國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

# 基於預訓練正規化之元學習

# Improving Meta-Learning by Regularized Pre-training

陳佳佑

Chia-You Chen

指導教授：林軒田博士

Advisor: Hsuan-Tien Lin, Ph.D.

中華民國 109 年 5 月

May, 2020

doi:10.6342/NTU202001626

doi:10.6342/NTU202001626

# 國立臺灣大學碩士學位論文
# 口試委員會審定書

## 基於預訓練正規化之元學習

## Improving Meta-Learning by Regularized Pre-training

本論文係陳佳佑君（學號 R07922001）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 109 年 5 月 26 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

林軒田
（指導教授）

陳縕儂

李宏毅

系主任 莊永裕

doi:10.6342/NTU202001626

doi:10.6342/NTU202001626

# 誌謝

在各種意外與巧合下，才有這篇論文，三月時，一切都毫無頭緒，甚至連四月的自己是否還在就學都無法確定，一路上有許多人的幫助，才有現今的成果。感謝我的父母，信任我在碩士旅程上的各個決定; 感謝 CLLAB 的各位，在互相的討論下，慢慢釐清出方向; 感謝 RIKEN 的 Gang 哥，提供給我元學習上的各種建議，指正我在研究上過於相信前人作品的想法; 感謝韋霖，把我拉入了元學習的領域，並對於實驗結果給予了各式各樣的建議; 感謝 Vivian 跟宏毅老師，在百忙之中，抽空幫忙口試; 感謝建民，在我毫無頭緒的日子，頂著時差給予我各式的方向，重新審視以前失敗的方向; 感謝侑廷，在 RIKEN 實習時，幫助我快速適應了生活上的種種，在研究上也給予了切實的建議; 最後，感謝 HT，提供給我自由的研究環境，在大學時一頭栽進 lab 少有涉略的強化學習，甚至是同意我在碩二上，又栽入了元學習的領域，此外，還有給予我機會去日本見識不一樣的研究風氣，與不同文化背景的人交流，而在研究之外，也給予我機會接觸系統管理，嘗試各式各樣的系統與架構，而在研究以外，也在我求學的道路上給予過來人的指點。這條路雖然走得彎彎曲曲，但在大家的幫助下，還是一步一步的走完了。

doi:10.6342/NTU202001626

doi:10.6342/NTU202001626

# 摘要

本論文提出一種新的改善元學習在少樣本學習 (few-shot learning) 的方法。近年來，元學習方法在少樣本學習有不俗的表現。相比於傳統的遷移式學習因為參數過多容易過擬和，元學習透過在訓練時模擬測試時的設定，而有較好的泛化效果。在這一兩年內，一些研究提出透過精細調整遷移式學習的方法，可以達到一樣的泛化效果。更甚者，有研究提出利用一般遷移式學習中的預訓練參數，可以當成元學習的初始化參數，並達成更好的學習效果。然而，我們發現這些研究都只是單純把預訓練的權重拿來使用，並且還是將重心放在元學習的架構調整上，而忽略預訓練改進之潛力。我們透過預訓練正規化在元學習上達到更快的收斂，在較淺的網路上有更好的收斂結果。並且，我們認為相比元學習，預訓練的方法更需要大家去改進。

**關鍵字：** 元學習, 預訓練, 正規化

doi:10.6342/NTU202001626

doi:10.6342/NTU202001626

# Abstract

This thesis presents a new dimension to improve the meta-learning in the few-shot learning field. In recent years, meta-learning has become one of the best ways to solve few-shot learning tasks. Traditional transfer learning style algorithms would easily overfit in the meta-train dataset and lead to poor performance on the meta-test dataset. However, in these two years, researchers have found that pre-training with some sophisticated fine-tuning may lead to competitive performance with the meta-learning approach. Moreover, utilizing the pre-training classifier weight on the original dataset could improve the meta-learning approach. Furthermore, pre-training is more time-efficient than episodic learning for meta-learning due to the sampling problem. We find that recent years of study mainly focus on the meta-learning part. However, the pre-training part has been less studied. We provide a naive regularization for pre-training with respect to Prototypical Network in the meta-learning stage. It leads to faster convergence speed and competitive performance which is comparable to classical Prototypical Network. The result implies that the classical meta-learning algorithm is good enough and it's possible to transfer some computation burden to the pre-training part.

**Keywords:** Meta-Learning, Pre-training, Regularization

doi:10.6342/NTU202001626

doi:10.6342/NTU202001626

# Contents

doi:10.6342/NTU202001626

doi:10.6342/NTU202001626

# List of Figures

doi:10.6342/NTU202001626

doi:10.6342/NTU202001626

# List of Tables

doi:10.6342/NTU202001626

doi:10.6342/NTU202001626

# Chapter 1

# Introduction

In recent years, deep learning methods have outperformed most of the traditional methods in supervised learning, especially in image classification. However, deep learning methods have lots of parameters and require lots of labeled data. In contrast, traditional methods such as support vector machine [6] requires less data than deep learning methods to perform well. To conquer the data issue, researchers have tried to incorporate some semi-supervised [2] [24] [1] assumptions to reduce the number of labeled data. However, in a more realistic setting, the unlabeled data may still be precious. For example, an ornithologist takes some photos of a new kind of bird occasionally. Since the data for the bird is limited, it's hard for semi-supervised learning methods to work. To conquer the limited data issue, researchers develop few-shot learning algorithms. In few-shot learning, we have a training dataset that consists of data for base classes. In the testing phase, the method should perform well on novel classes, which are unseen in base classes, with limited labeled data points.

Currently, there are two main frameworks, meta-learning [9] [20] [4] and transfer learning [8], that deal with the few-shot learning problem. For transfer learning, the main idea is to train a traditional classifier on the meta-train dataset. In the testing phase, these methods finetune on the limited datapoints for the labeled novel classes. For meta-learning frameworks, it contains a concept called episodic training [23]. For the testing phase of few-shot learning, the learning method is given N novel classes each contains K labeled data for finetune and Q query data for each class to evaluate the performance. Unlike

doi:10.6342/NTU202001626

transfer learning algorithms, episodic training tries to simulate the testing literature in the training phase by sampling. Due to episodic training, meta-learning methods generalize better than traditional transfer-like methods for the novel classes.

In these two years, some transfer-learning methods with sophisticated design in the finetuning [8] part have a competitive performance to the meta-learning approaches. Moreover, researchers have pointed out that combining both the global classifier pre-train part in the transfer learning framework and the episodic training concept for the meta-learning framework could lead to better performance. Yet, currently for the usages of pre-training are only limited to a simple cross-entropy loss classifier and most of the approaches focus on the meta-learning part.

In the past, meta-learning methods are more attractive due to better generalization ability. However, once the transfer learning methods catch up, the flaw of meta-learning algorithms have emerged. Episodic training is the key to make meta-learning outstood, however, it's also the time bottleneck since the episodic sampling is time-consuming. To make meta-learning faster, the most intuitive way is to reduce the number of episodes. Currently, there are only limited researches on reducing the number of episodes. One of the methods is to apply a better weight initialization that we have mentioned priorly. Another method is to mimic how people learn. For example, when we are learning dynamic programming, given a knapsack problem with simple constraint and the one with strong constraint, we will learn much more when we solve the strong constraint. [21] follows the latter idea and crafts the hard episode to make necessary episodes fewer.

In this work, we provide another aspect to reduce the number of episodes. Moreover, we also show that the classical meta-learning algorithm, ProtoNet, may be strong enough. The pre-training part still has room for improvement. We propose a simple regularization in the pre-training phase and have competitive performance and better convergence speed about the normal pre-training one. The original pre-training with meta-learning algorithms would simply remove the last layer of the global classifier and use it as an initialization weight for the meta-learning algorithms. We try to capture the high-level concept of Protonet, "data points in the same class should be closer". To capture the concept in

doi:10.6342/NTU202001626

pre-training, we add an auxiliary loss on the second last layer output, which constrains the distance between each point with in the same class indirectly. On the other hand, our approach is another kind of regularization for the weight of the pre-training weight. The simple regularization makes Protonet more competitive, which also supports our claim that what we need is a better initialization weight.

To sum up, there are three main contributions in this work.

1. A naive regularization that sharpens the classical meta-learning algorithms.

2. A new aspect for reducing the necessary episodic training episodes.

3. A brief study of the relation between pre-train weight and the meta-learning task.

The remainder of the thesis is organized as follows. In Chapter 2, we briefly introduce some main-stream meta-learning approaches in recent years. In Chapter 3, we give a formal problem definition of few-shot learning. The proposed method is illustrated in Chapter 4. Finally, we present the experiment results in Chapter 5 and conclude the work in Chapter 6.

doi:10.6342/NTU202001626

# Chapter 2

# Related Work

Few-shot learning tries to mimic the human ability to generalize novel classes with limited datapoint. In the following, we briefly discuss the transfer-learning framework and two categories of the meta-learning framework. In the end, we discuss the under-studied episode reduction method.

## 2.1 Transfer-Learning Framework

In the training phase, the transfer-learning framework would train a classifier on the general classification task across all base classes instead of utilizing episodic learning. And for the testing phase, transfer-learning methods finetune with the limited labeled data. There are several kinds of tricks. [16] proposes a method to append the mean of the embedding with a given class as a final layer of the classifier. [17] uses the parameter of the last activation output to dynamic predict the classifier for novel classes. [11] proposes a similar concept with [17]. [11] also embeds the weight of base classes during the novel class prediction. Moreover, it introduces an attention mechanism instead of directly averaging among the parameters of each shot. Besides embedding base classes weight to the final classifier, [8] utilizes label propagation by the uncertainty on a single prediction to prevent overfitting in the finetune stage, which is quite similar to the classical classification tasks.

doi:10.6342/NTU202001626

## 2.2 Meta-Learning Framework

For meta-learning like framework, the main difference is the concept of learning to learn and episodic training [23]. Learning to learn refers to learn from a lot of tasks to benefit the new task learning. To prevent confusion, the original train and test phase would be regarded as ”meta-train” and ”meta-test”. The term ”train” and ”test” would be referred to the one in each small task. Episodic training is the process of mimicking the task structure in meta-test during training. If the meta-test phase consists K support examples and Q query examples from N classes, then we will sample lots of tasks that also have K support examples and Q query examples from N classes in the meta-train phase. Meta-learning algorithms have developed rapidly in recent years. We will briefly categorize them into two categories, optimization-based methods and metric-based methods.

**Optimization-based Method** Optimization-based methods try to get an embedding that could easily fit subtasks by adding some extra layers. [9] proposed MAML (Model-Agnostic Meta-Learning), which tries to get a network that is the closest to all the best model in the low-way low-shot tasks. However, MAML may be quite inefficient due to the computation of Hessian matrix. To leverage the issue, iMAML [18] and foMAML [14] provides different approximation to avoid the heavy computation. However, MAML still suffers from the high dimensional overfitting issue. LEO model [19] solves the overfitting issue by learning a low dimensional latent space. Instead of aiming to get an embedding that could benefit the latter fully connected layer, MetaOptNet [13] aims to get an embedding that could benefit the latter differentiable support vector machine.

**Metric-based Method** Instead of learning an embedding that could benefit the latter additional layer, metric-based methods aim to get an embedding that could easily classify the classes by simple metrics. Matching Networks [23] conducts a cosine similarity metric with a Full Context Encoding module(FCE). Prototypical Networks [20] replaces the cosine similarity metric with the squared Euclidean distance and computes the mean of the embedding in the support-set as the prototype. Relation Network [22] embeds a relation

doi:10.6342/NTU202001626

module in the learning metric. Instead of using a consistent metric in the task, TADAM [15] designs a task-dependent metric that could dynamically fit new class combinations.

## 2.3 Mixed Framework

Some recent works have found that using a global pre-trained classifier as the initialization weight could lead to a better meta-learning result. [21] uses the pre-trained classifier weight as initialization weight and launches a simple gradient-based with restricting the learning process as shift and scale. Meta-Baseline[5] also follows the initialization literature and applies cosine similarity metric for the following learning process. [4] changes the original pre-trained network structure into a weight imprinting taste and a simple gradient-based method for the episodic learning part.

## 2.4 Episode Reduction Methods

Due to the parallelism property, we could see that normal training literature is much faster than episodic training. Moreover, recent researchers have found that a pre-trained classifier would lead to better meta-learning results. On the other hand, we could reduce the amount of heavy time-consumed episodes by using a pre-trained classifier. Besides utilizing the pre-training weight to reduce the number of episodes, Meta Transfer Learning [21] proposes the concept of hard episode. For each normal episode, MTL would add the class with the worst performance in a pool. After collecting for a while, MTL would create hard episodes by sampling from the pool.

Instead of crafting hard episodes, our approach tries to utilize more in the pre-training phase. We propose a simple regularization that could reduce the difference between the embeddings of the pre-trained classifier and the episodic training one. It has significantly reduced the number of episodes and achieves a similar (even better) performance for the original algorithms. Moreover, for shallow and deep backbones, it increases the final accuracy.

doi:10.6342/NTU202001626

# Chapter 3

# Background and Problem Setup

## 3.1 Few-Shot Learning

Since the method we propose based on metric learning, we follow the meta-learning literature. For each episode (small task), it is N-way K-shot. N-way K-shot refers to have K labeled instance for each N class. Also, there are Q query data points for each class that are unlabeled and are used for evaluating the performance of the method. Moreover, support set $S$ refers to the N-way K-shot labeled data. The query set $Q$ refers to the Q unlabeled data for each class.

The data is splitted in $D_{train}$, $D_{valid}$ and $D_{test}$. The classes for $D_{train}$, $D_{valid}$ and $D_{test}$ will not overlap. In the meta-training phase, we create episodes (small tasks) from $D_{train}$. In the meta-test phase, we create 600 episodes from $D_{test}$ and calculate the average performance.

In the meta-training phase, a learning algorithm is provided with episodes generated from $D_{train}$ composed of the support set $S$ and the query set $Q$. The learning algorithm trains on the support set and gets a score from the query set $Q$. After that, it meta-updates the network by the score, which is exactly the concept of learning to learn. For meta-testing phase, a learning algorithm is evaluated with the tasks constructed from the $D_{test}$ and record the average accuracy.

doi:10.6342/NTU202001626

## 3.2 Problem Setup

Episode construction may cost too much time since it is hard to parallelize. For the problem setup, we follow the standard few-shot learning algorithms. Besides improving the accuracy of the methods, we also focus on improving the convergence speed. Our goal is to provide a simple method to reduce the amount of necessary episode, which is quite similar to the concept of active learning. However, we try to reduce the amount of the training episode instead of the numbers of queries for the unlabeled data.

doi:10.6342/NTU202001626

# Chapter 4

# Proposed Method

## 4.1 Notation

In a N-way K-shot few-shot classification task, we are given a small support set of $N\dot{K}$ labeled data $S = \{(x_1, y_1), ..., (x_{N\dot{K}}, y_{N\dot{K}})\}$ where $x_i \in \mathbb{R}^D$ and $y_i \in \{1, ..., K\}$. $S_k$ denotes the set of examples labeled with class $k$.

## 4.2 Prototypical Network (ProtoNet)

Prototypical Network is one of the classical metric-based meta-learning algorithms. First, it computes prototypes $c_i$, which are $M$-dimensional representation $c_i \in R^M$ by averaging the output of the embedding function $f_\theta : \mathbb{R}^D \to \mathbb{R}^M$ from the same class set $S_i$.

$$c_i = \frac{1}{|S_i|} \sum_{(x,y) \in S_i} f_\theta(x) \tag{4.1}$$

Then, the prototypical network calculates a probability distribution for a given data $x$ by softmax over the Euclidean distance $d$ between each prototype and the embedding $f_\theta(x)$.

$$p_\theta(y = k|x) = \frac{exp(-d(f_\theta(x), c_k))}{\sum\limits_{k' \in \{1,...,K\}} exp(-d(f_\theta(x), c_{k'}))} \tag{4.2}$$

doi:10.6342/NTU202001626

**Algorithm 1** Regularized Pre-trained Prototypical Network (RP-Proto)

```
Require: Training set $D = \{(x_1, y_1), ...., (x_N, y_N)\}$, where each $y_i \in \{1, ..., K\}$.
 1: $M$: metric for judging distance between two vector.
 2: $f_\theta$: stands for the feature extractor which may output $e$ dimension vector.
 3: $g$: maps the output of $f_\theta$ to k dimension.
 4: $b$: batch sizes for pretraining.
 5: $p$: pretrained iteration number.
 6: $\alpha$: weight vector for the regularization loss.
 7: $L$: cross entropy loss
 8: Initialize $W_i$ with $e$ dimension for $i \in \{1, ..., K\}$
 9: for $i \in range(p)$ do
10:     $(X, Y) \leftarrow$ SAMPLE$(D, b)$;
11:     $l_c \leftarrow L(g(f_\theta(X)), Y)$
12:     $l_{reg} \leftarrow \frac{1}{b} \sum_{(x,y) \in (X,Y)} M(f_\theta(x), W_y)$
13:     $l_{tolal} \leftarrow l_c + \alpha \times l_r$
14:     $f_\theta, g \leftarrow$ Backpropogate$(l_{total}, (f_\theta, g))$
15: end for
16: $f_\theta \leftarrow$ ProtoNet$(f_\theta, D)$
17: return $f_\theta$
```

## 4.3 Regularized Pretrained Prototypical Network (RP-Proto)

Our goal is to reduce the number of episodes during episode training. Meanwhile, the pre-trained initialization weight works pretty well in recent researches. Thus, we come out with the idea of transferring the computation burden to the pre-training phase. A naive way is to mimic the loss of episodic learning in each pre-training batch. One straight forward way to implement is to calculate the mean of the embedding of each class in the pre-training phase and add a loss with the same one as prototypical networks. However, the step of calculating the exact mean of each class may cause some issues. It's hard to make sure that each batch consists uniform amount of instances in each class. Though it's possible to design a delicate uniform sampler, when it comes to large label sets, it may require a large batch. Instead of calculating the real mean, we introduce a surrogate loss that could reduce the order and approximate the idea.

We capture the higher concept of metric learning, ”instance with the same class should gather together in the embedding space”. For example, PROTONET uses euclidean distance to achieve the concept. We compute $l_{reg}$ base on the distance from $f_\theta(x)$ and $W_y$

doi:10.6342/NTU202001626

where $W_y$ is a $M$ dimension learnable parameter.

$$l_{reg}(x, y) = d(f_\theta(x), W_y) \tag{4.3}$$

For the pre-training phase on the ordinary classification task, $f_\theta$ corresponds to the network right before mapping to a final $K$ dimension. Then, we will add $l_{reg}$ to the ordinary classification loss. Instead of directly summing, we adopt a weighted approach of $\alpha$.

$$l_{total} = l_{classification}(x, y) + \alpha \times l_{reg}(x, y) \qquad (\alpha = 10^C) \tag{4.4}$$

$W_y$ is the surrogate mean of each class. Instead of directly summing up all the embeddings, $W_y$ could be calculated by backpropagation. Moreover, when $W_y$ equals to zero, it could be considered as an l2 regularization on the feature space, which scales the embedding. When $W_y$ is learnable, it makes each instance in the same class closer which satisfies the higher concept of ”metric learning”. We have described the detail in 1.

Recap with the flaw of the naive approach. In this auxiliary loss schema. A data point doesn’t need to consider the distance of data points in other classes, as a result, it could avoid the large computation effort of the cross-classes datapoint distance and the non-uniform amounts of datapoints problem.

doi:10.6342/NTU202001626

# Chapter 5

# Experiments

## 5.1 Dataset Details

**MiniImagenet** Proposed by [23] is a widely used benchmark for few-shot learning. It's a subset of ILSVRC-2015 [7] with 100 classes, 64 for meta-train, 16 for meta-validation and 20 for meta-test. Each class contains 600 images of size 84x84.

**CIFAR-FS** Proposed by [3] random splits between the original classes of CIFAR100 [12]. Since it's randomly split among classes and there are inter-class relationships in CIFAR100 [12], it has less semantic drift between the base classes and the novel classes. However, the image has only 32 x 32 resolution which makes the task harder to fast prototype. CIFAR-FS also follows the miniImagnet structure with 64 base classes, 16 validation classes, and 20 novel classes.

**FC100** Proposed by [15] tries to create a harder dataset that has the hardness of both the low resolution and the semantic drift between the base classes and the novel classes. Based on CIFAR100 [12], FC100 splits novel classes and base classes according to the 20 super-classes which are grouped by the original 100 classes. Moreover, it also has the same number of base, validation, and test classes split.

doi:10.6342/NTU202001626

| function | parameters |
|---|---|
| RandomSizedCrop | image_size |
| RandomHorizontalFlip | |
| Normalize Mean | (0.485, 0.456, 0.406) |
| Normalize Std | (0.229, 0.224, 0.225) |

Table 5.1: Preprocess pipeline

## 5.2 Experiment Setup

The first part experiment is to show the improvement of our methods and the original prototypical network with standard pre-trained backbone. We launch the experiment on all the datasets above. We set the pre-trained epoch to 100, which means that all the backbone should be well-trained. Also, we search the weighted hyperparameter $\alpha = 10^C$ from $C = -1$ to $-3.5$ for each backbone on each dataset.

The second part experiment is to show that adding the regularization on pre-trained may be possible. We reuse the pre-trained backbone without regularization loss and analyze the soft-nearest-neighborhood loss during episodic training.

The third part experiment is to study the relationship between different weighted hyperparameter $\alpha$ and the weight of the backbone. We analyze the backbone weight with different $\alpha$ among the pre-training phase.

The forth part experiment is to study the reason why the regularization loss could sharpen the performance. We analyze accuracy in the episodic learning phase with different pre-trained epoch and different $\alpha$.

## 5.3 Implementation Details

**Model Architecture** For the backbone of ResNet10 and ResNet18, we follow the preprocessing from [4]. We use image sizes with 224x224 for both pre-trained prototypical network and regularization pre-trained prototypical network. For ResNet12 backbone, we follow the implementation from [21] and use image sizes with 80x80. For the Conv4 backbone, we follow the implementation in the original work with image size 84x84.

doi:10.6342/NTU202001626

**Pre-training** For pre-training the backbone, we follow the settings in [5]. We use a stochastic gradient descent optimizer with a learning rate as 0.001 for all the backbones and the batch size is 128.

**Episodic Learning** For the episodic-learning part, we also follow the settings in [5]. We use stochastic gradient descent optimizer with learning rate as 0.001 and turn off the data augmentation part in the preprocess pipeline Table 5.1 to reach a more stable result.

**Evaluation** For the evaluation phase, we turn off all the data augmentation and record the mean of the performance and the 95% confidence intervals among 600 randomly sampled episodes.

## 5.4 Performance Evaluation

In the miniImagenet dataset result in Table. 5.2, our method has a competitive performance with other methods. We find that adding the regularized term could make the shallow network has a similar performance with the large one. Moreover, when the backbone is shallow, we have outperformed all other methods with the same backbone.

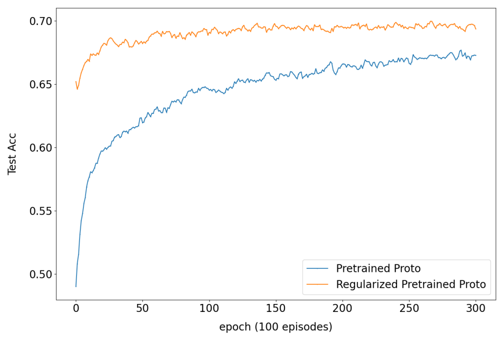
For our major goal ”Reduce the number of episodes”, we do a one by one comparison between standard pre-trained Prototypical Networks and the regularized pre-trained one by recording the meta-test performance after every 100 episodes in Fig. 5.1. Instead of ResNet18, all the regularized pre-trained backbones lead to a faster convergent speed by providing better initial accuracy. However, for ResNet18 it has a similar performance.

For experiment results on fc100 Table. 5.4 and cifar-fs Table. 5.3, the methods have improved the performance of the one with the shallow and deep network. The experiment is similar to the result on miniImagnet Table. 5.2.

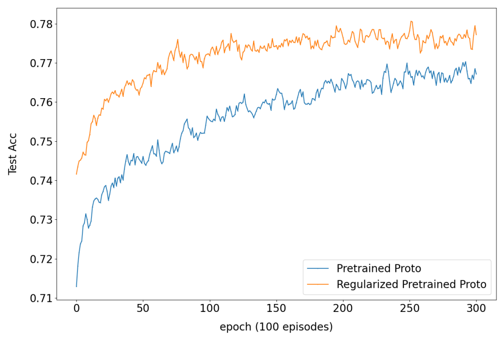
doi:10.6342/NTU202001626

|  | Backbone | 1 shot | 5 shot |
|---|---|---|---|
| Matching Networks | Conv4 | 43.56 ± 0.84 | 55.31 ± 0.73 |
| Prototypical Networks | Conv4 | 48.70 ± 1.84 | 63.11 ± 0.92 |
| LEO | WRN-28-10 | 61.76 ± 0.08 | 77.59 ± 0.12 |
| Chen et al. | ResNet18 | 51.87 ± 0.77 | 75.68 ± 0.63 |
| SNAIL | ResNet12 | 55.71 ± 0.99 | 68.88 ± 0.92 |
| AdaResNet | ResNet12 | 56.88 ± 0.62 | 71.94 ± 0.57 |
| TADAM | ResNet12 | 58.50 ± 0.30 | 76.70 ± 0.30 |
| MTL | ResNet12 | 61.2 ± 1.80 | 75.50 ± 0.80 |
| MetaOptNet | ResNet12 | 62.64 ± 0.61 | 78.63 ± 0.46 |
| Prototypical Networks (Pre) | Conv4 | 42.08 ± 0.75 | 67.77 ± 0.68 |
| Prototypical Networks (Pre) | ResNet10 | 55.34 ± 0.81 | 76.97 ± 0.67 |
| Prototypical Networks (Pre) | ResNet12 | 56.83 ± 0.83 | 76.45 ± 0.64 |
| Prototypical Networks (Pre) | ResNet18 | 57.80 ± 0.84 | 77.71 ± 0.63 |
| RP-Proto (euclidean) (C: -1.5) | Conv4 | 50.38 ± 0.80 | 69.56 ± 0.71 |
| RP-Proto (euclidean) (C: -1.5) | ResNet10 | 56.92 ± 0.84 | 77.45 ± 0.65 |
| RP-Proto (euclidean) (1-shot C: -2.5) (5-shot C: -3.5) | ResNet12 | 57.84 ± 0.82 | 77.08 ± 0.70 |
| RP-Proto (euclidean) (1-shot C: -1.0) (5-shot C: -3.5) | ResNet18 | 57.88 ± 0.86 | 77.43 ± 0.63 |

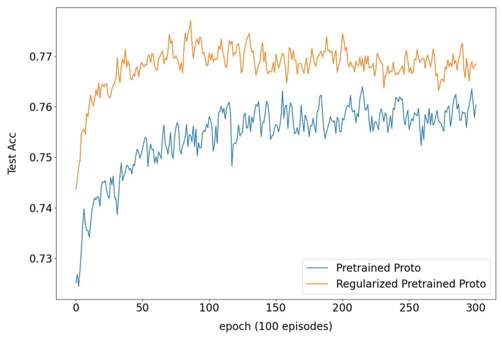
Table 5.2: 5-way 5-shot and 5-way 1-shot performance on miniImagenet.



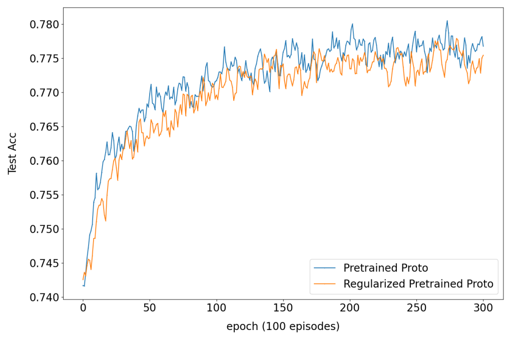
(a) Conv4



(b) ResNet10



(c) ResNet12



(d) ResNet18

Figure 5.1: Comparison for the converge speed between standard pre-trained and regularized pre-trained on 5-way 5-shot miniImagenet.

doi:10.6342/NTU202001626

| | Backbone | 1 shot | 5 shot |
|---|---|---|---|
| MAML | Conv4 | $58.9 \pm 1.9$ | $71.5 \pm 1.0$ |
| Prototypical Networks | Conv4 | $55.5 \pm 0.7$ | $72.0 \pm 0.6$ |
| Relation Networks | 64-96-128-256 | $55.0 \pm 1.0$ | $69.3 \pm 0.8$ |
| R2D2 | 96-192-384-512 | $65.3 \pm 0.2$ | $79.4 \pm 0.1$ |
| MetaOptNet | ResNet12 | $72.8 \pm 0.7$ | $85.0 \pm 0.5$ |
| Prototypical Networks (Pre) | Conv4 | $39.94 \pm 0.81$ | $74.19 \pm 0.73$ |
| Prototypical Networks (Pre) | ResNet10 | $63.45 \pm 0.93$ | $81.62 \pm 0.65$ |
| Prototypical Networks (Pre) | ResNet12 | $67.58 \pm 0.92$ | $83.25 \pm 0.64$ |
| Prototypical Networks (Pre) | ResNet18 | $65.65 \pm 0.92$ | $82.80 \pm 0.62$ |
| RP-Proto (euclidean) (C: -1) | Conv4 | $54.66 \pm 0.89$ | $75.16 \pm 0.75$ |
| RP-Proto (euclidean) (C: -1) | ResNet10 | $66.20 \pm 0.93$ | $83.77 \pm 0.69$ |
| RP-Proto (euclidean) (C: -3) | ResNet12 | $67.52 \pm 0.93$ | $83.30 \pm 0.63$ |
| RP-Proto (euclidean) (C: -3) | ResNet18 | $65.34 \pm 0.90$ | $82.98 \pm 0.68$ |

Table 5.3: 5-way 5-shot and 5-way 1-shot performance on CIFAR-FS.

| | Backbone | 1 shot | 5 shot |
|---|---|---|---|
| Prototypical Networks | Conv4 | $35.3 \pm 0.6$ | $48.6 \pm 0.6$ |
| TADAM | ResNet12 | $40.1 \pm 0.4$ | $56.1 \pm 0.4$ |
| MetaOptNet | ResNet12 | $47.2 \pm 0.6$ | $62.5 \pm 0.6$ |
| Prototypical Networks (Pre) | Conv4 | $35.63 \pm 0.72$ | $51.80 \pm 0.73$ |
| Prototypical Networks (Pre) | ResNet10 | $38.55 \pm 0.70$ | $57.71 \pm 0.71$ |
| Prototypical Networks (Pre) | ResNet12 | $39.48 \pm 0.66$ | $57.18 \pm 0.72$ |
| Prototypical Networks (Pre) | ResNet18 | $39.02 \pm 0.69$ | $56.57 \pm 0.73$ |
| RP-Proto (euclidean) (C: -1.5) | Conv4 | $37.36 \pm 0.68$ | $53.91 \pm 0.70$ |
| RP-Proto (euclidean) (1-shot C: -1) (5-shot C: -2) | ResNet10 | $39.03 \pm 0.69$ | $57.96 \pm 0.71$ |
| RP-Proto (euclidean) (C: -3) | ResNet12 | $39.73 \pm 0.69$ | $56.11 \pm 0.72$ |
| RP-Proto (euclidean) (C: -3) | ResNet18 | $38.88 \pm 0.67$ | $56.93 \pm 0.73$ |

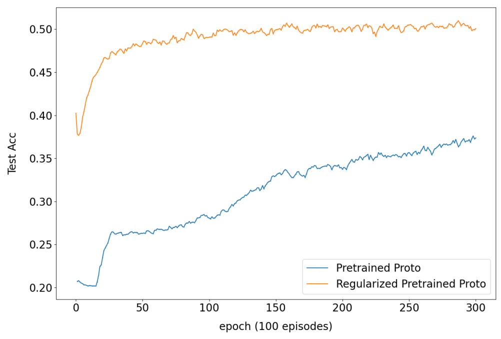
Table 5.4: 5-way 5-shot and 5-way 1-shot performance on fc100.

## 5.5 Pre-train Weight and Meta-learning Tasks

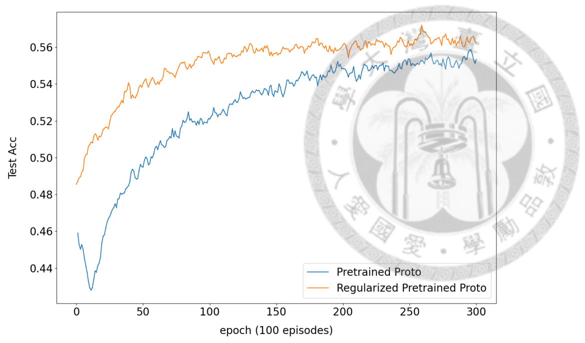
In this part, we design experiments to dig deeper why regularized pre-train may work by digging more among the relationship between pre-trained weight and the meta-learning task.

In [10], they propose soft-nearest-neighbor 5.1 loss to measure the disentanglement of data points. Moreover, they find that for the ResNet-like network, all the layers instead of the last layer won't increase the disentanglement. That is to say, most of the layers focus on capturing the common part of the image that is important for classification. And the

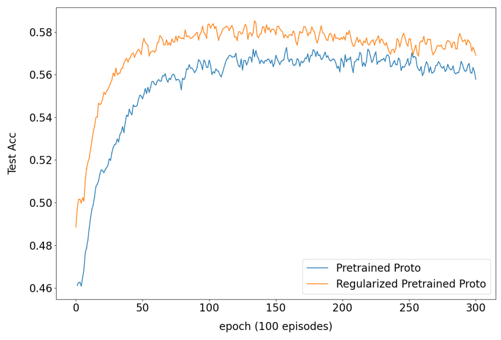
doi:10.6342/NTU202001626



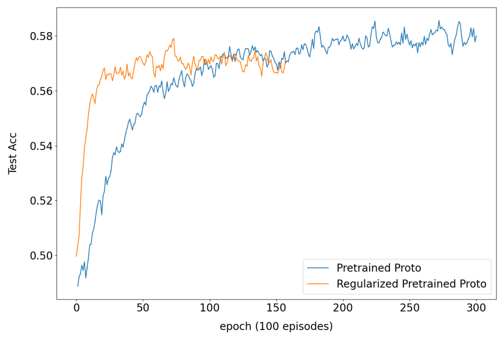
(a) Conv4



(b) ResNet10



(c) ResNet12



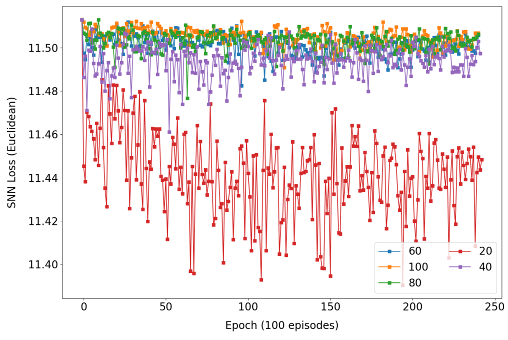
(d) ResNet18

Figure 5.2: Comparison for the converge speed between standard pre-trained and regularized pre-trained on 5-way 1-shot miniImagenet.

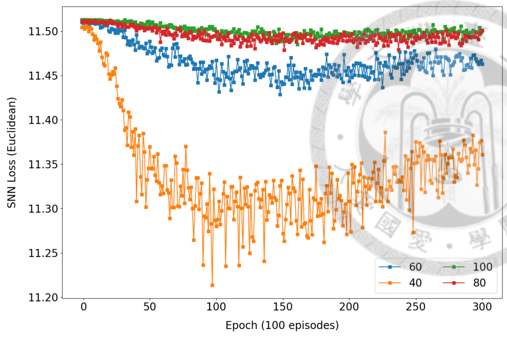
last layer does the classification.

Compared to the general classification task, the pre-trained metric-based method replace the last layer with some simple metric computation. That is to say, the metric computation works for disentangling. From this aspect, we measure the soft-nearest-neighbor loss for the backbone. In detail, we set $T = 1$ to measure the loss, though in the original approach they set $T$ as a learnable variable. However, it's pretty strange when regarding it as a metric instead of an optimization loss. From 5.3, though some of the lines decrease during episodic training, we find that those are all from low pre-trained epochs, which are not well trained on the original supervised learning task. As a result, the metric learning part has to pay effort to fit for some classification properties. However, for the well pre-trained part, the soft-nearest-neighbor loss doesn't decrease during episodic training. As a result, this may imply that the direction for meta-learning may be orthogonal to the pre-training direction. It means that it's possible to optimize both the loss together in the

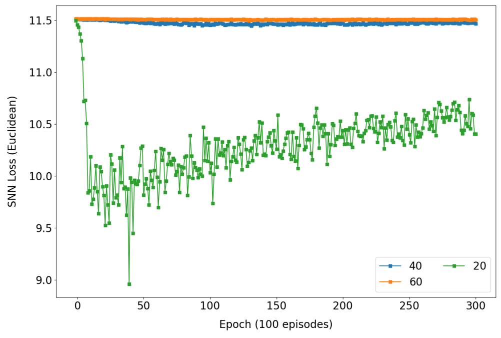
doi:10.6342/NTU202001626



(a) Conv4



(b) ResNet10



(c) ResNet18

Figure 5.3: Comparison for the SNNLoss (Euclidean) during the meta-learning process on 5-way 5-shot miniImagenet.

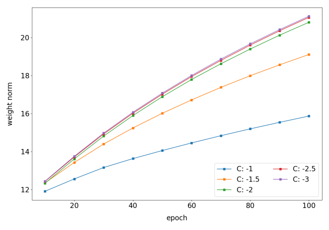
pre-training phase.

$$\textbf{Soft-Nearest-Neighbor-Loss} := -\frac{1}{b}\sum_{i\in 1..b}\frac{\sum_{j\in 1..b, j\neq i, y_i=y_j} exp(-\frac{|x_i-x_j|^2}{T})}{\sum_{k\in 1..b, k\neq i} exp(-\frac{|x_i-x_k|^2}{T})} \tag{5.1}$$

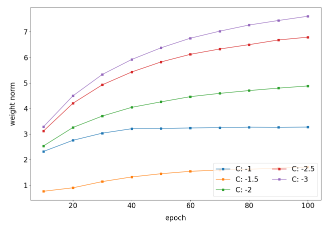
## 5.6 Relationship With Weight Scale

In this section, we analyze the sensitivity of the learning method and the weighted parameter $C$. As we have claimed in the proposed method section, when surrogate means $W_i$ is zero and the metric is Euclidean distance, it could be recognized as an l2 regularization. We plot the scale of the norm of the last Conv layer and the last batch normalization layer for all the Conv4 and ResNet10 backbone with various $C$. From Fig.5.4, a larger $C$ would lead to a smaller weight norm, which supports our claim that the term has a regulariza-

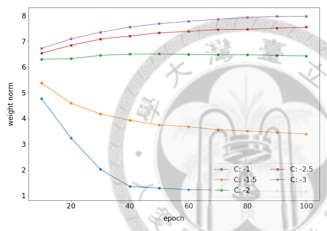
doi:10.6342/NTU202001626



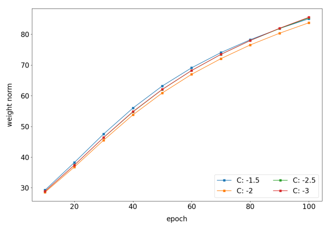
(a) ConvNet Norm



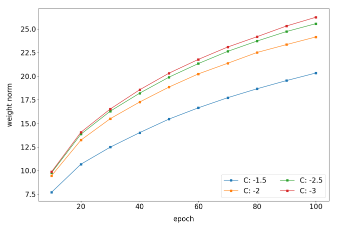
(b) BatchNorm Bias Norm



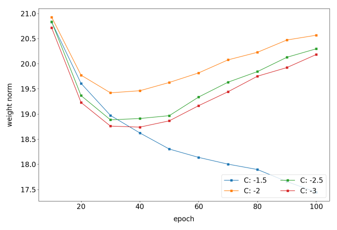
(c) BatchNorm Weight Norm



(d) ConvNet Norm



(e) BatchNorm Bias Norm



(f) BatchNorm Bias Weight

Figure 5.4: Comparison for the weight norm of each layer in Conv4 and ResNet10 on miniImagenet During Pre-training. (a)(b)(c) stands for Conv4 and (d) (e) (f) stands for ResNet10

tion effect. And for the reason why different $C$ lead to a quite different performance, we could observe the most effect takes in the weight of Batchnorm. There is a huge difference among different $C$ in the norm of batch normalization layer.

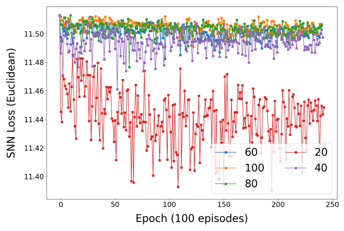
Instead of the effect on the weight norm scale, we also find that the regularization may be potentially biased. From Fig.5.4, the norm of the ConvNet in both ResNet10 and Conv4 are keep growing. A possible way to leverage the issue may be to decay on the regularization tradeoff weight $C$. However, due to the time issue, we leave it as future work.

## 5.7 Root Cause for Improvement

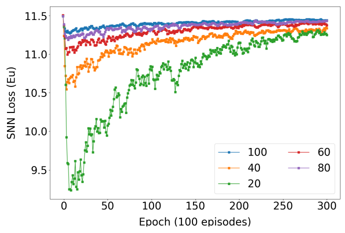
In this section, we analyze the reason why our methods could improve performance. Currently, there are two potential reasons for the improvement. First, the regularized pre-train leads the model to start in a better initialization point with higher accuracy. Second, the regularized term helps the model to start in an initialization space with better gradient property to the optimal weight parameter.

To distinguish from the two possible reasons, we analyze the accuracy and entangle-

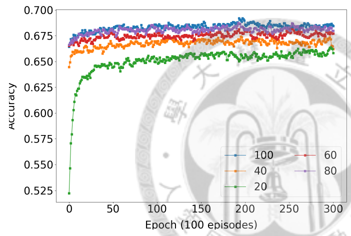
doi:10.6342/NTU202001626



(a) Origin SNN-Loss

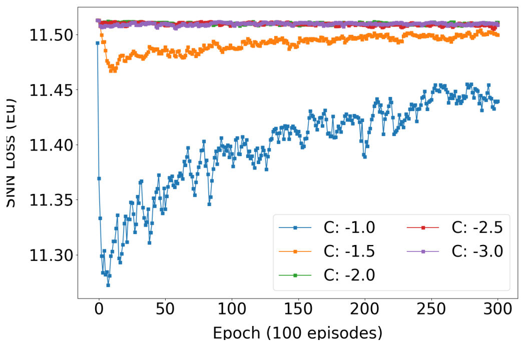


(b) RP SNN-Loss

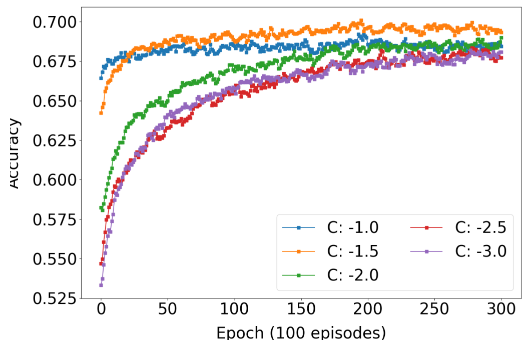


(c) Accuracy

Figure 5.5: Accuracy and SNN Loss with different pre-trained epoch during episodic training



(a) SNN-Loss



(b) Accuracy

Figure 5.6: Accuracy and SNN Loss with different weighted parameter during episodic training

ment during the episodic learning phase with different pre-trained epoch and different regularized weighted parameters.

**Regularized Pre-trained Epoch** In this part, we analyze the effect of the number of pre-trained epoch between normal pre-training (Fig. 5.5a) and regularized pre-training (Fig. 5.5b). The two plot has some similar properties. For larger pre-trained epoch the soft nearest loss would be larger and for smaller pre-trained epoch the soft nearest neighbor loss would be smaller. However, there are also some differences. Overall, the soft-nearest loss with regularized pre-training is smaller than the original one. That is to say, the regularized term pushes the embedding to disentanglement. And for the accuracy (Fig. 5.5c) among different regularized pre-trained epochs. Though the blue line, purple line, and the red line has almost the same initialized accuracy. During the episodic training, the order is always a blue line, purple line than the red one, which is also the order of the number of the pre-trained epoch. This supports the second explanation that the model is initialized

doi:10.6342/NTU202001626

in a better space that is easier to reach the optimum.

**Weighted Parameter** We plot the soft nearest loss among different weighted parameters and keeping other settings consistent. From Fig. 5.6a, we could see that larger weighted parameter leads to disentanglement. From Fig. 5.6b, we could see that there is a large gap for the initialization accuracy for different weighted parameters. However, better initialization accuracy doesn't imply a better final accuracy. For example, the orange line has a worse initialization accuracy but converges to the best weight in the end. Furthermore, we could regard the improvement as giving an extra dimension to choose the degree of disentanglement. That is to say, in the original soft-nearest work they claim that a full entangled second last layer would be the most suitable weight for the classifier. However, in metric learning with a pre-trained backbone, the situation is a little different. The layer is not behind another layer but a metric. The complexity of a metric may not be as many as a layer. Therefore, some works of disentanglement in the original last layer should be transferred to the second last layer.

doi:10.6342/NTU202001626

# Chapter 6

# Conclusion and Future Works

The regularization of the pre-trained classifier could benefit the convergence speed of the metric learning methods. That is to say, there is still room for improvement for the pre-training part of the meta-learning classifier. On the other hand, we have provided another concept to reduce the number of episodes, which is encoding the episodic-level property to the pre-training phase. Moreover, we have a detailed analysis of the regularization loss. Combining with the disentanglement theory, our approach could be viewed as an extra dimension for managing the disentanglement of the embedding. And the main contribution from the regularized loss may be a better space for optimization instead of a higher initialized point accuracy.

For future work, there are other metrics for metric learning that are not studied in this work. Analyzing different metrics in the schema and learning the relationship for different pre-training metrics and episodic learning metrics may be an interesting topic. For example, pre-training with cosine similarity regularization loss and evaluating with euclidean distance. Moreover, mutual information and self-supervised learning loss are also a good direction to study and combining with the scheme. There may be a huge potential for the schema to develop.

doi:10.6342/NTU202001626

# Bibliography

[1] D. Berthelot, N. Carlini, E. D. Cubuk, A. Kurakin, K. Sohn, H. Zhang, and C. Raffel. Remixmatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *International Conference on Learning Representations*, 2020.

[2] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. A. Raffel. Mixmatch: A holistic approach to semi-supervised learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 5049–5059. Curran Associates, Inc., 2019.

[3] L. Bertinetto, J. F. Henriques, P. Torr, and A. Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019.

[4] 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.

[5] Y. Chen, X. Wang, Z. Liu, H. Xu, and T. Darrell. A New Meta-Baseline for Few-Shot Learning. *arXiv e-prints*, page arXiv:2003.04390, Mar. 2020.

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

doi:10.6342/NTU202001626

[7] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

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

[9] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[10] N. Frosst, N. Papernot, and G. E. Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In *ICML*, pages 2012–2020, 2019.

[11] S. Gidaris and N. Komodakis. Dynamic few-shot visual learning without forgetting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[12] A. Krizhevsky, V. Nair, and G. Hinton. cifar100.

[13] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

[14] A. Nichol, J. Achiam, and J. Schulman. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999, 2018.

[15] B. Oreshkin, P. Rodríguez López, and A. Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 721–731. Curran Associates, Inc., 2018.

doi:10.6342/NTU202001626

[16] H. Qi, M. Brown, and D. G. Lowe. Low-shot learning with imprinted weights. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[17] S. Qiao, C. Liu, W. Shen, and A. L. Yuille. Few-shot image recognition by predicting parameters from activations. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[18] A. Rajeswaran, C. Finn, S. M. Kakade, and S. Levine. Meta-learning with implicit gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 113–124. Curran Associates, Inc., 2019.

[19] 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.

[20] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc., 2017.

[21] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele. Meta-transfer learning for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[22] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[23] O. Vinyals, C. Blundell, T. Lillicrap, k. kavukcuoglu, and D. Wierstra. Matching networks for one shot learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc., 2016.

doi:10.6342/NTU202001626

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

doi:10.6342/NTU202001626