Supplementary Materials of Exploring Hierarchical Graph Representation for Large-Scale Zero-Shot Image Classification

Kai Yi[1]0000-0003-0415-3584, Xiaoqian Shen[1], Yunhao Gou[2]0000-0002-1352-794X, and Mohamed Elhoseiny[1]0000-0001-9659-1551

[1] King Abdullah University of Science and Technology (KAUST)
[2] University of Electronic Science and Technology of China (UESTC)
{kai.yi, xiaoqian.shen, yunhao.gou, mohamed.elhoseiny}@kaust.edu.sa

The supplementary material provides:

- Section 1: Additional details on dataset splits;
- Section 2: Description on reconstructed hierarchical structure;
- Section 3: Additional training details;
- Section 4: Complementary experimental results on ImageNet-21K-P, ImageNet 2-hops and ImageNet 3-hops;
- Section 5: Experimental results on low-shot classification;
- Section 6: Complementary strategy ablation study results and analysis.
- Section 7: Additional ablations and comment.

1 Dataset Description and Reconstructed Hierarchical Structure Details

To make all of the classes fit into an appropriate location in the hierarchical Directed Acyclic Graph (DAG), we remove `fall11misc` (Miscellaneous synsets not in the major subtrees in the ImageNet 2011 Fall Release) from the original hierarchical structure but add `food` and its sub-branches. And then, according to the reconstructed hierarchical structure, those irrelevant classes are also removed from the ImageNet-1K and ImageNet-21K (winter-2021 release), resulting in our ImageNet-21K-D dataset. The processed result is presented in Tab. 1.

Class-wise Dataset	Train	Test	Train+Test	2-hops	3-hops
Original	1,000	20,841	21,841	1,549	7,860
Processed	983	17,295	18,278	1,533	6,898

Table 1. Comparison between original ImageNet-21K (Original) and our ImageNet-21K-D (Processed).

First four lines in Tab. 2 show our ImageNet-21K-D dataset splits. Please note that the official validation set, 50 images for each seen class, is used neither in ZSL training nor in ZSL validation. Our ZSL validation set is randomly sampled

Dataset	Description	Setting	Train	Val	Test
ImageNet-21K-D	# of Classes	seen	983	-	-
		unseen	-	17,295	17,295
	# of Images	seen	1,259,303	-	-
		unseen	-	792,510	11,337,589
ImageNet-21K-P	# of Classes	seen	975	-	-
		unseen	-	9,046	9,046
	# of Images	seen	1,252,157	-	-
		unseen	-	452,300	9,847,116

Table 2. ImageNet-21K-D and ImageNet-21K-P dataset split for ZSL.

from unseen classes with at most 50 images per class, and all the images from unseen classes are used for ZSL testing.

ImageNet-21K-P [9] is a pre-processed dataset from ImageNet21K by removing infrequent classes, reducing the number of total numbers by half but only removing only 13% of the original images. The original ImageNet-21K-P contains 12,358,688 images from 11,221 classes. After the above-mentioned pre-processing, class- and instance-wise splits are demonstrated in the rest four lines in Tab. 2.

2 Hierarchical Structure

Tab. 3 shows several examples of the reconstructed hierarchical tree. More general classes reside in the shallow layers, while the deeper layers contain more specific ones.

Layer	Example of Classes
1	plant, sport, artifact, animal, person
2	domestic animal, beach, painter
3	wildflower, ice field, vertebrate
...	...
12	Atlantic bottlenose dolphin, mouflon, Asian wild ox

Table 3. Example of classes in different layers. Noticed that we denote the root node as layer 0.

Fig. 1 shows the imbalanced distribution of classes per layer in our reconstructed hierarchical tree. Although there are 12 layers in the reconstructed hierarchical tree, most nodes locate in $2^{th} - 6^{th}$ layers.

3 Implementation and Training Details

We choose ResNet-50 [2] provided by CLIP [8] as image encoder, which uses the ResNet-D improvements from [3] and the antialiased rect-2 blur pooling

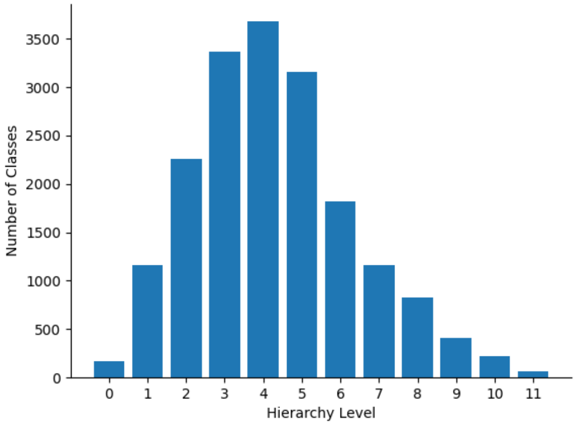

Fig. 1. Distribution of classes in different layers on the ImageNet-21K-D dataset.

from [13], and replaces the global average pooling layer with an attention mechanism. The dimension of the extracted feature representation is 1024. Moreover, our HGR-Net leverages two dimensions of hierarchical information. First, the ground truth class and all the ancestor nodes of this class tracing from the hierarchical tree are appended to form a list. We then set an outer ratio from this list to select part of the parent nodes as candidates for the outer loop. Similarly, within the outer loop, the inner ratio filters part of the parent nodes of the anchor node for the inner loop. In the inner loop, negative nodes of the outer-loop anchors are sampled through TopM sampling strategies. These negative classes contrast with the current layer's ground truth to guide the learning representation. Text descriptions of negative classes, as well as the ground truth one, are encoded into tokens and bracketed with start tokens and end tokens based on byte pair encoding (BPE) [10] with the max length of 77. For text embedding, we use CLIP [8] Transformer to extract semantic vectors with the same dimensions as feature representation. We obtain the logits with L2-normalized image and text features and calculate InfoNCE loss [7] layer by layer with an adaptive re-weighting strategy. More specifically, a learnable parameter with a size equivalent to the depth of the hierarchical tree is used to adjust the weights adaptively in both the outer and inner loops.

We use the AdamW optimizer [5] applied to all weights except the adaptive attention layer with a learning rate 3e-7 and a default weight decay. The reason for choosing such a small learning rate is that we finetune CLIP [8] with a hierarchical structure. Furthermore, a cosine scheduler is implemented to decay the learning rate for each step. In addition, we use the SGD optimizer separately for the adaptive layer with a learning rate of 1e-4. A learnable temperature parameter τ is initialized as 0.07 from [12] to scale the logits, and gradient clipping is utilized to avoid training instability. Besides, to accelerate training and avoid additional memory, mixed-precision [6] is used, and the weights of the model are only transformed into float32 for AdamW [5] optimization. The maximum number of sampled contrastive classes is set as 256. Training and testing are conducted on a Tesla V100 GPU with a batch size of 256 and 512, respectively.

4 Performance Comparison

Here we show complementary comparisons on more variances of the ImageNet dataset. We first show the results on the ImageNet-21K-P dataset [9]. The results show that our method achieved significantly better performance compared with baselines.

Models	**Hit@ k**(%)					**TOR**	**POR**
	1	2	5	10	20		
SGCN(w2v) [4]	3.70	6.39	12.00	17.84	25.16	6.26	11.90
DGP(w2v w/o) [4]	3.88	6.62	11.85	17.54	25.14	5.11	12.04
DGP(w2v) [4]	4.01	6.72	12.10	17.93	25.77	8.14	13.30
SGCN(Tr) [4]	6.41	10.75	18.76	27.06	36.71	10.61	16.44
DGP(Tr w/o) [4]	7.09	11.67	20.32	28.53	38.43	9.57	18.58
DGP(Tr) [4]	7.20	11.95	20.98	29.81	39.62	14.59	19.33
CNZSL(w2v w/o CN) [11]	1.25	2.22	4.54	7.56	12.21	3.04	5.17
CNZSL(w2v w/o INIT) [11]	2.58	4.27	7.94	12.27	18.36	4.96	6.78
CNZSL(w2v) [11]	2.58	4.27	8.01	12.44	18.58	3.88	6.58
CNZSL(Tr w/o CN) [11]	3.27	5.59	10.69	16.17	23.33	5.32	7.68
CNZSL(Tr w/o INIT) [11]	7.90	12.77	21.40	29.50	38.63	11.23	12.56
CNZSL(Tr) [11]	7.97	12.81	21.75	29.92	38.97	11.50	12.62
FREE(w2v) [1]	3.95	6.32	11.85	16.57	24.93	5.76	8.31
FREE(Tr) [1]	8.15	12.90	21.37	30.29	40.62	11.82	13.34
CLIP [8]	19.33	28.07	41.66	53.77	61.23	20.08	20.27
HGR-Net(Ours)	**20.08**	**29.35**	**42.49**	**52.47**	**62.00**	**23.43**	**23.22**

Table 4. Result of ImageNet21K-P [9]. DGP(w/o) [4] means without separating adjacency matrix into ancestors and descendants, `CN` and `INIT` in CNZSL [11] means class normalization and proper initialization respectively, and Tr is Transformer of CLIP for short.

We also conduct the performance comparison on two smaller ImageNet-21K variants, i.e., 2-hops, 3-hops. Tab. 5 proves the effectiveness of our method on smaller datasets. The performance drops for all models on "3-hops" test set than "2-hops" since seen classes are similar to unseen classes on "2-hops", while distant from unseen classes on "3-hops". However, our method still outperforms others when unseen classes are dominant in number and share less resemblance with seen classes, which proves efficiency in knowledge transferring.

Test Set	Method	Hit@ k(%)					TOR	POR
		1	2	5	10	20		
2-hops	SGCN(w2v) [4]	24.47	37.84	57.22	69.68	79.41	32.76	36.38
	SGCN(Tr) [4]	28.19	42.57	61.69	72.89	81.48	37.74	38.10
	DGP(w2v) [4]	24.57	37.67	56.88	69.60	79.17	34.94	37.04
	DGP(Tr) [4]	29.47	43.87	62.79	74.65	83.14	39.98	41.25
	CNZSL(w2v) [11]	11.99	19.11	32.46	44.31	56.40	17.37	16.70
	CNZSL(Tr) [11]	27.17	40.20	57.45	67.86	76.08	32.27	24.29
	CLIP [8]	35.24	48.51	65.01	74.61	81.96	39.34	41.99
	HGR-Net(Ours)	**36.11**	**49.46**	**65.90**	**75.69**	**82.98**	**40.87**	**42.63**
3-hops	SGCN(w2v) [4]	4.87	8.73	17.21	25.46	35.55	8.20	24.05
	SGCN(Tr) [4]	8.31	13.44	23.59	33.51	44.57	13.50	28.42
	DGP(w2v) [4]	4.95	8.81	16.91	25.64	36.14	10.40	26.85
	DGP(Tr) [4]	9.81	15.85	26.78	36.95	48.11	18.2	30.13
	CNZSL(w2v) [11]	3.71	6.11	11.31	17.21	24.95	7.14	16.08
	CNZSL(Tr) [11]	10.31	16.37	27.08	36.60	46.47	14.57	21.97
	CLIP [8]	22.46	31.58	44.49	54.27	63.57	24.93	32.22
	HGR-Net(Ours)	**23.23**	**32.53**	**45.74**	**55.70**	**65.05**	**26.54**	**32.44**

Table 5. Performance comparison among SoTA on 2-hops and 3-hops. Tr means text encoder is CLIP Transformer.

5 Low-Shot Classification

We have presented the performance in the main paper, but we select all the methods with Transformer of ClIP [8] in an independent graph to make it more clear as Fig. 2 shows.

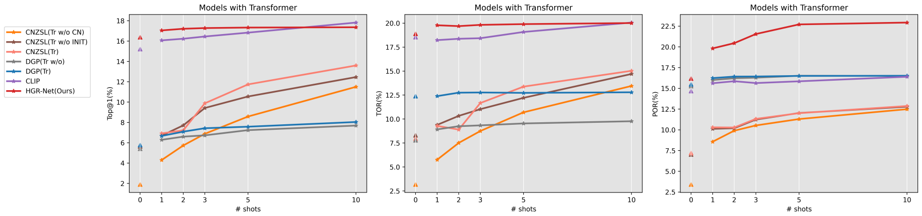

Fig. 2. Top@1, Top-Overlap Ratio (TOR) and Point-Overlap Ratio (POR) results among different Transformer of ClIP [8] based methods. DGP(w/o) [4] means without separating adjacency matrix into ancestors and descendants and Tr means the text encoder is Transformer of ClIP [8].

6 Complementary Ablation Study of Weighting Strategies

Fig. 3 4 5 6 7 show Top@1, Top-Overlap Ratio (TOR) and Point-Overlap Ratio (POR) evaluation among *Equal*, *Increasing*, *Decreasing*, ↑ *non-linear* and ↓ *non-linear* weighting strategies in different outer ratio (`K`) and inner ratio (`M`).

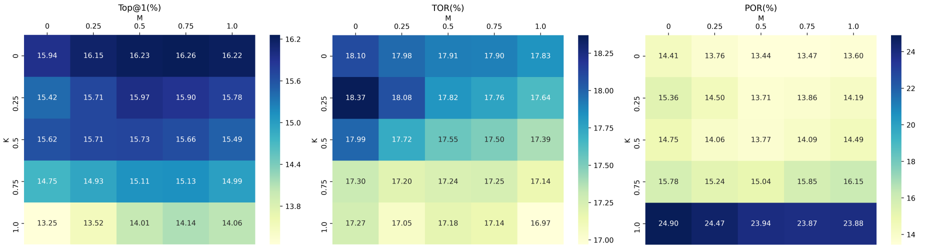

Fig. 3. Different outer ratio (K) and inner ratio (M) with weighting strategy *Equal*

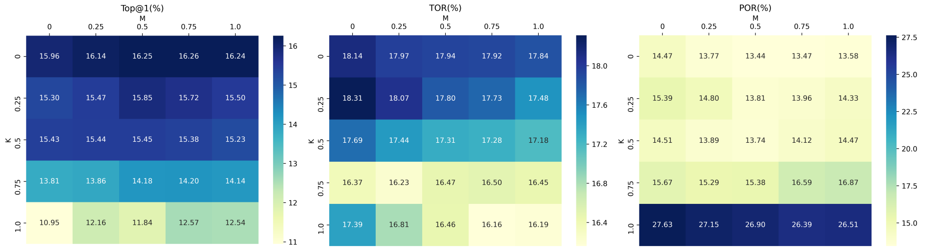

Fig. 4. Different outer ratio (K) and inner ratio (M) with weighting strategy *Increasing*

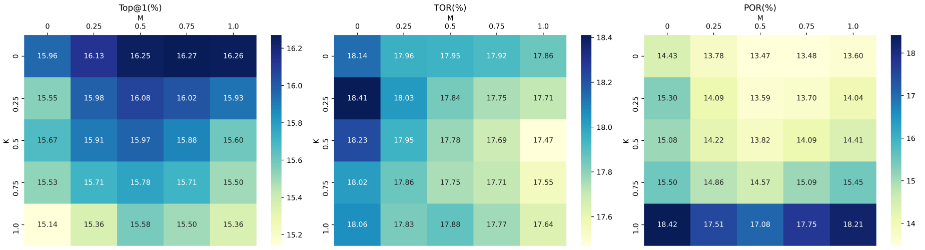

Fig. 5. Different outer ratio (K) and inner ratio (M) with weighting strategy *Decreasing*

Based on extensive experiments, the *adaptive* weighting strategy with a learnable parameter obtained the best performance. The observed result of learned weights firstly decreases and then increases as the depth goes deeper, which can be deemed as inversely proportional to the number of classes in each layer. Therefore, the *Adaptive* weighting can be simplified as: let N_j be the total number of classes in each layer j, and n be the number of layers, the weight for layer j is defined as $\frac{\frac{1}{N_j}}{\sum_{i=0}^{n} \frac{1}{N_i}}$, in order to simplify the optimization and to achieve a stable result. The imbalanced number of classes can explain this result in different layers as Fig. 1 and the model learns to activate layers with fewer classes more frequently to balance classes among different hierarchies.

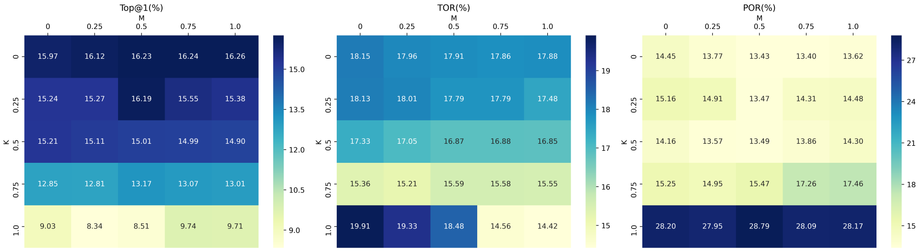

Fig. 6. Different outer ratio (K) and inner ratio (M) with weighting strategy ↑ *non-linear*

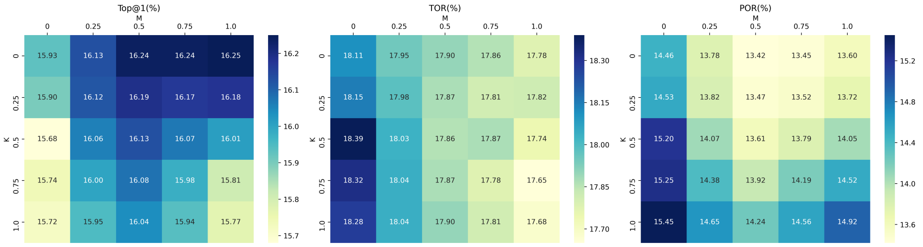

Fig. 7. Different outer ratio (K) and inner ratio (M) with weighting strategy ↓ *non-linear*

7 Additional Ablations and Comment

Loss ablations. We found it could be interesting to consider more ablations on our loss design, e.g., we consider two variants that both contain one single loop. The first one only considers the inner loss regarding the ground truth as the real label, while the second traces the ancestors of the ground truth as real labels and searches negative for each one. Experimental results demonstrate the performance drops by 5.61% and 1.40% respectively (see Table. 6 top part) and prove our outer-inner loops are necessary.

Experimental comparison fairness. Our model and the baseline DGP [4] both introduced the hierarchical semantic knowledge differently, but we demonstrated significantly better performance, which shows that a better-designed hierarchical graph can be critical to achieving good performance. The standard version of other baselines does not introduce a hierarchical graph in the training step. So we directly add HGR to the baseline CNZSL [11], dubbed CNZSL+HGR. As Table 6 shows, leveraging our HGR can increase the CNZSL [11] performance by 3.18% on average. We suggest it doesn't improve much because CNZSL [11] has a shallower architecture than CLIP [8] and builds on top visual features instead of raw images. Besides, our method HGR-Net still shows significantly better performance compared with CNZSL+HGR.

Method	**Hit@ k**(%)					**TOR**	**POR**
	1	2	5	10	20		
HGR-Net(Ours)	**16.39**	**24.19**	**35.66**	**44.68**	**53.71**	**18.90**	**16.19**
- outer loss	15.47$^{-0.92}$	22.80$^{-1.49}$	33.65$^{-2.01}$	42.23$^{-2.45}$	50.97$^{-2.74}$	17.00$^{-1.90}$	15.80$^{-0.39}$
- inner loss	16.16$^{-0.23}$	24.00$^{-0.19}$	35.47$^{-0.19}$	44.38$^{-0.30}$	53.46$^{-0.25}$	18.49$^{-0.41}$	16.09$^{-0.10}$
CNZSL(w2v)	1.94	3.17	5.88	9.12	13.73	3.93	4.03
+HGR	1.96$^{+0.02}$	3.20$^{+0.03}$	5.95$^{+0.07}$	9.28$^{+0.16}$	13.92$^{+0.19}$	4.38$^{+0.45}$	5.15$^{+1.12}$
CNZSL(Tr)	5.77	9.48	16.49	23.25	31.00	8.32	7.22
+HGR	5.91$^{+0.14}$	9.54$^{+0.06}$	16.79$^{+0.50}$	23.83$^{+0.58}$	31.71$^{+0.71}$	8.56$^{+0.24}$	10.28$^{+3.06}$

Table 6. First part explores our method with different loss, (e.g., - outer loss only calculates the inner loss). Second part compares CNZSL w/ or w/o additional graph knowledge.

References

1. Chen, S., Wang, W., Xia, B., Peng, Q., You, X., Zheng, F., Shao, L.: Free: Feature refinement for generalized zero-shot learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 122–131 (2021) 4
2. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 2
3. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 558–567 (2019) 2
4. Kampffmeyer, M., Chen, Y., Liang, X., Wang, H., Zhang, Y., Xing, E.P.: Rethinking knowledge graph propagation for zero-shot learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11487–11496 (2019) 4, 5, 7
5. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (2019) 3, 4
6. Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al.: Mixed precision training. arXiv preprint arXiv:1710.03740 (2017) 4
7. Van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv e-prints pp. arXiv–1807 (2018) 3
8. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020 (2021) 2, 3, 4, 5, 7
9. Ridnik, T., Ben-Baruch, E., Noy, A., Zelnik-Manor, L.: Imagenet-21k pretraining for the masses. arXiv preprint arXiv:2104.10972 (2021) 2, 4
10. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (2016) 3
11. Skorokhodov, I., Elhoseiny, M.: Class normalization for zero-shot learning. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=7pgFL2Dkyyy 4, 5, 7
12. Veeling, B.S., Linmans, J., Winkens, J., Cohen, T., Welling, M.: Rotation equivariant cnns for digital pathology. CoRR (2018) 3
13. Zhang, R.: Making convolutional networks shift-invariant again. In: International conference on machine learning. pp. 7324–7334. PMLR (2019) 3