# Conditional Self-Supervised Learning for Few-Shot Classification

**Yuexuan An**[1,2] , **Hui Xue**[1,2*] , **Xingyu Zhao**[1,2] and **Lu Zhang**[1,2]
[1]School of Computer Science and Engineering, Southeast University, Nanjing, 210096, China
[2]MOE Key Laboratory of Computer Network and Information Integration (Southeast University), China
{yx_an, hxue, xyzhao, lu_Zhang}@seu.edu.com

## Abstract

How to learn a transferable feature representation from limited examples is a key challenge for few-shot classification. Self-supervision as an auxiliary task to the main supervised few-shot task is considered to be a conceivable way to solve the problem since self-supervision can provide additional structural information easily ignored by the main task. However, learning a good representation by traditional self-supervised methods is usually dependent on large training samples. In few-shot scenarios, due to the lack of sufficient samples, these self-supervised methods might learn a biased representation, which more likely leads to the wrong guidance for the main tasks and finally causes the performance degradation. In this paper, we propose conditional self-supervised learning (CSS) to use prior knowledge to guide the representation learning of self-supervised tasks. Specifically, CSS leverages inherent supervised information in labeled data to shape and improve the learning feature manifold of self-supervision without auxiliary unlabeled data, so as to reduce representation bias and mine more effective semantic information. Moreover, CSS exploits more meaningful information through supervised learning and the improved self-supervised learning respectively and integrates the information into a unified distribution, which can further enrich and broaden the original representation. Extensive experiments demonstrate that our proposed method without any fine-tuning can achieve a significant accuracy improvement on the few-shot classification scenarios compared to the state-of-the-art few-shot learning methods.

## 1 Introduction

Different from previous deep learning based methods that require a large number of manually labeled training data, few-shot learning methods can mimic human to recognize new classes with very few examples. The recent work of visual few-shot learning can learn a transferable feature representation from training a collection of tasks on base classes and generalize the representation to novel (unseen) classes with very few examples [Chen *et al.*, 2019b]. However, due to the data scarcity, the obtained supervised information mainly focuses on the differences of the base class samples and ignores the semantic information within samples valuable for novel classes. Therefore, more semantic information for good representation from limited samples should be extracted for few-shot classification problems.

Self-supervised learning has emerged as an important training paradigm for exploring a strong visual representation, without relying on pre-defined annotations [He *et al.*, 2020; Chen *et al.*, 2020]. Usually, the self-supervised representation is learned from a pretext task by constructing correlated views with augmentation operations (e.g., rotation or exemplar) on original data and making a distinction between augmented views and the original one. Another method of self-supervision adopts contrastive loss, which brings representations of different views of the same data closer ("positive pairs"), and spreads representations of views from different data ("negative pairs") apart [He *et al.*, 2020].

Recently, self-supervised learning considered to offer additional semantic information is applied to the few-shot classification [Gidaris *et al.*, 2019; Su *et al.*, 2020; Lee *et al.*, 2020]. These self-supervision based few-shot learning methods use self-supervised tasks as auxiliary tasks and original few-shot classification tasks as main tasks to jointly learn a same representation. However, self-supervision is usually dependent on the availability of large training samples and unsuitable for few-shot scenarios. The direct application of previous self-supervision learning tasks for few-shot scenarios might easily fall into the learning of undesirable shortcuts [Noroozi and Favaro, 2016a], e.g., color histograms and edge continuity instead of key semantic information. Therefore, the learning of self-supervision might be biased, which might lead to the wrong guidance for the main classification tasks and finally cause the performance degradation.

In this paper, we propose conditional self-supervised learning (CSS) that can be more resilient to few-shot classification. CSS treats supervised few-shot and self-supervised tasks equally and learns two feature representations by these two paradigms respectively. For self-supervised part, CSS leverages supervised information as a teacher to guide the

*Contact Author

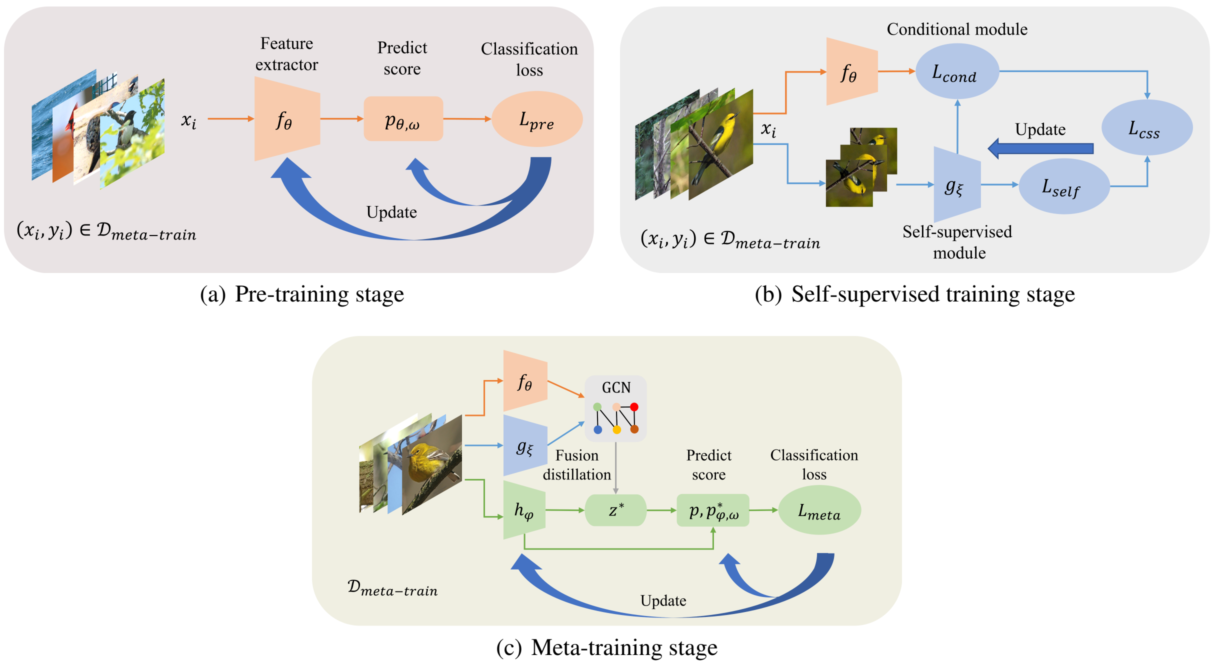

Figure 1: Architecture of conditional self-supervised learning for few-shot classification.

learning of self-supervision. Finally, all the information is integrated into a unified distribution which can further enrich and broaden the original representation. Therefore, the knowledge learned by CSS can infer other things from one fact and further improve the generalization performance. It is worth noting that our approach is fundamentally different from semi-supervised learning methods, and does not require any auxiliary unlabeled data.

The contributions of our work are:

- We explore a new pattern of self-supervision for few-shot learning. CSS learns two different representations by supervised and self-supervised learning respectively instead of the previous shared representation pattern, and fuses these representations into a new and augmented one to further enrich and broaden the semantic representation capacity for few-shot learning.
- We design a novel conditional module applied to self-supervised learning. The module can shape the learning feature manifold of self-supervision and reduce representation bias. To the best of our knowledge, it is the first time to leverage supervised information to guide the training of self-supervision in few-shot learning.
- We propose CSS with a 3-stage training pipeline: supervised pre-training stage, self-supervised training stage and meta-training stage. In the last stage, we design a novel knowledge distillation (FD) to reinforce the fusion of representations learning from the first two stages.
- We perform adequate experiments to demonstrate that our approach is totally superior to original and self-supervision-based few-shot classification methods. Systematic studies by varying the combinations of different stages are conducted, which verifies the effectiveness of the proposed three-stage training method.

## 2 Related Work

**Few-shot classification.** Few-shot classification aims to acquire transferable visual representation by learning to recognize unseen novel classes from few samples with abundant training on base classes [Chen *et al.*, 2019b]. Many efforts have been devoted to overcoming the data efficiency issue and can be mainly divided into two categories: the gradient-based model and metric-based model. The gradient-based model [Andrychowicz *et al.*, 2016; Finn *et al.*, 2017; Rusu *et al.*, 2019] can rapidly adapt a model to a given task via a small number of gradient update steps. MAML [Finn *et al.*, 2017] is a representative as the gradient-based model. This method advocates learning a suitable initialization of model parameters by learning from base classes, and transfers these parameters to novel classes in a few gradient steps. Metric-based model that leverages similarity information between samples to identify novel classes with few samples [Koch *et al.*, 2015; Vinyals *et al.*, 2016; Snell *et al.*, 2017; Sung *et al.*, 2018]. Prototypical Network [Snell *et al.*, 2017] is a representative of metric-based model. It takes the mean of support samples i.e., training samples of a class as its class prototype and classifies a query, i.e., test sample according to the distance to each class prototype. In our work, we mainly consider the classification model combined with Prototypical Networks and cosine classifiers.

**Self-supervised learning.** Without the expensive manual annotations, self-supervision learns a feature representation by constructing pseudo-labels for annotation-free pretext tasks with only input signals and learning to predict them.

The recent advances of self-supervised learning usually consist of two components: image generation and contrastive learning. The former dedicates to design the pretext tasks to exploit compact semantic visual representation such as relative rotation prediction [Chen *et al.*, 2019a], jigsaw puzzles [Noroozi and Favaro, 2016b], relative patch location prediction [Doersch *et al.*, 2015] and colorization [Zhang *et al.*, 2016]. However, these pretext tasks rely on specific transformations and heuristics and harm the generalization of learned representations [Chen *et al.*, 2020]. Contrastive learning [Chen *et al.*, 2020; Grill *et al.*, 2020] currently achieves state-of-the-art performance in self-supervised learning. Contrastive learning aims to learn the feature representation of samples by bringing the representations of positive pairs closer, and spreading representations of negative pairs apart. Wu et al. [Wu *et al.*, 2018] adopts a memory bank to store the instance class representation vector. Momentum Contrast (MoCo) [He *et al.*, 2020] trains a visual representation encoder by matching an encoded query $q$ to a dictionary of encoded keys with contrastive loss. SimCLR [Chen *et al.*, 2020] uses the normalized temperature-scaled cross-entropy loss as contrast loss. BYOL [Grill *et al.*, 2020] only relies on positive pairs rather than negative pairs to learn the feature representation. SimSiam [Chen and He, 2020] removes the momentum encoder and designs a simpler one based on BYOL. In this paper, we use the state-of-the-art SimSiam to extract semantic information of samples for simplicity and flexibility, which is more resilient to few-shot classification.

Some researchers are devoted to using self-supervision to squeeze out generalizable features and structural information from low data regimes. [Gidaris *et al.*, 2019] proposes a multi-task method combining self-supervised pretext as an auxiliary loss with the main few-shot loss. [Su *et al.*, 2020] uses self-supervised learning as a regularizer and uses importance weights to select data for self-supervision. SLA [Lee *et al.*, 2020] treats the pretext task as augmented labels and avoids the optimization of the weighted summation loss between supervised and self-supervised tasks. These methods directly apply self-supervision as an auxiliary task to the main few-shot task. However, due to limited samples, undesirable shortcuts might be learned by self-supervision, which inevitably hurts the learning of key semantic information.

Different from the previous work, we use different feature representations for different tasks. Firstly, a supervised representation is trained by the few-shot classification task. Then, we use the supervised representation as a teacher to guide the self-supervision to avoid exploiting shortcuts to solve self-supervised tasks. Finally, we learn a reinforced representation from the above representations with a novel fusing distillation method (FD). In that case, our method can benefit from these different paradigms respectively and learn a better semantic representation.

## 3 Method

### 3.1 Preliminary

Few-shot learning uses the episode training process and each episode samples a task. The meta train dataset and test dataset are represented by $\mathcal{D}_{tr} = \{\mathcal{T}_i\}_{i=1}^{I}$ and $\mathcal{D}_{te} = \{\mathcal{T}_i^*\}_{i=1}^{I^*}$ respectively, where $\mathcal{T}$ denotes the task. Each task instance contains a support set and a query set. The goal of each task is to estimate the class of samples in the query set with support set. For the meta train dataset $\mathcal{D}_{tr} = \{\mathcal{T}_i\}_{i=1}^{I} = \{S_i, Q_i\}_{i=1}^{I}$, the support set $S_i$ is composed of $N \times K$ samples that $N$ classes are randomly selected from all classes of $\mathcal{D}_{tr}$ and $K$ samples are extracted for each class. The query set $Q_i$ is composed of $N \times M$ samples that $M$ samples for each class are unseen in $S_i$. The model is trained by randomly selecting tasks from the task set $\{\mathcal{T}_i\}_{i=1}^{I}$. Similar to $\mathcal{D}_{tr}$, the test dataset $\mathcal{D}_{te} = \{\mathcal{T}_i^*\}_{i=1}^{I^*}$ is considered as a task set, from which a new task $\mathcal{T}_i^* = \{S_i^*, Q_i^*\}$ is sampled and the classes of $S_i^*$ and $Q_i^*$ are from $\mathcal{D}_{te}$ but unseen in $\mathcal{D}_{tr}$.

### 3.2 Methodology

As illustrated in Figure 1, the proposed CSS uses a 3-stage training pipeline. Firstly, in the pre-training stage, CSS learns an initial feature extractor $f_\theta(\cdot)$ through the original supervised learning method. In the next self-supervised training stage, CSS uses the learned $f_\theta$ as a prior condition to optimize the learning of self-supervision model $g_\xi(\cdot)$. In the final meta-training stage, CSS distills the knowledge of $f_\theta(\cdot)$ and $g_\xi(\cdot)$ learned in the first two stages into the final feature embedding network $h_\varphi(\cdot)$ through a novel fusion distillation method. It is worth noting that in all the three training stages, only $\mathcal{D}_{tr}$ is needed, without introducing any auxiliary datasets. In the remaining part of the section, we will describe these training stages in detail.

### 3.3 The Pre-Training Stage

In the pre-training stage as shown in Figure 1(a), feature extractor $f_\theta(\cdot)$ is learned with original few-shot classification tasks. For $N$-way $K$-shot problem, in each training episode, a few-shot task $\mathcal{T}_i = \{S_i, Q_i\}$ is sampled. Given the feature extractor $f_\theta(\cdot)$, the prototype for each class $k$ can be computed as

$$c_k = \frac{1}{\left|S_i^k\right|} \sum_{(x_t, y_t) \in S_i^k} f_\theta(x_t), \tag{1}$$

where $\left|S_i^k\right|$ denotes the number of support samples of the $k$-th class. Then, given a new sample $x_q$ from $Q_i$, the classifier outputs the normalized classification score for each class $k$

$$p_{\theta,\omega}(y = k \,|x_q, S_i) = \frac{\exp\left(sim_\omega\left(f_\theta\left(x_q\right), c_k\right)\right)}{\sum_{k'} \exp\left(sim_\omega\left(f_\theta\left(x_q\right), c_{k'}\right)\right)}, \tag{2}$$

where $sim_\omega(\cdot,\cdot)$ is a similarity function parameterized by $\omega$. In CSS, we consider the following similarity function:

$$sim_\omega(A, B) = \lambda \cos\left(F_\omega(A), F_\omega(B)\right), \tag{3}$$

where $F_\omega(\cdot)$ is a single layer neural network parameterized by $\omega$ and $\lambda$ is the inverse temperature parameter. Then CSS can update $\theta$ and $\omega$ according to the following classification loss:

$$L_{pre}(\theta, \omega) = \mathop{\mathbb{E}}_{(x_q, y_q) \in Q_i} \left[-\log p_{\theta,\omega}\left(y = y_q \,|x_q, S_i\right)\right]. \tag{4}$$

### 3.4 The Self-Supervised Training Stage

The core idea of the self-supervised training stage is to leverage the knowledge learned from the supervised pre-training stage to reduce representation bias and improve the learning of self-supervised tasks. For the above consideration, we propose the conditional self-supervision model, which consists of a self-supervised module and a conditional module. More details are illustrated in Figure 1(b).

In this stage, CSS trains the conditional self-supervision model by using all the data (removing labels) in $\mathcal{D}_{tr}$ without auxiliary unlabeled data. CSS trains a self-supervised feature extractor from scratch, rather than fine-tuning the feature extractor trained in the pre-training stage, which enables self-supervision to learn a diverse feature representation for few-shot learning.

**Self-supervised module.** We consider SimSiam [Chen and He, 2020] as a self-supervised task for simplicity and flexibility, other self-supervised methods are equally acceptable. Given a input image $x$, two augmented views $x_1$ and $x_2$ are generated according to random augment methods. The two views are processed by an encoder network consisting of the self-supervised feature extractor $g_\xi(\cdot)$ and a projection MLP head $\sigma$. A prediction MLP head, denoted as $\delta$, encode the output of one view and matches it to the other view. Representing the two output vectors as $p_1 \triangleq \delta(\sigma(g_\xi(x_1)))$ and $z_2 \triangleq \sigma(g_\xi(x_2))$, the negative cosine similarity of two embedding vectors is obtained:

$$D(p_1, z_2) = -\frac{p_1}{\|p_1\|_2} \cdot \frac{z_2}{\|z_2\|_2}, \tag{5}$$

where $\|\cdot\|_2$ is $l_2$-norm. Besides, a stop-gradient [Grill *et al.*, 2020] operation is implemented on $z_2$. Similarly, switching these two views and obtaining the distance between the two vectors, the symmetrized loss can be defined as:

$$L_{self}(x) = D(p_1, z_2) + D(p_2, z_1). \tag{6}$$

**Conditional module.** CSS takes the features learned in the pre-training stage as a prior condition to optimize the feature manifold learned in the self-supervised module. CSS minimizes the negative cosine similarity between $f_\theta(x)$ and $g_\xi(x)$ as the condition of self-supervised training:

$$L_{cond}(x) = -\frac{f_\theta(x)}{\|f_\theta(x)\|_2} \cdot \frac{g_\xi(x)}{\|g_\xi(x)\|_2}. \tag{7}$$

Then, the final loss function combining condition loss with self-supervised loss can be obtained by:

$$L_{css}(x) = \mathop{\mathbb{E}}_{x \in \mathcal{D}_{tr}} [L_{self}(x) + \gamma L_{cond}(x)], \tag{8}$$

where $\gamma$ is a positive constant trading off the importance of the self-supervised and the conditional terms of the loss.

### 3.5 The Meta-Training Stage

In the meta-training stage, shown in Figure 1(c), CSS distills the knowledge learned in the first two stages with a novel fusion distillation (FD) to improve the quality of the representation. Specifically, the knowledge extracted from $f_\theta(\cdot)$ and $g_\xi(\cdot)$ is adopted to augment the representations of samples with graph convolution network (GCN) [Kipf and Welling, 2017]. In particular, for the sample $x_i$, CSS first calculates its two corresponding embedding vectors $f_\theta(x_i)$ and $g_\xi(x_i)$, and then uses an augmentation operation $z_i = C(f_\theta(x_i), g_\xi(x_i))$ (e.g. concatenation) to connect them. By calculating $z_i$ of different samples, we can obtain the corresponding feature matrix $Z = [z_1, \cdots, z_n]^T$ of these samples, where $n$ is the number of samples. After that, we can calculate the cosine similarity between two samples and generate a graph $S$, where each vertex represents feature of a sample

$$S_{i,j} = \begin{cases} \frac{Z_{i,:} Z_{j,:}^T}{\|Z_{i,:}\|_2 \|Z_{j,:}\|_2}, i \neq j \\ 0, i = j \end{cases}, \tag{9}$$

where $S_{i,j}$ represents feature similarity between the $i$-th and $j$-th sample. Then CSS normalizes the graph matrix to obtain an adjacency matrix:

$$E = D^{-\frac{1}{2}} S D^{-\frac{1}{2}}, \tag{10}$$

where $D$ is the degree diagonal matrix and $D_{ii} = \sum_j S_{i,j}$. Inspired by [Li and Liu, 2020], CSS takes $E$ as the Laplacian matrix in GCN to aggregate information among vertices. Then for the sample $x_i$, the embedding vector of $x_i$ with FD method can be obtained by

$$z_i^* = \sum_{j=1}^{n} (\alpha I + E)_{i,j}^{\nu} h_\varphi(x_j), \tag{11}$$

where $I$ is the identity matrix, $\nu$ is the number of times for aggregating feature, $\alpha$ is a trade-off parameter to trade off the representation of neighbors and itself one. $(\alpha I + E)_{i,j}^{\nu}$ is the $j$-th element of the $i$-th row of $(\alpha I + E)^{\nu}$.

After that, for a support set $S_i$ and a query sample $x_q$, the classification score for class $k$ can be obtained:

$$p_{\varphi,\omega}^{*}(y = k | x_q, S_i) = \frac{\exp(sim_\omega(z_q^*, \mathbb{C}_k))}{\sum_{k'} \exp(sim_\omega(z_q^*, \mathbb{C}_{k'}))}, \tag{12}$$

where

$$\mathbb{C}_k = \frac{1}{|S_i^k|} \sum_{(x_t, y_t) \in S_i^k} z_t^* \tag{13}$$

is the prototype of class $k$. Besides, the classification score for class $k$ without FD can also be caculated:

$$p_{\varphi,\omega}(y = k | x_q, S_i) = \frac{\exp(sim_\omega(h_\varphi(x_q), \mathbf{C}_k))}{\sum_{k'} \exp(sim_\omega(h_\varphi(x_q), \mathbf{C}_{k'}))}, \tag{14}$$

where

$$\mathbf{C}_k = \frac{1}{|S_i^k|} \sum_{(x_t, y_t) \in S_i^k} h_\varphi(x_t). \tag{15}$$

Then the final loss is given by:

$$L_{meta}(\varphi, \omega) = \mathop{\mathbb{E}}_{(x_q, y_q) \in Q_i} [-\log p_{\varphi,\omega}^{*}(y = y_q | x_q, S_i) - \eta \log p_{\varphi,\omega}(y = y_q | x_q, S_i)] \tag{16}$$

where $\eta$ is a positive constant trading off the importance of the first and the second terms of the loss.

When predicting, CSS uses $h_\varphi$ as the feature extractor to extract the features of samples. Then CSS calculates the average feature of the support set samples corresponding to each class as a prototype of the class. Then the similarity between the query sample and the prototype of each class is calculated by the similarity function $sim_\omega(\cdot, \cdot)$. Finally, the class with the highest similarity score is obtained as the classification result.

## 4 Experiments

In this section, the effectiveness of CSS is verified by various experiments. Three standard few-shot classification datasets (CIFAR-FS [Bertinetto *et al.*, 2019], CUB-200 [Wah *et al.*, 2011], *mini*-ImageNet [Vinyals *et al.*, 2016]) are selected to compare the performance of our approach with previous few-shot learning methods. All the computations are performed on a GPU server with NVIDIA TITAN RTX, Intel Core i7-8700 CPU 3.20GHz processor and 32 GB memory.

### 4.1 Implementation Details

For the fair comparison, we implement all methods by PyTorch. Since [Chen *et al.*, 2019b] shows that the gap among different methods drastically reduces as the backbone gets deeper, all algorithms use the Conv-4-64 [Vinyals *et al.*, 2016] as the backbone and the feature embedding dimension is set to 1600. In CSS, $f_\theta$, $g_\xi$ and $h_\varphi$ are implement by the Conv-4-64 backbone, and the output dimension of $F_\omega$ is 2048. The projection MLP head $\sigma$ is a composite function, which is composed of $F_\omega$ and a multi-layer neural network with $\{1600, 2048, 2048\}$ units and batch normalization in each layer. The prediction MLP head $\delta$ is parameterized by a three-layer neural network with 512 hidden units and batch normalization in the hidden layer. All hidden layers use ReLU function [Glorot *et al.*, 2011] as the activation function.

### 4.2 Comparison with Prior Work

In order to verify the effectiveness of CSS, several competitive and state-of-the-art fully supervised few-shot classification methods are compared, including Baseline [Chen *et al.*, 2019b], Baseline++ [Chen *et al.*, 2019b], MAML [Finn *et al.*, 2017], Matching networks [Vinyals *et al.*, 2016], Prototypical networks [Snell *et al.*, 2017], Relation networks [Sung *et al.*, 2018], FEAT [Ye *et al.*, 2020] and Deep Kernel Transfer (DKT) [Patacchiola *et al.*, 2020]. Besides, we also compare the performance of CSS with several state-of-the-art self-supervision-based few-shot learning methods. ProtoTransfer [Medina *et al.*, 2020] pre-trains a representation on self-supervision and adapts it to original few-shot classification tasks. CC+rot, PN+rot, SLA and ProtoNet+rot [Gidaris *et al.*, 2019; Lee *et al.*, 2020; Su *et al.*, 2020] are multi-task few-shot learning and its varient methods.

We perform experiments to illustrate the average classification accuracies over 600 test episodes with 95% confidence intervals of different methods on CIFAR-FS, CUB-200 and *mini*-ImageNet datasets under 5-way 5-shot and 1-shot settings respectively. For each setting, the best result is highlighted in bold. CSS (pre-training), CSS(SSL-training) and CSS(meta-training) compose the 3-stage training pipeline proposed by us, which represent the classification model of CSS using $f_\theta$, $g_\xi$ and $h_\varphi$ as feature extractors respectively.

| Mehods | 5-shot | 1-shot |
|---|---|---|
| Baseline [Chen *et al.*, 2019b] | 68.43±0.73 | 47.01±0.78 |
| Baseline++ [Chen *et al.*, 2019b] | 72.85±0.72 | 51.55±0.80 |
| ProtoNet. [Snell *et al.*, 2017] | 68.78±0.77 | 43.65±0.86 |
| MatchingNet. [Vinyals *et al.*, 2016] | 68.93±0.74 | 51.32±0.85 |
| RelationNet. [Sung *et al.*, 2018] | 65.01±0.79 | 46.76±0.86 |
| MAML [Finn *et al.*, 2017] | 70.30±0.77 | 52.42±0.91 |
| FEAT [Ye *et al.*, 2020] | 69.39±0.77 | 50.69±0.86 |
| DKT [Patacchiola *et al.*, 2020] | 67.81±0.73 | 50.94±0.85 |
| ProtoTransfer [Medina *et al.*, 2020] | 67.95±0.76 | 42.46±0.77 |
| CC+rot [Gidaris *et al.*, 2019] | 69.75±0.76 | 52.02±0.87 |
| PN+rot [Gidaris *et al.*, 2019] | 70.52±0.72 | 48.53±0.88 |
| SLA [Lee *et al.*, 2020] | 68.62±0.75 | 45.94±0.87 |
| ProtoNet+rot [Su *et al.*, 2020] | 69.46±0.75 | 46.81±0.87 |
| CSS(pre-training) | 70.64±0.72 | 51.07±0.89 |
| CSS(SSL-training) | 69.96±0.76 | 51.48±0.89 |
| CSS(meta-training) | **74.59±0.72** | **56.49±0.93** |

Table 1: Comparison with prior work on CIFAR-FS. Average 5-way accuracies (%) with 95% confidence intervals (600 episodes).

| Mehods | 5-shot | 1-shot |
|---|---|---|
| Baseline [Chen *et al.*, 2019b] | 68.30±0.68 | 46.09±0.70 |
| Baseline++ [Chen *et al.*, 2019b] | 79.21±0.64 | 60.34±0.90 |
| ProtoNet. [Snell *et al.*, 2017] | 74.68±0.70 | 51.47±0.88 |
| MatchingNet. [Vinyals *et al.*, 2016] | 74.88±0.66 | 60.08±0.89 |
| RelationNet. [Sung *et al.*, 2018] | 77.45±0.64 | 62.14±0.94 |
| MAML [Finn *et al.*, 2017] | 76.71±0.69 | 61.73±0.95 |
| FEAT [Ye *et al.*, 2020] | 80.73±0.60 | 64.82±0.90 |
| DKT [Patacchiola *et al.*, 2020] | 74.57±0.68 | 53.56±0.91 |
| ProtoTransfer [Medina *et al.*, 2020] | 75.14±0.66 | 33.28±0.58 |
| CC+rot [Gidaris *et al.*, 2019] | 75.89±0.71 | 61.65±0.91 |
| PN+rot [Gidaris *et al.*, 2019] | 76.43±0.71 | 49.06±0.86 |
| SLA [Lee *et al.*, 2020] | 71.30±0.72 | 48.43±0.82 |
| ProtoNet+rot [Su *et al.*, 2020] | 75.18±0.66 | 47.37±0.83 |
| CSS(pre-training) | 77.96±0.65 | 58.09±0.89 |
| CSS(SSL-training) | 77.07±0.70 | 57.21±0.85 |
| CSS(meta-training) | **81.84±0.59** | **66.01±0.90** |

Table 2: Comparison with prior work on CUB-200. Average 5-way accuracies (%) with 95% confidence intervals (600 episodes).

From Tables 1, 2 and 3, we can find that in most of the few-shot classification settings, CSS already achieves almost the same performance as state-of-the-art methods only through the pre-training stage and self-supervised training stage, while the meta-learning stage can further improve the accuracy of our model. In all cases, after the meta-training finishes, we achieve state-of-the-art results surpassing previous methods with a significant margin. For instance, on CIFAR-FS, CUB-200, and *mini*-ImageNet datasets, CSS has around 6%, 7% and 4% performance improvements compared with the vanilla prototypical networks under the 5-shot setting, while it has 13%, 15% and 6% performance improvements under the 1-shot setting. At the same time, in all set-

| Mehods | 5-shot | 1-shot |
|---|---|---|
| Baseline [Chen *et al.*, 2019b] | 60.78±0.67 | 39.96±0.69 |
| Baseline++ [Chen *et al.*, 2019b] | 67.07±0.66 | 47.89±0.73 |
| ProtoNet. [Snell *et al.*, 2017] | 64.69±0.69 | 44.42±0.75 |
| MatchingNet. [Vinyals *et al.*, 2016] | 62.75±0.75 | 47.06±0.73 |
| RelationNet. [Sung *et al.*, 2018] | 63.96±0.69 | 48.11±0.82 |
| MAML [Finn *et al.*, 2017] | 63.37±0.70 | 47.44±0.80 |
| FEAT [Ye *et al.*, 2020] | 67.07±0.71 | 47.44±0.72 |
| DKT [Patacchiola *et al.*, 2020] | 62.43±0.72 | 46.89±0.74 |
| ProtoTransfer [Medina *et al.*, 2020] | 63.47±0.68 | 38.20±0.73 |
| CC+rot [Gidaris *et al.*, 2019] | 64.71±0.68 | 48.19±0.77 |
| PN+rot [Gidaris *et al.*, 2019] | 64.66±0.68 | 47.46±0.79 |
| SLA [Lee *et al.*, 2020] | 63.32±0.68 | 44.95±0.79 |
| ProtoNet+rot [Su *et al.*, 2020] | 64.77±0.69 | 45.78±0.77 |
| CSS(pre-training) | 65.01±0.73 | 48.31±0.78 |
| CSS(SSL-training) | 65.25±0.69 | 48.03±0.84 |
| CSS(meta-training) | **68.08±0.73** | **50.85±0.84** |

Table 3: Comparison with prior work on *mini*-ImageNet. Average 5-way accuracies (%) with 95% confidence intervals (600 episodes).

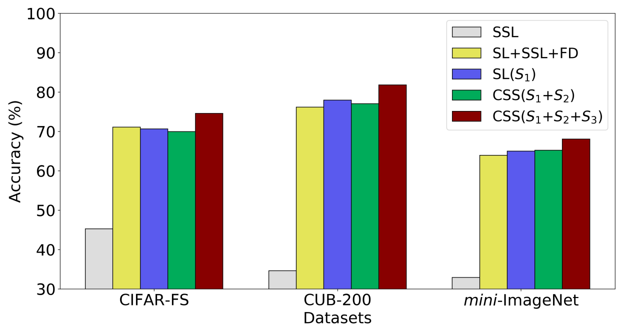

Figure 2: Classification results in different cases (5-way 5-shot).

tings, our approach outperforms around 2% to 5% higher than all previous leading methods.

### 4.3 Ablation Study

Now we explore the importance of the condition module in self-supervised learning and the effects of different stages. We design five cases respectively to investigate the performance when applying different combinations of stages. Then, we test the performance of these methods in the standard few-shot classification settings. To be brief, we use $S_1$, $S_2$ and $S_3$ to represent CSS(pre-training), CSS(SSL-training) and CSS(meta-training) respectively.

- SSL: Discard $S_1$ and $S_3$ and replace $S_2$ with a vanilla self-supervised learning without a conditional module.
- SL+SSL+FD: Change $S_2$ and abandon the conditional module in our self-supervised learning method.
- SL ($S_1$): Only execute supervised learning, i.e., using $f_\theta$ in CSS as the feature extractor for classification.
- CSS ($S_1$+$S_2$): execute the first two stages without $S_3$, i.e., using $g_\xi$ in CSS as the feature extractor.
- CSS ($S_1$+$S_2$+$S_3$): Execute the whole three stages of CSS, using $h_\varphi$ as the feature extractor for classification.

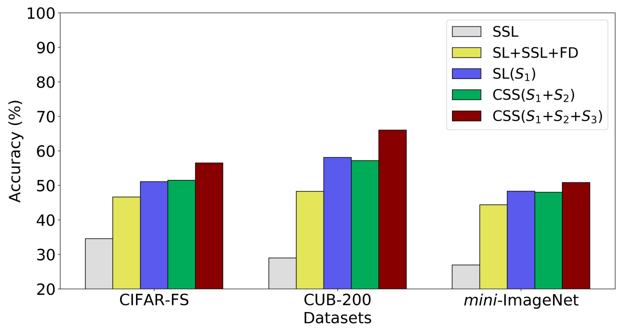

Figure 3: Classification results in different cases (5-way 1-shot).

The classification performance in different cases are shown in Figures 2 and 3. From Figures 2 and 3, we find that the performance of vanilla self-supervised learning (SSL) degrades significantly compared with other settings, which proves it difficult to study an effective representation in the absence of supervised learning with limited samples. The combination of supervised learning, self-supervised learning and fusion distillation (SL+SSL+FD) can improve the classification accuracy compared with SSL. However, the performance of SL+SSL+FD almost can not surpass the supervised learning model (SL) and the negative impact of SSL with limited samples still continues. In addition, by comparing the results of CSS($S_1$+$S_2$) with SSL and SL, we find that the conditional module can significantly improve the performance of self-supervised learning, which can achieve comparable results with the supervised model (SL). Furthermore, CSS with entire process can achieve an obvious performance improvement and surpass other cases with large margin. The experiments illustrate that the condition module plays a crucial role in self-supervision and the effective feature fusion can further improve the model performance.

## 5 Conclusion

In this work, we propose conditional self-supervised learning (CSS) with a 3-stage training pipeline: the supervised pre-training stage, the self-supervised training stage and the meta-training stage, and each training stage is conducive to the improvements of the model performance. For the self-supervised training stage, CSS leverages the supervised information learned by the pre-training stage to guide the learning of self-supervision, thus more resilient to low data regimes. For the meta-train stage, CSS utilizes a novel fusion distillation (FD) to integrate the information from the first two stages into a unified distribution so as to enrich and broaden the original representation. Detailed experiments reveal that our approach indeed leads to significant performance improvements over recent approaches and achieves state-of-the-art results.

## Acknowledgments

This work was supported by National Natural Science Foundation of China (Grant No.62076062) and National Key R&D Program of China (Grant No.2017YFB1002801). Furthermore, the work was also supported by Collaborative Innovation Center of Wireless Communications Technology.

## References

[Andrychowicz *et al.*, 2016] Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, pages 3981–3989, 2016.

[Bertinetto *et al.*, 2019] Luca Bertinetto, João F. Henriques, Philip H. S. Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019.

[Chen and He, 2020] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *CoRR*, abs/2011.10566, 2020.

[Chen *et al.*, 2019a] Ting Chen, Xiaohua Zhai, Marvin Ritter, Mario Lucic, and Neil Houlsby. Self-supervised gans via auxiliary rotation loss. In *CVPR*, pages 12154–12163, 2019.

[Chen *et al.*, 2019b] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. In *ICLR*, 2019.

[Chen *et al.*, 2020] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, volume 119, pages 1597–1607, 2020.

[Doersch *et al.*, 2015] Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, pages 1422–1430, 2015.

[Finn *et al.*, 2017] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, volume 70, pages 1126–1135, 2017.

[Gidaris *et al.*, 2019] Spyros Gidaris, Andrei Bursuc, Nikos Komodakis, Patrick Pérez, and Matthieu Cord. Boosting few-shot visual learning with self-supervision. In *ICCV*, pages 8058–8067, 2019.

[Glorot *et al.*, 2011] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *AISTATS*, volume 15, pages 315–323, 2011.

[Grill *et al.*, 2020] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent - A new approach to self-supervised learning. In *NeurIPS*, 2020.

[He *et al.*, 2020] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9726–9735, 2020.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[Koch *et al.*, 2015] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, 2015.

[Lee *et al.*, 2020] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Self-supervised label augmentation via input transformations. In *ICML*, volume 119, pages 5714–5724, 2020.

[Li and Liu, 2020] Jianyi Li and Guizhong Liu. Few-shot image classification via contrastive self-supervised learning. *CoRR*, abs/2008.09942, 2020.

[Medina *et al.*, 2020] Carlos Medina, Arnout Devos, and Matthias Grossglauser. Self-supervised prototypical transfer learning for few-shot classification. *CoRR*, abs/2006.11325, 2020.

[Noroozi and Favaro, 2016a] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, volume 9910, pages 69–84, 2016.

[Noroozi and Favaro, 2016b] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, volume 9910, pages 69–84, 2016.

[Patacchiola *et al.*, 2020] Massimiliano Patacchiola, Jack Turner, Elliot J. Crowley, Michael F. P. O'Boyle, and Amos J. Storkey. Bayesian meta-learning for the few-shot setting via deep kernels. In *NeurIPS*, 2020.

[Rusu *et al.*, 2019] Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

[Snell *et al.*, 2017] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087, 2017.

[Su *et al.*, 2020] Jong-Chyi Su, Subhransu Maji, and Bharath Hariharan. When does self-supervision improve few-shot learning? In *ECCV*, volume 12352, pages 645–666, 2020.

[Sung *et al.*, 2018] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, pages 1199–1208, 2018.

[Vinyals *et al.*, 2016] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, pages 3630–3638, 2016.

[Wah *et al.*, 2011] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[Wu *et al.*, 2018] Zhirong Wu, Yuanjun Xiong, Stella X. Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pages 3733–3742, 2018.

[Ye *et al.*, 2020] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Few-shot learning via embedding adaptation with set-to-set functions. In *CVPR*, pages 8805–8814, 2020.

[Zhang *et al.*, 2016] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *ECCV*, volume 9907, pages 649–666, 2016.