encoder network with a new classifier. As representation learning implies, the encoder network becomes discriminative during the first step of the training procedure; therefore, the fine-tuning process is relatively short and is guided by a lower learning rate. Note that we fine-tuned the feature extractor with the base class and cross-entropy, which was pre-trained in

the first stage of supervised contrastive learning. The fine-tuning process in supervised contrastive learning is optional; we can skip the process that does not touch the feature extractor because we only use the feature extractor at the end. When we follow the linear evaluation protocol, we keep the feature extractor intact, which implies that we skip the fine-tuning process. We chose to use the fine-tuning approach because it produces better results than no fine-tuning as shown in Table 5.1. After the supervised contrastive learning, optional fine-tuning follows. One-shot and five-shot five-way classification accuracy on Mini-ImageNet is reported. Our results are averaged over 10,000 episodes.

In addition, Figure 5.1 shows the details about proposed pretraining approach. The first stage of supervised contrastive learning uses supervised contrastive loss and projection head with the base dataset to learn visual representations. The second stage of supervised contrastive learning uses conventional cross-entropy loss with the base dataset to fine-tune the feature extractor. This two-stage supervised contrastive learning comprises the first stage of few-shot learning. The second stage of few-shot learning uses TIM [9] loss and the feature extractor fixed with the novel dataset to perform TIM adaptation. If the supervised fine-tuning becomes standard supervised training and the supervised contrastive pretraining is skipped, then the entire pipeline is the same as in the baseline method [9]

In our experiment, we added a supervised contrastive learning approach as an additional pretraining step in the first few-shot training

Table 5.1 Summary results for the fine-tuning setting.

fine-tune	1-shot	5-shot
✓	78.83	87.76
✗	75.94	86.16

stage. Furthermore, we fine-tuned the feature extractor with cross-entropy loss using the base class dataset.

5.4 Experiments

An implementation of our SPTA is publicly available[1].

Datasets: We examined three few-shot learning datasets, namely Mini-ImageNet, Tiered-ImageNet, and CUB. The **Mini-ImageNet** dataset [16] is composed of 100 classes from the ImageNet [10] dataset. It has 64/16/20 base/validation/novel classes, respectively, with 600 84×84 sized images per class following the split proposed by [107]. The **Tiered-ImageNet** [17] is composed of 608 classes from the ImageNet dataset. It has 351/97/160 base/validation/novel classes, respectively, with 779,165 84×84 sized images in total following the split proposed by [17]. Finally, the **Caltech-UCSD Birds 200-2011** [15] (CUB) dataset is composed of 200 classes and 11,788 images in total. It has 100/50/50 base/validation/novel classes, respectively, with 84×84 sized images following the split proposed by [121].

[1] https://github.com/taemin-lee/SPTA

Evaluations: We evaluate the algorithm's score by comparing the final predicted label at the second stage with the ground truth label. For each few-shot learning episode, N-way K-shot tasks with T queries per class were randomly selected from the dataset of novel classes. We chose $N = 5$, $T = 15$, and $K = 1$ for 1-shot or $K = 5$ for 5-shot classification. We followed the evaluation protocol in [9].

Implementation details: We examined mainly three different backbone network models, namely ResNet-18, MobileNet, and WRN28-10, following the implementation of [9,122]. Note that the number after ResNet indicates the depth of the network. Nevertheless, we report ResNet variants in one group following the convention of [8, 9]. We mainly investigated the alternating direction method (ADM) version of the TIM algorithm, which is faster than the gradient descent (GD) version[2]. We have added a prototype estimation technique [7, 8] to TIM. This further improved the 1-shot classification accuracy. We used a PyTorch [69] re-implementation of RandAugment[3] on the preprocessing stage of supervised contrastive learning[4] with $N = 3$ and $M = 20$ to implement modified stacked RandAugment. When pretraining, we used the training epochs of 1,000 for the supervised contrastive learning, and this was followed by five epochs of fine-tuning. Our method is referred as SPTA following the name of combined methods (i.e., supervised contrastive

[2]https://github.com/mboudiaf/TIM
[3]https://github.com/ildoonet/pytorch-randaugment
[4]https://github.com/HobbitLong/SupContrast

learning, prototype estimation, and TIM-ADM).

5.4.1 Comparison to the State-of-the-Art

We evaluated the 5-way 1-shot and 5-shot classification accuracy of our method on the Mini-ImageNet and CUB datasets. Our results were averaged over 10,000 episodes following [9, 122] and are summarized in Table 5.2. In the table, we present methods with 1-shot accuracy over 60% on Mini-ImageNet, and the methods are arranged in ascending order. The results are categorized according to the backbone network the algorithms use. The bold values are the best results within the algorithms that use the same backbone network. We excluded results from a semi-supervised setting because these methods require additional data at test time. We observed that consistent accuracy gains over the existing methods, regardless of the backbone network models. For example, 1-shot accuracy improved by more than 6% whereas 5-shot accuracy improved by more than 5% with the MobileNet network backbone on Mini-ImageNet surpassing all the existing methods with the ResNet network backbone model on Mini-ImageNet. With the ResNet-18 network backbone model on Mini-ImageNet, the 1-shot accuracy improved by almost 5% whereas the 5-shot accuracy improved by more than 2% surpassing all the existing methods with the WRN28-10 network backbone model on Mini-ImageNet. Therefore, the gain of our method is comparable to the selection of a better network architecture on Mini-ImageNet in improving

Table 5.2 Accuracy comparison to the state-of-the-art methods for five-way classification on Mini-ImageNet and CUB.

		Mini-ImageNet		**CUB**	
Method	Backbone	1-shot	5-shot	1-shot	5-shot
SimpleShot [122]	MobileNet	61.30	78.37	-	-
LaplacianShot [8]	MobileNet	70.27	80.10	-	-
Ours-SPTA	MobileNet	**76.57**	**85.82**	**83.76**	**89.01**
TEAM [100]	ResNet-18	60.07	75.90	80.16	87.17
MTL [123]	ResNet-12	61.2	75.5	-	-
vFSL [124]	ResNet-12	61.23	77.69	-	-
Neg-cosine [125]	ResNet-18	62.33	80.94	72.66	89.40
AFHN [126]	ResNet-18	62.38	78.16	70.53	83.95
MetaOpt [127]	ResNet-12	62.64	78.63	-	-
SimpleShot [122]	ResNet-18	62.85	80.02	-	-
Distill [128]	ResNet-12	64.82	82.14	-	-
ConstellationNet [129]	ResNet-12	64.89	79.95	-	-
DeepEMD [130]	ResNet-12	65.91	82.41	75.65	88.69
FEAT [131]	ResNet-12	66.78	82.05	-	-
IEPT [132]	ResNet-12	67.05	82.90	-	-
TRAML [133]	ResNet-12	67.10	79.54	-	-
CAN+T [134]	ResNet-12	67.19	80.64	-	-
MELR [135]	ResNet-12	67.40	83.40	-	-
DPGN [111]	ResNet-12	67.77	84.60	75.71	91.48
SIB+IFSL [136]	ResNet-10	68.85	80.32	-	-
LaplacianShot [8]	ResNet-18	72.11	82.31	80.96	88.68
TIM-GD [9]	ResNet-18	73.9	85.0	82.2	90.8
Ours-SPTA	ResNet-18	**78.83**	**87.76**	**88.81**	**93.11**
LEO [137]	WRN28-10	61.76	77.59	-	-
CC+rot [104]	WRN28-10	62.93	79.87	-	-
AWGIM [138]	WRN28-10	63.12	78.40	-	-
SimpleShot [122]	WRN28-10	63.50	80.33	-	-
FEAT [131]	WRN28-10	65.10	81.11	-	-
Transductive tuning [101]	WRN28-10	65.73	78.40	-	-
Logistic Regression with DC [139]	WRN28-10	68.57	82.88	79.56	90.67
SIB [112]	WRN28-10	70.0	79.2	-	-
BD-CSPN [7]	WRN28-10	70.31	81.89	-	-
SIB+IFSL [136]	WRN28-10	73.51	83.21	-	-
LaplacianShot [8]	WRN28-10	74.86	84.13	-	-
TIM-GD [9]	WRN28-10	77.8	87.4	-	-
Ours-SPTA	WRN28-10	**80.32**	**88.76**	-	-

Table 5.3 Accuracy comparison to the state-of-the-art methods for five-way classification on Tiered-ImageNet.

		Tiered-ImageNet	
Method	Backbone	1-shot	5-shot
SimpleShot [122]	MobileNet	69.47	85.17
LaplacianShot [8]	MobileNet	79.13	86.75
Ours-SPTA	MobileNet	**79.17**	**87.16**
MetaOpt [127]	ResNet-12	65.99	81.56
SimpleShot [122]	ResNet-18	69.09	84.58
FEAT [131]	ResNet-12	70.80	84.79
DeepEMD [130]	ResNet-12	71.16	86.03
Distill [128]	ResNet-12	71.52	86.03
MELR [135]	ResNet-12	72.14	87.01
IEPT [132]	ResNet-12	72.24	86.73
DPGN [111]	ResNet-12	72.45	87.24
CAN+T [134]	ResNet-12	73.21	84.93
SIB+IFSL [136]	ResNet-10	78.03	85.43
LaplacianShot [8]	ResNet-18	78.98	86.39
TIM-GD [9]	ResNet-18	79.9	**88.5**
Ours-SPTA	ResNet-18	**81.16**	88.43

performance.

The results for the Tiered-ImageNet are presented in Table 5.3. The results are categorized according to the backbone network the algorithms use. The bold values are the best results within the algorithms that use the same backbone network. Again, our results are averaged over 10,000 episodes, and our method scores competitive accuracy results compared to the existing methods. By comparison, the score gain on the Tiered-ImageNet is not as high as that of Mini-ImageNet and CUBs (i.e., less than or approximately 1%). We assume that the Tiered-ImageNet is a very large dataset compared to Mini-ImageNet and CUB, and thus the visual representation learned from the Tiered-ImageNet is sufficiently discriminative with conventional cross-entropy loss. This implies that our method is data-efficient in terms of the dataset size. Thus, it works particularly well with small datasets, reducing the cost of data preparation.

5.4.2 Ablation Study

We evaluated the influence of prototype estimation and supervised contrastive learning on the final accuracy of the method. Instead of the simple mean of support set examples, the prototype estimation technique calculates better initialization points by combining support set examples and query set examples. The results are reported in Table 5.4, and all of them used ResNet-18 as a backbone network model. Note that proto refers to prototype estimation and supcon refers to supervised contrastive

Table 5.4 Ablation study on the influence of prototype estimation and supervised contrastive learning.

				Mini-ImageNet	
Method	Backbone	proto	supcon	1-shot	5-shot
TIM-ADM [9]	ResNet-18			73.6	85.0
	ResNet-18	✓		74.86	84.95
	ResNet-18		✓	77.38	**87.82**
Ours-SPTA	ResNet-18	✓	✓	**78.83**	87.76
				Tiered-ImageNet	
Method	Backbone	proto	supcon	1-shot	5-shot
TIM-ADM [9]	ResNet-18			80.0	**88.5**
	ResNet-18	✓		**81.34**	88.41
	ResNet-18		✓	80.22	88.49
Ours-SPTA	ResNet-18	✓	✓	81.16	88.43
				CUB	
Method	Backbone	proto	supcon	1-shot	5-shot
TIM-ADM [9]	ResNet-18			81.9	90.7
	ResNet-18	✓		83.66	90.72
	ResNet-18		✓	87.63	93.08
Ours-SPTA	ResNet-18	✓	✓	**88.81**	**93.11**

Table 5.5 Summary of domain-shift setting results.

Method	domain	Backbone	1-shot	5-shot
TIM-GD [9]	CUB →CUB	ResNet-18	82.2	90.8
	Mini-ImageNet →CUB	ResNet-18	53.04	71.04
	Tiered-ImageNet →CUB	ResNet-18	82.58	91.39
Ours-SPTA	CUB →CUB	ResNet-18	**88.81**	**93.11**
	Mini-ImageNet →CUB	ResNet-18	51.50	68.69
	Tiered-ImageNet →CUB	ResNet-18	82.80	90.70

learning. The bold values are the best results among the methods. Our results are averaged over 10,000 episodes. From the TIM-ADM baseline method, prototype estimation and supervised contrastive learning were added one by one. We observe that most of the accuracy gain on the Mini-ImageNet and CUB datasets is from the supervised contrastive learning, and the prototype estimation improves 1-shot accuracy further, while it has a marginal impact on 5-shot accuracy. For example, 1-shot accuracy improved by almost 7%, whereas 5-shot accuracy improved by more than 2% on the CUB dataset. Most of the gain in 1-shot accuracy on the CUB dataset is from supervised contrastive learning (i.e., more than 5%), whereas the gain of the prototype estimation is less than 2%. Similarly, most of the gain in 5-shot accuracy on the CUB dataset is from supervised contrastive learning, whereas the gain of the prototype estimation is negligible. We assume that a 5-shot setting provides sufficient information to build a proper prototype for each class, even without the prototype estimation method.

5.4.3 Domain-Shift

We measure the impact of the domain-shift and report the results in Table 5.5. Note that no domain-shift and domain-shift from the larger dataset are presented. The bold values are the best results among the domain settings. All results used ResNet-18 as a backbone network model, and our results were averaged over 10,000 episodes. Domain A $\rightarrow$B implies that the feature extractor is trained on dataset A, whereas the few-shot learning method is evaluated on dataset B, similar to the setting from [121]. Domain CUB $\rightarrow$CUB is the baseline result without a domain-shift. Note that the domain-shift from a slightly large-sized dataset to a smaller one (i.e., Mini-ImageNet $\rightarrow$CUB) drastically degrades the accuracy of the few-shot learning method. The results show a drop in 1-shot accuracy of approximately 29% and 19% in 5-shot accuracy. By comparison, the domain-shift from a much larger dataset (i.e., Tiered-ImageNet $\rightarrow$CUB) is slightly better than the no domain-shift (i.e., CUB $\rightarrow$CUB) baseline setting. It improves 1-shot accuracy by approximately 1%. The results show that the existing method requires a much larger dataset in the source domain to build an effective feature extractor under a domain-shift. By contrast, the proposed method provides better feature extraction when using a smaller dataset. Indeed, with our data-efficient augmentation method, CUB $\rightarrow$CUB accuracy increases by a large margin surpassing that of Tiered-ImageNet $\rightarrow$CUB setting. Our method improves 1-shot

Table 5.6 Results on increasing the number of ways on Mini-ImageNet.

		10-way		**20-way**	
Method	Backbone	1-shot	5-shot	1-shot	5-shot
baseline [121]	ResNet-18	-	55.00	-	42.03
baseline++ [121]	ResNet-18	-	63.40	-	50.85
TIM-ADM [9]	ResNet-18	56.0	72.9	39.5	58.8
TIM-GD [9]	ResNet-18	56.1	72.8	39.3	59.5
Ours-SPTA	ResNet-18	**61.15**	**77.12**	**43.29**	**64.23**

accuracy by approximately 6%, and 5-shot accuracy by more than 2%. Therefore, if our method is applied, it is possible to prepare a small base class dataset, and it can still achieve superior accuracy without resorting to the very large base class dataset. Note that our method suffers from more degradation with domain-shift. We conjecture that our method is highly dependent on the base dataset as discussed in Section 5.4.6.

5.4.4 Increasing the Number of Ways

We investigated the effect of increasing the number of ways on Mini-ImageNet and report the results in Table 5.6. The bold values represent the best results among the algorithms. All results used ResNet-18 as a backbone network model, and our results were averaged over 10,000 episodes. These settings are more challenging than 5-way few-shot classification because there is a greater chance of misclassifying the input image. Our method's 10-way and 20-way few-shot classification accuracy scores are higher than those of existing methods by a large margin. For example, it improves the 10-way 1-shot accuracy by approximately

Table 5.7 Summary results for the runtime analysis.

Methods	1-shot	5-shot
ResNet-18 TIM-ADM baseline	73.6	85.0
MobileNet + ours	75.13	85.01

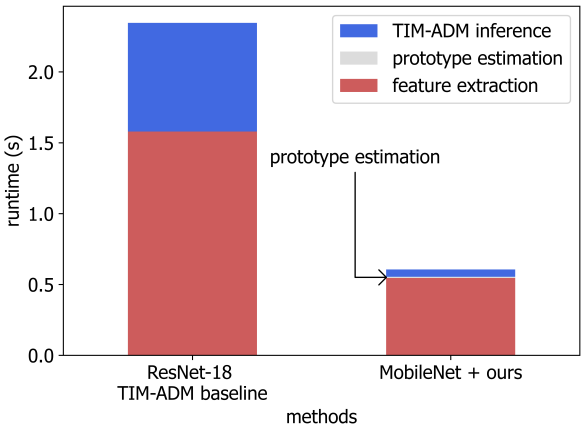

Figure 5.2 Runtime breakdown on NVIDIA Jetson TX2.

5%, 10-way 5-shot accuracy by more than 4%, 20-way 1-shot accuracy by approximately 4%, and 20-way 5-shot accuracy by more than 4% compared to the existing best method. This implies that our method improves the overall generalization performance of the few-shot learning method.

5.4.5 Runtime Analysis

The accuracy gain of our method can be utilized for runtime reduction in few-shot learning, which could be especially useful in resource-constrained contexts such as mobile settings. We measured the latency

of the methods on an NVIDIA Jetson TX2 to quantify the runtime impacts. The evaluation protocol included 100 warm-up runs, followed by 100 execution runs, and we reported the average over the execution runs. Figure 5.2 shows a breakdown of the algorithm runtime. Note that the runtime of prototype estimation is negligible. The runtime is measured in 5-shot classification (i.e., the feature extraction batch size is 100, and the algorithm assumes 5-shot classification). The baseline method is the TIM-ADM algorithm with the ResNet-18 backbone, which scores a 1-shot accuracy of 73.6 and a 5-shot accuracy of 85.0 as reported in Table 5.7. Note that supervised contrastive learning and prototype estimation are used to compensate for accuracy loss induced by the runtime reduction methods. Our accuracy results are averaged over 10,000 episodes. Note that the feature extraction latency is larger than the TIM-ADM inference runtime for target task training, which confirms the importance of backbone network selection. We chose to use the MobileNet backbone network with our method under early stopping (i.e., 10 TIM-ADM iterations instead of 150 iterations) and obtained a 1-shot accuracy of 75.13 and a 5-shot accuracy of 85.01, which is still higher than the baseline. Thus, the accuracy gain enabled by our method could be translated to a runtime reduction of $3.87\times$ without loss of accuracy.

5.4.6 Limitations

In domain-shift experiment, we observed that the feature extractor trained by our method did not improve the accuracy of the few-shot learning under the domain-shift setting (i.e., the last two rows in Table 5.5). This implies that our method is highly dependent on the base dataset as it consumes a high number of epochs (i.e., 1,000 epochs for supervised contrastive learning) with the base dataset. Therefore, we suggest that our method's application is limited to scenarios only when a domain-shift is not present. No domain-shift setting encourages a smaller base dataset in real-world implementations.

The cost of supervised contrastive learning is another limitation. A batch size larger than the number of classes in the base dataset is recommended to provide a sufficient number of positives in a single multiviewed batch. This implies many graphic processing units (GPUs) are required to implement and hinder extensive experiments. Specifically, we used two GTX 2080 Ti GPUs to six P100 GPUs to support a single run of the appropriate batch size for supervised contrastive learning. Therefore, we emphasize that a server with sufficient computing power is necessary to implement the pretraining stage. Note that once the pretraining stage and fine-tuning are completed, the remaining algorithm can be implemented in a resource-constrained environment.

In summary, both limitations indicate that our method has insufficient

scalability in terms of dataset size, and hence, it is effective in small-scale applications (e.g., few-shot learning). Reducing the cost of supervised contrastive learning is left for future work.

5.5 Conclusion

We proposed applying supervised contrastive learning for pretraining in the first stage of few-shot learning. The feature extractor was trained using supervised contrastive loss followed by fine-tuning, whereas the classifier performed adaptation using TIM loss. We report that our method is data-efficient (i.e., works well with a small dataset) while retaining competitive accuracy performance with a large dataset. Our experiment shows that we achieved new state-of-the-art results on Mini-ImageNet and CUB datasets.

Chapter 6

Conclusion

In this dissertation, two deep learning vision applications are considered, namely unsupervised domain adaptation and few-shot learning. I present 1) co-optimization of backbone network and parameter selection in unsupervised domain adaptation for edge device and 2) augmenting few-shot learning with supervised contrastive learning. Both methods aim to address low labeled data count in different settings.

The first method is to boost unsupervised domain adaptation by co-optimization of backbone network and parameter selection for edge device. Combining a large feature extractor and the unsupervised domain adaptation method that does not update the feature extractor at runtime, we can achieve new state-of-the-art accuracy result. Furthermore, we experiment using small pre-trained ImageNet models for edge device Predictor-guided evolutionary search is implemented to optimize the total latency end-to-end. We show that our method is Transferable between Office datasets without large accuracy drop. We also present pre-extraction of source feature by storing source features for several OFA

networks not specialized OFA networks. We also conduct more realistic scenario for edge device such as smaller target domain data and object detection. Lastly, We conduct an experiment to utilize intermediate domain data to reduce the algorithm latency further. We report 5.99× and 9.06× latency reduction on Office31 and Office-Home dataset, respectively.

The second method is to augment few-shot learning with supervised contrastive learning. Following the few-shot learning protocol, we use base dataset to train the feature extractor from scratch instead of using pre-trained ImageNet model. We propose to augment the feature extractor using supervised contrastive learning. After the supervised contrastive learning, fine-tuning process follows to boost the accuracy. Supervised contrastive learning with information maximization and prototype estimation methods achieves state-of-the-art accuracy result. After that, the accuracy gain can be translated to total runtime reduction by using lightweight feature extractor and early stopping. We achieve 3.87× latency reduction few-shot learning scenarios.

Our two stage approach which consists of accuracy boosting and latency reduction achieves a goal toward fast adaptation of deep learning vision applications with limited data for edge device. Note that the search space and search technique we use can be improved by using more advanced network model and algorithm, which is left for future work.

Bibliography

[1] Y. Zhang and B. D. Davison, "Impact of imagenet model selection on domain adaptation," in *2020 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, (Los Alamitos, CA, USA), pp. 173–182, IEEE Computer Society, mar 2020.

[2] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 09–15 Jun 2019.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[4] Q. Wang and T. Breckon, "Unsupervised domain adaptation via structured prediction based selective pseudo-labeling," *Proceed-*

ings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 6243–6250, Apr. 2020.

[5] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *International Conference on Learning Representations*, 2020.

[6] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 18661–18673, Curran Associates, Inc., 2020.

[7] J. Liu, L. Song, and Y. Qin, "Prototype rectification for few-shot learning," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 741–756, Springer International Publishing, 2020.

[8] I. Ziko, J. Dolz, E. Granger, and I. B. Ayed, "Laplacian regularized few-shot learning," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 11660–11670, PMLR, 13–18 Jul 2020.

[9] M. Boudiaf, I. Ziko, J. Rony, J. Dolz, P. Piantanida, and I. Ben Ayed, "Information maximization for few-shot learn-

ing," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 2445–2457, Curran Associates, Inc., 2020.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[11] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Computer Vision – ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), (Berlin, Heidelberg), pp. 213–226, Springer Berlin Heidelberg, 2010.

[12] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5018–5027, 2017.

[13] X. Peng, B. Usman, K. Saito, N. Kaushik, J. Hoffman, and K. Saenko, "Syn2real: A new benchmark forsynthetic-to-real visual domain adaptation," *CoRR*, vol. abs/1806.09755, 2018.

[14] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceed-*

ings of the IEEE International Conference on Computer Vision, pp. 1406–1415, 2019.

[15] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The Caltech-UCSD Birds-200-2011 Dataset," Tech. Rep. CNS-TR-2011-001, California Institute of Technology, 2011.

[16] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[17] M. Ren, S. Ravi, E. Triantafillou, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," in *International Conference on Learning Representations*, 2018.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[19] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[20] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[21] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8697–8710, 2018.

[22] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, p. 4278–4284, AAAI Press, 2017.

[23] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

[24] E. D. Cubuk, B. Zoph, J. Shlens, and Q. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Advances in Neural Information Processing Systems*

(H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 18613–18624, Curran Associates, Inc., 2020.

[25] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[26] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607, PMLR, 13–18 Jul 2020.

[27] J. Liang, D. Hu, and J. Feng, "Do we really need to access the source data? Source hypothesis transfer for unsupervised domain adaptation," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 6028–6039, PMLR, 13–18 Jul 2020.

[28] B. Gong, Y. Shi, F. Sha, and K. Grauman, "Geodesic flow kernel for unsupervised domain adaptation," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2066–2073, 2012.

[29] M. Jiang, W. Huang, Z. Huang, and G. G. Yen, "Integration of global and local metrics for domain adaptation learning via dimensionality reduction," *IEEE Transactions on Cybernetics*, vol. 47, no. 1, pp. 38–51, 2017.

[30] J. Wang, W. Feng, Y. Chen, H. Yu, M. Huang, and P. S. Yu, "Visual domain adaptation with manifold embedded distribution alignment," in *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, (New York, NY, USA), p. 402–410, Association for Computing Machinery, 2018.

[31] Y. Zhang, S. Xie, and B. D. Davison, "Transductive learning via improved geodesic sampling," in *Proceedings of the 30th British Machine Vision Conference (BMVC 2019)*, 2019.

[32] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, (New York, NY, USA), p. 1096–1103, Association for Computing Machinery, 2008.

[33] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," 2014.

[34] M. Long, Y. Cao, J. Wang, and M. I. Jordan, "Learning transferable features with deep adaptation networks," in *Proceedings of*

the 32nd International Conference on International Conference on Machine Learning - Volume 37, ICML'15, p. 97–105, JMLR.org, 2015.

[35] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. March, and V. Lempitsky, "Domain-adversarial training of neural networks," *Journal of Machine Learning Research*, vol. 17, no. 59, pp. 1–35, 2016.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, p. 84–90, May 2017.

[37] M. Long, H. Zhu, J. Wang, and M. I. Jordan, "Deep transfer learning with joint adaptation networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 2208–2217, JMLR.org, 2017.

[38] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2011.

[39] F. Dorri and A. Ghodsi, "Adapting component analysis," in *2012 IEEE 12th International Conference on Data Mining*, pp. 846–851, 2012.

[40] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu, "Transfer joint matching for unsupervised domain adaptation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1410–1417, 2014.

[41] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, and B. Schölkopf, "Domain adaptation with conditional transferable components," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 2839–2848, JMLR.org, 2016.

[42] J. Wang, Y. Chen, L. Hu, X. Peng, and P. S. Yu, "Stratified transfer learning for cross-domain activity recognition," in *2018 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10, 2018.

[43] J. Wang, Y. Chen, S. Hao, W. Feng, and Z. Shen, "Balanced distribution adaptation for transfer learning," in *2017 IEEE International Conference on Data Mining (ICDM)*, pp. 1129–1134, 2017.

[44] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, (Sydney, Australia), pp. 120–128, Association for Computational Linguistics, July 2006.

[45] M. Ghifary, W. B. Kleijn, M. Zhang, and D. Balduzzi, "Domain generalization for object recognition with multi-task autoencoders," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[46] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[47] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems* (Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, eds.), vol. 27, Curran Associates, Inc., 2014.

[48] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan, "Domain separation networks," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[49] X. Chen, S. Wang, M. Long, and J. Wang, "Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation," in *Proceedings of the 36th International Conference*

on Machine Learning (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 1081–1090, PMLR, 09–15 Jun 2019.

[50] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[51] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[52] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size," 2016.

[53] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[54] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6848–6856, 2018.

[55] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[56] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollar, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[57] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations*, 2017.

[58] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4780–4789, Jul. 2019.

[59] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[60] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019.

[61] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 678–687, PMLR, 10–15 Jul 2018.

[62] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *International Conference on Learning Representations*, 2019.

[63] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "Mnasnet: Platform-aware neural architecture search for mobile," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2815–2823, 2019.

[64] B. Wu, X. Dai, P. Zhang, Y. Wang, F. Sun, Y. Wu, Y. Tian, P. Vajda, Y. Jia, and K. Keutzer, "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[65] X. He and P. Niyogi, "Locality preserving projections," in *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, (Cambridge, MA, USA), p. 153–160, MIT Press, 2003.

[66] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[67] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[68] 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 Édouard Duchesnay, "Scikit-learn: Machine learning in

python," *Journal of Machine Learning Research*, vol. 12, no. 85, pp. 2825–2830, 2011.

[69] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[70] M. Xu, J. Zhang, B. Ni, T. Li, C. Wang, Q. Tian, and W. Zhang, "Adversarial domain adaptation with domain mixup," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 6502–6509, Apr. 2020.

[71] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 1180–1189, PMLR, 07–09 Jul 2015.

[72] C. Chen, Z. Fu, Z. Chen, S. Jin, Z. Cheng, X. Jin, and X.-s. Hua, "Homm: Higher-order moment matching for unsupervised

domain adaptation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 3422–3429, Apr. 2020.

[73] S. Xie, Z. Zheng, L. Chen, and C. Chen, "Learning semantic representations for unsupervised domain adaptation," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 5423–5432, PMLR, 10–15 Jul 2018.

[74] R. Xu, G. Li, J. Yang, and L. Lin, "Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[75] Z. Deng, Y. Luo, and J. Zhu, "Cluster alignment with a teacher for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[76] M. Long, Z. CAO, J. Wang, and M. I. Jordan, "Conditional adversarial domain adaptation," in *Advances in Neural Information Processing Systems* (S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, eds.), vol. 31, Curran Associates, Inc., 2018.

[77] K. You, X. Wang, M. Long, and M. Jordan, "Towards accurate model selection in deep unsupervised domain adaptation," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 7124–7133, PMLR, 09–15 Jun 2019.

[78] Y. Wu, D. Inkpen, and A. El-Roby, "Dual mixup regularized learning for adversarial domain adaptation," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 540–555, Springer International Publishing, 2020.

[79] Y. Zhang, H. Tang, K. Jia, and M. Tan, "Domain-symmetric networks for adversarial domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[80] M. Chen, S. Zhao, H. Liu, and D. Cai, "Adversarial-learned loss for domain adaptation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 3521–3528, Apr. 2020.

[81] Y. Zhang, T. Liu, M. Long, and M. Jordan, "Bridging theory and algorithm for domain adaptation," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and

R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 7404–7413, PMLR, 09–15 Jun 2019.

[82] H. Tang and K. Jia, "Discriminative adversarial domain adaptation," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pp. 5940–5947, Apr. 2020.

[83] Y. Jin, X. Wang, M. Long, and J. Wang, "Minimum class confusion for versatile domain adaptation," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 464–480, Springer International Publishing, 2020.

[84] L. Hu, M. Kan, S. Shan, and X. Chen, "Unsupervised domain adaptation with hierarchical gradient synchronization," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4042–4051, 2020.

[85] Y. Zhang and B. D. Davison, "Modified distribution alignment for domain adaptation with pre-trained inception resnet," 2019.

[86] G. Kang, L. Jiang, Y. Yang, and A. G. Hauptmann, "Contrastive adaptation network for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[87] H. Tang, K. Chen, and K. Jia, "Unsupervised domain adaptation via structurally regularized deep clustering," in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[88] X. Gu, J. Sun, and Z. Xu, "Spherical space domain adaptation with robust pseudo-label loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[89] J. Na, H. Jung, H. J. Chang, and W. Hwang, "Fixbi: Bridging domain spaces for unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1094–1103, June 2021.

[90] Y. Zhang and B. D. Davison, "Efficient pre-trained features and recurrent pseudo-labeling in unsupervised domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 2719–2728, June 2021.

[91] S. Cui, S. Wang, J. Zhuo, C. Su, Q. Huang, and Q. Tian, "Gradually vanishing bridge for adversarial domain adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[92] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015.

[93] T. Lee and S. Yoo, "Augmenting few-shot learning with supervised contrastive learning," *IEEE Access*, vol. 9, pp. 61466–61474, 2021.

[94] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 843–852, 2017.

[95] L. Fe-Fei, Fergus, and Perona, "A bayesian approach to unsupervised one-shot learning of object categories," in *Proceedings Ninth IEEE International Conference on Computer Vision*, pp. 1134–1141 vol.2, 2003.

[96] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[97] B. M. Lake, R. Salakhutdinov, J. Gross, and J. B. Tenenbaum, "One shot learning of simple visual concepts," in *Conference of the Cognitive Science Society (CogSci)*, 2011.

[98] X. Li, Q. Sun, Y. Liu, Q. Zhou, S. Zheng, T.-S. Chua, and B. Schiele, "Learning to self-train for semi-supervised few-shot classification," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[99] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. Hwang, and Y. Yang, "LEARNING TO PROPAGATE LABELS: TRANSDUCTIVE PROPAGATION NETWORK FOR FEW-SHOT LEARNING," in *International Conference on Learning Representations*, 2019.

[100] L. Qiao, Y. Shi, J. Li, Y. Tian, T. Huang, and Y. Wang, "Transductive episodic-wise adaptive metric for few-shot learning," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 3602–3611, 2019.

[101] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto, "A baseline for few-shot image classification," in *International Conference on Learning Representations*, 2020.

[102] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations*, 2018.

[103] H. Touvron, A. Vedaldi, M. Douze, and H. Jegou, "Fixing the train-test resolution discrepancy," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[104] S. Gidaris, A. Bursuc, N. Komodakis, P. Perez, and M. Cord, "Boosting few-shot visual learning with self-supervision," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[105] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, PMLR, 06–11 Aug 2017.

[106] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," 2018.

[107] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *International Conference on Learning Representations*, 2017.

[108] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML Deep Learning Workshop*, 2015.

[109] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[110] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7278–7286, 2018.

[111] L. Yang, L. Li, Z. Zhang, X. Zhou, E. Zhou, and Y. Liu, "Dpgn: Distribution propagation graph network for few-shot learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13387–13396, 2020.

[112] S. X. Hu, P. G. Moreno, Y. Xiao, X. Shen, G. Obozinski, N. Lawrence, and A. Damianou, "Empirical bayes transductive meta-learning with synthetic gradients," in *International Conference on Learning Representations*, 2020.

[113] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *2018*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3733–3742, 2018.

[114] O. J. Hénaff, A. Srinivas, J. D. Fauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. van den Oord, "Data-efficient image recognition with contrastive predictive coding," 2020.

[115] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *International Conference on Learning Representations*, 2019.

[116] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," 2020.

[117] P. Sermanet, C. Lynch, J. Hsu, and S. Levine, "Time-contrastive networks: Self-supervised learning from multi-view observation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 486–487, 2017.

[118] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia

Laguna Resort, Sardinia, Italy), pp. 297–304, PMLR, 13–15 May 2010.

[119] A. Mnih and K. Kavukcuoglu, "Learning word embeddings efficiently with noise-contrastive estimation," in *Advances in Neural Information Processing Systems* (C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013.

[120] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[121] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang, "A closer look at few-shot classification," in *International Conference on Learning Representations*, 2019.

[122] Y. Wang, W.-L. Chao, K. Q. Weinberger, and L. van der Maaten, "Simpleshot: Revisiting nearest-neighbor classification for few-shot learning," 2019.

[123] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[124] J. Zhang, C. Zhao, B. Ni, M. Xu, and X. Yang, "Variational few-shot learning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[125] B. Liu, Y. Cao, Y. Lin, Q. Li, Z. Zhang, M. Long, and H. Hu, "Negative margin matters: Understanding margin in few-shot classification," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 438–455, Springer International Publishing, 2020.

[126] K. Li, Y. Zhang, K. Li, and Y. Fu, "Adversarial feature hallucination networks for few-shot learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[127] K. Lee, S. Maji, A. Ravichandran, and S. Soatto, "Meta-learning with differentiable convex optimization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[128] Y. Tian, Y. Wang, D. Krishnan, J. B. Tenenbaum, and P. Isola, "Rethinking few-shot image classification: A good embedding is all you need?," in *Computer Vision – ECCV 2020* (A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, eds.), (Cham), pp. 266–282, Springer International Publishing, 2020.

[129] W. Xu, yifan xu, H. Wang, and Z. Tu, "Attentional constellation nets for few-shot learning," in *International Conference on Learning Representations*, 2021.

[130] C. Zhang, Y. Cai, G. Lin, and C. Shen, "Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[131] H.-J. Ye, H. Hu, D.-C. Zhan, and F. Sha, "Few-shot learning via embedding adaptation with set-to-set functions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[132] M. Zhang, J. Zhang, Z. Lu, T. Xiang, M. Ding, and S. Huang, "Iept: Instance-level and episode-level pretext tasks for few-shot learning," in *International Conference on Learning Representations*, 2021.

[133] A. Li, W. Huang, X. Lan, J. Feng, Z. Li, and L. Wang, "Boosting few-shot learning with adaptive margin loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[134] R. Hou, H. Chang, B. MA, S. Shan, and X. Chen, "Cross attention network for few-shot classification," in *Advances in Neu-*

ral Information Processing Systems (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[135] N. Fei, Z. Lu, T. Xiang, and S. Huang, "Melr: Meta-learning via modeling episode-level relationships for few-shot learning," in *International Conference on Learning Representations*, 2021.

[136] Z. Yue, H. Zhang, Q. Sun, and X.-S. Hua, "Interventional few-shot learning," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 2734–2746, Curran Associates, Inc., 2020.

[137] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, and R. Hadsell, "Meta-learning with latent embedding optimization," in *International Conference on Learning Representations*, 2019.

[138] Y. Guo and N.-M. Cheung, "Attentive weights generation for few shot learning via information maximization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[139] S. Yang, L. Liu, and M. Xu, "Free lunch for few-shot learning: Distribution calibration," in *International Conference on Learning Representations*, 2021.

국문초록

딥 러닝 기반 방법의 놀라운 성공은 주로 많은 양의 분류된 데이터로 달성되었다. 전통적인 기계 학습 방법과 비교해서 딥러닝 방법은 아주 큰 데이터셋으로부터 좋은 성능을 가진 모델을 학습할 수 있다. 하지만 고품질의 분류된 데이터는 만들기 어렵고 프라이버시 문제로 만들 수 없을 때도 있다. 게다가 사람은 아주 큰 분류된 데이터가 없어도 훌륭한 일반화 능력을 보여준다.

엣지 장비는 서버와 비교해서 제한적인 계산 능력을 가진다. 특히 학습 과정을 엣지 장비에서 수행하는 것은 매우 어렵다. 하지만, 도메인 변화 문제와 프라이버시 문제를 고려했을 때 엣지 장비에서 학습 과정을 수행하는 것은 바람직하다. 본 논문에서는 계산능력이 작은 엣지 장비를 위해 적응 과정을 전통적인 학습 과정 대신 고려한다.

전통적인 분류 문제는 학습 데이터와 테스트 데이터가 동일한 분포에서 파생되었음과 많은 양의 학습 데이터를 가정한다. 비지도 도메인 어댑테이션은 테스트 데이터가 학습데이터와 다른 분포에서 파생되는 상황을 가정하며 기존의 분류된 데이터와 학습된 모델을 이용해 새로운 데이터를 분류하는 문제이다. 퓨샷 학습은 적은 양의 학습 데이터를 가정하며 소수의 분류된 데이터만을 가지고 새로운 데이터를 분류하는 문제이다. 엣지 장비를 위해 이미지넷에서 미리 학습된 모델을 통해 비지도 도메인 어댑테이션 성능을 강화하는 방법과 지도 컨트라스티브

학습을 통해 퓨샷 학습 성능을 강화하는 방법을 제안하였다. 두 방법은 모두 적은 분류된 데이터 문제를 다루며 다만 서로 다른 시나리오를 가정한다.

첫 번째 방법은 엣지 장비를 위해 네트워크 모델과 파라미터 선택의 동시 최적화를 통해 비지도 도메인 어댑테이션 성능을 강화하는 방법이다. 이미지넷에서 미리 학습된 모델은 Office 데이터셋과 같이 작은 데이터셋을 다룰때 매우 중요하다. 특징 추출기를 갱신하지 않는 비지도 도메인 어댑테이션 알고리즘을 사용하고 아주 큰 이미지넷에서 미리 학습된 모델을 조합하는 방법으로 높은 정확도를 얻을 수 있다. 더 나아가 엣지 장비를 위해 작고 가벼운 이미지넷에서 미리 학습된 모델을 실험하였다. 지연시간을 줄이기 위해 예측기를 도입한 진화 알고리즘으로 방법의 시작부터 끝까지 최적화하였다. 그리고 프라이버시를 지키기 위한 비지도 도메인 어댑테이션 시나리오에 대해 고려하였다. 또한 엣지 장비에서 좀 더 현실적인 시나리오인 작은 데이터셋과 object detection 에 대해서도 실험하였다. 마지막으로 연속적인 데이터가 입력될 때 중간 데이터를 활용하여 지연시간을 더 감소시키는 방법을 실험하였다. Office31과 Office-Home 데이터셋에 대해 각각 5.99배와 9.06배 지연시간 감소를 달성하였다.

두 번째 방법은 지도 컨트라스티브 학습을 통해 퓨샷 학습 성능을 강화하는 방법이다. 퓨샷 학습 벤치마크에서는 베이스 데이터셋으로 특징 추출기를 학습하기 때문에 이미지넷에서 미리 학습된 모델을 사용할 수 없다. 대신에, 지도 컨트라스티브 학습을 통해 특징 추출기를 강화한다. 지도 컨트라스티브 학습과 정보 최대화 그리고 프로토타입

추정 방법을 조합하여 아주 높은 정확도를 얻을 수 있다. 특징 추출기와 미리 끝내기를 통해 이렇게 얻은 정확도를 수행시간 감소로 바꿀 수 있다. 트랜스덕티브 5-웨이 5-샷 학습 시나리오에서 3.87배 지연시간 감소를 달성하였다.

본 방법은 정확도를 증가시킨 후 지연시간을 감소시키는 방법으로 요약할 수 있다. 먼저 이미지넷에서 미리 학습된 모델을 쓰거나 지도 컨트라스티브 학습을 통해 특징 추출기를 강화해서 높은 정확도를 얻는다. 그 후 진화 알고리즘을 통해 시작부터 끝까지 최적화하거나 미리 끝내기를 통해 지연시간을 줄인다. 정확도를 증가시킨 후 지연시간을 감소시키는 두 단계 접근 방식은 엣지 장비를 위한 한정된 데이터를 가지는 딥러닝 비전 어플리케이션의 빠른 적응을 달성하는데 충분하다.

주요어: 뉴럴네트워크, 엣지장비, 비지도 도메인 어댑테이션, 퓨샷 학습, 수도 레이블링, 컨트라스티브 학습, 정보 최대화
학번: 2015-31052