

# Supplementary Material: Relational Generalized Few-Shot Learning

Xiahao Shi<sup>1</sup>

xiahao.shi@de.bosch.com

Leonard Salewski<sup>1</sup>

leonard.salewski@gmail.com

Martin Schiegg<sup>1</sup>

martin.schiegg@de.bosch.com

Max Welling<sup>2</sup>

m.welling@uva.nl

<sup>1</sup> Bosch Center for Artificial Intelligence

Robert-Bosch-Campus 1,

Renningen, Germany

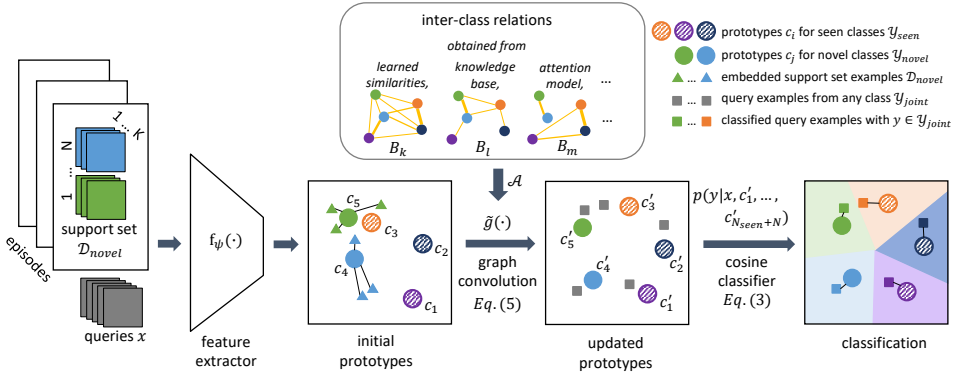
<https://www.bosch-ai.com/>

<sup>2</sup> University of Amsterdam

Science Park 904,

Amsterdam, the Netherlands

<http://amlab.science.uva.nl/>



**Figure 2: Our framework:** In an  $N^+$ -way  $K$ -shot episode, the task is to discriminate the joint label space  $\mathcal{Y}_{\text{joint}} = \mathcal{Y}_{\text{seen}} \cup \mathcal{Y}_{\text{novel}}$ . While  $\mathcal{Y}_{\text{seen}}$  contains (a large number of) seen classes,  $\mathcal{Y}_{\text{novel}}$  consists of  $N$  previously *unseen* classes with only  $K$  labeled *support set* examples per class. The goal of GFSL is to perform well across a series of such  $N^+$ -way  $K$ -shot episodes with *varying*  $\mathcal{Y}_{\text{novel}}$ . GcGPN addresses this challenge by explicitly modeling relationships between *all* classes in  $\mathcal{Y}_{\text{joint}}$  as graphs, where edges (yellow) represent inter-class relations: First, a feature extractor  $f_\psi$  maps all support set and query examples to a feature space, where classes are represented by prototypes. While novel class prototypes (solid circles) are initialized by averaging the corresponding support set feature representations (triangles), initial seen class prototypes (shaded circles) are modeled as learnable parameters. Then, GcGPN employs graph-convolutions to propagate information among classes according to the given inter-class relations, resulting in jointly updated prototypes. At last, the queries (grey rectangles) are classified according to their cosine similarity to all prototypes. The model is trained end-to-end in a meta-learning setup.

## A Implementation details

### A.1 Reproducibility details

For the sake of reproducibility, we provide comprehensive implementation details of our method in this section. Figure 2 depicts the detailed pipeline of our framework and Algorithm 1 provides the step-by-step recipe how our model GcGPN is used to perform GFSL.

---

**Algorithm 1**  $N^+$ -way  $K$ -shot classification with GcGPN

---

- 1: Input:  $N_{\text{seen}}, N, \mathcal{A} := \{B_1, B_2, B_3, \dots\}$   $\triangleright$  Number of classes, number of shots, semantic operators
  - 2: Initialize parameters:  $\psi, c_1, \dots, c_{N_{\text{seen}}}, \theta_B, s_B, \tau, \forall B \in \mathcal{A}$
  - 3: **for**  $episode = 1, 2, \dots$  **do**
  - 4:    $\mathcal{Y}_{\text{novel}}, \mathcal{Y}_{\text{seen}}, \mathcal{D}_{\text{novel}}, \mathcal{Q}_{\text{joint}} \leftarrow$  Algorithm 2  $\triangleright$  Sample a  $N^+$ -way  $K$ -shot episode
  - 5:    $z_{n,i} \leftarrow f_{\psi}(\mathcal{D}_{\text{novel}}), n = N_{\text{seen}} + 1, \dots, N_{\text{seen}} + N, i = 1, \dots, K$   $\triangleright$  Apply feature extractor  $f_{\psi}$  to all support sets
  - 6:    $Z \leftarrow f_{\psi}(\mathcal{Q}_{\text{joint}})$   $\triangleright$  Apply feature extractor  $f_{\psi}$  to all query examples
  - 7:    $c_n \leftarrow \frac{1}{K} \sum_{i=1}^K \bar{z}_{n,i}, n = N_{\text{seen}} + 1, \dots, N_{\text{seen}} + N$   $\triangleright$  Average normalized support sets to initial novel prototypes
  - 8:    $C \leftarrow (c_1, \dots, c_{N_{\text{seen}}}, c_{N_{\text{seen}}+1}, \dots, c_{N_{\text{seen}}+N})$   $\triangleright$  Concatenate seen and novel initial prototypes
  - 9:    $C' = (c'_1, \dots, c'_{N_{\text{seen}}}, c'_{N_{\text{seen}}+1}, \dots, c'_{N_{\text{seen}}+N}) \leftarrow \tilde{g}(C, \{\theta_B, \mathcal{A}, s_B\}_{B \in \mathcal{A}})$   $\triangleright$  Update all prototypes with graph-conv.
  - 10:    $p(y = n | x, c'_1, \dots, c'_{N_{\text{seen}}+N}) \leftarrow \frac{\exp(\tau \cos(z, c'_n))}{\sum_{m=1}^{N_{\text{seen}}+N} \exp(\tau \cos(z, c'_m))}, \forall z \in Z$   $\triangleright$  Predict class probabilities for all queries
  - 11:   **if** training **then**
  - 12:     Compute loss  $L$ , take gradient  $\delta L$  w.r.t. parameters, perform SGD update
  - 13:     Adjust learning rate, check early stopping
  - 14:   **end if**
  - 15: **end for**
- 

**Sampling of GFSL episodes:** The sampling at training time is given in Algorithm 2. At test time, instead of simulating novel and seen classes from  $\mathcal{Y}_{\text{train}}$ , the seen label space is given by all training classes, *e.g.*  $\mathcal{Y}_{\text{seen}} = \mathcal{Y}_{\text{train}}$ , while  $\mathcal{Y}_{\text{novel}}$  and  $\mathcal{D}_{\text{novel}}$  are sampled from a larger test set of novel classes.

**Application of the feature extractor:** The feature extractor  $f_{\psi}$  maps from input space into a  $d$ -dimensional feature space. For comparability, we adopt the same feature extractor architecture as in [10] and [9] with 4 convolutional blocks and 128 output feature maps, where each block consists of a  $3 \times 3$  convolution layer followed by batch normalization, ReLU and  $2 \times 2$  max-pooling.

**Initial prototypes:** Seen class initial prototypes  $c_n \in \mathbb{R}^d, n = 1, \dots, N_{\text{seen}}$  are parameters

of the model. Novel class 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  $\bar{z}_{n,k} = \frac{f_\psi(x_{n,k})}{\|f_\psi(x_{n,k})\|}$ ,  $n = N_{\text{seen}} + 1, \dots, N_{\text{seen}} + N$  with  $x_{n,k}$  denoting the  $k$ -th labeled support set examples of class  $n$ . The  $(N_{\text{seen}} + N) \times d$  matrix  $C = (c_1, \dots, c_{N_{\text{seen}}}, c_{N_{\text{seen}}+1}, \dots, c_{N_{\text{seen}}+N})$  contains all initial prototypes with the upper block corresponding to seen and the lower block to novel classes.

**Obtaining operators:** As discussed in sec. 4.1, a set  $\mathcal{A}$  of operators can be extracted from different kinds of inter-class relations. Here, we describe how the operators used in our experiments are obtained. As mentioned in sec. 5, the semantic operator  $B$  is at the core of all model variants we evaluated.

For *miniImageNet*, the  $i, j$ -th entry is obtained as *WordNet* path similarity between class  $i$  and class  $j$ . More precisely, we use the `path_similarity` method from the NLTK library [9] with default parameters. This measures class similarities based on the shortest path distances between the class labels in the *WordNet* taxonomy. Fig. 4 visualizes such a semantic operator on an example 5<sup>+</sup>-way episode.

For *CUB*, the  $i, j$ -th entry is obtained as pairwise cosine similarity between class-level attributes, which are 312-dimensional vectors describing visual characteristics of the respective bird species. These attributes are annotations that come together with the dataset.

We normalize the rows of the semantic operators by a softmax with learnable temperature (initialized to 1.0). Fig. 5 visualizes such a semantic operator on an example 5<sup>+</sup>-way episode. For the semantic-only model variant of GcGPN,  $\mathcal{A} = \{B\}$ . For the *-split* variant, we split  $B$  into four individual operator  $\mathcal{A} = \{B_{\text{ss}}, B_{\text{sn}}, B_{\text{ns}}, B_{\text{nn}}\}$  with one activated block each. This is to study the effect of learning specialized post-convolution transforms for each block. For the *-aux* variant, we additionally include auxiliary operators  $\hat{B}_1$  and  $\hat{B}_2$  defined in eq. (5), e.g.  $\mathcal{A} = \{B, \hat{B}_1, \hat{B}_2\}$  or  $\mathcal{A} = \{B_{\text{ss}}, B_{\text{sn}}, B_{\text{ns}}, B_{\text{nn}}, \hat{B}_1, \hat{B}_2\}$ . This is to study the effect of the ability to trade-off between self-connection and neighboring prototypes.

**Application of graph convolution:** We apply graph convolution  $\tilde{g}$  to the initial prototypes  $C$  to obtain the updated prototypes  $C' = \tilde{g}(C) = \tilde{g}^L(\dots \tilde{g}^1(C))$ , where  $\tilde{g}^l, l = 1, \dots, L$  is defined in sec. 4.1. In our experiments we use one graph-convolution layer and study two variants for the post-convolution transforms  $\theta_B$ : either as diagonal matrix or as full matrix (latter variant indicated by *-fc* $\theta_B$ ) with learnable entries. In both cases,  $\theta_B$  is initialized to be the identity matrix.

**Performing classification:** For each query  $x \in \mathcal{Q}_{\text{joint}}$ , we obtain conditional class probabilities using the updated prototypes  $C'$  according to eq. (3). At training time, we compute the cross-entropy loss on these softmax probabilities and take the gradient to update all trainable parameters of the model.

**Training:** The learnable parameters of GcGPN include the weights  $\psi$  of the feature extractor, the seen class initial prototypes  $c_1, \dots, c_{N_{\text{seen}}}$ , the weights of the post-convolutional transform  $\theta_B$  and the corresponding scaling factor  $s_B$  for each operator  $B \in \mathcal{A}$ , temperatures in any operator normalization (if applicable) and the cosine classifier. All parameters are learned end-to-end and the model trained from scratch, unlike the two-phase training used in [2] or approaches using pre-trained image features such as [9, 12, 16]. All models are trained for 75 epochs on *miniImageNet* and 45 epochs on *CUB* using an SGD optimizer with a momentum of 0.9, a weight decay parameter of 0.0005 and an initial learning rate of 0.1

---

**Algorithm 2** Sampling of an  $N^+$ -way  $K$ -shot episode from a set of training data  $\mathcal{D}_{\text{train}}$ , where  $\mathcal{D}^{(i)}$  only contains elements of class  $i$ .  $N$  denotes the number of novel classes per episode,  $K$  the number of instances in the support set,  $Q$  the number of query instances and finally  $B$  the number of instances per seen class.  $\text{RANDOMSAMPLE}(S, N)$  describes uniform random sampling of  $N$  elements without replacement from a set  $S$

---

```

1: Input:  $N_{\text{train}}, N, K, Q, B$ 
2:  $\mathcal{Y}_{\text{novel}} \leftarrow \text{RANDOMSAMPLE}(\{1, \dots, N_{\text{train}}\}, N)$   $\triangleright$  Sample “fake” classes for episode
3:  $\mathcal{Y}_{\text{seen}} \leftarrow \{1, \dots, N_{\text{train}}\} \setminus \mathcal{Y}_{\text{novel}}$   $\triangleright$  Store remaining seen classes
4: for  $i$  in  $\mathcal{Y}_{\text{novel}}$  do
5:    $D_{\text{novel}}^{(i)} \leftarrow \text{RANDOMSAMPLE}(\mathcal{D}_{\text{train}}^{(i)}, K)$   $\triangleright$  Sample support instances
6:    $Q_{\text{novel}}^{(i)} \leftarrow \text{RANDOMSAMPLE}(\mathcal{D}_{\text{train}}^{(i)} \setminus D_{\text{novel}}^{(i)}, Q)$   $\triangleright$  Sample novel query instances
7: end for
8: for  $j$  in  $\mathcal{Y}_{\text{seen}}$  do
9:    $Q_{\text{seen}}^{(j)} \leftarrow \text{RANDOMSAMPLE}(\mathcal{D}_{\text{train}}^{(j)}, B)$   $\triangleright$  Sample seen query instances
10: end for
11:  $Q_{\text{joint}} \leftarrow Q_{\text{novel}} \cup Q_{\text{seen}}$ 
12: Output:  $\mathcal{D}_{\text{novel}}, Q_{\text{joint}}$   $\triangleright$  Output support sets and query sets

```

---

that was reduced after 20, 40, 50 and 60 epochs. Performance is monitored on the validation set for early stopping.

## A.2 Pseudo code for GFSL episodic sampling

We provide pseudo code for episodic sampling at training time under the GFSL setup in Algorithm 2. For test time evaluation, instead of simulating novel and seen classes from  $\mathcal{Y}_{\text{train}}$ , the seen label space is given by all training classes, e.g.  $\mathcal{Y}_{\text{seen}} = \mathcal{Y}_{\text{train}}$ , while  $\mathcal{Y}_{\text{novel}}$  and  $\mathcal{D}_{\text{novel}}$  are sampled from a larger test set of novel classes.

# B Experimental details

## B.1 Ablation study for GcGPN without side information

In Section 4.1, we discussed that our framework accommodates different kinds of operators to model inter-class relationships. As we mentioned, a simple choice can be any distance or similarity measure on the prototype space, i.e., the operator entry  $B_{m,n}$  is given by  $\text{dist}(c_m, c_n)$ . Here, we provide experimental results for GcGPN using such simple distance operators, in particular L2 distance (*GcGPN-l2*) and cosine similarity (*GcGPN-cos*). Similar to 5, we also consider variants with and without the auxiliary operators  $\hat{B}_1$  and  $\hat{B}_2$  defined in eq. (5) (variant indicated by *-aux*).

Table 3 and 4 show the results on the *miniImageNet* and *CUB* datasets, respectively. Generally, L2 distance seems to have slight advantage over cosine similarity and the use of auxiliary operators increases performance overall. As discussed in 4.2, the competitor model DFSLwoF [2] can be seen as a special case of our framework with  $\mathcal{A} = \{\hat{B}_1, \hat{B}_2, \hat{B}_{\text{key}}\}$

1-shot	Seen-Seen	FSL		GFSL		
		Novel-Novel	Joint-Joint	Seen-Joint	Novel-Joint	H-Mean
GcGPN-cos	65.14±0.44%	55.08±0.75%	47.25±0.41%	54.65±0.46%	39.86±0.75%	45.24±0.52%
GcGPN-l2	53.22±0.46%	52.45±0.77%	40.14±0.41%	35.70±0.47%	44.59±0.75%	38.83±0.40%
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%
GcGPN-l2-aux	70.08±0.41%	54.46±0.75%	48.21±0.40%	62.78±0.43%	33.65±0.68%	43.09±0.59%
5-shot	Seen-Seen	FSL		GFSL		
		Novel-Novel	Joint-Joint	Seen-Joint	Novel-Joint	H-Mean
GcGPN-cos	55.44±0.46%	68.52±0.65%	50.62±0.41%	43.56±0.49%	57.68±0.72%	49.04±0.40%
GcGPN-l2	57.53±0.44%	69.25±0.66%	51.17±0.41%	47.98±0.46%	54.36±0.75%	50.29±0.40%
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%
GcGPN-l2-aux	67.79±0.43%	72.37±0.62%	57.81±0.41%	59.30±0.46%	56.32±0.68%	57.29±0.43%

Table 3: Test set accuracies for 5<sup>+</sup>-way 1-shot and 5<sup>+</sup>-way 5-shot classification on *mIN*.

1-shot	Seen-Seen	FSL		GFSL		
		Novel-Novel	Joint-Joint	Seen-Joint	Novel-Joint	H-Mean
GcGPN-cos	44.19±0.56%	60.86±0.93%	38.06±0.48%	39.15±0.56%	36.97±0.84%	36.85±0.51%
GcGPN-l2	44.10±0.55%	60.00±0.91%	38.84±0.49%	36.82±0.56%	40.86±0.88%	37.58±0.50%
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%
GcGPN-l2-aux	45.99±0.56%	60.30±0.93%	41.88±0.52%	35.47±0.56%	48.28±0.87%	40.00±0.50%
5-shot	Seen-Seen	FSL		GFSL		
		Novel-Novel	Joint-Joint	Seen-Joint	Novel-Joint	H-Mean
GcGPN-cos	43.10±0.53%	74.82±0.81%	45.79±0.49%	37.22±0.52%	54.36±0.86%	43.40±0.47%
GcGPN-l2	43.05±0.57%	74.47±0.80%	45.82±0.51%	37.23±0.56%	54.41±0.85%	43.42±0.49%
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%
GcGPN-l2-aux	51.47±0.55%	74.65±0.75%	48.20±0.49%	47.52±0.56%	48.88±0.79%	47.44±0.50%

Table 4: Test set accuracies for 5<sup>+</sup>-way 1-shot and 5<sup>+</sup>-way 5-shot classification on *CUB*.

where  $\hat{B}_{key}$  is an operator based on a learned key space (see 4.1, graph-conv. operators (3)). More precisely, the pairwise class similarities of the GcGPN variants here are computed on the class prototypes  $c_n$ ,  $n = 1, \dots, N_{seen} + N_{novel}$ , whereas those of DFSLwoF are computed between the class keys  $k_n$ ,  $n = 1, \dots, N_{seen} + N_{novel}$ , which are optimized model parameters *in addition* to the prototypes. Thus, DFSLwoF has higher modeling capacity and flexibility for the inter-class relations than our simple GcGPN variants. While it maintains an edge over GcGPN-\*-aux of around 2% on *miniImageNet*, the GcGPN-\*-aux variants outperform it on *CUB* in terms of both Joint-Joint and H-Mean accuracy with a margin of about 2 to 3% on the 5-shot task and about 4 to 6% on the 1-shot task. This shows that with our framework, we can potentially obtain the same performance with a much simpler inter-class relationship model.

## B.2 Comparison to FSL methods

In sec. 5 of the paper, we discussed the major requirements for GFSL models, which are (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. Although FSL models can address (1), they cannot be easily extended to cover requirements (2) and (3). Therefore, sec. 5 focuses on comparing our approach GcGPN with relevant *GFSL* methods  $PN^+$  (naive extension of PN [10] to GFSL) and DFSLwoF [2]. Nevertheless, we can compare the *FSL* performance of GFSL models, which is captured by the performance measure Novel-Novel, to recent FSL models. Note that all FSL models are trained with the few-shot objective in eq. (1), whereas the GFSL models (DFSLwoF and GcGPN) are trained with the

	5-way 1-shot	5-way 5-shot
Matching Network [13]	46.6%	60.0%
PN [14]	46.61±0.78%	65.77±0.70%
MAML [15]	48.07±1.75%	63.15±0.91%
Meta-LSTM [8]	43.44±0.77%	60.60±0.71%
Meta-SGD [9]	50.47±1.87%	64.03±0.94%
REPTILE [6]	49.97±0.32%	65.99±0.58%
VERSA [3]	53.40±1.82%	67.37±0.86%
CAVIA [16]	51.82±0.65%	65.85±0.50%
Relation Net [17]	50.44±0.82%	65.32±0.70%
Parameter prediction [18]	54.53±0.40%	67.87±0.20%
PN <sup>+</sup> (sec. 5)	53.88±0.78%	70.84±0.66%
DFSLwoF [19]	55.80±0.78%	72.59±0.62%
GcGPN-cos-aux (ours)	54.00±0.77%	71.22±0.65%
GcGPN (ours)	55.67±0.73%	71.53±0.63%
GcGPN-aux (ours)	56.59±0.75%	71.81±0.64%
GcGPN-split (ours)	55.68±0.76%	71.83±0.62%
GcGPN-aux-split (ours)	<b>60.40±0.71%</b>	<b>73.31±0.62%</b>

Table 5: FSL performance (Novel-Novel) on 5-way 1-shot and 5-way 5-shot classification on *miniImageNet*. The upper part of the table contains FSL methods and the lower part GFSL methods. Numbers for the FSL models are as reported in [14] and [17] and numbers for the GFSL models are obtained from our own experiments.

objective in eq. (2). Table 5 show the results for 5-way 1-shot and 5-way 5-shot classification on *miniImageNet*. The numbers suggest that (a) GFSL methods outperform FSL methods even on FSL tasks, and (b) additionally exploiting inter-class relations further improves performance.

### B.3 Definition of performance measures

For the experimental results in Table 1 and 2 from sec. 5, we report several different performance measures according to the conventions in GFSL and GZSL. We provide the definitions here: **Novel-Novel** measures the accuracy when classifying novel class queries in the novel class label space. **Seen-Seen** measures the accuracy when classifying seen class queries in the seen class label space. **Joint-Joint** measures the accuracy when classifying seen and novel queries in the joint label space. **Novel-Joint** measures the accuracy when classifying novel class queries in the joint label space. **Seen-Joint** measures the accuracy when classifying seen class queries in the joint label space. **Harmonic Mean (H-Mean)** is the harmonic mean of Novel-Joint and Seen-Joint, where  $H(x_1, x_2) = \frac{2 \cdot x_1 \cdot x_2}{x_1 + x_2}$  denotes the harmonic mean of two numbers  $x_1$  and  $x_2$ .

The performance measures Novel-Novel, Seen-Seen and Joint-Joint accuracies are reported in [14]. In addition to them, we also adopt the convention in GZSL (generalized zero-shot learning) [13] and report Novel-Joint and Seen-Joint accuracies together with their harmonic mean. As [13] points out, the Novel-Joint performance is of particular interest because GZSL models often fail here drastically in spite of good Novel-Novel performance. Further, the harmonic mean is often preferred over Joint-Joint accuracy which is easily dominated by the seen class performance. This is because queries are much more likely to stem

from the seen classes, thus the Joint-Joint accuracy correlates heavily with the Seen-Seen accuracy.

## B.4 Varying the number of shots

We study the model’s behavior under different few-shot settings with varying numbers  $K$  of available labeled examples per novel class. Fig. 3 shows the Joint-Joint accuracy for  $5^+$ -way  $K$ -shot classification on the *CUB* dataset. The GcGPN variants are explained in sec. 5 and A. We train separate models for each  $K$  and evaluate their performance. The results show that the GcGPN variants consistently outperform the baseline DFSLwoF [4].

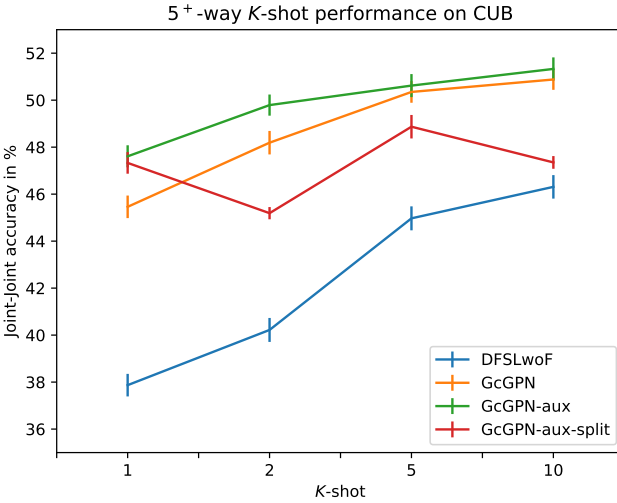


Figure 3:  $5^+$ -way  $K$ -shot classification accuracy (Joint-Joint) on the *CUB* dataset for different  $K$ .

## B.5 Semantic operators

In this section we show the semantic operators  $B$  for the *miniImageNet* [13] dataset (Figure 4) and for the Caltech-UCSD Birds-200-2011 (*CUB*) [14] dataset (Figure 5). For both visualizations operator temperature is at 1 leaving the operators unmodified.

Each row and column represents one class. In both visualizations brighter color indicates higher inter-class similarity and block structures arise when similar classes are listed next to each other. Note that the colormap clips the largest values on the diagonal to visualize the off-diagonal structure of the side information. Since graph-convolution operators usually require normalization, we apply row-wise softmax with a learnable temperature such that each row sums up to 1.

The blue lines divide the operator into four blocks corresponding to relations among seen classes in the upper left (Seen-Seen), novel classes in the lower right (Novel-Novel) and mixed relations in the other two blocks (Seen-Novel and Novel-Seen).

The figures indicate that relational structures are more prominent in *CUB* as compared to *miniImageNet*. This reflects that in *WordNet*, the 100 *miniImageNet* classes are a small subset from a much larger taxonomy and are almost equally related to each other. In contrast to that,

the dedicated fine-grained attributes in *CUB* provides more structural and discriminative information, which proves to be particularly beneficial.

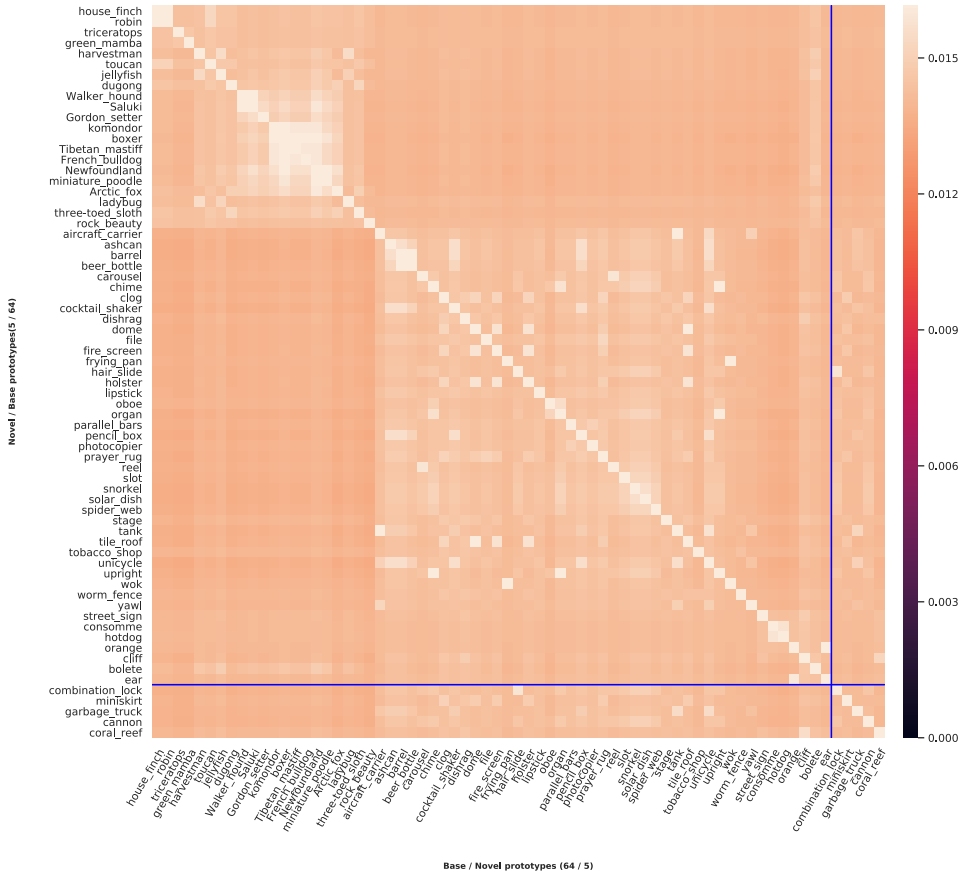


Figure 4: Softmax normalized semantic operators for the *miniImageNet* [14] dataset for a typical GFSL  $5^+$ -way  $K$ -shot task. Two large blocks are visible, which indicate the similarities of animate (classes *house\_finch* to *three-toed\_sloth*) and inanimate things (remaining classes). (Best viewed in color)



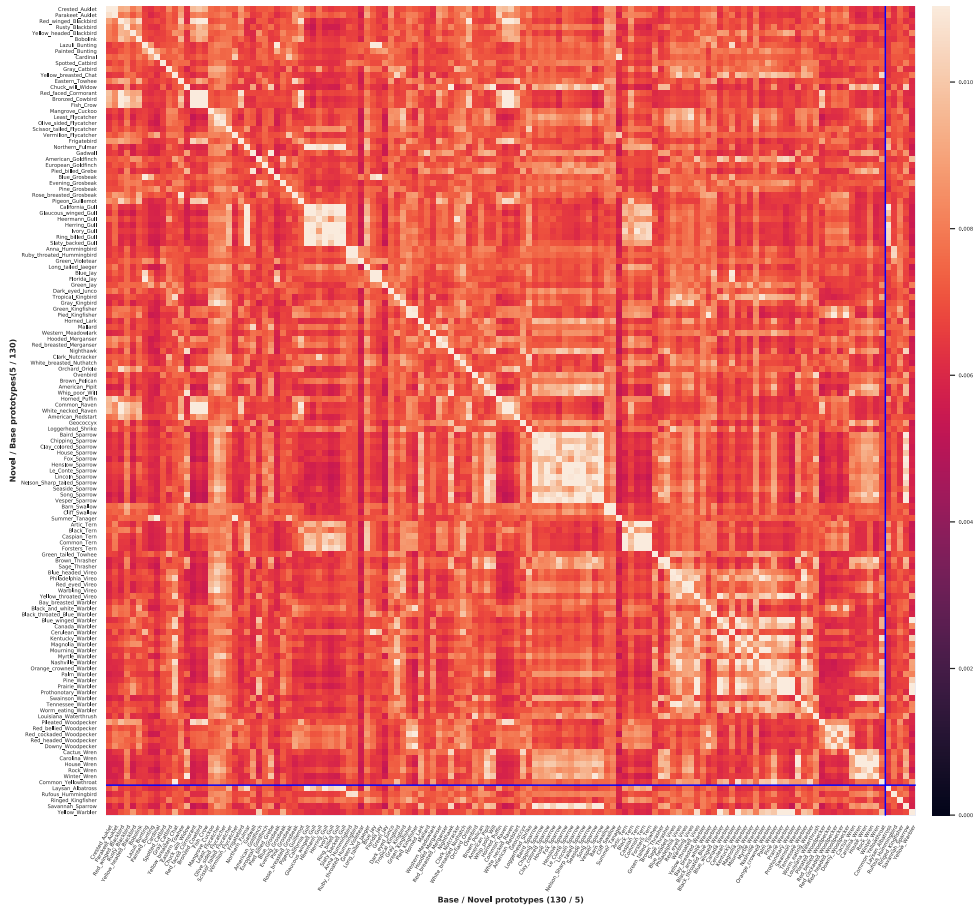


Figure 5: Softmax normalized semantic operators for the Caltech-UCSD Birds-200-2011 (CUB) [14] dataset for a typical GFSL 5<sup>+</sup>-way  $K$ -shot task. The largest continuous block are several different sparrow species (classes Baird\_Sparrow to Vesper\_Sparrow). (Best viewed in color)

## References

- [1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.
- [2] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *CVPR*, 2018.
- [3] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard E Turner. Meta-learning probabilistic inference for prediction. *ICLR*, 2019.
- [4] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.
- [5] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2, 2018.
- [6] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet: Similarity: measuring the relatedness of concepts. In *HLT-NAACL*, 2004.
- [7] 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.
- [8] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.
- [9] Edgar Schönfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero- and few-shot learning via aligned variational autoencoders. *CVPR*, 2019.
- [10] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, 2017.
- [11] 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.
- [12] Vinay K. Verma and Piyush Rai. A simple exponential family framework for zero-shot learning. In *ECML-PKDD*, 2017.
- [13] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NIPS*, 2016.
- [14] 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.
- [15] 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.
- [16] Yongqin Xian, Saurabh Sharma, Bernt Schiele, and Zeynep Akata. f-vaegan-d2: A feature generating framework for any-shot learning. In *CVPR*, 2019.
- [17] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. 2019.