GIFSL - Grafting based Improved Few-Shot Learning

Pratik Mazumder[1]*, Pravendra Singh[1], Vinay P. Namboodiri[1,2]

[1]*Department of Computer Science and Engineering, Indian Institute of Technology Kanpur, Kanpur, India*
[2]*University of Bath, United Kingdom*

Abstract

A few-shot learning model generally consists of a feature extraction network and a classification module. In this paper, we propose an approach to improve few-shot image classification performance by increasing the representational capacity of the feature extraction network and improving the quality of the features extracted by it. The ability of the feature extraction network to extract highly discriminative features from images is essential to few-shot learning. Such features are generally class agnostic and contain information about the general content of the image. Our approach improves the training of the feature extraction network in order to enable them to produce such features. We train the network using filter-grafting along with an auxiliary self-supervision task and a knowledge distillation procedure. Particularly, filter-grafting rejuvenates unimportant (invalid) filters in the feature extraction network to make them useful and thereby, increases the number of important filters that can be further improved by using self-supervision and knowledge distillation techniques. This combined approach helps in significantly improving the few-shot learning performance of the model. We perform experiments on several few-shot learning benchmark datasets such as mini-ImageNet, tiered-ImageNet, CIFAR-FS, and FC100 using our approach. We also present various ablation studies to validate the proposed approach. We empirically show that our approach performs better than other state-of-the-art few-shot learning methods.

Keywords: Few-Shot Learning, Grafting, Self-Supervision, Distillation, Deep

*Corresponding author.
Email addresses: `pratikm@iitk.ac.in` (Pratik Mazumder[1]), `psingh@iitk.ac.in` (Pravendra Singh[1]), `vinaypn@iitk.ac.in` (Vinay P. Namboodiri[1,2])

Learning, Object Recognition

1. Introduction

Deep learning techniques are now used to tackle several types of problems. They have become very popular because they achieve high performances for various tasks. They have even surpassed human performance in many scenarios. However, current deep learning techniques are still not human-like. Deep learning methods generally require training a neural network using a large amount of labeled data. In the presence of limited labeled data, they generally do not perform well. This hunger for training data is not a trivial problem since data, specially labeled data, is not always available and is usually very costly to obtain. It may also be the case that even if sufficient data is available for a few categories, some categories of data may have extremely few samples available for training. Humans, on the other hand, can learn a new category from very few examples. We can learn what a cat looks like from a few pictures and then identify them in the wild with very high accuracy. It has been the goal of researchers to enable deep learning networks to achieve this capability. Few-shot learning is an approach in this direction.

Few-shot learning methods [1, 2, 3] employ special training techniques for deep networks that enable them to perform relatively well even for categories that have very few training samples. Generally, these methods aim to transfer the knowledge gained by training the network on the classes with many training samples to help classify classes with very few training samples. They also aim to make the networks generic enough such that they can even extract good features from images belonging to categories that they were not trained on.

Few-shot learning techniques generally consider an episodic framework for the few-shot learning problem, i.e., the networks operate on a small episode at a time (Fig. 1). An episode can be thought of as a mini-dataset with a small set of classes. Each class has a few labeled examples that are known as support examples. Most few-shot methods train and test on episodes [1, 4]. However, some methods can be modified to only carry out the testing using episodes while the network can be trained on the full train set [2].

Prototypical network [2] finds a prototype embedding for each class in the episode and uses the nearest neighbor classification technique to determine the nearest class prototype for each query example. MAML [4] trains the few-shot classifier in such a way that it can adapt within a few iterations to a new task or set of classes. TADAM [5] learns an embedding that best represents the current episode and uses it as an attention to produce better features for the examples in the episode. MetaOptNet [6] focuses on learning features that are more compatible with linear classifiers.

Most few-shot learning methods use a feature extraction network that extracts useful features from the images and a classification module that performs the few-shot classification. For the classification module to be successful, the feature extraction network should produce good features/representation for the input image. Our proposed method improves the training process of the feature extraction module to enable it to extract better and more discriminative features from images.

In the feature extraction network, not all filters contribute substantially to the output representation produced by it. Filters that do not significantly affect the output representation can be thought of as unimportant/invalid filters [7]. Therefore, the representational capacity of the network is determined by the quantity of important/valid filters. Some methods discard the invalid filters [8, 9], but this will not improve the representational capacity of the network. [10] proposes a filter-grafting technique that converts invalid filters into valid ones by grafting valid filters weights on to these invalid filters. This process will increase the number of valid filters in the feature extraction network and hence improve its representational capacity.

The strength of a feature extraction network lies in the quality of intermediate representation extracted by its filters. Therefore, improving the quality of representation produced by the network will improve the performance of the network on few-shot learning. This improvement can be achieved by using techniques such as self-supervision and knowledge distillation. Self-supervised learning [11, 12, 13, 14, 15] involves training the network on labels that are generated from the training data itself. Self-supervision methods are used to improve the discriminative powers of networks by training them on artificial tasks that force the network to learn more about the struc-

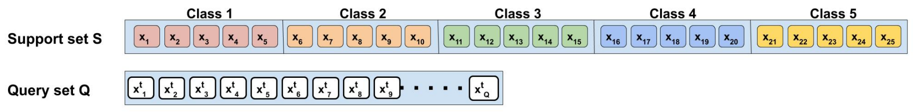

Figure 1: Episodic setup for few-shot learning. Figure depicts a 5-way 5-shot episode. Each episode consists of 5 classes with 5 support examples each and multiple query examples that belong to one of these 5 classes.

ture of the input data. In [16], the authors use self-supervision as an auxiliary loss to improve few-shot classification. Knowledge distillation [17] is a popular method for transfer learning. It can also be used to train a student network with the same architecture as the teacher, and this helps the student network learn some more general features that help improve its performance.

The important filters will mainly contribute to the quality of the representation produced by the feature extraction network. Since filter-grafting increases the proportion of important filters in the network, it will help boost the effectiveness of applying self-supervision and knowledge distillation to the feature extraction module. This process will significantly improve the performance of the feature extraction module.

Our proposed approach combines the techniques of filter-grafting, self-supervision, and knowledge distillation to improve the training of the feature extraction model. We use the filter-grafting setting that grafts filter from another network [10]. For this setting, we train two models with the same architecture while performing grafting of important parameters/filters weights into unimportant ones from one network to the other. This increases the proportion of important/useful parameters in both the networks. We also use self-supervision through rotation [16] as an auxiliary task in parallel to the classification training, in order to improve the networks' discriminative power. Finally, we choose one of these networks to perform distillation to another network with the same architecture in order to obtain a network with better discriminative powers. The trained feature extraction network is used to extract features for the support examples for a class. The extracted features of the support examples are used to learn a linear classification model, which is then used to classify the query examples. Through our ablation experiments, we show that each component of our method helps in improving the network performance. Our method is described in detail in Sec. 3.

We perform experiments on several few-shot learning benchmark datasets and compare our method to existing methods for few-shot classification. We empirically show that our method performs better than existing methods on few-shot learning.

Our contributions are as follows:

- We propose a novel approach to few-shot learning that uses filter-grafting to improve the representational capacity of the feature extraction network, which is then exploited by using a self-supervision auxiliary task and knowledge distillation to improve the discriminative power of the network.
- We empirically show that our network performs better than existing methods on several few-shot learning benchmark datasets.

2. Background

2.1. Few-shot Learning

Deep learning generally requires a large amount of labeled data for training networks. However, there are many real-life scenarios where labeled data is very scant. Few-shot learning is used to train networks that perform well under such circumstances. Many approaches to this problem have been explored by researchers [2, 18, 19, 20, 21, 22, 23, 24].

The work in [25] trains a siamese network to find an embedding space where similar images are closer to each compared to dissimilar images. Prototypical networks [2] is a very popular few-shot learning method. It first computes the class representatives or "prototypes" by averaging the extracted features of the support examples of each class. Then, for each query image feature/embedding, the nearest class prototype is predicted to be the output class. In [26], the authors modify the prototypical network to work in a semi-supervised setup where it makes use of labeled and unlabeled examples in each episode. MAML [4] trains the network to adapt to a new episode within a few training iterations quickly. In [27], the authors propose to achieve rapid generalization by shifting inductive biases via fast parameterization. The method proposed in [28] applies a graph neural network architecture to the few-shot learning problem.

RelationNet [29] learns to predict and use relation scores between query images and support images of the classes. TADAM computes a task-based embedding that represents the current task/episode. This embedding is used as an attention to the convolutional layers of the feature extraction network to modify the image embeddings in such a way that best suits the classification process in that episode.

Learning without forgetting [30] learns a weight generator for the few-shot classifier. The weight generator uses classifier weights of the base classes and support examples of the novel classes to generate weights for the classifier. R2D2 [3] proposes to use fast convergent methods like ridge regression as the main adaptation mechanism for few-shot learning. LEO [31] learns an embedding of model parameters, and in this parameter space, it performs optimization-based meta-learning. SNAIL [32] proposes a simple meta-learner architecture that combines temporal convolutions and soft attention. The method proposed in [33] predicts parameters from activations in order to adapt a pre-trained network to new classes.

MetaOptNet [6] proposes to use linear predictors to learn better representations for few-shot learning. In [16], the authors use a self-supervision task as an auxiliary task while training the network for few-shot learning. This helps the feature extraction network in learning better representations, leading to improved performance. TPN [34] proposes a transductive inference setting where the entire test set is classified at once, and a graph-based module is employed to utilize the structure of the test data.

The method proposed in [35] uses conditional Wasserstein Generative Adversarial Networks (cWGAN) to hallucinate discriminative features. In [36], the authors propose to extract latent information from the base classes and combine them with features support examples to generate a diverse set of features for the novel classes. The work in [37] develops a variational metric scaling framework for learning a metric scaling parameter, which boosts the performance of metric-based meta-learning algorithms.

2.2. Grafting

Deep neural networks contain unimportant filters that do not contribute significantly to the network output, as evident in [8, 9]. In order to make maximum use of all the filters available in the network, we can use filter-grafting that has been proposed in [10]. Filter-grafting changes the filter weights of the unimportant/invalid filters during training, in order to re-initialize them and eventually make them valid. This will increase the representational capacity of the network.

In order to find the invalid/unimportant filters, an entropy-based selection criterion has been proposed in [10], which considers filters having weights with low entropy as unimportant. Other methods can also be used to find unimportant filters such as $l1-$norm used in [7, 38], which considers filters having weights with a low $l1-$norm as unimportant filters.

Since filter-grafting involves re-initializing the weights of the invalid filter during training, a simple method would be to use a Gaussian noise to modify it. However, this may make the network harder to converge. Another option is to use valid filters from the same network as a source to graft into invalid filters. However, [10] shows that this does not introduce new information to the network and might not end up being very useful.

Another technique is to graft valid filters from another network into the invalid filters in the current network. This process will not suffer from poor convergence like the noise-based model and will also bring in new information as compared to the same network filter-based model. This, however, needs the two models to be trained in parallel. This method is proposed in [10], and it performs grafting layer-wise to maintain layer-wise consistency, i.e., all filters of a layer in a network are grafted on to the same layer in the other network.

However, if the filter is entirely replaced, then some base information may also be lost. Therefore, [10] proposes to perform a weighted addition of the filter weights from the two networks. The weights are determined by the information content of the filter/layer. The higher the information content of the filter/layer of the network, the more weight is given to it, in order to preserve the information of the network.

2.3. Self-Supervised Learning

Self-supervised learning involves training networks in the absence of labeled data. Instead of real labels, such methods use synthesized labels. These labels are generated using basic knowledge about the data that is readily available. The aim of such training is that in the process of learning from such labels, the network should learn about the structure and basic content of the data. Self-supervision helps the network to learn good features and also perform well on downstream tasks. The term self-supervision is used since the supervising signal, i.e., the synthesized label, has been generated from the same set of data.

There have been many works related to self-supervised learning. Image in-painting/ completion was proposed in [11], wherein the network learns to predict missing patches inside images. The methods in [12, 13] focus on using image colorization as a self-supervision signal. The work in [14] proposes a method in which a network is given as input two patches of the same image, and the network has to learn to predict the relative position of one patch with respect to the other. In [15], the authors propose to train a network to rearrange and solve a jigsaw puzzle.

The method proposed in [39] uses rotation angle prediction as a self-supervision task. Images are rotated by a fixed set of angles, and the network has to be trained to predict the angle of rotation of the rotated images. Surrogate classes are created in [40] by modifying images using different transformations, and the network is trained to predict the class of the network. The method proposed in [41] forces the network to learn additional discriminative features in addition to predicting the rotation angle. It achieves this by training the network to additionally bring different versions of the same image, that differ in their rotation angle, closer in the feature space. The authors claim that this helps the network to produce features that are better at instance-level discrimination.

The method proposed in [42] uses self-supervised learning to train generative adversarial networks. Besides images, self-supervision is also being used to train deep neural networks for videos. Some techniques that are used for this include using self-motion of moving objects in videos [43, 44], temporal coherence [45, 46], and even ambient sounds [47].

2.4. Distillation

Knowledge distillation involves training a student network in such a way that the probability distribution of the soft labels produced by it matches that of the teacher network. Knowledge distillation has been used extensively in transfer learning and has also provided improvements in this area. The authors in [17], show that by distilling knowledge from an ensemble model into a single model, the performance of the smaller single model can be improved significantly.

The Jacobian matrices of the teacher and student network are matched in [48]. The method proposed in [49] minimizes the Maximum Mean Discrepancy (MMD) between the distributions of neuron selectivity patterns of the teacher and student networks. Similarly, [50] uses gram matrix for this purpose. In [51], the authors analyze how self-distillation, i.e., knowledge distillation between two networks having the same architecture, results in improved performance of the network.

3. Method

3.1. Problem Setting

In the few-shot learning setting, the train and test splits of the dataset have a disjoint set of classes. The train classes are referred to as base classes, and the test classes are referred to as novel classes. In this setting, networks operate on data in the form of episodes. An episode can be thought of as a mini-dataset, and it consists of a mini-train set and a mini-test set. Each episode can have data points from a fixed small set of N classes (N-way), and each class can have only K labeled examples (K-shot). Therefore, each episode is referred to as an N-way K-shot episode (Fig. 1). The K labeled examples for each class are known as support examples i.e. $S = \{(x_1, y_1), (x_2, y_2), (x_3, y_3)...(x_{K\times N}, y_{K\times N})\}$ where x_i refer to images and $y_i \in \{1..N\}$ refer to labels. The mini-test set consists of query/test examples that also belong to one of the N classes i.e. $Q = \{(x^t_1, y^t_1), (x^t_2, y^t_2), (x^t_3, y^t_3)...(x^t_q, y^t_q)\}$, where t denotes test examples. In this setting, the objective of any method is to classify the query data points in the episode, using the few support examples per class present in the episode.

3.2. Training setup

The few-shot model that is to be trained is divided into the feature extraction network E and the few-shot classification module C^{few}. During training, we do not use

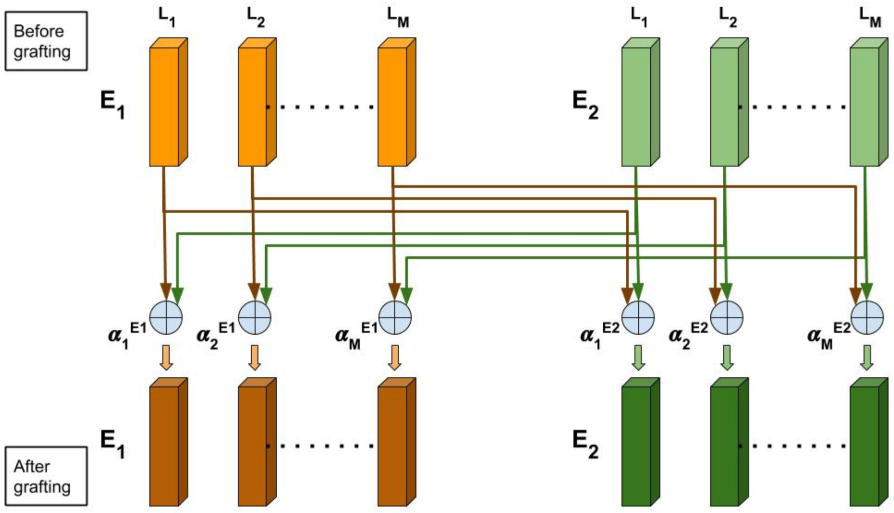

Figure 2: Mutual Filter-Grafting between two feature extraction networks E_1 and E_2 with the same architecture. L_i represents the i^{th} layer in each network and $i \in [1, M]$. M is the total number of layers in E_1 and E_2. A weighted addition of the filter weights of layer i in E_1 and the filter weights of the same layer in E_2 is carried out using the contribution weight $\alpha_i^{E_1}$, to give the grafted weight of layer i in E_1. The same process is repeated for E_2 using layer specific $\alpha_i^{E_2}$. After grafting, proportion of valid filters get increased in both networks.

C^{few} and instead use a fully-supervised classification module C. We also use a rotation classification module R.

3.3. *Filter Grafting*

Filter grafting rejuvenates invalid filters in deep neural networks. For performing filter-grafting, we train two networks simultaneously for classification on the full train dataset. We graft all the filters in a layer of the network so that layer-wise consistency is maintained as proposed in [10]. Let E_1 and E_2 be the two feature extraction networks. Let $W_i^{E_1}$ and $W_i^{E_2}$ be the weights of the i^{th} layer of E_1 and E_2 respectively. Let $W_i^{E_1*}$ and $W_i^{E_2*}$ denote the weights of the i^{th} layer in E_1 and E_2 after grafting. The grafting process can be represented as follows:

$$W_i^{E_1*} = \alpha_i^{E_1} W_i^{E_1} + (1 - \alpha_i^{E_1}) W_i^{E_2} \tag{1}$$

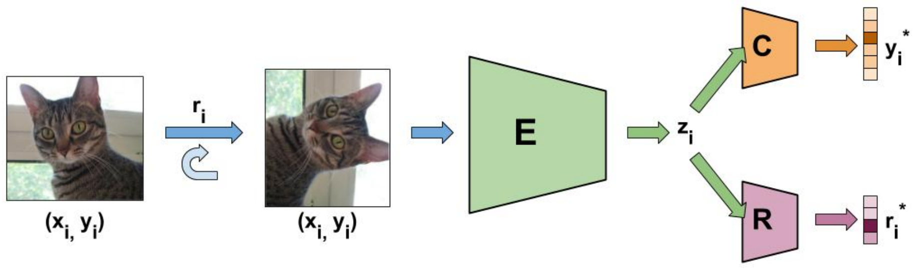

Figure 3: Self-supervised auxiliary classification task. Input image x_i is rotated through a angle r_i. The feature extraction network E produces feature z_i from x_i which is used in parallel by the fully-supervised classification network C and the auxiliary self-supervised rotation prediction network R.

$$W_i^{E_2*} = \alpha_i^{E_2} W_i^{E_2} + (1 - \alpha_i^{E_2}) W_i^{E_1} \tag{2}$$

where, $\alpha_i^{E_1}$ is an adaptive coefficient for the i^{th} layer of E_1 that should be greater than 0.5 if $W_i^{E_1}$ is more informative than $W_i^{E_2}$ [10] and similarly, $\alpha_i^{E_2}$ is an adaptive coefficient for the i^{th} layer of E_2.

Therefore, we perform a weighted addition of the filter-weights of the corresponding layers in the two networks. The $\alpha_i^{E_1}$ adaptive coefficient, determines the contribution of the i^{th} layer of E_1 and E_2 towards calculating the new weights of the i^{th} layer of E_1. In order to calculate $\alpha_i^{E_1}$ the entropy of the i^{th} layer of E_1 and E_2 are used. If the entropy of the i^{th} layer of E_1 is more than that of the i^{th} layer of E_2 then the contribution of the i^{th} layer of E_1 should be more. Therefore, $\alpha_i^{E_1}$ should be more than 0.5. Similarly, the $\alpha_i^{E_2}$ adaptive coefficient, determines the contribution of the i^{th} layer of E_2 and E_1 towards calculating the new weights of the i^{th} layer of E_2. The detailed description of these adaptive coefficients is provided in [10].

3.4. Self-Supervised Auxiliary Task

We use the rotation based self-supervision [39] as an auxiliary task in our method. The input image is rotated through $0^o, 90^o, 180^o, 270^o$ degree, and the objective for the network is to predict the angle of rotation of the given image. We add a network R for rotation angle prediction. For any given input image, we take the output of the feature extraction network E and feed it to R to predict the rotation angle. This training is carried out in parallel to the full classification training. The self-supervision (SS) auxiliary task loss can be defined as

$$L_{SS} = \Sigma_i F_{CE}(r_i^*, r_i) \tag{3}$$

where i refers to the i^{th} data point in the mini-batch, F_{CE} refers to the cross-entropy loss function, r_i^* refers to the predicted rotation angle, and r_i refers to the actual rotation angle.

3.5. Knowledge Distillation

We perform Knowledge Distillation (KD) [17] from a teacher network to a student network, both of which have the same architecture. This is done so that in the process of transferring knowledge to the student network, the student network also learns more generic features than the teacher, which only serves as a guiding signal. The knowledge distillation loss L_{KD} calculates Kullback-Liebler (KL) divergence between the soft predictions of the teacher and the student networks and can be defined as follows:

$$L_{KD} = \Sigma_i KL(F_{\texttt{softmax}}(C^S(E^S(x_i))/\kappa), F_{\texttt{softmax}}(C^S(E^T(x_i))/\kappa)) \tag{4}$$

where, $F_{softmax}$ refers to the softmax function, κ is the temperature hyper-parameter that is used to dampen the logits in order to avoid peaky output probability distribution, KL refers to the KL-Divergence function. E^S, E^T refer to the feature extraction networks of the student and the teacher respectively. C^S, C^T refer to the fully-supervised classification networks of the student and the teacher respectively.

3.6. Method Overview

Our proposed Grafting based Improved Few-Shot Learning (GIFSL) method uses filter-grafting to increase the representational capacity of the feature extraction network of a few-shot model, which can then be exploited by a self-supervised auxiliary task and a knowledge distillation procedure to improve the performance of the few-shot model further.

Our training consists of 2 stages. In the first stage, we train two networks on a full classification task and an auxiliary self-supervised classification task and perform

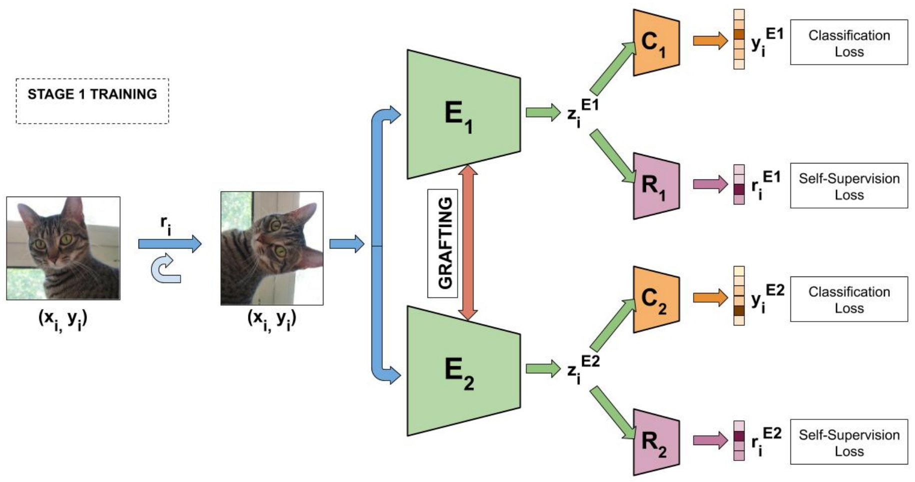

Figure 4: Stage 1 Training. Feature extraction networks (E_1 and E_2) having the same architecture produce features $z_i^{E_1}$ and $z_i^{E_2}$ from the input image x_i respectively. $z_i^{E_1}$ and $z_i^{E_2}$ are fed to the corresponding classification networks C_1 and C_2 in order to perform training for fully supervised classification using cross-entropy loss. $z_i^{E_1}$ and $z_i^{E_2}$ are also fed to the corresponding rotation prediction networks R_1 and R_2 in order to perform training for self-supervised auxiliary classification. Filter-grafting is carried out from E_1 and E_2 and vice-versa after each training epoch.

filter-grafting between the two networks after each epoch. In the second stage, we randomly choose one of the networks from the first stage as a teacher. We again train two student networks using a full classification task and an auxiliary self-supervised classification task. We perform a knowledge distillation procedure from the teacher network to the two student networks and also perform filter-grafting between the two student networks.

3.6.1. First Stage Training

For the first stage of training (Fig. 4), we take two feature extraction networks E_1 and E_2 with the same architecture. We train them along with their corresponding fully-supervised classification modules, C_1 and C_2 respectively, for classification on the entire training dataset using cross-entropy loss. The total fully-supervised classification loss L_S^{tot} is given as follows:

$$L_S^{tot}(E_1, C_1, E_2, C_2) = \Sigma_i(F_{CE}(y_i^{C_1*}, y_i) + F_{CE}(y_i^{C_2*}, y_i)) \quad (5)$$

where, y_i is the real label for the i^{th} data point, $y_i^{C_1*}$ and $y_i^{C_2*}$ refer to the label predicted by C_1 and C_2 respectively.

We also train E_1 and E_2 on the rotation based self-supervision auxiliary tasks. We use rotation prediction networks R_1 and R_2 for E_1 and E_2, respectively. The total self-supervision auxiliary loss L_{SS}^{tot} can be derived from Eq. 3 and is as follows:

$$L_{SS}^{tot}(E_1, R_1, E_2, R_2) = \Sigma_i(F_{CE}(r_i^{R_1*}, r_i) + F_{CE}(r_i^{R_2*}, r_i)) \tag{6}$$

where, r_i is the real rotation label for the i^{th} data point, $y_i^{R_1*}$ and $y_i^{R_2*}$ refer to the label predicted by R_1 and R_2 respectively.

Therefore, the combined loss of stage 1 can be given as

$$L_{stage1} = L_S^{tot}(E_1, C_1, E_2, C_2) + \lambda L_{SS}^{tot}(E_1, R_1, E_2, R_2) \tag{7}$$

where, λ is a hyper-parameter which controls the contribution of the auxiliary self-supervision loss.

During this stage, after each epoch we perform filter-grafting using Eqs. 1,2 on both E_1 and E_2 i.e. we graft filter weights from E_1 to E_2 and vice versa.

3.6.2. Second Stage Training

For the second stage of training (Fig. 5), we use one of the two extraction units (randomly) from the first stage as the teacher network E^T along with its fully-supervised classification module C^T. We take two student feature extraction networks E_1^S and E_2^S with the same architecture. We train them on the fully-supervised classification loss Eq. 5, on the auxiliary self-supervision loss Eq. 6. We also train them on the Knowledge Distillation (KD) loss using the teacher network. Using Eq. 4, the total knowledge distillation loss L_{KD}^{tot} can be given as

$$\begin{aligned} L_{KD}^{tot}(E_1^S, C_1^S, E_2^S, C_2^S, E^T, C^T) = \\ \Sigma_i(KL(F_{\texttt{softmax}}(C_1^S(E_1^S(x_i))/\kappa), F_{\texttt{softmax}}(C^T(E^T(x_i))/\kappa)) \\ + KL(F_{\texttt{softmax}}(C_2^S(E_2^S(x_i))/\kappa), F_{\texttt{softmax}}(C^T(E^T(x_i))/\kappa))) \end{aligned} \tag{8}$$

where, C_1^S, C_2^S are the fully-supervised classification modules attached to E_1^S, E_2^S respectively. C^T is the fully-supervised classification module attached to E^T.

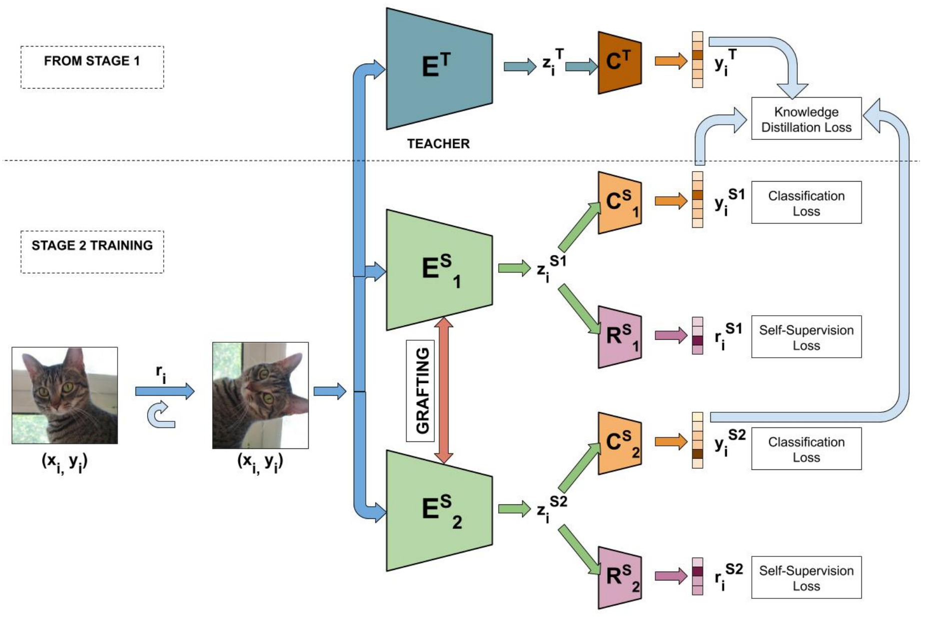

Figure 5: Stage 2 Training. One of the trained extraction units from stage 1 is randomly chosen as teacher E^T, along with its corresponding classification network C^T. Two student feature extraction networks (E_1^S and E_2^S) having the same architecture are taken along with the corresponding classification networks (C_1^S and C_2^S) and rotation prediction networks (R_1^S and R_2^S). The student networks are trained on the fully supervised classification loss and auxiliary self-supervision loss as in the stage 1 training along with knowledge distillation loss from the teacher network. Filter-grafting is carried out between the student networks after each training epoch.

Therefore, the combined loss of stage 2 can be given as

$$\begin{aligned} L_{stage2} = \beta(L_S^{tot}(E_1^S, C_1^S, E_2^S, C_2^S) + \lambda L_{SS}^{tot}(E_1^S, R_1^S, E_2^S, R_2^S)) \\ + \gamma L_{KD}^{tot}(E_1^S, C_1^S, E_2^S, C_2^S, E^T, C^T) \end{aligned} \quad (9)$$

where, R_1^S, R_2^S are the rotation classification networks attached to E_1^S, E_2^S respectively. λ is a hyper-parameter which controls the contribution of the auxiliary self-supervision loss. β, γ are hyper-parameters that control the contribution of the classification losses (supervised and self-supervised) and the knowledge distillation loss, respectively.

Even during this stage, after each epoch we perform filter-grafting using Eqs. 1,2

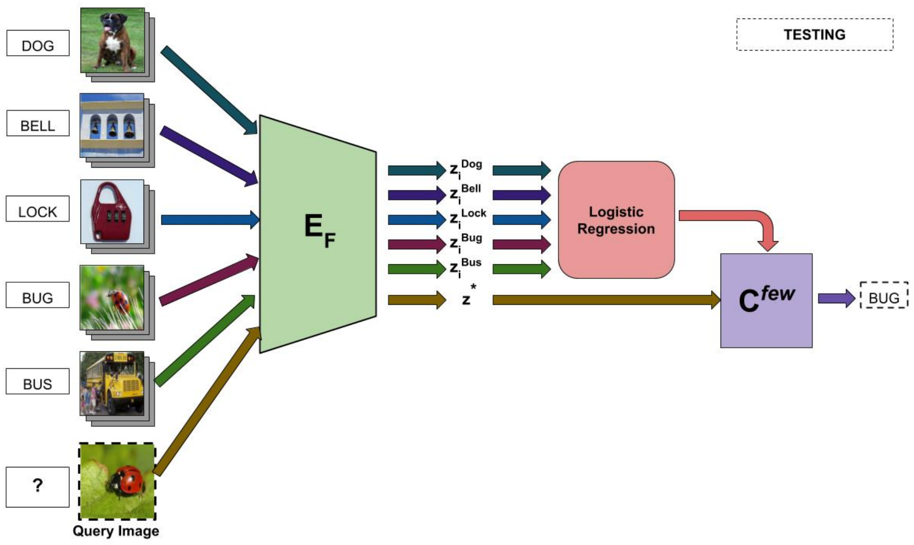

Figure 6: Testing. During testing, each N-way K-shot episode contains K support images for each of the N classes in the episode. One of the student networks from stage 2 is randomly chosen as the final extraction network E_F. E_F extracts features from the support images. These features are used to train a linear model C_{few} using logistic regression. E_F extracts feature z^* from the query image, which is used by C_{few} to predict the class (out of the N classes in the episode) that the query image belongs to.

on both E_1^S and E_2^S i.e. we graft filter weights from E_1^S to E_2^S and vice versa.

3.6.3. Testing Stage

After the completion of stage 2 training, we randomly choose one of the student extraction networks as our final extraction network E_F. We use E_F to perform few-shot testing.

We perform few-shot testing in the episodic setting, just like other few-shot methods. For each episode, we first extract the features for the support examples for each class, using E_F.

$$z_i = E_F(x_i) \tag{10}$$

where, x_i refers to a support image and z_i refers to the feature extracted by E_F from that image.

Now, using the extracted features of the support images, we learn a linear classification model C^{few} using logistic regression. Next, we extract features for the query examples using E_F,

$$z_i^* = E_F(x_i^*) \tag{11}$$

where, x_i^* refers to a query image and z_i^* refers to the feature extracted by E_F from that image.

Using C^{few}, we predict the class for the query images (Fig. 6). We validate the different components of our network using ablation experiments.

4. Experiments

4.1. Datasets

We report the results for few-shot classification experiments on 4 benchmark datasets: mini-ImageNet [1], tiered-ImageNet [26], CIFAR-FS [3] and FC100 [5].

mini-ImageNet [1] is one of the most popular few-shot learning dataset and is derived from the ImageNet [52] dataset. It consists of 100 classes, each of which has around 600 images of size 84×84 pixels. There are 64 train classes, 16 validation classes, and 20 test classes.

tiered-ImageNet [26] is also subset of ImageNet with 351 train, 97 validation, and 60 test classes. It is structured in such a way that the training classes differ significantly from the testing classes as compared to mini-ImageNet.

CIFAR-FS is derived from CIFAR-100 [53] classes and consists of 64 train, 16 validation, and 20 test classes with images of size 32×32 pixels. CIFAR-FS allows faster processing than mini-ImageNet while presenting a hard task due to its smaller size. Few-shot-CIFAR100 (FC100) is also derived from CIFAR-100. FC100 is split in such a way that there is a minimum overlap of similar classes across the splits, making it more difficult to perform few-shot classification. The 100 classes of CIFAR-100 are grouped into 20 superclasses. The dataset is then split using superclasses to ensure minimum overlap of similar classes across splits. The 60 train classes belong to 12 superclasses, and the 20 classes of validation and test splits belong to 5 superclasses each. Each image is of size 32×32 pixels.

4.2. Implementation Details

We use the ResNet-12 architecture [54] for the feature extraction network for all the experiments. The fully-supervised classifier module C is a fully connected network with 1 layer having an input of size 640 and output of size 64 for mini-ImageNet and CIFAR-FS, 60 for FC100, 351 for tiered-ImageNet. The rotation classification network R is a convolutional neural network with 4 convolutional blocks of kernel size 3×3, padding 1, and stride 1. Each convolutional block is followed by a batch normalization module and a ReLU activation function. Each convolutional block has 640 output filters. An adaptive average pooling block is added after the last convolutional block, and it is followed by a fully-connected layer with input size 640 and output size 4. We use $\lambda = 1, \beta = 0.5$, $\gamma = 1$ and $\kappa = 4$ as the hyper-parameters. We report the average accuracy of 1000 test episodes with 95% confidence intervals. We perform experiments for 5-way 1-shot and 5-shot episodes. The experiments have all been performed in PyTorch [55].

An important point to note is that apart from grafting, self-supervision, and knowledge distillation, GIFSL also performs full training of the network on the entire train set and testing using logistic regression on the extracted features. Both these components have been used separately in different works e.g., [56, 57] make use of pre-training the network on the entire train set and [6, 58] make use of linear classifiers such as SVM and logistic regression. The authors in [56] show how training on the entire train set alone can outperform many recent few-shot learning methods. Therefore, we have included these two techniques in our approach too. We show in Sec. 5.2 that our proposed approach (with grafting, self-supervision, and knowledge distillation) can achieve around 3.81% (1-shot) and 3.06% (5-shot) absolute increase over our method that uses only training on the entire train+val set and logistic regression (LR).

4.3. mini-ImageNet results

Table 1 reports the experimental results for few-shot classification on mini-ImageNet dataset for 5-way 1-shot and 5-shot results. The results indicate that our method GIFSL performs significantly better than the other methods in the 1-shot setting as well as in

Table 1: Average 1/5-shot 5-way few-shot classification accuracy over test images from the novel classes of mini-ImageNet. * indicate methods that train on a union of train and validation (train+val) set. Bold values indicate the best results obtained by our method in the comparison.

Models	Backbone	1-shot	5-shot
MAML [4] (ICML'17)	Conv-4-64	48.70 ± 1.84%	63.10 ± 0.92%
Prototypical Nets [2] (NIPS'17)	Conv-4-64	49.42 ± 0.78%	68.20 ± 0.66%
Warp-MAML [24] (ICLR'20)	Conv-4-64	52.30 ± 0.80%	68.4 ± 0.60%
LwoF [30] (CVPR'18)	Conv-4-64	56.20 ± 0.86%	72.81 ± 0.62%
RelationNet [29] (CVPR'18)	Conv-4-64	50.40 ± 0.80%	65.30 ± 0.70%
GNN [28] (ICLR'18)	Conv-4-64	50.30%	66.40%
SNAIL [32] (ICLR'18)	ResNet-12	55.71 ± 0.99%	68.88 ± 0.92%
TPN [34] (ICLR'19)	Conv-4-64	55.51 ± 0.86%	69.86 ± 0.65%
Qiao et al. [33]* (CVPR'18)	WRN-28-10	59.60 ± 0.41%	73.74 ± 0.19%
TADAM [5] (NIPS'18)	ResNet-12	58.50 ± 0.30%	76.70 ± 0.30%
R2-D2 [3] (ICLR'19)	Conv-4-64	49.50 ± 0.20%	65.40 ± 0.20%
R2-D2 [3] (ICLR'19)	Conv-4-512	51.80 ± 0.20%	68.40 ± 0.20%
Munkhdalai et al. [27] (ICML'17)	ResNet-12	57.10 ± 0.70%	70.04 ± 0.63%
STANet [20] (AAAI'19)	ResNet-12	58.35 ± 0.57%	71.07 ± 0.39%
IdeMe-Net [21] (CVPR'19)	ResNet-18	59.14 ± 0.86	74.63 ±0.74
Shot-Free [22] (ICCV'19)	ResNet-12	59.04%	77.64%
SalNet [23] (CVPR'19)	ResNet-101	62.22 ± 0.87%	77.95 ± 0.65%
LEO* [31] (ICLR'19)	WRN-28-10	61.76 ± 0.08%	77.59 ± 0.12%
CC+rot [16] (ICCV'19)	WRN-28-10	62.93 ± 0.45%	79.87 ± 0.33%
MetaOptNet [6] (CVPR'19)	ResNet-12	62.64 ± 0.61%	78.63 ± 0.46%
MetaOptNet* [6] (CVPR'19)	ResNet-12	64.09 ± 0.62%	80.00 ± 0.45%
TADAM+D-SVS [37] (AAAI'20)	ResNet-12	60.16 ± 0.47%	77.25 ± 0.15%
Deep DTN [36] (AAAI'20)	ResNet-12	63.45 ± 0.86%	77.91± 0.62%
AFHN [35] (CVPR'20)	ResNet-18	62.38 ± 0.72%	78.16 ± 0.56%
AWGIM [59] (CVPR'20)	WRN-28-10	63.12 ± 0.08%	78.40 ± 0.11%
GIFSL (Ours)	ResNet-12	**65.47** ± 0.63%	**82.75** ± 0.42%
GIFSL* (Ours)	ResNet-12	**67.02** ± 0.61%	**83.89** ± 0.42%

Table 2: Average 1/5-shot 5-way few-shot classification accuracy over test images from the novel classes of tiered-ImageNet. * indicate methods that train on a union of train and validation (train+val) set. Bold values indicate the best results obtained by our method in the comparison.

Models	Backbone	1-shot	5-shot
MAML [4] (ICML'17)	Conv-4-64	$51.67 \pm 1.81\%$	$70.30 \pm 0.08\%$
Prototypical Nets [2] (NIPS'17)	Conv-4-64	$53.31 \pm 0.89\%$	$72.69 \pm 0.74\%$
RelationNet [29] (CVPR'18)	Conv-4-64	$54.48 \pm 0.93\%$	$71.32 \pm 0.78\%$
TPN [34] (ICLR'19)	Conv-4-64	$59.91 \pm 0.94\%$	$73.30 \pm 0.75\%$
Warp-MAML [24] (ICLR'20)	Conv-4-64	$57.20 \pm 0.90\%$	$74.1 \pm 0.70\%$
LEO* [31] (ICLR'19)	WRN-28-10	$66.33 \pm 0.05\%$	$81.44 \pm 0.09\%$
MetaOptNet [6] (CVPR'19)	ResNet-12	$65.99 \pm 0.72\%$	$81.56 \pm 0.53\%$
MetaOptNet* [6] (CVPR'19)	ResNet-12	$65.81 \pm 0.74\%$	$81.75 \pm 0.53\%$
Shot-Free [22] (ICCV'19)	ResNet-12	66.87%	82.64%
CC+rot [16] (ICCV'19)	WRN-28-10	$70.53 \pm 0.51\%$	$84.98 \pm 0.36\%$
AWGIM [59] (CVPR'20)	WRN-28-10	$67.69 \pm 0.11\%$	$82.82 \pm 0.13\%$
GIFSL (Ours)	ResNet-12	**72.39** $\pm 0.66\%$	**86.91** $\pm 0.44\%$
GIFSL* (Ours)	ResNet-12	**73.85** $\pm 0.67\%$	**88.22** $\pm 0.45\%$

the 5-shot setting. GIFSL outperforms AFHN [35] significantly, with an absolute increase of 3.09% and 4.59% in the 1-shot and 5-shot settings. GIFSL (train+val) trains the network on the entire train and validation set. This model performs even better on mini-ImageNet for both settings. On removing grafting, self-supervision and knowledge distillation the performance of our method is significantly decreased by absolute percentages of 3.64% and 3.43% for 1-shot and 5-shot settings when the training is done on only the train set and by absolute percentages of 3.81% and 3.06% for 1-shot and 5-shot settings when the training is done on the train+val set.

4.4. tiered-ImageNet results

Table 2 reports the experimental results for few-shot classification on tiered-ImageNet dataset for 5-way 1-shot and 5-shot results. The results indicate that our method GIFSL outperforms other state-of-the-art methods in the 1-shot and 5-shot settings for tiered-

Table 3: Average 1/5-shot 5-way few-shot classification accuracy over test images from the novel classes of CIFAR-FS. * indicate methods that train on a union of train and validation (train+val) set. †: results from [3]. ‡: results from [16]. Bold values indicate the best results obtained by our method in the comparison.

Models	Backbone	1-shot	5-shot
PN [2]† (NIPS'17)	Conv-4-64	55.50 $\pm$ 0.70%	72.00 $\pm$ 0.60%
PN [2]† (NIPS'17)	Conv-4-512	57.90 $\pm$ 0.80%	76.70 $\pm$ 0.60%
PN [2]‡ (NIPS'17)	Conv-4-64	62.82 $\pm$ 0.32%	79.59 $\pm$ 0.24%
PN [2]‡ (NIPS'17)	Conv-4-512	66.48 $\pm$ 0.32%	80.28 $\pm$ 0.23%
MAML [4]† (ICML'17)	Conv-4-64	58.90 $\pm$ 1.90%	71.50 $\pm$ 1.00%
MAML [4]† (ICML'17)	Conv-4-512	53.80 $\pm$ 1.80%	67.60 $\pm$ 1.00%
RelationNet [29]† (CVPR'18)	Conv-4-64	55.00 $\pm$ 1.00%	69.30 $\pm$ 0.80%
GNN [28]† (ICLR'18)	Conv-4-64	61.90%	75.30%
GNN [28]† (ICLR'18)	Conv-4-512	56.00%	72.50%
R2-D2 [3] (ICLR'19)	Conv-4-64	62.30 $\pm$ 0.20%	77.40 $\pm$ 0.20%
R2-D2 [3] (ICLR'19)	Conv-4-512	65.40 $\pm$ 0.20%	79.40 $\pm$ 0.20%
Shot-Free [22] (ICCV'19)	ResNet-12	69.15%	84.70%
MetaOptNet [6] (CVPR'19)	ResNet-12	72.60 $\pm$ 0.70%	84.30 $\pm$ 0.50%
MetaOptNet* [6] (CVPR'19)	ResNet-12	72.80 $\pm$ 0.70%	85.00 $\pm$ 0.50%
CC+rot [16] (ICCV'19)	WRN-28-10	73.62 $\pm$ 0.31%	86.05 $\pm$ 0.22%
GIFSL (Ours)	ResNet-12	**74.58** $\pm$ 0.38%	**87.68** $\pm$ 0.23%
GIFSL* (Ours)	ResNet-12	**76.02** $\pm$ 0.40%	**88.87** $\pm$ 0.26%

ImageNet. GIFSL outperforms [16] significantly, with an absolute increase of 1.86% and 1.93% in the 1-shot and 5-shot settings. GIFSL (train+val) achieves higher performance on tiered-ImageNet for both the settings. On removing grafting, self-supervision and knowledge distillation the performance of our method is significantly decreased by absolute percentages of 2.94% and 2.79% for 1-shot and 5-shot settings when the training is done on only the train set and by absolute percentages of 2.99% and 2.63% for 1-shot and 5-shot settings when the training is done on the train+val set.

Table 4: Average 1/5-shot 5-way few-shot classification accuracy over test images from the novel classes of FC100. * indicate methods that train on a union of train and validation (train+val) set. Bold values indicate the best results obtained by our method in the comparison.

Models	Backbone	1-shot	5-shot
PN [2] (NIPS'17)	Conv-4-64	35.30 $\pm$ 0.60%	48.60 $\pm$ 0.60%
TADAM [5] (NIPS'18)	ResNet-12	40.10 $\pm$ 0.40%	56.10 $\pm$ 0.40%
MetaOptNet [6] (CVPR'19)	ResNet-12	41.10 $\pm$ 0.60%	55.50 $\pm$ 0.60%
MetaOptNet* [6] (CVPR'19)	ResNet-12	47.20 $\pm$ 0.60%	62.50 $\pm$ 0.60%
MTL [18] (CVPR'19)	ResNet-12	45.1 $\pm$ 1.80%	57.60 $\pm$ 0.90%
DC [19] (CVPR'19)	ResNet-12	42.04 $\pm$ 0.17%	57.63 $\pm$ 0.23%
GIFSL (Ours)	ResNet-12	**45.35** $\pm$ 0.32%	**61.71** $\pm$ 0.34%
GIFSL* (Ours)	ResNet-12	**52.25** $\pm$ 0.33%	**69.25** $\pm$ 0.35%

4.5. CIFAR-FS results

Table 3 reports the experimental results for few-shot classification on CIFAR-FS dataset for 5-way 1-shot and 5-shot results. The results indicate that our method GIFSL achieves higher performance than other state-of-the-art methods in the 1-shot and 5-shot settings for CIFAR-FS. GIFSL (train+val) performs even better for both the settings. On removing grafting, self-supervision and knowledge distillation the performance of our method is significantly decreased by absolute percentages of 3.32% and 1.79% for 1-shot and 5-shot settings when the training is done on only the train set and by absolute percentages of 2.97% and 2.31% for 1-shot and 5-shot settings when the training is done on the train+val set.

4.6. FC100 results

Table 4 reports the experimental results for few-shot classification on FC100 dataset for 5-way 1-shot and 5-shot settings. The results indicate that our methods GIFSL and GIFSL (train+val) perform better than other methods in both the settings for FC100. On removing grafting, self-supervision and knowledge distillation the performance of our method is significantly decreased by absolute percentages of 3.12% and 2.76% for 1-shot and 5-shot settings when the training is done on only the train set and by absolute percentages of 2.62% and 2.98% for 1-shot and 5-shot settings when the training is done on the train+val set.

5. Ablation

We perform various ablations such as choice of the classifier for few-shot testing, contribution of each component of our method and others. We use the ResNet-12 architecture for the feature extraction module for all the experiments in this section.

5.1. Classifier module

We check which type of linear classifier best suits our method. We experiment on the mini-ImageNet dataset with four types of classifiers: prototype-based classifier [2], single-layer neural network (Linear NN), logistic regression (LR), and linear SVM (LSVM). In the case of the single-layer neural network, for each episode, we train the classification module on the extracted features of the support examples for 1000 epochs and then use it to classify the query features. The prototype-based classification module finds the prototype for each class in the episode by performing an average on the extracted features of the support examples of each class. The nearest class prototype to the query feature is predicted to be the output class [2]. The results in Table 5 indicate that the logistic regression-based classifier performs the best from among the compared classifiers.

5.2. Significance of each component

In our proposed method, we make use of filter-grafting, self-supervision, and knowledge distillation. We train the network on the entire train/train+val set and use logistic regression for testing. In this section, we perform experiments to validate the contribution of each component. We also show that combining filter-grafting with self-supervision and knowledge distillation significantly boosts the network's performance. In our experiments where logistic regression is not used for testing, the nearest prototype-based testing is used as in [2].

The results in Table 6 indicate that when the network is trained on the entire train+val set, it performs significantly better than when the network is trained on episodes

Table 5: Experimental results for 1/5-shot 5-way classification on mini-ImageNet using different types of classifier with training on train+val set.

shot	Proto. based	Linear NN	LR	LSVM
1	66.21 ± 0.63%	66.19 ± 0.61%	67.02 ± 0.64%	65.66 ± 0.61%
5	83.45 ± 0.40%	83.21 ± 0.42%	83.89 ± 0.45%	82.51 ± 0.44%

Table 6: Average 1/5-shot 5-way few-shot classification accuracy over test images from the novel classes of mini-ImageNet for different components of our method with ResNet-12 architecture. The training was carried out on the train+val set. FT refers to training on the entire train+val set. LR refers to logistic regression. KD refers to knowledge distillation

FT	LR	Grafting	Self-Supervision	KD	1-shot	5-shot
✗	✗	✗	✗	✗	58.12%	75.46%
✓	✗	✗	✗	✗	62.06%	79.92%
✓	✓	✗	✗	✗	63.21%	80.83%
✓	✓	✓	✗	✗	63.85%	81.25%
✓	✓	✗	✓	✗	63.98%	81.36%
✓	✓	✗	✗	✓	65.29%	82.54%
✓	✓	✓	✓	✗	65.35%	82.86%
✓	✓	✗	✓	✓	65.73%	82.88%
✓	✓	✓	✓	✓	**67.02%**	**83.89%**

with an absolute increase in performance by 3.94% (1-shot) and 4.46% (5-shot). When logistic regression is used along with training on the entire train+val set, the performance of the network improves further. When grafting is used along with these two components, there is a slight increase in the performance of the model. This means that adding grafting does not trivially lead to huge performance gains in few-shot learning. Self-supervision and knowledge distillation provide some increase in the performance of the model. However, when we combine them with grafting, a significant increase in performance is noticed. This means that the increase in representational capacity caused by the grafting helps the network to utilize self-supervision and knowledge dis-

Table 7: Average 1/5-shot 5-way few-shot classification novel class accuracy for the CIFAR-FS dataset using GIFSL with different types of auxiliary self-supervised tasks using the ResNet-12 architecture. Please note that we use grafting and knowledge distillation for all the experiments and only replace the self-supervision technique. The network is trained on the entire train set.

Auxiliary Self-Supervision Task	1-shot	5-shot
Patch [14]	74.03%	87.12%
SimCLR [60]	74.15%	87.05%
Rotation [39]	**74.58%**	**87.68%**

tillation better. Our proposed approach (with grafting, self-supervision, and knowledge distillation) can achieve around 3.81% (1-shot) and 3.06% (5-shot) absolute increase over simply using only training on the entire train+val set and logistic regression (LR). This validates the choice of components for our method.

5.3. Self-Supervision Techniques

We compare different self-supervision techniques when used as an auxiliary self-supervision task in GIFSL. We perform experiments with the auxiliary task as rotation angle prediction [39], relative patch location prediction [14] (Patch) and SimCLR [60]. SimCLR is a recent self-supervision technique that uses contrastive learning. These methods were chosen as they can be easily used as an auxiliary task in GIFSL.

We perform experiments on the CIFAR-FS dataset using the ResNet-12 architecture. Table 7 shows that GIFSL with rotation prediction performs better than patch location prediction and SimCLR. SimCLR is a state-of-the-art self-supervision method, but it is still unable to outperform rotation prediction when used as an auxiliary task for few-shot learning.

5.4. Representational Capacity

We compute the total entropy of all filters in the network with and without our components. The more the entropy, the more is the representational capacity of the network, as shown in [10]. We use the ResNet-12 architecture for these experiments and train the network on the entire train+val set. In the case of miniImageNet, the total entropy of GIFSL without grafting, self-supervision, and knowledge distillation is 9.05,

Table 8: Average 1/5-shot 5-way few-shot classification accuracy over test images from the novel classes of CIFAR-FS, FC100 and CUB datasets using a ResNet-12 network trained on mini-ImageNet using our method with and without grafting (GF), self-supervision (SS) and knowledge distillation (KD).

Method	CIFAR-FS		FC100		CUB	
	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
GIFSL w/o GF, SS, KD	58.50%	76.85%	42.01%	58.26%	48.46%	66.44%
GIFSL	**62.79%**	**80.44%**	**45.18%**	**61.72%**	**49.68%**	**69.23%**

while the total entropy of GIFSL with these components is 18.40. Similarly, in the case of tieredImageNet, the total entropy of GIFSL without grafting, self-supervision, and knowledge distillation is 6.78, while the total entropy of GIFSL with these components is 15.07. Therefore, the model trained with grafting, self-supervision, and knowledge distillation has more information and better representational capacity.

5.5. Cross-Domain Few-Shot Learning

We also perform cross-domain few-shot learning experiments to validate the effectiveness of our method. We train the ResNet-12 network on mini-ImageNet and perform few-shot testing on the CIFAR-FS [3], FC100 [5] and CUB [61] datasets. We also experiment without grafting, self-supervision, and knowledge distillation and compare the results. Table 8 shows that using grafting, self-supervision, and knowledge distillation improves the network performance significantly for cross-domain few-shot classification on multiple datasets e.g., in CIFAR-FS using grafting, self-supervision, and knowledge distillation achieves absolute performance increases of 4.29% (1-shot) and 3.59% (5-shot).

6. Conclusion

In this work, we proposed a novel few-shot learning approach called Grafting based Improved Few Shot Learning (GIFSL). Our method combines filter-grafting with self-supervision based auxiliary loss and knowledge distillation. We show how filter-grafting can improve the representational capacity of the feature extraction network in a few-shot model. We also show that this increased representational capacity can be exploited by the network to draw more improvements from auxiliary self-supervised loss and knowledge distillation loss. Through multiple experiments using multiple benchmark datasets, we show that our method performs better than state-of-the-art few-shot learning methods. We also validated the components of our method using ablation experiments.

References

[1] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al., Matching networks for one shot learning, in: Advances in neural information processing systems, 2016, pp. 3630–3638.

[2] J. Snell, K. Swersky, R. Zemel, Prototypical networks for few-shot learning, in: Advances in Neural Information Processing Systems, 2017, pp. 4077–4087.

[3] L. Bertinetto, J. F. Henriques, P. Torr, A. Vedaldi, Meta-learning with differentiable closed-form solvers, in: International Conference on Learning Representations, 2019.
URL `https://openreview.net/forum?id=HyxnZh0ct7`

[4] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 1126–1135.

[5] B. Oreshkin, P. R. López, A. Lacoste, Tadam: Task dependent adaptive metric for improved few-shot learning, in: Advances in Neural Information Processing Systems, 2018, pp. 721–731.

[6] K. Lee, S. Maji, A. Ravichandran, S. Soatto, Meta-learning with differentiable convex optimization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10657–10665.

[7] H. Li, A. Kadav, I. Durdanovic, H. Samet, H. P. Graf, Pruning filters for efficient convnets, in: International Conference on Learning Representations, 2017. URL `https://openreview.net/forum?id=rJqFGTslg`

[8] H. Zhuo, X. Qian, Y. Fu, H. Yang, X. Xue, Scsp: Spectral clustering filter pruning with soft self-adaption manners, in: arXiv preprint arXiv:1806.05320, 2018.

[9] X. Suau, L. Zappella, V. Palakkode, N. Apostoloff, Principal filter analysis for guided network compression, in: arXiv preprint arXiv:1807.10585, Vol. 2, 2018.

[10] F. Meng, H. Cheng, K. Li, Z. Xu, R. Ji, X. Sun, G. Lu, Filter grafting for deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[11] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A. A. Efros, Context encoders: Feature learning by inpainting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2536–2544.

[12] G. Larsson, M. Maire, G. Shakhnarovich, Learning representations for automatic colorization, in: European Conference on Computer Vision, Springer, 2016, pp. 577–593.

[13] R. Zhang, P. Isola, A. A. Efros, Colorful image colorization, in: European Conference on Computer Vision, Springer, 2016, pp. 649–666.

[14] C. Doersch, A. Gupta, A. A. Efros, Unsupervised visual representation learning by context prediction, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1422–1430.

[15] M. Noroozi, P. Favaro, Unsupervised learning of visual representations by solving jigsaw puzzles, in: European Conference on Computer Vision, Springer, 2016, pp. 69–84.

[16] S. Gidaris, A. Bursuc, N. Komodakis, P. Pérez, M. Cord, Boosting few-shot visual learning with self-supervision, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 8059–8068.

[17] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, in: NIPS Deep Learning and Representation Learning Workshop, 2014.
URL `https://fb56552f-a-62cb3a1a-s-sites.googlegroups.com/site/deeplearningworkshopnips2014/65.pdf`

[18] Q. Sun, Y. Liu, T.-S. Chua, B. Schiele, Meta-transfer learning for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 403–412.

[19] Y. Lifchitz, Y. Avrithis, S. Picard, A. Bursuc, Dense classification and implanting for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 9258–9267.

[20] S. Yan, S. Zhang, X. He, et al., A dual attention network with semantic embedding for few-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 9079–9086.

[21] Z. Chen, Y. Fu, Y.-X. Wang, L. Ma, W. Liu, M. Hebert, Image deformation meta-networks for one-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 8680–8689.

[22] A. Ravichandran, R. Bhotika, S. Soatto, Few-shot learning with embedded class models and shot-free meta training, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019, pp. 331–339.

[23] H. Zhang, J. Zhang, P. Koniusz, Few-shot learning via saliency-guided hallucination of samples, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 2770–2779.

[24] S. Flennerhag, A. A. Rusu, R. Pascanu, F. Visin, H. Yin, R. Hadsell, Meta-learning with warped gradient descent, in: International Conference on Learning Representations, 2020.
URL `https://openreview.net/forum?id=rkeiQlBFPB`

[25] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, in: ICML Deep Learning Workshop, Vol. 2, 2015.

[26] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, R. S. Zemel, Meta-learning for semi-supervised few-shot classification, in: arXiv preprint arXiv:1803.00676, 2018.

[27] T. Munkhdalai, H. Yu, Meta networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 2554–2563.

[28] V. G. Satorras, J. B. Estrach, Few-shot learning with graph neural networks, in: International Conference on Learning Representations, 2018.
URL `https://openreview.net/forum?id=BJj6qGbRW`

[29] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, T. M. Hospedales, Learning to compare: Relation network for few-shot learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 1199–1208.

[30] S. Gidaris, N. Komodakis, Dynamic few-shot visual learning without forgetting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4367–4375.

[31] A. A. Rusu, D. Rao, J. Sygnowski, O. Vinyals, R. Pascanu, S. Osindero, R. Hadsell, Meta-learning with latent embedding optimization, in: International Conference on Learning Representations, 2019.
URL `https://openreview.net/forum?id=BJgklhAcK7`

[32] N. Mishra, M. Rohaninejad, X. Chen, P. Abbeel, A simple neural attentive meta-learner, in: International Conference on Learning Representations, 2018.
URL `https://openreview.net/forum?id=B1DmUzWAW`

[33] S. Qiao, C. Liu, W. Shen, A. L. Yuille, Few-shot image recognition by predicting parameters from activations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7229–7238.

[34] Y. Liu, J. Lee, M. Park, S. Kim, E. Yang, S. Hwang, Y. Yang, LEARNING TO PROPAGATE LABELS: TRANSDUCTIVE PROPAGATION NETWORK FOR

FEW-SHOT LEARNING, in: International Conference on Learning Representations, 2019.
URL `https://openreview.net/forum?id=SyVuRiC5K7`

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

[36] M. Chen, Y. Fang, X. Wang, H. Luo, Y. Geng, X. Zhang, C. Huang, W. Liu, B. Wang, Diversity transfer network for few-shot learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020.

[37] J. Chen, L.-M. Zhan, X.-M. Wu, F. lai Chung, Variational metric scaling for metric-based meta-learning, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020.

[38] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, in: International Joint Conference on Artificial Intelligence (IJCAI), 2018, pp. 2234–2240.

[39] S. Gidaris, P. Singh, N. Komodakis, Unsupervised representation learning by predicting image rotations, in: International Conference on Learning Representations, 2018.
URL `https://openreview.net/forum?id=S1v4N2l0-`

[40] A. Dosovitskiy, J. T. Springenberg, M. Riedmiller, T. Brox, Discriminative unsupervised feature learning with convolutional neural networks, in: Advances in neural information processing systems, 2014, pp. 766–774.

[41] Z. Feng, C. Xu, D. Tao, Self-supervised representation learning by rotation feature decoupling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10364–10374.

[42] T. Chen, X. Zhai, M. Ritter, M. Lucic, N. Houlsby, Self-supervised generative adversarial networks, in: arXiv preprint arXiv:1811.11212, 2018.

[43] P. Agrawal, J. Carreira, J. Malik, Learning to see by moving, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 37–45.

[44] D. Pathak, R. Girshick, P. Dollár, T. Darrell, B. Hariharan, Learning features by watching objects move, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2701–2710.

[45] X. Wang, A. Gupta, Unsupervised learning of visual representations using videos, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2794–2802.

[46] H.-Y. Lee, J.-B. Huang, M. Singh, M.-H. Yang, Unsupervised representation learning by sorting sequences, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 667–676.

[47] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, A. Torralba, Ambient sound provides supervision for visual learning, in: European Conference on Computer Vision, Springer, 2016, pp. 801–816.

[48] S. Srinivas, F. Fleuret, Knowledge transfer with jacobian matching, in: arXiv preprint arXiv:1803.00443, 2018.

[49] Z. Huang, N. Wang, Like what you like: Knowledge distill via neuron selectivity transfer, in: arXiv preprint arXiv:1707.01219, 2017.

[50] J. Yim, D. Joo, J. Bae, J. Kim, A gift from knowledge distillation: Fast optimization, network minimization and transfer learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4133–4141.

[51] H. Mobahi, M. Farajtabar, P. L. Bartlett, Self-distillation amplifies regularization in hilbert space, in: arXiv preprint arXiv:2002.05715, 2020.

[52] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, in: International journal of computer vision, Vol. 115, Springer, 2015, pp. 211–252.

[53] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images.

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

[55] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch.

[56] Y. Chen, X. Wang, Z. Liu, H. Xu, T. Darrell, A new meta-baseline for few-shot learning (2020). arXiv:2003.04390.

[57] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, J.-B. Huang, A closer look at few-shot classification, in: International Conference on Learning Representations, 2019.
URL https://openreview.net/forum?id=HkxLXnAcFQ

[58] Y. Wang, C. Xu, C. Liu, L. Zhang, Y. Fu, Instance credibility inference for few-shot learning (2020). arXiv:2003.11853.

[59] Y. Guo, N.-M. Cheung, Attentive weights generation for few shot learning via information maximization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 13499–13508.

[60] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning (ICML), 2020.

[61] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The caltech-ucsd birds-200-2011 dataset.