Relational Generalized Few-Shot Learning

Xiahan Shi[1]
xiahan.shi@de.bosch.com
Leonard Salewski[1]
leonard.salewski@gmail.com
Martin Schiegg[1]
martin.schiegg@de.bosch.com
Max Welling[2]
m.welling@uva.nl

[1] Bosch Center for Artificial Intelligence
Robert-Bosch-Campus 1,
Renningen, Germany
https://www.bosch-ai.com/

[2] University of Amsterdam
Science Park 904,
Amsterdam, the Netherlands
http://amlab.science.uva.nl/

Abstract

Transferring learned models to novel tasks is a challenging problem, particularly if only very few labeled examples are available. Most proposed methods for this few-shot learning setup focus on discriminating novel classes only. Instead, we consider the extended setup of *generalized few-shot learning* (GFSL), where the model is required to perform classification on the *joint* label space consisting of both previously seen and novel classes. We propose a graph-based framework that explicitly models relationships between *all* seen and novel classes in the joint label space. Our model *Graph-convolutional Global Prototypical Networks* (GcGPN) incorporates these inter-class relations using graph-convolution in order to embed novel class representations into the existing space of previously seen classes in a globally consistent manner. Our approach ensures both fast adaptation and global discrimination, which is the major challenge in GFSL. We demonstrate the benefits of our model on two challenging benchmark datasets.

1 Introduction

Few-shot learning (FSL) [3, 10, 27, 41] is inspired by the human ability to learn new concepts from very few or even only one example. This extreme low-data setup is particularly challenging for deep neural networks, which require large amounts of data to ensure generalization. FSL has mostly been approached from the meta-learning perspective, focusing on the problem setup of *N-way K-shot classification*. For each N-way K-shot task, the model has to discriminate N novel few-shot classes with only K labeled examples

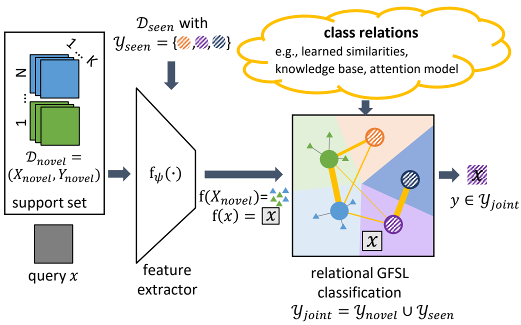

Figure 1: In each episode, the label space of seen classes is extended by novel classes from which only few samples are presented (support set). Our framework is able to exploit a relational graph between classes to improve the transfer from seen to novel classification tasks.

available per class. Unlike in standard transfer learning, meta-learning requires the model to adapt well across a *series of various* previously unknown tasks instead of a fixed, *specific* target task. Therefore, efficient *on-the-fly* model adaptation based on very few examples is at the core of most FSL models [11, 13, 14, 23, 35, 39, 45].

However, this FSL problem setup focuses only on discriminating novel classes from each other and offers no incentive for the model to remember classes previously seen during training or to maintain a globally consistent label space. However, we would like the model to *incorporate* few-shot novel classes into the label space of previously seen classes. This leads us to an extended scenario called *generalized few-shot learning* (GFSL), where the model has to discriminate the *joint* label space consisting of both previously seen and novel classes. This terminology is adopted from zero-shot learning (ZSL) and generalized zero-shot learning (GZSL), where novel classes come with no labeled examples at all and classification relies on side information such as attributes or semantic label embeddings [1, 48]. It is a well-known observation that many ZSL models fail dramatically at discriminating the joint label space (GZSL) despite good performance on novel label space (ZSL) [6, 48]. Similarly, GFSL is a more challenging task compared to FSL due to the trade-off between fast adaptation to novel classes and maintaining a global consistency across the joint label space.

We address the GFSL problem setup by explicitly modeling inter-class relationships as a weighted graph. We propose the *Graph-convolutional Global Prototypical Network* (GcGPN) which models representative prototypes for all novel and seen classes *jointly*. In particular, the prototypes are updated by graph convolutional operations [18] according to the relationship graph. Fig. 1 provides an illustration of our approach. To summarize, our **main contributions** are: We propose the first flexible framework for relational GFSL that

(1) considers an *arbitrary weighted graph* describing relations between classes (from any source of side information, attention mechanism or other similarity measures),

(2) applies graph-convolution for modeling prototypes and allows for end-to-end learning,

(3) accommodates previous (G)FSL methods [14, 39] as special cases and

(4) achieves state-of-the-art performance on GFSL tasks.

2 Related Work

Few-shot learning (FSL) has been approached from different perspectives including mimicking the human learning behavior by modeling high-level concepts [20], learning similarity measures [19] and extending deep neural networks by an external memory module to allow for direct incorporation of few-shot examples [16, 36]. Moreover, recent *meta-learning* approaches focus on the N-way K-shot setup and can be divided in two categories: Optimization-based methods [2, 11, 35] rely on a meta model that learns an optimal strategy which is carried out by an inner model to efficiently adapt to varying novel tasks. Distance-based methods such as *Matching Networks* [45] and *Prototypical Networks* [39] perform nearest-neighbor-based classification with a learned distance measure. Despite its simplicity, the method in [39] achieves excellent performance and has inspired extensions which parameterize the distance measure or the prototype mechanism in a more flexible way [40].

Generalized few-shot learning (GFSL) in the context of meta-learning is not yet a well-studied setup. Alternatives to meta-learning include matching seen and novel feature spaces [38, 44], modeling the global class structure in the joint label space [24], obtaining transferable features from a hierarchy between seen and novel classes via clustering [21], propagating labels from class-level to instance-level graphs [51], and learning generative models to synthesize additional image features for the few-shot classes [49] or additional samples [22]. These methods, however, require knowledge about all novel classes *a priori*, *e.g*. by leveraging a pre-trained feature space or graph construction, which is in contrast to the meta-learning setup where the model must be able to adapt on-the-fly to unknown and varying novel classes. The most relevant work to our target setup is *Dynamic FSL without Forgetting* [14] (DFSLwoF), which utilizes within a meta-learning setup an attention-based weight generator for novel classes to extend the classifier from seen classes to the joint label space. Its connections to our work is discussed in detail in Section 4.2. Their work is extended in [15] by a graph neural network (GNN) based denoising autoencoder to regularize the class prototypes, where the underlying graph is a special case of our fully connected setup. Further, the GNN layer consists of a separate neighbor-aggregation block based on Relation Networks [37] and an update-block to combine the prototypes with their respective neighbor-information. In contrast, our method uses a simpler model structure while providing a more general framework to include side information, and is trained end-to-end instead of the two-stage training procedure in [15, 37]. An orthogonal approach has been proposed in [34], where a pre-trained network can be extended to additional few-shot classes by predicting the final-layer parameters from the activations. This method is most notably useful when working with an existing, very strong model where further training is hard to realize.

Side Information plays a crucial role in zero-shot learning (ZSL), where no labeled examples are available for novel classes at all. In particular, ZSL methods typically build on side information from knowledge graphs [28], semantic word embeddings [26, 33] or visual attributes [8]. Generalization can be achieved by relating visual image features to semantic side information either through a learned mapping or a joint embedding space [1, 12, 30, 38]. Furthermore, graph-convolutions can be applied to distill information from relational knowledge graphs and class embeddings to predict the classifier weights for unseen classes [47]. Apart from ZSL, a range of FSL methods exist that incorporate side information, e.g., to regularize the feature space with textual embeddings for alignment on a distributional level [43], to use attention mechanisms for synthesizing additional training examples for few-shot classes [42], or to match a visual classifier with a knowledge-based representation [32]. Besides the low-shot data regime, relational information has also been used to improve loss functions for deep learning in general [50].

Graph-convolutional networks (GCN) [7, 9, 18] are a powerful tool to jointly learn node representations for inherently graph-structured data such as items in recommender systems or users of social networks [6]. Graph-based methods have been applied to FSL [13, 17] by representing *image instances* with graph nodes in an N-way K-shot classification setup. In contrast, we represent *classes* by graph nodes with a GFSL setup. Class-level graph-convolution has been used in a similar way for ZSL [47]. Alternatively, GCNs may exploit the manifold structure in the data to propagate labels from labeled to unlabeled images by using edge weights that depend on learned distances in the feature space [25].

3 (Generalized) Few-Shot learning

Few-shot learning (FSL) We consider N-way K-shot classification, which is the most widely studied problem setup for FSL. The classifier has to perform a series of N-way K-shot tasks, where each task consists of N previously unseen, novel classes with K labeled examples each (usually $K \leq 5$). More precisely, let $\mathcal{Y}_{\text{novel}}$ denote the novel class label space with $|\mathcal{Y}_{\text{novel}}| = N$, and let $\mathcal{D}_{\text{novel}} = \cup_{n=1}^{N} \{(x_{n,k}, y_n)\}_{k=1}^{K}$ denote the *support set*, where $x_{n,k}$ is the k-th labeled example of the class with label y_n. For a new query example x, the FSL prediction is given by

$$\hat{y} = \arg\max_{y \in \mathcal{Y}_{\text{novel}}} p_\psi(y|x, \mathcal{D}_{\text{novel}}). \tag{1}$$

N-way K-shot classification considers FSL from a *meta-learning* perspective. Unlike in standard transfer learning, the goal is not to generalize to a *specific* novel label space but to adapt and perform well across a series of *various* novel label spaces presented at test time. Therefore, most FSL methods adopt *episodic training* [45], where a new N-way K-shot task gets randomly sampled from a larger training set in every episode. This involves randomly selecting N simulated[1] novel classes and sampling K support set examples per class along with a batch $\mathcal{B}$ of query examples. The loss on this batch is given by $\frac{1}{|\mathcal{B}|} \sum_{(x,y) \in \mathcal{B}} -\log p_\psi(y|x, \mathcal{D}_{\text{novel}})$, which is used to train the model parameters ψ.

An FSL model is only concerned with discriminating the novel label space since all test time queries, by design of the task setup, come from one of the novel classes. Hence the $\arg\max$ in eq. (1) is only over $\mathcal{Y}_{\text{novel}}$. Classes seen during training no longer play a role at test time. This setup emphasizes fast adaptation to varying new tasks but does not encourage the model to accumulate knowledge, which may not always be very practical. Many real-world applications require the model to incorporate novel few-shot classes into the existing space of seen classes while maintaining global discrimination. Therefore, we consider the extended setup of *generalized few-shot learning* (GFSL) with test time queries that may come from both novel and seen classes.

Generalized few-shot learning (GFSL) In generalized N^+-way K-shot classification, the model has to discriminate the joint label space $\mathcal{Y}_{\text{joint}} = \mathcal{Y}_{\text{seen}} \cup \mathcal{Y}_{\text{novel}}$ consisting of the novel few-shot classes and all previously seen training classes. We denote the training set by $\mathcal{D}_{\text{seen}} = \cup_{n=1}^{N_{\text{seen}}} \{(x_{n,k}, y_n)\}_{i=1}^{K_n}$, where N_{seen} is the number of training classes and K_n is the number of labeled examples available for class $y_n \in \mathcal{Y}_{\text{seen}}$. In general, $N_{\text{seen}} \gg N$ and $K_n \gg K$. For a new query x, a GFSL model performs

$$\hat{y} = \arg\max_{y \in \mathcal{Y}_{\text{joint}}} p_\psi(y|x, \mathcal{D}_{\text{novel}}). \tag{2}$$

In contrast to eq. (1), the $\arg\max$ is over $\mathcal{Y}_{\text{joint}}$ instead of $\mathcal{Y}_{\text{novel}}$ since a query x may come from any of the seen and novel classes. In particular, GFSL requires discrimination of a much larger label space than FSL ($N^+ := N + N_{\text{seen}}$ instead of N). In addition, the model has to maintain a globally consistent joint label space while, at the same time, achieve fast adaptation to novel classes based on very few examples. In general, we cannot expect FSL models to perform well in GFSL since there is no explicit reward to remember the training classes and learn a well-separated joint label space.

4 Graph-convolutional Global Prototypical Networks

We propose *Graph-convolutional Global Prototypical Networks* (GcGPN) to perform relational GFSL. The key idea is to explicitly model and incorporate the relationships among *all*

[1] The "novel" classes at training time are randomly sampled from the *training* classes in order to simulate the test time setup, but they are *disjoint to the real novel classes at test time*.

(*e.g*. including seen and novel) classes through a weighted graph when learning class representations (so-called *prototypes*). This addresses the challenge of GFSL to maintain *global* consistency and discrimination when incorporating novel classes into an existing space of seen classes.

4.1 GcGPN: Model Overview

Fig. 1 illustrates our method, and more details on our framework are visualized in the supplementary. First, GcGPN maps all support set and query examples into a d-dimensional feature space by a feature extractor $f_\psi(\cdot)$. Next, initial prototypes $c_n \in \mathbb{R}^d$, $n = 1,\dots,N_{\text{seen}}+N$, are computed for all classes. While seen class prototypes are learned as model parameters, novel class prototypes are initialized on-the-fly since the novel label space varies at test time. The novel initial prototypes are given by the per-class average $c_n = \frac{1}{K}\sum_{i=1}^{K}\bar{z}_{n,i}$ of the normalized support set examples $\bar{z}_{n,k} = \frac{f_\psi(x_{n,k})}{||f_\psi(x_{n,k})||}$, $n = N_{\text{seen}}+1,\dots,N_{\text{seen}}+N$, as in [14, 39]. Then, a graph-convolution block $\tilde{g}(\cdot)$ as defined below updates these node initializations jointly according to the inter-class relationships specified in the edge weights. The updated prototypes c'_n, $n = 1,\dots,N_{\text{seen}}+N$, are then adapted representations of the joint label space of the N^+-way K-shot task at hand. Finally, the predicted class probabilities for a query x are obtained from its cosine similarities between its feature representation and the updated prototypes:

$$p(y=n|x,c'_1,\dots,c'_{N_{\text{seen}}+N}) = \frac{\exp\left(\tau\cos(f_\psi(x),c'_n)\right)}{\sum_{m=1}^{N_{\text{seen}}+N}\exp\left(\tau\cos(f_\psi(x),c'_m)\right)}, \tag{3}$$

where τ is a learnable temperature. In [14], this is referred to as the "cosine classifier". We adopt it since it was found to be preferable over the originally proposed L2 distance [39] when combining existing and novel classes [4, 14]. To train the model, we use the cross-entropy loss on eq. (3). Note that the sum in eq. (3) is over *all* class prototypes in the *joint* label space, which is in accordance with eq. (2) and differs from the FSL objective. Further, we apply episodic training for GFSL: For each episode, N out of N_{seen} training classes are sampled to act as novel classes and the remaining $N_{\text{seen}}-N$ are treated as the label space of seen classes. Contrary to an FSL episode, the GFSL query set $\mathcal{Q}$ must also contain examples from the seen classes, thus rewarding the model for maintaining global discrimination instead of focusing only on the novel classes. In every episode, the gradient of the loss is computed and all learnable parts of the model get updated including the parameters ψ of the feature extractor, the initial prototypes c_n, $n = 1,\dots,N_{\text{seen}}$ of seen classes, trainable components of the graph-convolution block $\tilde{g}(\cdot)$ and the classifier temperature τ. Unlike previous work [14], our model does not require multi-stage training but trains all parts of GcGPN jointly.

The graph-convolution block The graph-convolution block $\tilde{g}(\cdot)$ is at the core of GcGPN. To recap, a graph-convolution block [18] consists of L graph-convolution layers $g(\cdot) = g^L(g^{L-1}(\dots(g^1(\cdot))))$ on a graph of V nodes, which is given in its general form by

$$X^{(l+1)} = g^l(X^{(l)}) = \rho(\sum_{B\in\mathcal{A}} BX^{(l)}\theta_B^{(l)}), \tag{4}$$

where $l \in \{1,\dots,L\}$ indexes the layer of the block, $X^{(l)}$ is the $(V\times d_l)$-dimensional input matrix containing d_l-dimensional node features in its rows, $\mathcal{A}$ denotes a set of $(V\times V)$-dimensional linear node operators such as the adjacency or weight matrix of the graph, $\theta_B^{(l)}$

with $B \in \mathcal{A}$ denotes a $(d_l \times d_{l+1})$-dimensional matrix containing learnable parameters of the l-th layer and $\rho(\cdot)$ is a non-linearity. For example, if B is the adjacency matrix of the graph, the local convolutional operation $BX^{(l)}$ computes for each node the sum of its neighbors.

In our GFSL model, eq. (4) is applied to the class prototypes to model relations between them. More precisely, let C denote the $((N_{\text{seen}} + N) \times d)$-dimensional matrix, where the n-th row contains the initial prototype c_n of the n-th class. Further, let the operator entries $B_{m,n}$ encode a similarity score between classes represented by c_m and c_n. Then, L-layered graph-convolution block takes $X^{(0)} := C$ as input and computes the updated class prototypes as $C' = g(C) = g^L(\dots g^1(C))$ according to eq. (4).

Note that graph-convolution can be interpreted as performing two steps to update a class prototype: First, a weighted average of similar prototypes is computed with weights given in the *convolution operator* B and second, a non-linear *post-convolution transform* is applied given by $\theta_B^{(l)}$ and $\rho(\cdot)$. The first part models interactions among classes and operates only on node-level, while the second part operates only on feature-level by applying the same transform to all classes.

The general graph-convolution definition from eq. (4) operates in Euclidean spaces. We adopt the graph-convolution block to be consistent with cosine similarities as used in eq. (3) by intermediate normalizations $\bar{x} = \frac{x}{\|x\|}$ to keep the prototypes at unit length. Our graph-convolution block is thus defined by $C' = \tilde{g}(C) = \tilde{g}^L(\dots \tilde{g}^1(C))$ with $\tilde{g}^l(C^{(l)}) = \overline{\rho(\sum_{B \in \mathcal{A}} s_B B \bar{C}^{(l)} \theta_B^{(l)})}$, $l = 1, \dots, L$, where scalars s_B are introduced to trade-off between different operators in $\mathcal{A}$. The relational information between the classes is modeled by the operators $B \in \mathcal{A}$ and we call these the *semantic* operators.

Graph-convolutional operators: In typical applications for graph convolutional networks such as recommender systems or social network analysis, the adjacency matrix is a popular choice [18]. Inter-class relationship modeling suggests to more generally use a weighted graph, where entries express some notion of similarity. In general, there are several possible strategies to define the operator entries $B_{n,m}$:

(1) Any distance/similarity on the prototype space such as L2 distance or cosine similarity.

(2) Learned similarities, either using a standard measure in a learned transformed space or learning a flexible transform of the element-wise absolute differences as done in [13].

(3) Similarities within a learned key space as proposed in [14]. This means learning a key k_n for each class n and obtaining inter-class similarities by matching the corresponding keys in the key space.

(4) Side information from external sources of information such as knowledge graphs or semantic models. This can be extracted as relational scores (*e.g.* shortest-path distance between two class names in an ontology [47]) directly or obtained from per-class embeddings such as word vectors or attributes [1] by computing pairwise similarity.

Note that also sparse graphs (as arising from *e.g.* sparse knowledge graph structure, operator thresholding, adjacency) are covered and higher-order structures are easily incorporated, *e.g.* by adding higher-order versions of the adjacency matrices to $\mathcal{A}$ or using similarity scores that already contain higher-order information such as the path similarity in WordNet [47].

Due to the multi-operator design, our model can naturally combine multiple of the above strategies, resulting in a general and flexible framework for relational GFSL.

Post-convolutional transforms: The parameters θ_B are another crucial component of the model. Since the output of $\tilde{g}(\cdot)$ are updated prototypes, we choose θ_B to preserve the dimensionality. Using a learnable quadratic weight matrix is the most straight-forward approach, although restricting θ_B similar to [14] is also a competitive option.

4.2 Generalization of existing approaches

There is a naive extension of the state-of-the-art FSL method *Prototypical Networks* [39] (PN) to the GFSL setup: For a readily trained PN, seen class prototypes can be obtained as feature averages over all available training examples for that class[2], which can then be used to perform GFSL tasks. This extension referred to as PN$^+$ corresponds to the assumption that the learned feature extractor and prototype mechanism would naturally generalize over the joint label space. Thus, there is no explicit inter-class dependency modeling, which is equivalent to setting all $B \in \mathcal{A}$, $\rho(\cdot)$, s_B, $\theta_B^{(l)}$ to identity matrices or functions in our GcGPN framework. We will observe in the experiments later that this assumption is not appropriate.

The model in [14] addresses GFSL successfully by an attention-based weight generator that computes classifier weights w^* for novel classes based on their support sets and their similarities to seen classes. Both our model and theirs utilize a cosine classifier. However, while the cosine classifier in our model operates on representative prototypes in the feature space, theirs operates in the weight space and computes cosine similarities between seen class weights and transformed support set image features. The *Average Weight Generator* variant in [14] can be recovered in our framework by using a GcGPN with prototype initialization as in sec. 4.1 and one graph-convolution layer ($L = 1$) with $\mathcal{A} = \{\hat{B}_1, \hat{B}_2\}$ containing two block-structured operators

$$\hat{B}_1 = \begin{pmatrix} I_{N_{\text{seen}} \times N_{\text{seen}}} & 0_{N_{\text{seen}} \times N} \\ 0_{N \times N_{\text{seen}}} & 0_{N \times N} \end{pmatrix}, \qquad \hat{B}_2 = \begin{pmatrix} 0_{N_{\text{seen}} \times N_{\text{seen}}} & 0_{N_{\text{seen}} \times N} \\ 0_{N \times N_{\text{seen}}} & I_{N \times N} \end{pmatrix} \tag{5}$$

with identity matrix on the seen- and novel-class blocks respectively and zeros elsewhere. This corresponds to not modeling inter-class relations at all. $\theta_{\hat{B}_1}$ is the identity matrix and $\theta_{\hat{B}_2}$ is a learnable diagonal matrix. The *Attention Weight Generator* variant in [14] can be recovered by adding one more operator to $\mathcal{A}$, whose lower-left block contains attention weights that are obtained by matching the respective classes in a learned key space (see semantic operator option 3 above). This corresponds to an underlying relational graph with weighted directed edges from seen to novel classes, such that novel prototypes do not only depend on the support set but also on similar seen classes.

To summarize, GcGPN generalizes over [14] in several respects: *(i)* We use a fully connected graph, allowing not only relations from seen to novel classes but among *all* classes (i.e., operators in $\mathcal{A}$ may be full matrices); *(ii)* our framework accommodates any kind of similarity modeling (not only attention matching) and offers a natural way to combine multiple strategies (see semantic operators (1)–(4)); *(iii)* more general post-convolution transforms and layer stacking ($L \geq 1$) result in a more flexible joint model for class prototypes; *(iv)* all parameters can be trained end-to-end through a GFSL objective, thus does not require the 2-stage training procedure from [14].

[2]This is in contrast to our method, where prototypes are learnable parameters, *initialized* with an average over the support points.

5 Experiments

We evaluate our method on two widely used benchmark datasets. First, we use the FSL benchmark dataset **miniImageNet** [45], which consists of 100 classes and 600 images per class. We adopt the split specified in [35] to obtain 64 seen, 16 novel validation and 20 novel test classes. To obtain training, validation and test sets for the *seen* class label space, we further follow the approach in [14]. We enrich this dataset with side information based on conceptual semantics and lexical relations by mapping class names into the ontology *WordNet* [28]. In particular, we use *WordNet* path similarities [31] between class labels, which are scores based on the shortest path distances between words in the taxonomy. Second, we evaluate our method on **Caltech-UCSD Birds-200-2011** (*CUB*) [46], which is widely used for ZSL. This dataset contains 11,788 images across 200 classes of different bird species. Each class has 312 annotated continuous attributes describing visual characteristics of the respective bird species. We follow the standard split used in ZSL [29] to obtain 150 seen and 50 novel test classes. Further, we randomly select 20 from the 150 seen classes for validation. For each seen class, 25% of the images are hold out as seen class test set and 10% as seen class validation set. In this dataset, we obtain edge weights by computing pairwise cosine similarities between class attributes. These semantic operators $\tilde{B}$, where class similarities are used as edge weights, are depicted in the supplementary.

Evaluation protocol We follow the episodic testing evaluation protocol from previous meta-learning work in (G)FSL [11, 13, 14, 35, 39, 40, 45] and evaluate all models across 600 test episodes, where each test episode is an N^+-way classification task with *all* seen classes plus N novel classes randomly sampled from a larger test set containing more than N novel test classes.[3] The average performance with the 95% confidence interval is reported in Table 1 and 2. In addition to the evaluation measures Seen-Seen, Novel-Novel and Joint-Joint in [14], we adopt the convention in GZSL [48] and report Seen-Joint and Novel-Joint accuracies and their harmonic mean, which capture the *joint* label space classification performance separately for seen and novel classes, and the harmonic mean balances the unequal sizes of seen and novel classes. See the supplementary for details on the performance measures and pseudo-code for meta-testing.

	FSL		GFSL			
1-shot	Seen-Seen	Novel-Novel	Joint-Joint	Seen-Joint	Novel-Joint	H-Mean
PN$^+$ [39]	54.02±0.46	53.88±0.78	27.02±0.23	54.02±0.46	0.02±0.01	0.04±0.03
DFSLwoF [14]	69.93±0.41	55.80±0.78	49.42±0.41	58.54±0.43	40.30±0.74	46.95±0.55
GcGPN	63.68±0.42	55.67±0.73	46.82±0.41	51.08±0.46	42.57±0.72	45.68±0.48
GcGPN-aux	68.39±0.40	56.59±0.75	49.66±0.39	58.16±0.44	41.16±0.69	47.51±0.51
GcGPN-split	68.26±0.42	55.68±0.76	49.60±0.41	55.22±0.47	43.98±0.76	48.13±0.49
GcGPN-aux-split	68.13±0.43	60.40±0.71	**51.63±0.41**	54.68±0.46	48.59±0.72	**50.83±0.45**
GcGPN-cos-aux	69.86±0.41	54.00±0.77	47.94±0.40	62.39±0.45	33.50±0.67	42.88±0.59
5-shot	Seen-Seen	Novel-Novel	Joint-Joint	Seen-Joint	Novel-Joint	H-Mean
PN$^+$ [39]	60.42±0.45	70.84±0.66	31.70±0.25	60.41±0.45	2.99±0.20	5.54±0.34
DFSLwoF [14]	70.24±0.43	72.59±0.62	**59.08±0.40**	59.89±0.47	58.26±0.68	**58.58±0.41**
GcGPN	66.51±0.43	71.53±0.63	57.16±0.40	56.73±0.45	57.59±0.67	56.69±0.41
GcGPN-aux	68.89±0.43	71.81±0.64	58.03±0.39	60.56±0.45	55.50±0.67	57.41±0.42
GcGPN-split	68.69±0.43	71.83±0.62	57.87±0.38	57.78±0.46	57.96±0.67	57.36±0.39
GcGPN-aux-split	68.30±0.45	73.31±0.62	58.63±0.40	57.93±0.48	59.32±0.68	58.12±0.41
GcGPN-cos-aux	68.03±0.43	71.22±0.65	57.41±0.41	60.26±0.48	54.56±0.72	56.66±0.45

Table 1: Test set accuracies (in %) for 5^+-way 1-shot and 5^+-way 5-shot classification on *miniImageNet*.

Baselines Recall the three major requirements for GFSL models: *(1)* handle dynamic novel label space on-the-fly, *(2)* store and represent all seen classes at test time, and *(3)* consistently embed novel classes into the existing label space. Most FSL models satisfy (1) but cannot be easily extended to (2). Either the entire training set would have to be stored at test time (*e.g.* [13, 45]), or the model is tailored to N-way classification only (*e.g.* [4, 11]). In contrast,

[3] Note that in the meta-learning setup, N is usually smaller than the number of *all* available novel test classes since the label spaces should vary during episodic training.

PN [39] offers a straight-forward extension PN$^+$ to handle requirement (2) as discussed in 4.2. Since our paper aims at improving GFSL performance, the relevant baselines are PN$^+$ and DFSLwoF [14]. For the sake of completeness, we compare the Novel-Novel accuracy of a GFSL to the performance of FSL models in the supplementary.

Model setup for GcGPN To evaluate the ability of leveraging side information for relational GFSL, we explore multiple variants of GcGPN with different specifications for the graph-convolution block. At the core of almost all model variants is the *semantic* operator B containing all graph edge weights (similarities among all $N_{\text{seen}} + N$ classes). For reproducibility details on network architecture and hyperparameters see the supplementary. We exploit the model's flexibility to combine multiple operators and include variants where the operator set $\mathcal{A}$ is augmented by the two auxiliary operators $\hat{B}_1$ and $\hat{B}_2$ defined in eq. (5) (variant indicated by *-aux*). This allows the model to trade-off between self-connection and the effect of similar prototypes. Further, note that the operators have an inherent four-block structure with relations between seen-seen, seen-novel, novel-seen and novel-novel classes (similar to eq. (5)). We explore the effect of either utilizing only one semantic operator $\mathcal{A} = \{B\}$ with all class similarities or splitting B into four individual operators $\mathcal{A} = \{B_{\text{ss}}, B_{\text{sn}}, B_{\text{ns}}, B_{\text{nn}}\}$ with one activated block each. The latter variant, indicated by *-split*, allows the model to learn specialized post-convolution transforms for each block.

To further study the effect of the semantic operator and the post-convolution transform, we conduct two more ablation experiments on *CUB*: Variant GcGPN-aux-sn has reduced capacity in the operator by deactivating all inter-class relations other than the seen-novel block, whereas GcGPN-aux-fcθ_B has increased capacity in the post-convolution transform by using fully connected instead of diagonal θ_B.

We also explore GcGPN-cos-aux, a very simple way to make use of inter-class relationship modeling without using any side information. We obtain the operator entries by computing cosine similarity between the respective class prototypes (see 4.1, graph-conv. operators (1)). This also serves as an ablation to understand the effect of the graph-convolution based framework without the additional benefit of including side information. We provide an ablation study on different variants of this in the supplementary, including using L2-distance instead of cosine similarity and dropping the auxiliary operators.

For all variants, we use one graph-convolution layer and diagonal post-convolution transform θ_B with learnable entries.

Results and Discussion Tables 1 and 2 show results for generalized 5^+-way K-shot classification on *miniImageNet* (*mIN*) and *CUB*. Since PN$^+$ is only trained for FSL, its performance on novel class queries drops drastically when changing from the novel to the joint label space (*cf.* Novel-Novel and Novel-Joint). The novel classes are well-separated from each other but not consistently embedded into the seen label space.

GcGPN-cos-aux is the simplest variant with an explicit inter-class relationship model given by the cosine similarity between class prototypes. DFSLwoF [14] also relies on cosine similarity, but between *keys*. More precisely, every class has a d-dimensional key k_n, which are trainable model parameters *in addition* to the prototypes. Thus, DFSLwoF has higher modeling capacity and flexibility for the inter-class relations than GcGPN-cos-aux. While maintaining an edge on *mIN*, it is clearly outperformed by GcGPN-cos-aux on *CUB* in terms of both Joint-Joint accuracy and the harmonic mean. This shows that our graph-convolution based framework with an in general fully-connected graph can potentially obtain better performance with a much simpler inter-class relationship model.

	FSL		GFSL			
1-shot	Seen-Seen	Novel-Novel	Joint-Joint	Seen-Joint	Novel-Joint	H-Mean
PN$^+$ [39]	35.16±0.42	58.87±0.91	17.61±0.21	35.16±0.42	0.05±0.02	0.09±0.04
DFSLwoF [14]	47.02±0.56	59.87±0.93	37.87±0.48	41.55±0.56	34.19±0.82	36.34±0.56
GcGPN	43.96±0.55	70.49±0.81	45.46±0.48	34.92±0.54	56.00±0.84	42.21±0.47
GcGPN-aux	46.26±0.57	71.17±0.79	**47.61±0.47**	36.35±0.56	58.88±0.78	44.21±0.48
GcGPN-split	40.60±0.53	71.77±0.81	46.09±0.48	30.49±0.52	61.68±0.80	40.15±0.50
GcGPN-aux-split	50.99±0.53	71.51±0.75	47.33±0.46	45.64±0.53	49.01±0.77	46.53±0.47
GcGPN-cos-aux	51.79±0.55	59.80±0.95	44.06±0.52	41.25±0.57	46.87±0.88	42.90±0.52
Ablations						
GcGPN-aux-fcθ_B	51.88±0.55	72.72±0.80	47.49±0.46	47.33±0.55	47.66±0.74	**46.77±0.48**
GcGPN-aux-sn	38.71±0.56	70.25±0.84	44.67±0.48	29.26±0.54	60.09±0.81	38.61±0.52
5-shot	Seen-Seen	Novel-Novel	Joint-Joint	Seen-Joint	Novel-Joint	H-Mean
PN$^+$ [39]	43.04±0.44	75.81±0.67	25.26±0.26	42.90±0.44	7.62±0.32	12.45±0.44
DFSLwoF [14]	48.37±0.55	74.73±0.79	44.97±0.51	45.09±0.55	44.85±0.82	44.19±0.54
GcGPN	44.33±0.53	76.98±0.75	50.35±0.46	36.44±0.53	64.26±0.75	45.92±0.48
GcGPN-aux	50.73±0.56	75.87±0.74	**50.62±0.49**	45.92±0.54	55.33±0.79	**49.53±0.48**
GcGPN-split	52.31±0.53	76.49±0.74	49.16±0.48	48.37±0.54	49.95±0.78	48.42±0.49
GcGPN-aux-split	51.39±0.56	76.63±0.75	48.87±0.50	47.79±0.57	49.95±0.80	48.11±0.52
GcGPN-cos-aux	50.56±0.56	74.70±0.77	46.90±0.48	46.82±0.57	46.99±0.80	46.06±0.50
Ablations						
GcGPN-aux-fcθ_B	42.27±0.54	77.38±0.76	50.11±0.48	34.21±0.52	66.02±0.80	44.45±0.49
GcGPN-aux-sn	45.42±0.55	76.27±0.74	49.37±0.49	38.45±0.55	60.29±0.83	46.22±0.48

Table 2: Test set accuracies (in %) for 5^+-way 1-shot and 5^+-way 5-shot classification on *CUB*.

On *mIN*, GcGPN benefits from auxiliary operators and splitting on both tasks. Our best variant achieves state-of-the-art Joint-Joint accuracy and harmonic mean on the 1-shot task while being competitive with DFSLwoF [14] on the 5-shot task. On *CUB*, our model outperforms state-of-the-art by a wide margin on both 1-shot and 5-shot tasks and in terms of both Joint-Joint accuracy and harmonic mean performance. These improvements mainly stem from the model's enhanced ability to incorporate novel classes consistently into the seen class label space, which is suggested by the significant increase in Novel-Joint accuracy while the Seen-Joint accuracy remains comparable with [14]. Comparing to GcGPN-cos-aux, we see that side information has a clear beneficial effect on the accuracy of around 3%. Unlike on *mIN*, splitting was not beneficial. We do observe improvements from using auxiliary operators, however, the simplest GcGPN already outperforms the baselines significantly. Note that our model variants do not require learning an additional key space and an attention module as in DFSLwoF, but instead relies on side information only. Thus, the quality of the side information becomes crucial. The attributes on *CUB* provides fine-grained visual information which, according to our empirical results, proves to be a richer source of relational information compared to the taxonomy-based similarity for *mIN*.

We conducted a further ablation study for GcGPN on *CUB*, which suggests that increasing the post-convolution transformation capacity (GcGPN-aux-fcθ_B) improves the model's discriminative power in the 1-shot task. Restricting the relational graph to novel-seen dependencies turns out to harm the performance, which is in line with our key intuition that learning prototypes jointly by incorporating *all* inter-class relationships helps to handle the challenging trade-off in GFSL.

6 Conclusion and future work

We propose GcGPN which takes inter-class relationships defined by weighted graphs into account to consistently embed previously seen and novel classes into a joint prototype space. This allows for better generalization to novel tasks while maintaining discriminative power over not only novel but also all seen classes. Our model generalizes existing approaches in FSL and GFSL and achieves strong state-of-the-art results by leveraging side information.

Future work along this line would include an extensive analysis and comparison of different kinds of operators. Further, identifying useful external sources of side information would greatly leverage the benefits of using semantic operators for few-shot learning tasks.

References

[1] Zeynep Akata, Florent Perronnin, Zaid Harchaoui, and Cordelia Schmid. Label-embedding for image classification. *TPAMI*, 38(7), 2016.

[2] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your MAML. In *ICLR*, 2019.

[3] Evgeniy Bart and Shimon Ullman. Cross-generalization: Learning novel classes from a single example by feature replacement. In *CVPR*, 2005.

[4] Matthias Bauer, Mateo Rojas-Carulla, Jakub B. Świątkowski, Bernhard Schölkopf, and Richard E. Turner. Discriminative k-shot learning using probabilistic models. In *Second Workshop on Bayesian Deep Learning at NIPS*, 2017.

[5] Rianne van den Berg, Thomas N. Kipf, and Max Welling. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263*, 2017.

[6] Wei-Lun Chao, Soravit Changpinyo, Boqing Gong, and Fei Sha. An empirical study and analysis of generalized zero-shot learning for object recognition in the wild. In *ECCV*, 2016.

[7] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.

[8] Kun Duan, Devi Parikh, David Crandall, and Kristen Grauman. Discovering localized attributes for fine-grained recognition. In *CVPR*, 2012.

[9] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.

[10] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *TPAMI*, 28(4), 2006.

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

[12] Andrea Frome, Greg S Corrado, Jon Shlens, Samy Bengio, Jeff Dean, Tomas Mikolov, et al. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013.

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

[14] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018.

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

[16] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. In *ICLR*, 2017.

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

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

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

[20] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266), 2015.

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

[22] Aoxue Li, Tiange Luo, Tao Xiang, Weiran Huang, and Liwei Wang. Few-shot learning with global class representations. In *ICCV*, 2019.

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

[24] Xuechen Li, Will Grathwohl, Eleni Triantafillou, David Duvenaud, and Richard Zemel. Few-shot learning for free by modelling global class structure. *2nd Workshop on Meta-Learning at NeurIPS*, 2018.

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

[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[27] Erik G. Miller, Nicholas E Matsakis, and Paul A. Viola. Learning from one example through shared densities on transforms. In *CVPR*, 2000.

[28] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995.

[29] Pedro Morgado and Nuno Vasconcelos. Semantically consistent regularization for zero-shot recognition. In *CVPR*, 2017.

[30] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *ICLR*, 2014.

[31] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet: Similarity: measuring the relatedness of concepts. In *HLT-NAACL*, 2004.

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

[33] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, 2014.

[34] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *IEEE Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.

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

[36] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, 2016.

[37] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *NIPS*, 2017.

[38] Edgar Schönfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero- and few-shot learning via aligned variational autoencoders. *CVPR*, 2019.

[39] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.

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

[41] Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *NIPS*, 1996.

[42] Yao-Hung H. Tsai and Ruslan Salakhutdinov. Improving one-shot learning through fusing side information. *Learning with Limited Data Workshop at NIPS*, 2017.

[43] Yao-Hung H. Tsai, Liang-Kang Huang, and Ruslan Salakhutdinov. Learning robust visual-semantic embeddings. In *ICCV*, 2017.

[44] Vinay K. Verma and Piyush Rai. A simple exponential family framework for zero-shot learning. In *ECML-PKDD*, 2017.

[45] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.

[46] Cathrine 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.

[47] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *CVPR*, 2018.

[48] Yongqin Xian, Christoph H. Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *TPAMI*, 2018.

[49] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. f-vaegan-d2: A feature generating framework for any-shot learning. In *CVPR*, 2019.

[50] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. *arXiv preprint arXiv:1711.11157*, 2017.

[51] Chenrui Zhang, Xiaoqing Lyu, and Zhi Tang. TGG: Transferable graph generation for zero-shot and few-shot learning. In *ACM International Conference on Multimedia*, 2019.