This WACV 2021 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

AdarGCN: Adaptive Aggregation GCN for Few-Shot Learning

Jianhong Zhang[1*] Manli Zhang[1*] Zhiwu Lu[1,2†] Tao Xiang[3]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods
[3]University of Surrey, United Kingdom
jianhong@ruc.edu.cn manlizhang@ruc.edu.cn luzhiwu@ruc.edu.cn

Abstract

Existing few-shot learning (FSL) methods assume that there exist sufficient training samples from source classes for knowledge transfer to target classes with few training samples. However, this assumption is often invalid, especially when it comes to fine-grained recognition. In this work, we define a new FSL setting termed scarce-source few-shot learning (SSFSL), under which both the source and target classes have limited training samples. To overcome the source class data scarcity problem, a natural option is to crawl images from the web with class names as search keywords. However, the crawled images are inevitably corrupted by large amount of noise (irrelevant images) and thus may harm the performance. To address this problem, we propose a graph convolutional network (GCN)-based label denoising (LDN) method to remove the irrelevant images. Further, with the cleaned web images as well as the original clean training images, we propose a GCN-based FSL method. For both the LDN and FSL tasks, a novel adaptive aggregation GCN (AdarGCN) model is proposed, which differs from existing GCN models in that adaptive aggregation is performed based on a multi-head multi-level aggregation module. With AdarGCN, how much and how far information carried by each graph node is propagated in the graph structure can be determined automatically, therefore alleviating the effects of both noisy and outlying training samples. Extensive experiments demonstrate the superior performance of our AdarGCN under both the new SSFSL and the conventional FSL settings.

1. Introduction

Few-shot learning (FSL) [22, 5] aims to recognize a set of target classes by learning with sufficient labeled samples from a set of source classes and only few labeled samples from the target classes. Existing FSL methods [33, 37, 6, 47, 50, 48, 1, 34, 38, 21, 29, 11] employ deep neural networks (DNNs) [18, 56, 13]. They thus make the implicit assumption that there are sufficient training samples from at least the source classes for knowledge transfer to the target. However, this assumption is often invalid in practice especially when it comes to fine-grained recognition. For this problem, the source classes are also fine-grained, so collecting and labeling sufficient samples for each source class is challenging. For example, in the widely-used CUB dataset [51], each bird class has less than 60 samples. Without sufficient labeled samples from each source class, it becomes harder to recognize the target classes by knowledge transfer from source classes.

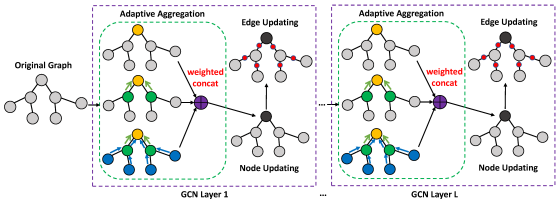


Figure 1. Illustration of the adaptive aggregation strategy included in our AdarGCN which is realized based on a multi-head multi-level module. In a GCN layer, each head performs aggregation for different iterations and all the heads are weighted concatenated to update the node feature, followed by an edge updating module.

In this work, we define a new setting termed scarce-source few-shot learning (SSFSL), where only few labeled samples from both source and target classes are available for training. To overcome the source class data scarcity problem under this new setting, a natural solution would be to crawl sufficient images from the web by searching with the name of each source class. However, although the crawled data contains extra relevant training images, it also inevitably consists of large quantities of irrelevant ones. To fully exploit the crawled noisy images of each source class for SSFSL, label denoising (LDN) is required as a preprocessing stage. Once the web images are cleaned by a LDN model, a SSFSL problem becomes a conventional

*Equal Contribution
†Corresponding Author

many-shot few-shot learning (MSFSL) problem for which a plethora of existing methods can be applied. However, no matter how strong the LDN method is, some wrongly labeled images will remain which pose additional challenges to the subsequent FSL stage. Dealing with noisily-labeled images thus becomes the key for solving the SSFSL problem in both stages.

We thus propose to use graph convolutional networks (GCNs) [17, 4, 46, 8] for both the LDN and FSL stages. This is because, treating each image as a graph node and using edges between nodes to represent their visual similarity in a feature space, a graph model is intrinsically suited for enforcing visual and label consistency across training samples (i.e., images of the same class labels should be visually similar and vice versa). Those nodes that do not conform to the consistency constraint can thus be flagged out for denoising. Their negative effects for label propagation on the graph can also be reduced for the 2nd stage few-shot recognition. Both tasks benefit from multiple GCN layers which learn better node and edge representations. Indeed, GCNs have been employed for both LDN [17, 14, 23] and FSL [41, 15, 10] (albeit without considering label noise). However, existing GCNs are intrinsically limited for both stages. In particular, they lack mechanisms to control how the information carried by a node (especially the noisy ones) can be propagated to the rest of the graph. Such a control ideally should be introduced in an instance-dependent and adaptive manner. This shortcoming also partly explains why in practice a GCN with more than three layers often becomes ineffective due to over-smoothing in the top layers [24]. Therefore, we propose a novel adaptive aggregation GCN (AdarGCN) which can perform adaptive aggregation based on a multi-head multi-level aggregation module (see Fig. 1). With AdarGCN, how much and how far the information carried by each node is propagated to the rest of the graph is determined by each head, making the propagation controllable and adaptive to each instance. An aggregation gate with learnable parameters is then used to dynamically determine the weight of each head when fusing multiple heads together. In this way, the negative impact of a noisy training sample can be limited to a small neighborhood of the corresponding node, thus effectively diminishing its detrimental effect.

As shown in Fig. 2, our AdarGCN is used in both stages of our SSFSL framework because dealing with noisy training images is the key for both. Importantly, we also empirically observe that with AdarGCN, (1) the GCN can be deeper than existing ones which are typically limited by the depth of layers due to over-smoothing, bringing additional performance gain, and (2) it beats the state-of-the-art alternatives even under the conventional FSL setting where no label noise is present. This indicates that AdarGCN can also be used to deal with the clean but outlying training samples. These samples are particularly harmful for FSL as each class is only represented with few samples.

Our contributions are: **(1)** We define a new FSL setting termed SSFSL, which is more challenging yet more realistic than the conventional FSL setting. **(2)** A two-stage solution is proposed for SSFSL: i) crawling sufficient source class training images from the web and performing label denoising on them; ii) solving the FSL problem after merging the cleaned web images with the original training samples. **(3)** Both the LDN and FSL tasks involved in our SSFSL setting are addressed by a novel GCN model termed AdarGCN. Extensive experiments show that our AdarGCN achieves state-of-the-art results under both FSL settings.

2. Related Work

Few-Shot Learning: Meta-learning based methods [33, 37, 6, 47, 50, 48, 32, 1, 19, 40, 54, 25, 27, 53] have dominated the recent FSL research. Apart from metric learning solutions [47, 50, 48, 1, 54, 25, 27, 53], another promising approach is learning to optimize [37, 6, 19, 40]. More recently, methods based on feature hallucination and synthesis [12, 44, 7, 52, 57, 49] or predicting parameters of the network [36, 35, 10, 9] have been developed. However, the promising performance of existing FSL methods is highly dependent on the assumption that there exist sufficient training samples from source classes. Note that this assumption is often invalid in practice, especially when it comes to fine-grained recognition. In this work, we thus focus on a new SSFSL setting (only with a few labeled samples per source class). Even though our AdarGCN model is designed for this new SSFSL setting, it is found to be extremely competitive under the conventional FSL setting (see Table 4). This suggests that the outlying samples problem, largely ignored by existing FSL methods so far, should also be addressed even if the source class data is ample.

Graph Convolutional Networks: GCN is designed to work directly on graphs and leverage their structural information [17, 4, 46, 8]. Recently, GCN has been employed in various problems [59, 58, 45, 31, 55, 42, 20, 26, 28, 60]. In particular, label denoising with GCN [14, 10] has attracted much attention. In [14], its focus is on fully exploiting sufficient noisily labeled samples from target classes for FSL (which is against the standard FSL setting), and the core transfer problem in FSL remains untouched. To overcome these drawbacks, we choose to study a new SSFSL setting in our current work. In [10], a GCN-based denoising autoencoder is proposed to generate the classification weights for both source and target classes under generalized FSL [43], but no GCN-based label denoising problem is concerned in [10]. Moreover, although GCN has been directly used in a number of recent FSL methods [41, 15, 10, 14], our AdarGCN is different in that it can perform adaptive aggregation for FSL and is able to cope with both noisy and

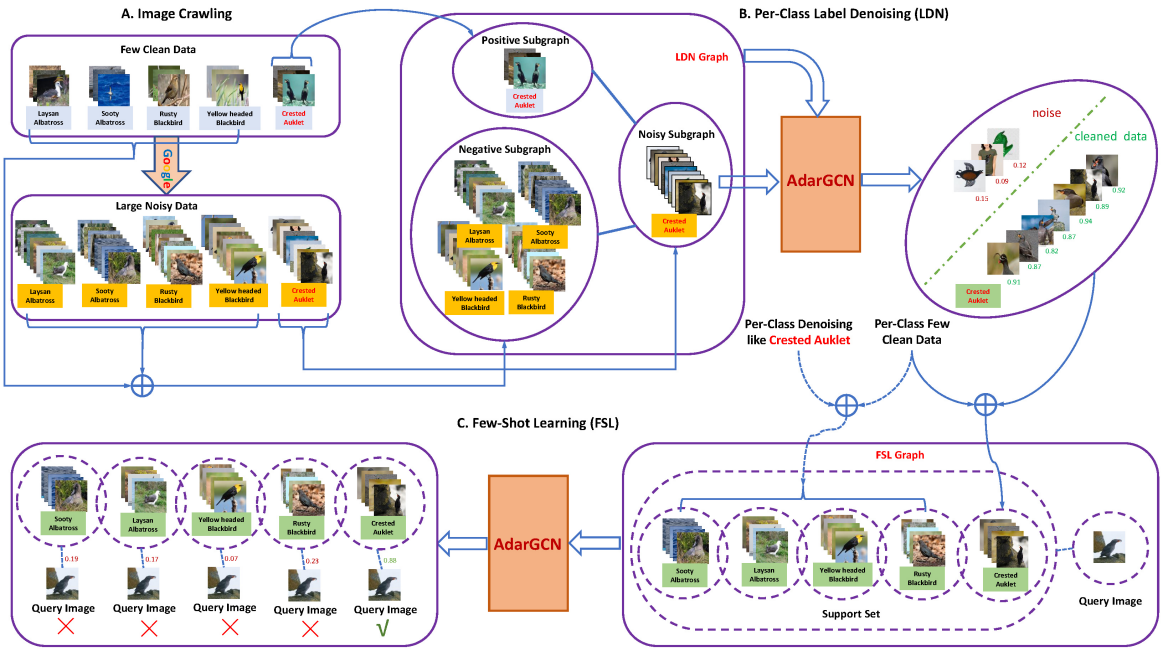


Figure 2. Schematic of our AdarGCN for scarce-source few-shot learning (SSFSL). A) We crawl sufficient images from the web by searching with the name of each source class. B) We use our GCN-based LDN model to clean the crawled web images which contain large amount of noise. C) With the cleaned web images as well as the original clean training images, we employ our GCN-based FSL model like under the conventional FSL setting. For both the LDN and FSL tasks, our AdarGCN is adopted.

outlying training samples. Our results show that the new GCN is clearly better than existing GCN based alternatives (see Table 4).

3. Methodology

3.1. Problem Definition

We formally define the scarce-source few-shot learning (SSFSL) problem as follows. Let C_s denote a set of source classes and C_t denote a set of target classes ($C_s \bigcap C_t = \emptyset$). We are given a k_1-shot sample set $\mathcal{D}_s$ from the source classes, a k-shot sample set $\mathcal{D}_t$ from the target classes, and a test set $\mathcal{T}$ from the target classes. The source class small sample set is denoted as $\mathcal{D}_s = \{(x_i, y_i)|y_i \in \mathcal{C}_s, i = 1, ..., N_s\}$, where y_i denotes the class label of sample x_i and N_s is the number of samples in $\mathcal{D}_s$. Since each source class from $\mathcal{D}_s$ has only k_1 labeled samples, we have $N_s = k_1|C_s|$. Similarly, we have $\mathcal{D}_t = \{(x_i, y_i)|y_i \in \mathcal{C}_t, i = 1, ..., N_t\}$, where $N_t = k|C_t|$ (each target class has k labeled samples). The goal of SSFSL is thus to train a model with $\mathcal{D}_s$ and $\mathcal{D}_t$ that can generalize well to $\mathcal{T}$. Note that our new SSFSL problem is clearly more challenging than the conventional FSL problem, since $\mathcal{D}_s$ only has few samples per class.

As in previous works on FSL [6, 47, 50, 48, 32], we train a FSL model with n-way k-shot classification tasks sampled from $\mathcal{D}_s$. Concretely, each n-way k-shot task is defined over a randomly-sampled episode $\{\mathcal{S}_e, \mathcal{Q}_e\}$, where $\mathcal{S}_e$ is the support set having n classes and k samples per class, and $\mathcal{Q}_e$ is the query set. Each episode is sampled as follows: we first sample a small set of source classes $C_e = \{C_i|i = 1, ..., n\}$ from C_s, and then generate $\mathcal{S}_e$ and $\mathcal{Q}_e$ by sampling k support samples and q query samples from each class in C_e, respectively. Formally, we have $\mathcal{S}_e = \{(x_i, y_i)|y_i \in C_e, i = 1, ..., n \times k\}$ and $\mathcal{Q}_e = \{(x_i, y_i)|y_i \in C_e, i = 1, ..., n \times q\}$, where $\mathcal{S}_e \bigcap \mathcal{Q}_e = \emptyset$. A FSL model is then trained by minimizing the gap between its predicted labels and the ground-truth labels over the query set $\mathcal{Q}_e$.

3.2. Framework Overview

To overcome the lack of training samples from source classes in SSFSL, we propose a two-stage framework shown in Fig. 2. (1) The first stage consists of image crawling and GCN-based label denoising (LDN), as depicted in Fig. 2(A) and Fig. 2(B) respectively. For each source class $c \in C_s$, we only have a small set of k_1 clean images initially: $X_c^s = \{x_i|(x_i, y_i) \in \mathcal{D}_s, y_i = c, i = 1, ..., N_s\}$. To augment the small set X_c^s, we then crawl another set of k_2 additional images from the web by image searching with the name of source class $c \in C_s$: $X_c^{web} = \{x_i|i = 1, ..., k_2\}$. As expected, there exists much noise in X_c^{web}. A GCN-based LDN method is thus deployed to reduce the noise in X_c^{web} and obtain a set of cleaned images $X_c^d \subset X_c^{web}$. We then define the set of cleaned samples as: $\mathcal{D}_d = \{(x, y)|x \in X_c^d, y = c, c = 1, ..., |C_s|\}$. (2) The second stage consists of GCN-based FSL, as shown in Fig. 2(C). We leverage both $\mathcal{D}_s$ and $\mathcal{D}_d$ to train our GCN-based FSL model. For

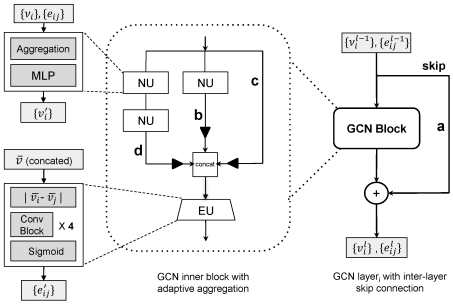


Figure 3. Illustration of the network architecture of our AdarGCN model. Notations: NU – node updating; EU – edge updating.

both stages, an adaptive aggregation GCN (AdarGCN) is used to deal with noisy training images.

3.3. Network Architecture of AdarGCN

Different from existing GCN models [41, 15, 10, 17, 23]), our AdarGCN, illustrated in Fig. 3, induces adaptive aggregation into GCN training to better control the information propagation from each node to the rest in the graph structure.

Formally, for the l-th GCN layer ($l = 1, ..., L$) of our AdarGCN, given the node feature matrix V^{l-1} and the edge feature matrix E^{l-1} as inputs, the output V^l can be obtained by adding the node feature matrix from the GCN inner block and that from the previous GCN layer:

$$V^l = V^{l-1} + V_{block}, \tag{1}$$

which is realized by the inter-layer skip-connection branch **a** (see Fig. 3). Within the GCN inner block of the l-th GCN layer, we design a multi-head multi-level aggregation module to aggregate the node features adaptively with different aggregation complexities (see Fig. 3). The node feature matrix and edge feature matrix are then updated successively.

Specifically, node feature updating is achieved by the adaptive aggregation among the three updating branches **c, b, d** with different degrees of aggregation. Branches **c, b, d** update the node features respectively with $0, 1, 2$ iterations: one iteration update is denoted by a Node Updating (NU) unit which consists of an aggregation module and a multi-layer perceptron (MLP) module. The outputs of branches **c, b, d** are given by:

$$\begin{aligned} &V_c^l = V^{l-1},\ \ V_b^l = f_{\theta_b}(E^{l-1} \cdot V^{l-1}), \\ &V_d^l = f_{\theta_{d1}}(E^{l-1} \cdot f_{\theta_{d2}}(E^{l-1} \cdot V^{l-1})), \end{aligned} \tag{2}$$

where θ_b collects the parameters of the MLP module in branch **b**, while θ_{d1} and θ_{d2} collect the parameters of the two MLP modules in branch **d**, respectively. As shown in Fig. 1, how far information can travel along each head/branch differs – **d** has the farthest influence whilst **c** the shortest (each node itself, to be precise). For adaptive aggregation, the adaptive weight of each of branches **c, b, d** is computed with a fully connected (FC) layer:

$$w_c = \text{FC}(V_c^l),\ w_b = \text{FC}(V_b^l),\ w_d = \text{FC}(V_d^l), \tag{3}$$

where $\text{FC}(\cdot)$ denotes the output of a FC layer, followed by a sigmoid function. The total node update within the GCN inner block is formulated as:

$$\bar{V}^l = \text{concat}(w_c \cdot V_c^l,\ w_b \cdot V_b^l,\ w_d \cdot V_d^l). \tag{4}$$

Note that more than three branches can be used here, but empirically we found that more branches bring no further gains. We thus use only three branches.

For edge feature updating, we denote it with an Edge Updating (EU) unit (see Fig. 3), which is used to learn the distance metric given node features as inputs. In this work, each EU unit includes a distance computing operation, 4 conv blocks, and a sigmoid activation function (see Fig. 3).

3.4. AdarGCN for LDN

Our AdarGCN is employed in both stages of our framework. In the first stage, it is for LDN over the noisy images crawled for each source class. Specifically, given a source class $c \in C_s$, we have a positive image set $X_c^+ = X_c^s$, a noisy image set $X_c^* = X_c^{web}$, and a negative image set $X_c^- = \{X_i^+ \cup X_i^* | i \in C_s, i \neq c\}$, as shown in Fig. 2(B). Before per-class LDN, we pretrain a simple embedding network (e.g. four-block ConvNet) on X_c^+ like ProtoNet [47] to extract d-dimensional image feature vectors, which is consistent with the second stage.

To construct an LDN graph for each source class $c \in C_s$, we generate a mini-batch by randomly selecting m^+ images from X_c^+, m^* images from X_c^*, and m^- images from X_c^- (see Fig. 2(B)). The image feature matrices of these three groups of samples are respectively denoted as $V_c^+ \in \mathbb{R}^{m^+ \times d}$, $V_c^* \in \mathbb{R}^{m^* \times d}$, and $V_c^- \in \mathbb{R}^{m^- \times d}$. The node feature matrix is thus defined as $V_c = [V_c^+; V_c^*; V_c^-] = [v_1; \cdots; v_M] \in \mathbb{R}^{M \times d}$, where $M = m^+ + m^* + m^-$. The initial symmetric adjacency matrix $A_c = \{a_{ij}\} \in \mathbb{R}^{M \times M}$ is formed as: $a_{ij} = 0$ if v_i and v_j respectively come from V_c^+ and V_c^-, and $a_{ij} = 1$ otherwise (see Fig. 2(B)). This way of constructing A_c ensures that the positive and negative samples cannot be directly confused by each other.

We denote the above LDN graph as $\mathcal{G}_c = (V_c, E_c)$, where the edge feature matrix $E_c = \{e_{ij}\} \in \mathbb{R}^{M \times M}$. To adapt the generic AdarGCN described in Sec. 3.3 to the LDN task, we make four modifications to its architecture: (1) For each edge updating (EU) unit, we set $e_{ij} = 0$ when $a_{ij} = 0$, after the sigmoid operation to avoid direct label confusion during label propagation. (2) Before the node updating (NU) unit of the first GCN layer, we perform EU to initialize E_c. (3) For each NU unit, we perform a linear transformation after the concat operation. (4) For the last

GCN layer, we drop the EU unit and use a sigmoid function to output the predicted score for each sample.

Note that our GCN-based LDN model can be regarded as a binary classifier, outputting 1 for positive samples and 0 for negative ones. However, unlike the traditional image classification, our model can make full use of uncertain samples (i.e. images from X_c^*) by aggregating similar nodes for more effective label propagation. Let $\hat{y}_i$ be the predicted score of each sample x_i ($i = 1, ..., M$). The loss for GCN-based LDN is defined as follows:

$$\mathcal{L}_{LDN} = -\frac{1}{m^+}\sum_{i=1}^{m^+}\log(\hat{y}_i) - \frac{1}{m^-}\sum_{i=M-m^-+1}^{M}\log(1-\hat{y}_i). \tag{5}$$

Although our GCN-based LDN model ignores the direct back-propagation w.r.t. the loss of noisy images (whose labels are uncertain), it can learn better representation for uncertain images by aggregating both certain and uncertain images, followed by back-propagation w.r.t. the loss of certain images. Our GCN-based LDN model is shown to outperform multi-layer perceptron (MLP) which copes with each sample independently (see Table 1), indicating the importance of certain and uncertain image aggregation.

Since each uncertain sample $x \in X_c^*$ can appear in multiple LDN graphs w.r.t. source class c, due to random sampling, we average the obtained multiple predicted scores as the probability of being positive for each sample x. If this probability is greater than a preset threshold, we then add (x, c) to $\mathcal{D}_d$.

3.5. AdarGCN for FSL

In the second stage, given $\mathcal{D}_s \cup \mathcal{D}_d$, we train our AdarGCN-based FSL model by episodic sampling. For each episode, we randomly select $n \times k$ samples to form the support set $\mathcal{S}_e$ and $n \times q$ samples to form the query set $\mathcal{Q}_e$. The embedding network f_φ is trained jointly with the GCN module to obtain the feature representations of all samples from $\mathcal{S}_e \cup \mathcal{Q}_e$: $v_i = f_\varphi(x_i)$, $i = 1, ..., n \times (k + q)$. Note that, although both transductive (with all query images in one trial) and non-transductive (with a single query image per trial) test strategies are followed in the state-of-the-art GCN-based FSL model in [15], we only adopt the non-transductive strategy. This is for fair comparison with most of the other state-of-the-art FSL methods which are non-transductive.

For each episode, we construct $n \times q$ graphs, each of which is constructed using $n \times k$ support samples and 1 query sample. Denote the $n \times q$ graphs as $\mathcal{G} = \{\mathcal{G}_t = (V_t, E_t) | t = 1, ..., n \times q\}$. For a single graph $\mathcal{G}_t$ in $\mathcal{G}$, $V_t = \{\overbrace{v_1^s, \cdots, v_{n\times k}^s}^{\mathcal{S}_e}, \overbrace{v_t^q}^{\mathcal{Q}_e}\}$. For symbol conciseness, we denote it as $V_t = \{v_i\}_{i=1}^{n\times k+1}$, along with $E_t = \{e_{ij}\}_{i,j=1,\cdots,n\times k+1}$, where v_i is the node feature obtained by the embedding network, $v_{n\times k+1}$ denotes the node feature of the query image, and e_{ij} is the edge feature w.r.t. v_i and v_j. For $v_i, v_j \in \mathcal{S}_e$, $e_{ij} = 1$ if v_i and v_j come from the same class and $e_{ij} = 0$ otherwise. When $v_i \in \mathcal{Q}_e$ or $v_j \in \mathcal{Q}_e$, we also set $e_{ij} = 0$ due to the unknown label of v_i or v_j.

The full GCN is composed of L GCN layers, each having the AdarGCN architecture shown in Fig. 3. In this work, we set $L = 3$ for fair comparison with other GCN based FSL methods [41, 15, 10] that also use three GCN layers. Given a graph $\mathcal{G}_t$ ($t = 1, ..., n \times q$), the inputs of the first GCN layer (i.e. node feature matrix V_t^0 and edge feature matrix E_t^0) are obtained in the way mentioned above (where we set $l = 0$). For the l-th GCN layer ($l = 1, ..., L$), the inputs V_t^{l-1} and E_t^{l-1} (from the previous layer) are updated to V_t^l and E_t^l.

For GCN training, we choose the binary cross-entropy loss between the ground-truth edge matrix $E^{gt} = \{e_{ij}^{gt}\}_{i,j=1,\cdots,n\times k+1}$ and the edge feature matrices of all L GCN layers $\{E_t^l = \{e_{ij}^l\}_{i,j=1,\cdots,n\times k+1}\}_{l=1}^L$, where $e_{ij}^{gt} = 1$ if x_i and x_j come from the same class and $e_{ij}^{gt} = 0$ otherwise. Formally, for each graph $\mathcal{G}_t$, the binary cross-entropy loss of the l-th GCN layer is defined as:

$$\mathcal{L}_l^t = -\sum_{i=1}^{n\times k+1}\sum_{j\neq i} e_{ij}^{gt}\cdot\log(e_{ij}^l) + (1-e_{ij}^{gt})\cdot\log(1-e_{ij}^l). \tag{6}$$

Taking all graphs (each for a query image) and all GCN layers on board, we compute the overall cross-entropy loss for a training episode as follows:

$$\mathcal{L}_{FSL} = \sum_{t=1}^{n\times q}\sum_{l=1}^{L}\lambda^l\cdot\mathcal{L}_l^t, \tag{7}$$

where λ^l denotes the loss weight of the l-th GCN layer.

For GCN inference, the edge feature matrix E_t^L of the last GCN layer can be used to predict the label of the unique query image. Concretely, the predicted scores of the query image are collected as $\hat{y} = \{\hat{y}_1, \cdots, \hat{y}_{n\times k}\}$, where $\hat{y}_i = e_{n\times k+1,i}^L$ ($0 \leq \hat{y}_i \leq 1$), being the predicted probability of the query image coming from the class to which the support sample x_i ($i = 1, ..., n \times k$) belongs. The classification probability of the query image is given by:

$$p_c = \sum_{i=1}^{n\times k}\mathbf{1}(y_i = c)\cdot\hat{y}_i / \sum_{i=1}^{n\times k}\hat{y}_i, \tag{8}$$

where $\mathbf{1}$ denotes the indicator function, y_i is the class label of support sample x_i, and c the c-th class label in the episode ($c = 1, ..., n$).

Method	*mini*ImageNet			CUB		
	k_1=10	k_1=20	k_1=50	k_1=10	k_1=20	k_1=50
FSL (w/o crawled noisy images)	40.91 ± 0.68	50.01 ± 0.66	55.04 ± 0.59	58.89 ± 0.71	68.33 ± 0.69	76.16 ± 0.62
FSL (w/ crawled noisy images)	59.20 ± 0.59	59.37 ± 0.61	59.64 ± 0.58	76.35 ± 0.57	76.83 ± 0.57	77.18 ± 0.56
FSL+LDN (LP)	60.21 ± 0.57	62.83 ± 0.55	64.74 ± 0.55	76.94 ± 0.49	77.98 ± 0.51	78.87 ± 0.54
FSL+LDN (MLP)	60.19 ± 0.58	62.88 ± 0.54	64.25 ± 0.60	77.10 ± 0.53	78.06 ± 0.48	78.92 ± 0.50
FSL+LDN (GCN [17])	61.22 ± 0.58	63.27 ± 0.59	65.36 ± 0.54	77.44 ± 0.52	78.56 ± 0.54	79.32 ± 0.49
FSL+LDN (ResGCN [23])	61.48 ± 0.61	63.79 ± 0.56	65.92 ± 0.56	77.52 ± 0.51	78.69 ± 0.52	79.69 ± 0.49
FSL+LDN (ours)	$\mathbf{63.37 \pm 0.53}$	$\mathbf{65.12 \pm 0.55}$	$\mathbf{66.85 \pm 0.52}$	$\mathbf{79.16 \pm 0.48}$	$\mathbf{79.82 \pm 0.51}$	$\mathbf{80.88 \pm 0.50}$

Table 1. Comparative results by various label denoising (LDN) methods under the new SSFSL setting (k_2=1,200). FSL denotes GCN-based FSL with our AdarGCN model.

4. Experiments

4.1. New SSFSL

Datasets and Settings. 1) Datasets. Two benchmarks are selected: (1) ***mini*-ImageNet**: This dataset [50] is derived from ILSVRC-12 [39]. It consists of 100 classes totally, with 600 images per class. As in [37], the training/validation/test split is set to 64/16/20 classes. (2) **CUB**: The fine-grained CUB [51] is particularly suitable for our new SSFSL setting. Since the number of images per class is less than 60, FSL on CUB is naturally a SSFSL problem. Although CUB has been widely used for FSL, we are the first to identify the problem and provide a solution. It consists of 200 bird species, and the training/validation/test split is set to 100/50/50 classes. In both datasets, each image is resized to 84×84.

2) SSFSL Settings. Let k_1 be the number of original clean images per training class (i.e. source class), and k_2 be the number of crawled noisy images per training class. In this work, we set k_1=10, 20, or 50, and k_2=1,200. As in the state-of-the-art GCN-based FSL models [41, 15], the four-block ConvNet network is used as the embedding network. For both LDN and FSL tasks involved in our new SSFSL setting, the same embedding network is used. As in [41, 15], the 5-way 5-shot accuracy is computed over 600 episodes randomly sampled from the test set: each test episode has 5 support images and 15 query images per class.

3) Implementation Details. (1) **GCN-Based LDN**: The four-block ConvNet network pretrained on the training set is used as the feature extractor. The dimensionality of the output features is 128. For GCN training over each training class, a mini-batch consists of three types of images from this class: 5 positive images, 5 negative images, and 50 crawled noisy images[1] (see Fig. 2). We construct an LDN graph over each mini-batch. We set a learning rate of 0.001, a dropout probability of 0.5, and a mini-batch size of 8. The Adam optimizer [16] is used over 500 training epochs. For each source class, we select the cleaned images with prediction scores ≥ 0.5 for the subsequent GCN-based FSL task.

[1]Out of these crawled web images, around 40% are noise. After LDN using our AdarGCN, this percent is reduced to around 10%. Some examples of the removed web images can be found in the suppl. material.

(2) **GCN-Based FSL**: With the same 4-block embedding network, our GCN model for FSL is trained by the Adam optimizer [16] with an initial learning rate of 0.001 and a weight decay of $1e-6$. We also use label smoothing as in [19]. During the training phase, we cut the learning rate in half every 10,000 episodes and set total training episodes as 50,000. In the 5-way 5-shot scenario, each mini-batch has 32 training episodes, and each episode consists of 25 support images and 5 query images (5 shot support samples and 1 query sample per class). Within a training episode, we construct a graph over 25 support images and 1 query image for each query image.

4) Compared Methods. Since our SSFSL setting includes both LDN and FSL tasks, we compare our AdarGCN-LDN and AdarGCN-FSL with various LDN and FSL alternatives, respectively. When comparing our AdarGCN-LDN with other LDN methods including label propagation (LP) [61, 62], MLP, GCN [17] and ResGCN [23], we employ the same subsequent FSL model (i.e. AdarGCN-FSL) for fair comparison. Note that a score threshold of 0.5 is selected for all LDN methods except LP to classify the positive and negative samples. For LP-based LDN, a score threshold of 0 is used otherwise, since the positive samples are labeled as '1' and the negative samples as '-1'. Similarly, when comparing our AdarGCN-FSL with various FSL baselines, we adopt the same LDN model (i.e. AdarGCN-LDN) for fair comparison. We focus on representative/state-of-the-art FSL methods including MatchingNet [50], MAML [6], ProtoNet [47], IMP [1], Baseline++ [3], GCN [41], wDAE-GNN [10], and EGCN [15].

Comparison to LDN Alternatives. The comparative results for the label denoising task are shown in Table 1. It can be seen that: (1) Adding the crawled images (although noisy) to the original few clean data leads to consistent and significant improvements for different values of k_1. The improvements are particularly salient when k_1 takes a smaller value. (2) All LDN methods can further improve the FSL performance (see 'FSL+LDN' vs. 'FSL (w/ crawled noisy images)') by imposing label denoising over the crawled images, showing the effectiveness of LDN under the SSFSL setting. (3) Our AdarGCN-LDN achieves the best label de-

Method	k_1	k_2	*mini*ImageNet	CUB
LDN+FSL (MatchingNet [50])	50	1,200	51.59 ± 0.62	66.12 ± 0.60
LDN+FSL (MAML [6])	50	1,200	59.67 ± 0.57	73.50 ± 0.54
LDN+FSL (ProtoNet [47])	50	1,200	64.73 ± 0.58	74.48 ± 0.56
LDN+FSL (IMP [1])	50	1,200	65.44 ± 0.57	78.61 ± 0.54
LDN+FSL (Baseline++ [3])	50	1,200	64.55 ± 0.55	77.90 ± 0.54
LDN+FSL (GCN [41])	50	1,200	64.80 ± 0.54	74.59 ± 0.49
LDN+FSL (wDAE-GNN [10])	50	1,200	63.26 ± 0.53	74.23 ± 0.50
LDN+FSL (EGCN [15])	50	1,200	65.12 ± 0.52	78.10 ± 0.51
LDN+FSL (ours)	50	1,200	$\mathbf{66.85 \pm 0.52}$	$\mathbf{80.88 \pm 0.50}$

Table 2. Comparative results by various FSL methods under the new SSFSL setting. LDN denotes GCN-based LDN with our AdarGCN.

GCN Model	k_1	k_2	LDN Task	FSL Task
AdarGCN (branch: b)	50	1,200	64.36 ± 0.57	63.13 ± 0.59
AdarGCN (branches: a, b)	50	1,200	65.92 ± 0.53	65.01 ± 0.55
AdarGCN (branches: a, b, c)	50	1,200	66.10 ± 0.52	65.88 ± 0.55
AdarGCN (branches: a, b, c, d)	50	1,200	$\mathbf{66.85 \pm 0.52}$	$\mathbf{66.85 \pm 0.52}$

Table 3. Ablation study results for our AdarGCN on both LDN and FSL tasks involved in new SSFSL setting over *mini*ImageNet.

noising results among all LDN methods including the GCN-based ones [17, 23]. This validates the effectiveness of our AdarGCN for LDN.

Comparison to FSL Alternatives. The comparative results for the FSL task are shown in Table 2. For fair comparison, all compared methods use the same set of cleaned samples obtained by our AdarGCN-LDN. We have the following observations: (1) Our AdarGCN-FSL performs the best among all FSL methods, validating the effectiveness of our AdarGCN for solving the FSL task. (2) Our AdarGCN-FSL clearly outperforms the latest GCN-based FSL methods [41, 15, 10], which suggests that adaptive aggregation indeed plays an important role when applying GCN to FSL. (3) Our AdarGCN-FSL also clearly leads to improvements over the state-of-the-art FSL baselines [1, 3].

Further Evaluations. 1) Ablation Study Results. The ablative results for our AdarGCN model on both tasks are presented in Table 3. We can observe from Table 3 that adding more branches leads to more performance improvements on both the LDN and FSL tasks, consistently demonstrating the contribution of each branch (a, b, c, or d in Fig. 3) in our AdarGCN model.

2) Effect of Different Values of k_2. To show the effect of different values of k_2 on our AdarGCN-LDN method, we gradually reduce k_2 from 1,200 to 300, and then evaluate the obtained LDN results by forwarding them to the subsequent FSL task. The results in Fig. 4 show that our AdarGCN-LDN method suffers from gradual performance degradation when k_2 decreases from 1,200 to 300.

3) Iterative Optimization for GCN-Based LDN. Note that the cleaned samples obtained by our AdarGCN-LDN can be easily exploited for another round of GCN-based LDN. In this work, for computational efficiency, we have not attempted such iterative optimization in the experiments reported so far. To show the effect of iterative optimization on our AdarGCN-LDN, we present the results by iterative optimization in Fig. 5. We see that our AdarGCN-LDN consistently achieves more improvements when more rounds of GCN-based LDN are included and becomes stable after three iterations/runs.

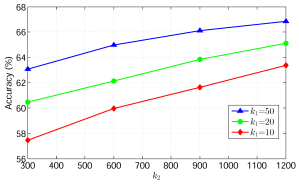


Figure 4. The effect of different values of k_2 on our AdarGCN-LDN method (k_1=10, 20, 50) over *mini*ImageNet.

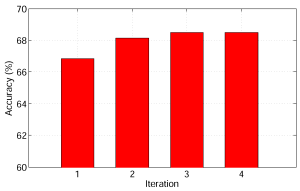


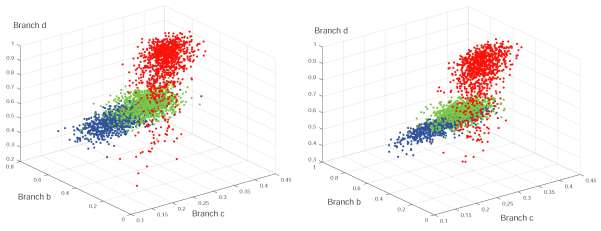
Figure 5. The effect of iterative optimization on our AdarGCN-LDN method ($k_1 = 50$, $k_2 = 1,200$) over *mini*ImageNet.

Models	GCN	*mini*ImageNet	CUB
MatchingNet [50]	no	55.30 ± 0.73	68.71 ± 0.81
ProtoNet [47]	no	65.77 ± 0.66	74.70 ± 0.57
R2-D2 [2]	no	68.40 ± 0.60	–
MAML [6]	no	63.11 ± 0.92	71.33 ± 0.71
Relation Net [48]	no	67.07 ± 0.63	69.66 ± 0.75
IMP [1]	no	68.10 ± 0.80	71.87 ± 0.69
Baseline++ [3]	no	66.43 ± 0.72	$75.39 \pm 0.62^{\dagger}$
TPN [30]	no	69.86 ± 0.58	–
MetaOptNet [19]	no	69.51 ± 0.51	77.10 ± 0.53
DN4 [27]	no	71.02 ± 0.64	74.92 ± 0.64
GCN [41]	yes	66.41 ± 0.60	74.07 ± 0.67
wDAE-GNN [10]	yes	65.91 ± 0.60	73.85 ± 0.74
EGCN [15]	yes	66.85 ± 0.62	74.58 ± 0.65
AdarGCN (ours)	yes	$\mathbf{71.48 \pm 0.48}^{\dagger}$	$\mathbf{78.04 \pm 0.45}$

Table 4. Comparative results for conventional FSL. † denotes that the result is reproduced since our data split of CUB is different from that in [3]. ‡ note that our AdarGCN achieves an even higher accuracy of **72.24** with six GCN layers (see Fig. 7).

4.2. Conventional FSL

Datasets and Settings. We further evaluate our AdarGCN-FSL under the conventional FSL setting. The full *mini*ImageNet and CUB are selected, where *mini*ImageNet has 600 samples per class and CUB has less than 60 samples per class. The non-transductive 5-way 5-shot test strategy is adopted under the conventional FSL setting. Note that only the 5-way 5-shot protocol is considered for fair comparison to the latest GCN-based FSL methods [41, 15] (the results with the 5-way 1-shot protocol can be found in the suppl. material). Moreover, the implementation details for GCN training remain largely unchanged compared to those described in Section 4.1. One exception is that: since the number of training samples in the CUB dataset is relatively small, we cut the learning rate in half every 5,000 episodes and set the total number of training episodes as 20,000 on CUB for better optimization.



(a) Training (b) Test

Figure 6. Illustration of weight distribution on the three branches **b, c, d** of different GCN layers obtained by our adaptive aggregation module over *mini*ImageNet. The visualization of adaptive aggregation is presented for both training and test phases. The red, green, and blue points denote the weights of GCN layer 1, 2, and 3, respectively.

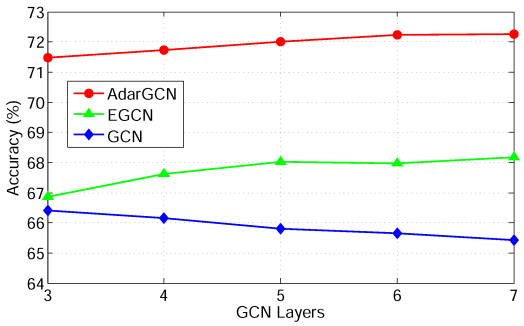


Figure 7. Comparative results among the three latest GCN-based FSL methods with deeper GCNs (GCN layers $\geq$ 3) for conventional FSL over *mini*ImageNet.

Comparison to FSL Baselines. Two groups of baselines are selected: (1) State-of-the-art GCN-based FSL methods [41, 15, 10]; (2) Representative/latest FSL methods (w/o GCN) [50, 47, 6, 48, 30, 2, 1, 3, 19]. The comparative results for conventional FSL are shown in Table 4. It can be seen that: (1) Our AdarGCN-FSL method yields 3–5% improvements over the latest GCN-based FSL methods [41, 15, 10], validating the effectiveness of adaptive aggregation for GCN-based FSL. (2) The improvements achieved by our AdarGCN-FSL method over the state-of-the-art FSL baselines [30, 2, 1, 3, 19] range from 1% to 6%, showing that AdarGCN has a great potential for FSL even with sufficient and clean training samples, due to its ability to limit the negative effect of outlying samples.

Further Evaluations. 1) Visualization of Adaptive Aggregation. By randomly sampling 1,000 query images respectively from the training set and the test set, we visualize the weights of the three branches **b, c, d** of different GCN layers obtained by our adaptive aggregation module (see Fig. 3). The visualization results over *mini*ImageNet are presented in Fig. 6. It shows that each GCN layer has a significantly different weight distribution. This provides direct evidence that adaptive aggregation is indeed needed in GCN-based FSL. Further, it is also noted that the weight of branch **c** is forced to be significantly larger than those of the other two branches for the outlying samples so that their negative effect can be effectively limited (see the suppl. material), demonstrating that the negative impact of a noisy or outlying training sample can be limited to a small neighborhood of the corresponding node, making the model more robust against the clean but outlying samples under the conventional FSL setting.

2) FSL with Deeper GCN. In the above experiments, all GCN-based FSL methods uniformly set the number of GCN layers to 3, for fair comparison. It is well-known that deeper GCNs often lead to performance degradation. However, since both adaptive aggregation and skip connection are included in our AdarGCN model, it is possible to go deeper with our AdarGCN. To demonstrate this, we provide the comparative results obtained using three GCN-based FSL methods (i.e. GCN [41], EGCN [15], and our AdarGCN) in Fig. 7, where the number of GCN layers ranges from 3 to 7. As expected, the performance of GCN [41] drops when it goes deeper. However, both EGCN [15] and our AdarGCN achieve performance improvements when more GCN layers are stacked, and our AdarGCN consistently outperforms EGCN. This can be explained as follows: our AdarGCN leverages both adaptive aggregation and skip connection, while only skip connection is concerned in EGCN. These results suggest that skip connection enables relative performance gains with deeper GCNs. But the adaptive aggregation (induced by our AdarGCN) is crucial for the absolute performance. One possible reason is that it helps avoid the over-smooth effect, because it can adaptively aggregate over different reaches of message passing on the graph.

5. Conclusion

In this paper, we have defined a new scarce-source few-shot learning (SSFSL) setting. To overcome the training data scarcity problem, we chose to augment the training data by crawling sufficient images from the web. Since the crawled images are noisy, we then proposed a GCN-based LDN method to clean the crawled noisy images. Further, with the cleaned web images and the original clean training images as the new training set, we proposed a GCN-based FSL method. For both the LDN and FSL tasks, we designed an AdarGCN model which can perform adaptive aggregation to deal with noisy training data more effectively. Extensive experiments show that our AdarGCN outperforms the state-of-the-art alternatives under both the new SSFSL and the conventional FSL settings.

Acknowledgement Zhiwu Lu is partially supported by National Natural Science Foundation of China (61976220 and 61832017), and Beijing Outstanding Young Scientist Program (BJJWZYJH012019100020098).

References

[1] Kelsey R. Allen, Evan Shelhamer, Hanul Shin, and Joshua B. Tenenbaum. Infinite mixture prototypes for few-shot learning. In *ICML*, pages 232–241, 2019.

[2] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019.

[3] 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.

[4] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016.

[5] Li Fe-Fei, Rob Fergus, and Pietro Perona. A bayesian approach to unsupervised one-shot learning of object categories. In *ICCV*, pages 1134–1141, 2003.

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

[7] Hang Gao, Zheng Shou, Alireza Zareian, Hanwang Zhang, and Shih-Fu Chang. Low-shot learning via covariance-preserving adversarial augmentation networks. In *NIPS*, pages 975–985, 2018.

[8] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *KDD*, pages 1416–1424, 2018.

[9] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, pages 4367–4375, 2018.

[10] Spyros Gidaris and Nikos Komodakis. Generating classification weights with GNN denoising autoencoders for few-shot learning. In *CVPR*, pages 21–30, 2019.

[11] Jiechao Guan, Zhiwu Lu, Tao Xiang, Aoxue Li, An Zhao, and Ji-Rong Wen. Zero and few shot learning with semantic feature synthesis and competitive learning. *TPAMI*, 2020.

[12] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *ICCV*, pages 3037–3046, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[14] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, Ondrej Chum, and Cordelia Schmid. Graph convolutional networks for learning with few clean and many noisy labels. *arXiv preprint arXiv:1910.00324*, 2019.

[15] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D. Yoo. Edge-labeling graph neural network for few-shot learning. In *CVPR*, pages 11–20, 2019.

[16] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[19] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, pages 10657–10665, 2019.

[20] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein. CayleyNets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2018.

[21] Aoxue Li, Tiange Luo, Zhiwu Lu, Tao Xiang, and Liwei Wang. Large-scale few-shot learning: Knowledge transfer with class hierarchy. In *CVPR*, pages 7212–7220, 2019.

[22] Fei-Fei Li, Robert Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(4):594–611, 2006.

[23] Guohao Li, Matthias Müller, Ali Thabet, and Bernard Ghanem. Can GCNs go as deep as CNNs? In *ICCV*, 2019.

[24] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[25] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *CVPR*, pages 1–10, 2019.

[26] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI*, pages 3538–3545, 2018.

[27] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*, pages 7260–7268, 2019.

[28] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Combinatorial optimization with graph convolutional networks and guided tree search. In *Advances in Neural Information Processing Systems*, pages 539–548, 2018.

[29] Guangzhen Liu and Zhiwu Lu. Discriminativeness-preserved domain adaptation for few-shot learning. *IEEE Access*, 8:168405–168413, 2020.

[30] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. In *ICLR*, 2019.

[31] Jianxin Ma, Peng Cui, Kun Kuang, Xin Wang, and Wenwu Zhu. Disentangled graph convolutional networks. In *ICML*, pages 4212–4221, 2019.

[32] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.

[33] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[34] Zhimao Peng, Zechao Li, Junge Zhang, Yan Li, Guo-Jun Qi, and Jinhui Tang. Few-shot image recognition with knowledge transfer. In *ICCV*, pages 441–449, 2019.

[35] Hang Qi, Matthew Brown, and David G. Lowe. Low-shot learning with imprinted weights. In *CVPR*, pages 5822–5830, 2018.

[36] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *CVPR*, pages 7229–7238, 2018.

[37] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[38] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *ICCV*, pages 331–339, 2019.

[39] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[40] 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.

[41] Victor Garcia Satorras and Joan Bruna. Few-shot learning with graph neural networks. In *ICLR*, 2018.

[42] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607, 2018.

[43] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero-and few-shot learning via aligned variational autoencoders. In *CVPR*, pages 8247–8255, 2019.

[44] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Abhishek Kumar, Rogerio Feris, Raja Giryes, and Alex Bronstein. Delta-encoder: an effective sample synthesis method for few-shot object recognition. In *Advances in Neural Information Processing Systems*, pages 2850–2860, 2018.

[45] Lei Shi, Yifan Zhang, Jian Cheng, and Hanqing Lu. Two-stream adaptive graph convolutional networks for skeleton-based action recognition. In *CVPR*, pages 12026–12035, 2019.

[46] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, pages 3693–3702, 2017.

[47] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087, 2017.

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

[49] Satoshi Tsutsui, Yanwei Fu, and David Crandall. Meta-reinforced synthetic data for one-shot fine-grained visual recognition. In *NIPS*, pages 3057–3066, 2019.

[50] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[51] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[52] Yu-Xiong Wang, Ross Girshick, Martial Hebert, and Bharath Hariharan. Low-shot learning from imaginary data. In *CVPR*, pages 7278–7286, 2018.

[53] Ziyang Wu, Yuwei Li, Lihua Guo, and Kui Jia. Parn: Position-aware relation networks for few-shot learning. In *ICCV*, pages 6659–6667, 2019.

[54] Chen Xing, Negar Rostamzadeh, Boris Oreshkin, and Pedro O O. Pinheiro. Adaptive cross-modal few-shot learning. In *NIPS*, pages 4847–4857, 2019.

[55] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, pages 974–983, 2018.

[56] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[57] Hongguang Zhang, Jing Zhang, and Piotr Koniusz. Few-shot learning via saliency-guided hallucination of samples. In *CVPR*, pages 2770–2779, 2019.

[58] Li Zhang, Xiangtai Li, Anurag Arnab, Kuiyuan Yang, Yunhai Tong, and Philip HS Torr. Dual graph convolutional network for semantic segmentation. *arXiv preprint arXiv:1909.06121*, 2019.

[59] Long Zhao, Xi Peng, Yu Tian, Mubbasir Kapadia, and Dimitris N Metaxas. Semantic graph convolutional networks for 3D human pose regression. In *CVPR*, pages 3425–3435, 2019.

[60] Ling Zhao, Yujiao Song, Chao Zhang, Yu Liu, Pu Wang, Tao Lin, Min Deng, and Haifeng Li. T-GCN: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[61] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, pages 321–328, 2004.

[62] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, pages 912–919, 2003.