



Universiteit
Leiden
The Netherlands

Exploring deep learning for data-efficient and generalizable visual recognition

He, K.

Citation

He, K. (2026, September 22). *Exploring deep learning for data-efficient and generalizable visual recognition*. Retrieved from <https://hdl.handle.net/1887/4310276>

Version: Publisher's Version

License: [Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden](#)

Downloaded from: <https://hdl.handle.net/1887/4310276>

Note: To cite this publication please use the final published version (if applicable).

Chapter 6

Dual Selective Knowledge Transfer for Few-Shot Classification

In the previous Chapters 3, 4 and 5, we have explored various aspects of feature representation and metric optimization for few-shot learning (FSL) tasks. Starting with this chapter, we research on a more fundamental challenge in knowledge transfer for FSL - the effective utilization of pre-trained knowledge while maintaining adaptability to novel tasks. Therefore, in this chapter, we focus on understanding how transferable knowledge can be selectively leveraged to build more generalizable feature representations for few-shot tasks (**RQ 4**).

Few-shot learning aims at recognizing novel visual categories from very few labelled examples. Different from the existing few-shot classification methods that are mainly based on metric learning or meta-learning, in this work we focus on improving the representation capacity of feature extractors. For this purpose, we propose a new two-stage dual selective knowledge transfer (DSKT) framework, to guide models towards better optimization. Specifically, we first exploit an improved multi-task learning approach to train a feature extractor with robust representation capability as a teacher model. Then, we design an effective dual selective knowledge distillation method, which enables the student model to selectively learn knowledge from the teacher model and current samples, thereby improving the student model’s ability to generalize on unseen classes. Extensive experimental results show that our DSKT achieves competitive performances on four well-known few-shot classification benchmarks.

This chapter is based on the following publication [36]:

- **He, K.**, Pu, N., Lao, M., Bakker, E. M., and Lew, M.S. “Dual selective knowledge transfer for few-shot classification.” Applied Intelligence, 2023.

6.1 Introduction

Deep learning models have achieved breakthroughs in many visual understanding tasks, such as image classification [304] and object detection [305]. However, their performances degrade significantly when few labelled examples are available and these large-capacity models usually have difficulties in transferring the learnt knowledge to unseen classes. As such, there has been increasing interest in few-shot learning (FSL) [1, 24, 101], which aims to develop well-generalized models to recognize new categories using only limited annotated samples.

Recently, meta-learning [2, 14, 115], *a.k.a.* learning-to-learn, has made significant advances in FSL. The primary idea of meta-learning is to exploit episodic training to transfer the knowledge learnt from a massive known meta-training set to novel classes. For example, Sun et al. [198] proposed Meta-Transfer Learning, which introduced the scaling and translation parameters to adjust the weight parameters, meeting the needs of new tasks. However, the success of transfer ability depends on the similarity between tasks. It’s difficult to transfer knowledge effectively when new tasks are significantly different from old ones. Another practical solution to FSL are metric-based methods [23, 116, 306], which aim to learn good representations through deep networks and generate final predictions based on the similarity or distance between support and query samples, such as RelationNet [24] and ProtoNet [1]. Though metric learning has become the prominent methods for FSL, it’s still challenging to select an appropriate distance metric for various tasks and datasets, as different metrics can yield different results.

In this chapter, we propose a new dual selective knowledge transfer (DSKT) framework for FSL tasks, following the transfer learning strategy. The framework and overall training process are illustrated in Figure 6.1. In the first stage of the DSKT framework (DSKT-1), we introduce a multi-task loss to mine useful signals or information from the limited labelled examples. This enables the model to reduce the risk of overfitting and to ensure heterogeneity in the prediction space, simultaneously.

Furthermore, to implement selective knowledge transfer, we propose an effective dual selective knowledge distillation method in the second stage (DSKT-2). Specifically, we first employ the learnt model in DSKT-1 as a teacher network and train a student model based on the teacher’s outputs. Then, different from the classical knowledge distillation (KD) [307] loss that considers different classes equally, we reformulate the KD loss into two independent components according to the class labels of samples, inspired by [308]. By separately fine-tuning the KD strength of each component, models are endowed with the ability to selectively transfer knowledge. Meanwhile, we introduce PolyLoss [309] to adjust the strengths of learning on current samples, which complements the knowledge transferred from the teacher model. After training, we freeze the student model up to the penultimate layer and

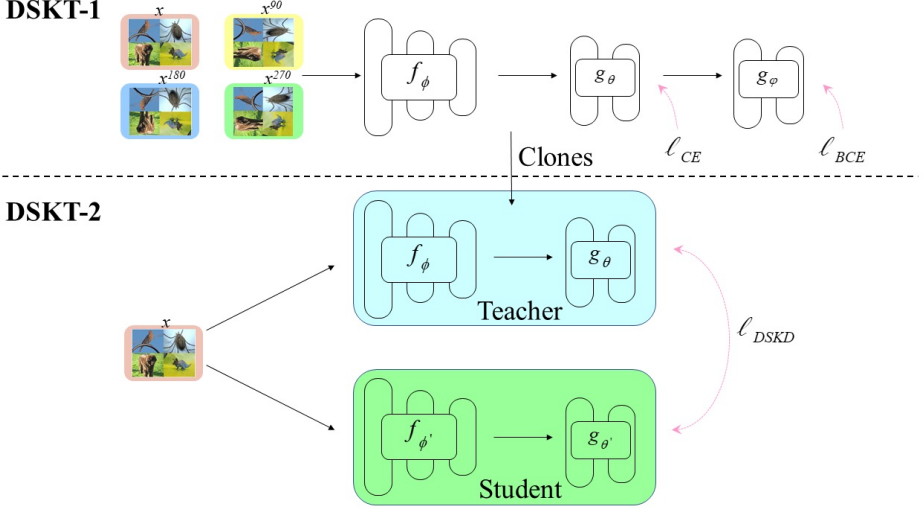


Figure 6.1: Overall training process of DSKT, where f_ϕ is a feature extractor, g_θ and g_φ are two different classifiers. DSKT-1 denotes the first process which includes a binary cross entropy (BCE) loss and a cross entropy (CE) loss, with the goal to learn a good feature extractor. DSKT-2 represents the stage of dual selective knowledge distillation (DSKD), which selectively transfers knowledge to ensure the student model’s generalization ability.

apply it as a feature extractor. During the evaluation, we fit a logistic regression classifier based on the frozen feature extractor without fine-tune, implemented as in [110].

Compared to previous work [269], we take similar action in the first stage to train a discriminative feature extractor. However, we design a distinct DSKD framework for knowledge distillation. The difference and superiority can be simply concluded as follows: 1) We only use the original samples during the knowledge distillation process, which reduces computational costs to some extent. 2) We propose a new DSKD framework for knowledge distillation, which integrates decoupled knowledge distillation (DKD) and PolyLoss, enabling the student model could effectively learn knowledge from both the teacher model and current samples, enhancing the student model’s capacity to generalize on new tasks.

The main contributions of this work are as follows: 1) We incorporate KD and PolyLoss as a knowledge transfer framework for FSL tasks. 2) We exploit DKD to improve the conventional KD, which enables the student model to selectively learn knowledge from the teacher. 3) the experiments on four popular benchmark datasets show that our approach achieves new state-of-the-art performances in all the 5-way 1-shot and 5-way 5-shot tasks.

6.2 Related Work

Few-shot learning (FSL): As a surprising research area in deep learning, FSL focus on learning patterns from a set of data (base dataset) and then adapting to a disjoint set (novel dataset) with few labelled samples. In order to evaluate the model’s performance, the novel dataset is split into a support set and a query set. FSL is also named N -way K -shot classification when the support set contains N classes and K samples per class.

Two main training methods have been explored in few-shot learning, namely meta-learning and metric learning. On the one hand, meta-learning consists of two phases, namely meta-training and meta-testing, where each phase involves a family of tasks. Meta-learning aims to perform well on new tasks based on the knowledge learnt from previous tasks. For example, Finn et al. [2] proposed a model-agnostic meta-learning (MAML) approach, which can adapt to new tasks quickly with a better initialization weight. With the goal to simplify MAML, Reptile [310] removes re-initialization for each task. MetaOptNet [237] employs SVM to replace its original linear classifier and solves the convex optimization problem by using quadratic programming (QP). Sun et al. [198] proposed a meta-transfer learning (MTL) method, which learns to adapt a deep NN for few-shot learning tasks. Bansal et al. [311] proposed a meta-learning model, which combines supervised learning and self-supervised learning for natural language classification tasks.

On the other hand, metric learning pays attention to finding an existing or learned metric space, where the support set can be simply matched with the query set. In metric learning, additional parameters are not required to learn once the metric is acquired. Snell et al. [1] proposed ProtoNet, which learns a metric space where classification can be performed by computing distances to prototype representations of each class. RelationNet [24] applies a network to learn the relation (similarity) between support and query images. Based on memory module, MatchingNet [14] uses cosine distance as a metric to classify query samples. To achieve better classification, CAN [22] is designed to highlight the proper region of interest by learning an attention module. Zhang et al. [37] proposed a DeepEMD algorithm, which adopts the Earth Mover’s Distance (EMD) as a distance metric to calculate the similarity. Different from these approaches, we propose a new two-stage knowledge transfer framework, further improving the model’s generalization ability for FSL tasks.

Knowledge distillation (KD): With the goal to deploy demanding deep models on devices with limited resources, KD is developed for model compression and acceleration. KD works by effectively transferring knowledge from a large and complex model, called teacher model, to a smaller and simple one, called student model. During the process of KD, the student’s learning performance is influenced by multiple factors, such as knowledge types, distillation strategies and teacher-student architectures.

Hinton et al. [307] first generalized and brought this idea into deep learning. By minimizing the loss between the teacher model’s and student model’s outputs, informative knowledge can be transferred. Tasks [312, 313] show KD’s benefits for knowledge transfer and optimization. Besides, sequential distillation [314] is introduced to improve the performance of teacher models. Mobahi et al. [315] presented a theoretical analysis of self-distillation. Lim et al. [316] proposed an Efficient-PrototypicalNet, which involves both transfer learning and knowledge distillation for few-shot learning tasks. Liu et al. [317] proposed learning a model through online self-distillation, which combines supervised training with knowledge distillation via a continuously updated teacher. A novel Supervised Masked Knowledge Distillation model (SMKD) [318] is also designed for few-shot Transformers, which incorporates label information into self-distillation frameworks. Unlike commonly-used KD which only transfers knowledge from the teacher model, we use a dual selective knowledge distillation method to encourage the student model to selectively learn knowledge from the teacher model and examples simultaneously.

6.3 Dual Selective Knowledge Transfer

To address the challenges caused by FSL, we propose a new dual selective knowledge transfer (DSKT) framework, which consists of two stages: multi-task learning and dual selective knowledge distillation (DSKD). In the first stage, we introduce multi-task learning and enforce the learnt representations equivariant to image transformations, which is beneficial for extracting low-level features. Furthermore, we propose an effective DSKD, which enables the student model to selectively learn knowledge from both the teacher model and the current samples.

6.3.1 Problem Formulation

In this work, we use a large-scale labelled base dataset for training a feature extractor. The base dataset is defined as $D^{base} = \{x_t^{base}, y_t^{base}\}_{t=1}^{N^{base}}$, with label $y_t^{base} \in C^{base}$. In order to learn a good feature extractor, we hold the assumption that both the amount of classes ($|C^{base}|$) and the number of examples (N^{base}) are large. For the novel dataset used for evaluation, is denoted as $D^{novel} = \{x_t^{novel}, y_t^{novel}\}_{t=1}^{N^{novel}}$, with label $y_t^{novel} \in C^{novel}$. Notice that base classes and novel classes are disjoint, which means $C^{base} \cap C^{novel} = \emptyset$. The testing of the learnt model is organized in episodes, in which each episode i contains a support set $D_i^{support} = \{x_{i,t}^{support}, y_{i,t}^{support}\}_{t=1}^{CK}$ and a query set $D_i^{query} = \{x_{i,t}^{query}, y_{i,t}^{query}\}_{t=1}^{CK'}$, which contains C classes, with K and K' examples per class from the novel dataset D^{novel} , respectively.

6.3.2 Multi-task Learning

In the first stage of our DSKT framework, we introduce a multi-task learning method, which consists of a category task and a rotation task to train a discriminative feature extractor with good representations on the base dataset. Following a similar training strategy [110], we adopt a neural network that contains a feature extractor f_ϕ , a category classifier g_θ and an additional rotation classifier g_φ .

First, we randomly sample a minibatch $B = \{x_i, y_i\}_{i=1}^m$ from the base dataset D^{base} , where m stands for the batch size. We impose the rotate transformation with 90, 180 and 270 degrees on the images x , to generate augmented copies x^{90} , x^{180} and x^{270} , respectively. Then, we stack the original images and their transformed versions together, resulting in a single tensor $\hat{x} = \{x, x^{90}, x^{180}, x^{270}\} \in \mathbb{R}^{(h \times w) \times (4 \times m)}$ with corresponding labels $\hat{y} \in \mathbb{R}^{4 \times m}$. According to the rotation direction, we also create one-hot encoded labels $\hat{r} = \{r_i \in \mathbb{R}^s\}_{4 \times m}$, where $s = 4$ indicates that four rotation directions are applied in our method.

Next, by using the feature extractor f_ϕ , the stacked tensor $\hat{x}$ is mapped to feature vectors $\hat{v} = f_\phi(\hat{x}) \in \mathbb{R}^{d \times (4 \times m)}$, where d denotes the feature map size. Then, we pass the feature vectors $\hat{v}$ through category classifier g_θ to obtain its corresponding logits $\hat{p} = g_\theta(\hat{v}) \in \mathbb{R}^{c \times (4 \times m)}$. Finally, we apply rotation classifier g_φ to map the logits $\hat{p}$ to the rotation logits $\hat{q} = g_\varphi(\hat{p}) \in \mathbb{R}^{s \times (4 \times m)}$.

Afterwards, to train all network modules (i.e., f_ϕ , g_θ and g_φ), we use two loss functions to jointly optimize three modules. One is a commonly-used cross entropy loss [110] for classifying categories of samples, which is denoted by ℓ_{CE} . The other is a binary cross entropy loss for rotation prediction, which serves as the rotation loss ℓ_{BCE} . The optimization problem is formulated as:

$$\begin{aligned} \phi, \theta, \varphi = \arg \min_{\phi, \theta, \varphi} \mathbb{E}_{(x, y) \sim D^{\text{base}}} & \left[\ell_{CE} \left(g_\theta \left(f_\phi \left(\hat{x} \right) \right), \hat{y} \right) \right. \\ & \left. + \alpha \ell_{BCE} \left(g_\varphi \left(g_\theta \left(f_\phi \left(\hat{x} \right) \right) \right), \hat{r} \right) \right]. \end{aligned} \quad (6.1)$$

where α is a weighting coefficient to control the strength of rotation loss.

Learning a good feature extractor is challenging in FSL as limited labelled samples are available during this process. In this work, we introduce multi-task learning and it enables the feature extractor to effectively learn the low-level features [319]. In addition, the risk of overfitting can be reduced through multi-task loss [319]. Experimental results in Table 6.4 demonstrate the superiority of our method.

6.3.3 Dual Selective Knowledge Distillation

To improve the model’s generalization ability by knowledge transfer, we develop an effective dual selective knowledge distillation (DSKD) to promote models in learn-

ing informative knowledge. Once the first stage is finished, we take one clone of the trained model as a teacher model, where weights are frozen and only used for inference. The knowledge distillation (KD) [307] loss is reformulated into two components, by separately adjusting the strength of each component, informative knowledge can be selectively transferred to a new student model, which only contains a feature extractor and a classifier. In addition, PolyLoss [309] is introduced to ensure the student model selectively learns knowledge from current training samples. By considering such dual learning strategies, our DSKD enables the student model to simultaneously learn knowledge from the teacher model and current samples.

To make the different components of the transferred knowledge controllable, we first reformulate the KD loss into target class knowledge distillation (TCKD) loss and non-target class knowledge distillation (NCKD) loss, inspired by [308]. For the i -th input image, $p_i \in \mathbb{R}^c$ denotes its output logits, and $p_{i,t}$ represents the logit of the t -th class. Hence, the probability that the i -th sample belongs to the t -th class $z_{i,t}$ and the probability that it belongs to any of the other classes $z_{i,\setminus t}$ can be formulated as:

$$z_{i,t} = \frac{e^{p_{i,t}}}{\sum_j e^{p_{i,j}}}, \quad z_{i,\setminus t} = \frac{\sum_{k=1, k \neq t}^c e^{p_{i,k}}}{\sum_j e^{p_{i,j}}} = 1 - z_{i,t}. \quad (6.2)$$

Meanwhile, the vector $\hat{z}_i$ denotes the probability among non-target classes (i.e., the t -th class is not considered),

$$\hat{z}_i = \left[\hat{z}_{i,1}, \dots, \hat{z}_{i,t-1}, \hat{z}_{i,t+1}, \dots, \hat{z}_{i,c} \right], \quad (6.3)$$

where $\hat{z}_{i,k} = \frac{e^{p_{i,k}}}{\sum_{j=1, j \neq t}^c e^{p_{i,j}}}.$

Thereby, TCKD loss and NCKD loss can be defined as:

$$\ell_{TCKD} = \text{KL}(b_i^T || b_i^S), \quad \ell_{NCKD} = \text{KL}\left(\hat{z}_i^T || \hat{z}_i^S\right), \quad (6.4)$$

where $b_i = [z_{i,t}, z_{i,\setminus t}]^T \in \mathbb{R}^{2 \times 1},$

where KL denotes Kullback-Leibler divergence, T and S represent the teacher and student model, respectively. By adjusting the strength of ℓ_{TCKD} and ℓ_{NCKD} , the student model can selectively learn the knowledge from well-predicted samples.

Apart from learning from the teacher model, we further explore adjustable learning from current samples by introducing PolyLoss [309]. PolyLoss provides a unified view on common loss functions for classification problems. It defines loss function as a linear combination of polynomial functions. Under polynomial expansion, focal loss is a horizontal shift of the polynomial coefficients compared to cross entropy loss. In this chapter, PolyLoss is used to calculate the loss between predictions and ground-truth labels so as to guide the student model's training toward good

optimization. The function of PolyLoss is defined as follows:

$$\ell_{Poly-N} = -\log(P_t) + \sum_{k=1}^N \beta_k (1 - P_t)^k, \quad (6.5)$$

where P_t denotes the model’s prediction probability of the target ground-truth class, N represents the number of leading coefficients. With this flexible form, PolyLoss can easily adjust the importance of different polynomial bases according to the targeting datasets and tasks, consistently improving the performance. In this work, we set $N = 1$ and the experimental results show it achieves significant improvement.

Finally, the optimization problem of DSKD can be written as,

$$\begin{aligned} \phi', \theta' = \arg \min_{\phi', \theta'} & \mathbb{E}_{(x,y) \sim D^{base}} \\ & [\eta_1 \ell_{Poly-1}(g_{\theta'}(f_{\phi'}(x)), y) \\ & + \eta_2 [\lambda_1 \ell_{TCKD}(g_{\theta'}(f_{\phi'}(x)), g_{\theta}(f_{\phi}(x)), y) \\ & + \lambda_2 \ell_{NCKD}(g_{\theta'}(f_{\phi'}(x)), g_{\theta}(f_{\phi}(x)), y)] \end{aligned} \quad (6.6)$$

where $\eta_2 = 1 - \eta_1$, λ_1 and λ_2 are the weights of ℓ_{TCKD} and ℓ_{NCKD} , respectively. Our proposed DSKD method enables the student model to selectively learn knowledge from the teacher model and current samples.

6.4 Experiments

In this section, we elaborate on the experimental configuration and evaluation. The description of the datasets and implementation details are first presented, followed by experimental results of our approach on popular benchmark datasets. Finally, an ablative analysis is presented.

6.4.1 Datasets and Implementation details

Datasets: We conduct extensive experiments on four popular FSL benchmark datasets, miniImageNet [14], tieredImageNet [62], Fewshot-CIFAR100 (FC100) [23] and Caltech-UCSD Birds-200-2011 (CUB) [113]. For miniImageNet, as a subset of ImageNet, it includes 100 classes and each class contains 600 images. We follow the splitting protocol proposed in [115], with 64 classes for training, 16 classes for validation and 20 classes for testing. The tieredImageNet contains 608 classes, which can be grouped into 34 high-level categories. We use 20 categories (351 classes), 6 categories (97 classes), and 8 categories (160 classes) for training, validation and testing, respectively. FC100 is derived from CIFAR100 dataset and employs a split similar to tieredImageNet, which can be divided into 60 training classes, 20 validation classes and 20 testing classes. Each class includes 600 images. The CUB was originally used for fine-grained bird classification, which has 11,788 images from 200

Table 6.1: Comparison of DSKT (our approach) to prior works on miniImageNet and tieredImageNet datasets, with mean accuracy (%) and 95% confidence interval. ‡ indicates a deeper backbone. DSKT-1 and DSKT-2 represent the first and second stage, respectively.

Model	Backbone	miniImageNet 5-way		tieredImageNet 5-way	
		1-shot	5-shot	1-shot	5-shot
ProtoNet [1] (NIPS’17)	ResNet-12	60.37 $\pm$ 0.83	78.02 $\pm$ 0.57	65.65 $\pm$ 0.92	83.40 $\pm$ 0.65
TADAM [23] (NIPS’18)	ResNet-12	58.50 $\pm$ 0.30	76.70 $\pm$ 0.30	-	-
MTL [198] (CVPR’19)	ResNet-12	61.20 $\pm$ 1.80	75.50 $\pm$ 0.80	65.62 $\pm$ 1.80	80.61 $\pm$ 0.90
MetaOptNet [237] (CVPR’19)	ResNet-12	62.64 $\pm$ 0.61	78.63 $\pm$ 0.46	65.99 $\pm$ 0.72	81.56 $\pm$ 0.53
TapNet [107] (ICML’19)	ResNet-12	61.65 $\pm$ 0.15	76.36 $\pm$ 0.10	63.08 $\pm$ 0.15	80.26 $\pm$ 0.12
Shot-Free [320] (ICCV’19)	ResNet-12	59.04 $\pm$ 0.43	77.64 $\pm$ 0.39	66.87 $\pm$ 0.43	82.64 $\pm$ 0.43
DeepEMD [37] (CVPR’20)	ResNet-12	65.91 $\pm$ 0.82	82.41 $\pm$ 0.56	71.16 $\pm$ 0.87	86.03 $\pm$ 0.58
DSN-MR [106] (CVPR’20)	ResNet-12	64.60 $\pm$ 0.72	79.51 $\pm$ 0.50	67.39 $\pm$ 0.83	82.85 $\pm$ 0.56
FEAT [239] (CVPR’20)	ResNet-12	66.78 $\pm$ 0.20	82.05 $\pm$ 0.14	70.80 $\pm$ 0.23	84.79 $\pm$ 0.16
Neg-Cosine [126] (ECCV’20)	ResNet-12	63.85 $\pm$ 0.81	81.57 $\pm$ 0.56	-	-
AssoAlign [54] (ECCV’20)	ResNet-18‡	59.88 $\pm$ 0.67	80.35 $\pm$ 0.73	69.29 $\pm$ 0.56	85.97 $\pm$ 0.49
RFS-distill [110] (ECCV’20)	ResNet-12	64.82 $\pm$ 0.82	82.41 $\pm$ 0.43	71.52 $\pm$ 0.69	86.03 $\pm$ 0.49
P-Transfer [67] (AAAI’21)	ResNet-12	64.21 $\pm$ 0.77	80.38 $\pm$ 0.59	-	-
ALFA+MeTA [196] (ICCV’21)	ResNet-12	66.61 $\pm$ 0.28	81.43 $\pm$ 0.25	70.29 $\pm$ 0.40	86.17 $\pm$ 0.35
MixtFSL [161] (ICCV’21)	ResNet-12	63.98 $\pm$ 0.79	82.04 $\pm$ 0.49	70.97 $\pm$ 1.03	86.16 $\pm$ 0.67
BML [256] (ICCV’21)	ResNet-12	67.04 $\pm$ 0.63	83.63 $\pm$ 0.29	68.99 $\pm$ 0.50	85.49 $\pm$ 0.34
FRN [122] (CVPR’21)	ResNet-12	66.45 $\pm$ 0.19	82.83 $\pm$ 0.13	71.16 $\pm$ 0.22	86.01 $\pm$ 0.15
SKD-GEN1 [269] (BMVC’21)	ResNet-12	67.04 $\pm$ 0.85	83.54 $\pm$ 0.54	72.03 $\pm$ 0.91	86.50 $\pm$ 0.58
APP2S [321] (AAAI’22)	ResNet-12	66.25 $\pm$ 0.20	83.42 $\pm$ 0.15	72.00 $\pm$ 0.22	86.23 $\pm$ 0.15
DCAP [322] (ACM’22)	ResNet-12	65.20 $\pm$ 0.67	80.93 $\pm$ 0.53	70.15 $\pm$ 0.74	85.33 $\pm$ 0.55
MDM-Net [323] (IJMLC’22)	ResNet-12	59.88 $\pm$ 0.42	76.60 $\pm$ 0.24	-	-
DSKT-1	ResNet-12	65.88 $\pm$ 0.81	83.09 $\pm$ 0.54	71.60 $\pm$ 0.90	86.61 $\pm$ 0.60
DSKT-2	ResNet-12	67.33 $\pm$ 0.82	84.19 $\pm$ 0.50	72.11 $\pm$ 0.89	86.69 $\pm$ 0.59

classes. We follow the split division in [55] that 100, 50, and 50 classes are grouped for training, validation and testing, respectively.

Implementation details: To make a fair comparison with recent works [110], we use ResNet12 as our backbone. We follow [237] to apply a Dropblock as a regularizer and adjust the amount of filters from (64, 128, 256, 512) to (64, 160, 320, 640). Besides, a 4-neuron fully-connected layer is applied after the final classification layer. Each batch contains 64 samples. We use SGD with a momentum of 0.9 and a weight decay of $5e-4$. The initial learning rate is set to 0.05 and decayed with a factor of 0.1. We train 100 epochs and decay twice for miniImageNet and CUB, we train 60 epochs and decay three times for tieredImageNet, and finally we train 65 epochs and decay once for FC100. The same learning schedule is applied during distillation. Furthermore, we use random cropping, color jittering and random horizontal flip for data augmentation during the whole process. For the hyper-parameters, we set the temperature coefficient to 4.0 and $\eta_1 = \eta_2 = 0.5$, where α , β_1 , λ_1 and λ_2 are tuned on the validation dataset. For evaluation, we train a N -way logistic regression classifier.

6.4.2 Evaluations

miniImageNet and tieredImageNet. Table 6.1 presents a comparison between our approach and the state-of-the-art methods in FSL tasks on the two ImageNet-

6. DUAL SELECTIVE KNOWLEDGE TRANSFER FOR FEW-SHOT CLASSIFICATION

Table 6.2: Comparison of DSKT (our approach) to prior works on FC100 dataset, with mean accuracy (%) and 95% confidence interval. ‡ indicates a deeper backbone. DSKT-1 and DSKT-2 represent the first and second stage, respectively.

Model	Backbone	FC100 5-way	
		1-shot	5-shot
ProtoNet [1] (NIPS’17)	ResNet-12	37.5 $\pm$ 0.6	52.5 $\pm$ 0.6
TADAM [23] (NIPS’18)	ResNet-12	40.1 $\pm$ 0.4	56.1 $\pm$ 0.4
MTL [198] (CVPR’19)	ResNet-12	45.1 $\pm$ 1.8	57.6 $\pm$ 0.9
MetaOptNet [237] (CVPR’19)	ResNet-12	41.1 $\pm$ 0.6	55.5 $\pm$ 0.6
DeepEMD [37] (CVPR’20)	ResNet-12	46.5 $\pm$ 0.8	63.2 $\pm$ 0.7
AssoAlign [54] (ECCV’20)	ResNet-18‡	45.8 $\pm$ 0.5	59.7 $\pm$ 0.6
RFS-distill [110] (ECCV’20)	ResNet-12	44.6 $\pm$ 0.7	60.9 $\pm$ 0.6
MixtFSL [161] (ICCV’21)	ResNet-12	44.9 $\pm$ 0.6	60.7 $\pm$ 0.7
Meta-Navigator [324] (ICCV’21)	ResNet-12	45.6 $\pm$ 0.8	59.9 $\pm$ 0.8
SKD-GEN1 [269] (BMVC’21)	ResNet-12	46.5 $\pm$ 0.8	63.1 $\pm$ 0.7
CORL [259] (WACV’22)	ResNet-12	44.8 $\pm$ 0.7	61.3 $\pm$ 0.5
MDM-Net [323] (IJMLC’22)	ResNet-12	43.6 $\pm$ 0.4	57.4 $\pm$ 0.3
DSKT-1	ResNet-12	45.4 $\pm$ 0.8	62.7 $\pm$ 0.7
DSKT-2	ResNet-12	46.6 $\pm$ 0.8	63.7 $\pm$ 0.7

based benchmarks. Our method is denoted as DSKT, where DSKT-1 and DSKT-2 represent the first and second stage, respectively. On both datasets and in both 1-shot and 5-shot scenarios, our approach yields state-of-the-art results. On mini-ImageNet, DSKT-1 achieves 65.88% and 83.09% in 5-way 1-shot and 5-way 5-shot tasks, respectively. This shows a gain of 1.06% and 0.68% over RFS-distill. The improvement becomes more substantial after distillation, with DSKT-2 producing a gain of 2.51% and 1.78% over RFS-distill under 5-way 1-shot and 5-way 5-shot settings, respectively. On tieredImageNet, DSKT-1 achieves improvements over RFS-distill by 0.08% and 0.58% in the 1-shot and 5-shot tasks, respectively. The improvements are 0.59% and 0.66% with DSKT-2.

FC100. Table 6.2 illustrates similar comparisons, this time on FC100. Here, DSKT provides accuracy improvements in all cases. For DSKT-1, the improvements over RFS-distill for 1-shot and 5-shot are 0.8% and 1.8%, respectively. Furthermore, the addition of distillation (DSKT-2) shows an exclusive improvement of 1.2% under 5-way 1-shot and 1.0% under 5-way 5-shot settings.

CUB. Table 6.3 compares our approach DSKT, against the state-of-the-art on CUB for fine-grained classification. Here, our method outperforms previous works even if they are implemented with deeper backbones. In particular, DSKT-2 achieves improvements over the best-reported numbers by 0.68% and 1.04% in 5-way 1-shot and 5-way 5-shot scenarios, respectively.

Table 6.3: Comparison of DSKT (our approach) to prior works on CUB dataset, with mean accuracy (%) and 95% confidence interval. ‡ indicates a deeper backbone. DSKT-1 and DSKT-2 represent the first and second stage, respectively.

Model	Backbone	CUB 5-way	
		1-shot	5-shot
MatchNet [14] (NIPS’16)	ResNet-12	71.87 ± 0.9	85.08 ± 0.6
ProtoNet [1] (NIPS’17)	ResNet-18‡	71.88 ± 0.9	86.64 ± 0.5
MAML [2] (ICML’17)	ResNet-18‡	68.42 ± 1.0	83.47 ± 0.6
RelationNet [24] (CVPR’18)	ResNet-18‡	67.59 ± 1.0	82.75 ± 0.6
DEML [325] (Arxiv’18)	ResNet-50‡	66.95 ± 1.0	77.11 ± 0.8
Robust-20 [326] (ICCV’19)	ResNet-18‡	58.67 ± 0.7	75.62 ± 0.5
DeepEMD [37] (CVPR’20)	ResNet-12	75.65 ± 0.8	88.69 ± 0.5
Neg-Margin [126] (ECCV’20)	ResNet-18‡	72.66 ± 0.9	89.40 ± 0.4
MixtFSL [161] (ICCV’21)	ResNet-18‡	73.94 ± 1.1	86.01 ± 0.5
APP2S [321] (AAAI’22)	ResNet-12	77.64 ± 0.1	90.43 ± 0.1
DSKT-1	ResNet-12	76.26 ± 0.8	90.76 ± 0.4
DSKT-2	ResNet-12	78.32 ± 0.8	91.47 ± 0.4

6.4.3 Ablative Analysis

Benefits of multi-task learning. To study the impact of multi-task learning, we evaluate our approach with and without the rotation loss on CUB. As shown in Table 6.4, we first simply train the DSKT-1 with cross entropy loss, which is similar to RFS-simple [110], the model achieves 71.74% and 87.23% in 5-way 1-shot and 5-way 5-shot tasks, respectively. Then, we train the DSKT-1 with additional rotation loss, the model performance improves to 76.26% and 90.76%, which presents an absolute gain of 4.52% and 3.53%. From the results, we can infer the importance of multi-task learning during the training process.

Table 6.4: FSL results on CUB, with different combinations of loss functions. For DSKT-2, the loss functions on the left side of the arrow are employed to train the DSKT-1 model.

Model	Loss function	CUB 5-way	
		1-shot	5-shot
DSKT-1	ℓ_{CE}	71.74	87.23
	$\ell_{CE} + \alpha \ell_{BCE}$	76.26	90.76
DSKT-2	$\ell_{CE} \rightarrow \ell_{KD}$	73.59	88.47
	$\ell_{CE} \rightarrow \lambda_1 \ell_{TCKD} + \lambda_2 \ell_{NCKD}$	74.05	88.67
	$\ell_{CE} \rightarrow \eta_1 \ell_{Poly-1} + \eta_2 (\lambda_1 \ell_{TCKD} + \lambda_2 \ell_{NCKD})$	74.60	89.20
	$\ell_{CE} + \alpha \ell_{BCE} \rightarrow \ell_{KD}$	77.87	91.08
	$\ell_{CE} + \alpha \ell_{BCE} \rightarrow \lambda_1 \ell_{TCKD} + \lambda_2 \ell_{NCKD}$	78.16	91.23
	$\ell_{CE} + \alpha \ell_{BCE} \rightarrow \eta_1 \ell_{Poly-1} + \eta_2 (\lambda_1 \ell_{TCKD} + \lambda_2 \ell_{NCKD})$	78.32	91.47

6. DUAL SELECTIVE KNOWLEDGE TRANSFER FOR FEW-SHOT CLASSIFICATION

Table 6.5: FSL results on CUB, with different processing methods in the first stage.

Processing methods	CUB 5-way	
	1-shot	5-shot
Puzzle	75.75 ± 0.8	89.50 ± 0.5
Fusion	73.93 ± 0.8	89.02 ± 0.5
Rotation	76.26 ± 0.8	90.76 ± 0.4

Table 6.6: FSL results on CUB, with different classifiers during the evaluation process.

Classifiers	CUB 5-way	
	1-shot	5-shot
Nearest classifier	77.92 ± 0.7	88.02 ± 0.5
Cosine classifier	77.67 ± 0.7	88.04 ± 0.5
Logistic Regression	78.32 ± 0.8	91.47 ± 0.4

Benefits of DSKD. To better evaluate the contribution of DSKD, we train the model DSKT-2 with different combinations of loss functions on CUB. From Table 6.4, we can find that, for models trained only with cross entropy loss in the first stage, DSKD gives 1.01% and 0.73% gains compared with KD (KD is defined as $\ell_{KD} = KL(p_i^T p_i^S)$, where p_i denotes the output logits of the i -th sample, and T and S represent the teacher and student model, respectively.) in 1-shot and 5-shot tasks, respectively. For models trained with both cross entropy and binary cross loss functions, DSKD still achieves 0.45% and 0.39% improvements than KD. These results confirm the effectiveness of DSKD.

Benefits of rotation and logistic regression classifier. To study the impact of rotation and logistic regression classifiers, we evaluate our approach with other processing methods and classifiers on CUB. Table 6.5 presents the results on CUB with different processing methods (i.e., puzzle and fusion) in the first stage (DSKT-1). From Table 6.5 we can find that rotation performs much better than fusion (Three channels of RGB images are fused and the coefficient of each channel is set to 0.5 in this work). When compared to puzzle (split the original image into pieces, randomly sort and then reassemble. The puzzle size is set to 7 on CUB 5-way tasks.), rotation still obtains 0.51% and 1.26% improvements on CUB 5-way 1-shot and 5-shot tasks, respectively. Table 6.6 shows the FSL results with different classifiers on CUB during the evaluation process. From Table 6.6, we can find that logistic regression classifier achieves the best performance when compared to Nearest classifier (make predictions based on the euclidean distances between query and support samples) and Cosine classifier (make predictions based on the cosine similarities between query and support samples) on both CUB 5-way 1-shot and 5-shot tasks during the evaluation process.

Table 6.7: FSL results on CUB, with different inputs for the rotation classifier in the first stage.

Input of the rotation classifier	CUB 5-way	
	1-shot	5-shot
The feature extractor’s output	75.37 ± 0.8	90.10 ± 0.4
The category classifier’s output	76.26 ± 0.8	90.76 ± 0.4

Table 6.8: Results of different degrees of the polynomial used in PolyLoss on CUB.

N	CUB 5-way	
	1-shot	5-shot
1	78.32 ± 0.8	91.47 ± 0.4
2	77.57 ± 0.8	91.10 ± 0.4
3	77.42 ± 0.8	91.11 ± 0.4

Table 6.9: Results of different applications of rotation loss on CUB.

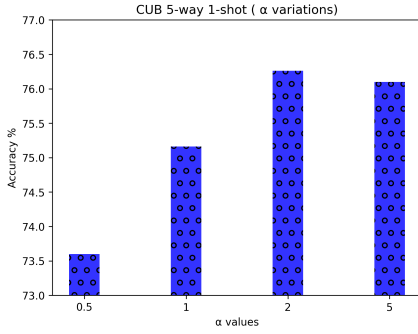
Application of the rotation loss	CUB 5-way	
	1-shot	5-shot
Applied on the features	75.73 ± 0.8	89.84 ± 0.4
Applied on the category logits	76.26 ± 0.8	90.76 ± 0.4

Benefits of using the output of the category classifier as input for the rotation classifier. In this section, we exploit the output of category classifier as input for the rotation classifier, rather than that of the feature extractor. This can help improve the model’s performance on complex data. Table 6.7 present the comparison results on CUB. From Table 6.7, we can find that applying the output of the category classifier gains 0.89% and 0.66% improvements in CUB 5-way 1-shot and 5-shot tasks, respectively.

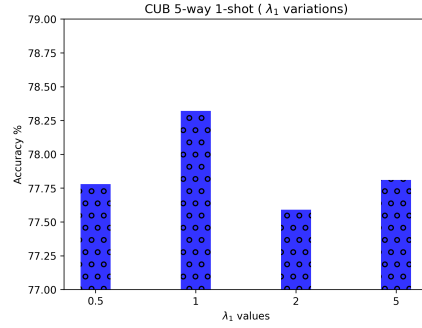
Different degrees of polynomial in PolyLoss. The degree of the polynomial used in PolyLoss N is set to 1 in this work. To study the impact of the degree, we assign the degree with different values in CUB 5-way tasks. From Table 6.8, we can find that when $N = 1$, our method achieves around 0.8% and 0.3% improvements compared to $N = 2$ or $N = 3$ in CUB 5-way 1-shot and 5-shot tasks, respectively.

Application of the rotation loss. In this work, the rotation loss is directly applied to the predicted logits of the category classifier. To learn its contribution, we make a comparison with applying rotation loss on the features in CUB 5-way tasks. Table 6.9 shows that compared with applying multi-task loss on the features, applying on the category logits achieves 0.53% and 0.92% improvements on CUB 5-way 1-shot and 5-shot tasks, respectively.

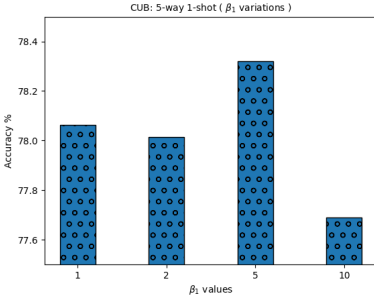
6. DUAL SELECTIVE KNOWLEDGE TRANSFER FOR FEW-SHOT CLASSIFICATION



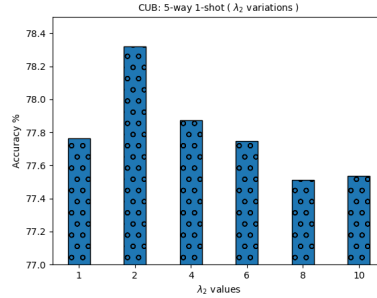
(a) Ablation study on the sensitivity of hyper-parameter α on CUB 5-way 1-shot tasks.



(b) Ablation study on the sensitivity of hyper-parameter λ_1 on CUB 5-way 1-shot tasks.



(c) Ablation study on the sensitivity of hyper-parameter β_1 on CUB 5-way 1-shot tasks.



(d) Ablation study on the sensitivity of hyper-parameter λ_2 on CUB 5-way 1-shot tasks.

Figure 6.2: Ablative Analysis.

Variations of Hyper-parameters: There are totally seven hyper-parameters in this work, t , α , β_1 , λ_1 , λ_2 , η_1 and η_2 . t denotes the temperature coefficient, α controls the contribution of rotation loss during DSKT-1, β_1 is used to adjust the contribution of the first polynomial base, λ_1 , λ_2 , η_1 and η_2 are the weights of different loss functions. We mainly investigate the variants of hyper-parameters α , β_1 , λ_1 and λ_2 , where the default values of t , η_1 and η_2 are 4.0, 0.5 and 0.5, respectively. Figure 6.2a shows the DSKT-1 performance on CUB 5-way 1-shot tasks by changing α . It's obviously that our model performance increases from 0.5 to 2, and then decreases when $\alpha = 5$, which indicates the importance of multi-task learning. Figure 6.2c shows the DSKT-2 performance on CUB 5-way 1-shot tasks by changing β_1 . We observe that DSKT-2 achieves over 78% when β_1 increases from 1 till 5, while $\beta_1 = 10$ performs the lowest. Note that the performance drops only by about 0.6%, which indicates that it is not sensitive to the value of β_1 . Figure 6.2b presents the DSKT-2 performance on CUB 5-way 1-shot tasks by changing λ_1 . When λ_1 ranges from 0.5 till 5, DSKT-2 can always obtain over 77% accuracies and achieves the highest

accuracy when $\lambda_1 = 1.0$, indicating it is not sensitive to the value of λ_1 . Figure 6.2d presents the DSKT-2 performance on the same tasks by changing λ_2 . We find that the performance increases from 1 to 2, then decreases with larger values of λ_2 . These results indicate that it is not a good idea to blindly increase the weights of ℓ_{NCKD} .

6.5 Chapter Conclusions

In this work, we aim to raise awareness of the importance of training a well-generalized feature extractor for FSL tasks by proposing a new two-stage dual selective knowledge transfer framework. First, we use multi-task learning to enforce the feature extractor to learn robust low-level features. Then, we propose an effective dual knowledge distillation method, which enables the student model to selectively learn knowledge from the teacher model and current examples, further improving the model’s generalization ability. Extensive experimental results demonstrate the importance of strong feature extractors for FSL and our approach outperforms the state-of-the-art on four popular FSL benchmark datasets.

Future Work. Despite the promising results in our study, there are some areas that are worth further investigation. We only apply our method on classification tasks in this chapter, object detection tasks will be our future research direction. The correlation between the distillation performance and polynomial function coefficients β_i is not fully investigated, we will expand upon our work in future researches.