UNIVERSITY OF CALIFORNIA SAN DIEGO

Toward Realistic Classifier for Long-Tail Distributions

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Electrical Engineering (Signal and Image Processing)

by

Tz-Ying Wu

Committee in charge:

Professor Nuno Vasconcelos, Chair
Professor Sonia Martinez
Professor Truong Nguyen
Professor Bhaskar Rao
Professor Xiaolong Wang

2024

The Dissertation of Tz-Ying Wu is approved, and it is acceptable in quality and form for publication on microfilm and electronically.

University of California San Diego

2024

DEDICATION

To my family.

EPIGRAPH

Stay hungry. Stay foolish.

-*Steve Jobs*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

My Ph.D. journey at UCSD has been an incredible chapter in my life. It would not have been possible without the unwavering support and guidance of many people. I am profoundly grateful to those who have walked beside me, enabling me to achieve this significant milestone.

First, I would like to express my deepest appreciation to my Ph.D. advisor, Professor Nuno Vasconcelos. His advisement has been pivotal in my growth and development. Through his guidance, I have not only broadened my knowledge but also refined my critical thinking skills, encouraging me to delve deeper into the essence of problems. He consistently challenges me to explore fundamental questions without becoming mired in established approaches. His insights into crafting compelling paper structures and presentations have enhanced my ability to convey my research's significance effectively. I am also immensely thankful to Professors Sonia Martinez, Truong Nguyen, Bhaskar Rao, and Xiaolong Wang for their invaluable mentorship and constructive feedback as committee members. This thesis owes its completion to their guidance and support.

In addition, I am grateful to all my mentors and collaborators over the years. Prof. Min Sun was my Master's advisor at National Tsing Hua University (NTHU). I sincerely thank him for his patience and dedication in establishing the foundation for my research endeavors. He ignited my passion for Computer Vision, inspiring me to pursue a Ph.D after completing my Master's degree. I also appreciate Prof. Mi-Chang Chang at NTHU and Prof. Juan Carlos Niebles at Stanford for supporting my Ph.D. applications, and Prof. Jing-Jia Liou at NTHU for introducing research to me for the first time as my undergraduate project advisor. I could not be here without them. Outside of academics, I am thankful to all my industrial collaborators during my graduate studies. I would like to thank Gurumurthy Swaminathan, Zhizhong Li, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto for their energetic mentoring and constructive guidance during my internship at Amazon AI. I also cherished the time interning at Intel AI Lab, where I worked with Subarna Tripathi, Kyle Min, Hector Valdez, Sainan Liu, and Somdeb Majumdar. Thank them for providing invaluable guidance and thoughtful critiques. A shout-out

to Subarna for being so dedicated and supportive during the long-term collaborations. I was also fortunate to learn from Boqing Gong from Google and Patrick Liu from Xpeng Motors, who provided insightful perspectives as industrial professionals.

I am also grateful to have wonderful lab mates and student collaborators at SVCL, Chih-Hui Ho, Pedro Morgado, Pei Wang, Yunsheng Li, Alakh Desai, Yi Li, Jiacheng Cheng, Jia Wan, Zhiyuan Hu, Jiteng Mu, Brandon Leung, Bo Liu, Zhaowei Cai, Deepak Sridhar, Shaopeng Guo, Xudong Wang, Yiran Xu, Xiaoyin Yang, Lihang Gong, Hsuan-Chu Lin, Zhihang Ren, Amir Persekian, Jiawen Zeng, Yuze Song, Parth Kumar, and Ashish Farande. It was a pleasure discussing research ideas and sharing school life at UCSD with them. I also thank Tarun Kalluri and Prof. Manmohan Chandraker for the inter-research-topic discussion. In addition, I want to thank my English conversation leaders from the UCSD EIA program, Chelsea Young, Eileen Mehrabian, and Yuvadee Srijongsirikul. They helped me overcome the language barrier and build confidence. I would also like to thank Professors Nuno Vasconcelos, Manuela Vasconcelos, Sonia Martinez, and Yuanyuan Shi for providing TA opportunities to me. I gained a lot of teaching and communication skills from these experience.

Last but not least, I would like to express my deepest gratitude to my family for their unconditional support throughout my studies. While I grew up in the countryside, my parents greatly valued education and always encouraged me to acquire knowledge and explore new things. I would like to thank them for their endless love and support, allowing me to pursue my dream overseas. I am also lucky to have my brother and sister accompanying me during my growth. They made the family full of happiness. A unique acknowledgment goes to my husband, whose unwavering support and companionship have been a constant source of joy in my life, helping me go through all the ups and downs of this journey. I am also thankful to my extended family for their warm support of my study.

Chapter 2 is, in full, based on the material as it appears in the publications of "Learning of Visual Relations: The Devil is in the Tails", Tz-Ying Wu*, Alakh Desai*, Subarna Tripathi, and Nuno Vasconcelos, in Proceedings of IEEE International Conference on Computer Vision

(ICCV), 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 3 is, in full, based on the material as it appears in the publications of "Solving Long-tailed Recognition with Deep Realistic Taxonomic Classifier", Tz-Ying Wu, Pedro Morgado, Pei Wang, Chih-Hui Ho, and Nuno Vasconcelos, in Proceedings of European Conference on Computer Vision (ECCV), 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 4 is, in full, based on the material as it appears in the publications of "ProTeCt: Prompt Tuning for Taxonomic Open Set Classification", Tz-Ying Wu*, Chih-Hui Ho*, and Nuno Vasconcelos, in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2024. The dissertation author was the primary investigator and author of this paper.

VITA

2016	B.S. in Electrical Engineering, National Tsing Hua University, Taiwan
2018	M.S. in Electrical Engineering, National Tsing Hua University, Taiwan
2024	Ph. D. in Electrical Engineering (Signal and Image Processing), University of California San Diego, United States

PUBLICATIONS

Tz-Ying Wu, Kyle Min, Subarna Tripathi, and Nuno Vasconcelos, "Ego-VPA: Egocentric Video Understanding with Parameter-efficient Adaptation". *Under submission*, 2024.

Tz-Ying Wu*, Chih-Hui Ho*, and Nuno Vasconcelos, "ProTeCt: Prompt Tuning for Taxonomic Open Set Classification". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Alakh Desai, **Tz-Ying Wu**, Subarna Tripathi, and Nuno Vasconcelos, "Single-Stage Visual Relationship Learning using Conditional Queries". In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

Tz-Ying Wu, Gurumurthy Swaminathan, Zhizhong Li, Avinash Ravichandran, Nuno Vasconcelos, Rahul Bhotika and Stefano Soatto," Class-Incremental Learning with Strong Pre-trained Models". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Tz-Ying Wu*, Alakh Desai*, Subarna Tripathi, and Nuno Vasconcelos, "Learning of Visual Relations: The Devil is in the Tails". In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

Tz-Ying Wu, Pedro Morgado, Pei Wang, Chih-Hui Ho, and Nuno Vasconcelos, "Solving Long-tailed Recognition with Deep Realistic Taxonomic Classifier". In *European Conference On Computer Vision (ECCV)*, 2020.

Yiran Xu, Xiaoyin Yang, Lihang Gong, Hsuan-Chu Lin, **Tz-Ying Wu**, Yunsheng Li, and Nuno Vasconcelos, "Explainable Object-induced Action Decision for Autonomous Vehicles". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Chih-Hui Ho, Bo Liu, **Tz-Ying Wu**, and Nuno Vasconcelos, "Exploit Clues from Views: Self-Supervised and Regularized Learning for Multiview Object Recognition". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

ABSTRACT OF THE DISSERTATION

Toward Realistic Classifier for Long-Tail Distributions

by

Tz-Ying Wu

Doctor of Philosophy in Electrical Engineering (Signal and Image Processing)

University of California San Diego, 2024

Professor Nuno Vasconcelos, Chair

Machine learning models, despite their widespread use in everyday applications, often suffer from unreliable performance due to the distribution shifts between training and inference. Distribution shifts are ubiquitous, occurring in both low-level features and high-level semantics, exacerbated by the non-uniformity of real-world data, particularly in long-tailed distributions where some classes appear much more frequently than others. This imbalance results in non-uniform model performance across classes, posing risks for applications requiring precise information. In contrast, humans are adept at adapting to such challenges. Inspired by this, we focus on addressing the distribution shifts in vision tasks caused by long-tail distributions to make machine learning classifiers more realistic like humans.

In this thesis, we aim to redefine long-tail recognition more broadly and concentrate on crafting a classifier that mirrors human adaptability to distribution shifts, a crucial aspect lacking in modern classifiers that is essential for constructing reliable AI systems. Expanding beyond the traditional framework, we extend long-tail recognition to encompass combinatorial label spaces. Furthermore, we explore a hierarchical label space within a single long-tail distribution, offering adaptable control for user-defined systems based on the model's competent level or the desired label space of the user. By delving into the core of the long-tail concept, we demonstrate that significant performance enhancements are attainable through appropriate data sampling techniques, even with straightforward architectures. We also identify hierarchical consistency as a key factor for building a model aligned with human cognition.

Chapter 1

Introduction

Machine learning models have been widely adopted in many daily applications, e.g., unlocking a door with facial recognition, classifying plants with a mobile app, and self-checkout at grocery stores. While showing promising results, these models are not always reliable since they typically rely on two impractical assumptions: i) consistent data distributions between training and inference and ii) balanced data distributions. In practice, these assumptions are frequently violated, and model performance can degrade significantly.

In fact, the distribution shift between training and inference is ubiquitous and can occur in both low-level features and high-level semantics. The former distribution shift occurs when the training data and testing data are sampled from different visual domains (i.e. web-collected images vs image sketches). The latter distribution shift results from the data collection pipeline, especially for those collected on the Internet, where the object classes and their class names that appeared during training do not align with that during inference. This is referred to as the *label space* distribution shift between training and testing.

These challenges are compounded by the non-uniformity of real-world data, which is naturally *long-tailed* distributed, where data from different classes may appear at significantly different frequencies. For example, while classes like "car" are very frequent on the road, classes like "Formosan black bear" rarely appear. This imbalance will result in non-uniform model performance across classes, leading to unreliable predictions, particularly for rare classes. A more critical issue is that these models may provide incorrect information unconsciously, which is dangerous for applications that require precise information, such as security systems, autonomous driving, and medical purposes.

Humans, however, are realistic and less affected by these issues since they are apt at adapting to environmental change and react smartly when facing challenging questions. Motivated by this, this thesis focuses on addressing the distribution shifts in vision tasks incurred by long-tail distributions, aiming to make machine learning models *realistic* like humans.

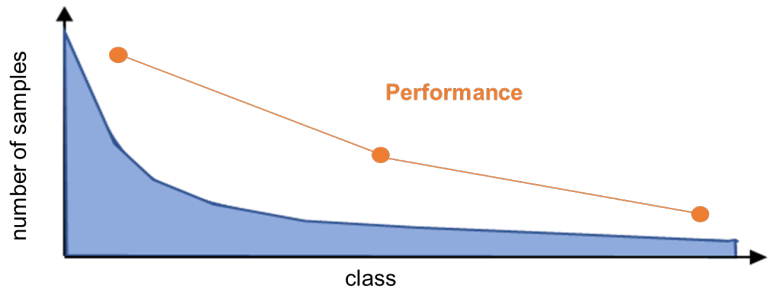


Figure 1.1. Real-world data are long-tailed distributed by nature. Model performance (orange) degrades with class frequency.

1.1 Long-tail Recognition

1.1.1 Definition

A long-tail distribution is an extreme case of an imbalanced distribution. The class frequency differs significantly, where the head classes have many samples while the number of samples in the tail classes is very limited. To quantize the skewness of the data distribution, prior work [20] defines the *imbalance factor* as the ratio of the number of samples in the most populated class over the rarest class. The lower the imbalance factor typically means the classification problem is harder, as the training distribution is more skewed. Training deep learning classifiers with such a data unbalance is non-trivial, and the model performance per class degrades notably from the head classes to the tail classes [80], as illustrated in Figure 1.1. The long-tail recognition research aims to solve this performance degradation problem.

1.1.2 Challenges of long-tail recognition

The modern classifier design typically involves a feature extractor $h(;\Phi) \in \mathrm{R}^k$ with parameter Φ and a softmax classifier $\sigma(;\mathbf{W})$ with a linear layer parameterized by a weight matrix $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_C] \in \mathrm{R}^{k \times C}$, i.e. $f(\mathbf{x}) = \sigma(\mathbf{W}^T h(\mathbf{x};\Phi))$, which maps an input sample to the label space $\mathscr{Y}$, where $C = |\mathscr{Y}|$ denotes the number of possible labels, and $h(;\Phi)$ can be implemented as a Convolutional Neural Network (CNN) [95, 44] or Transformers [105, 26]. The model

assigns the class posterior probability of class j as

$$f(\mathbf{x})_j = \frac{P_{X,Y}(\mathbf{x},j)}{\sum_{l=1}^{C} P_{X,Y}(\mathbf{x},l)} = \frac{P_{X|Y}(\mathbf{x}|j)P_Y(j)}{\sum_{l=1}^{C} P_{X|Y}(\mathbf{x}|l)P_Y(l)} = \frac{\exp\left(\mathbf{w}_j^T h(\mathbf{x};\Phi)\right)}{\sum_{l=1}^{C}\exp\left(\mathbf{w}_l^T h(\mathbf{x};\Phi)\right)}\,, \tag{1.1}$$

where the network parameters $\mathbf{W},\Phi$ are optimized with a classification loss on the training data $\mathscr{D}^{tr} = \{(\mathbf{x}_i^{tr}, y_i^{tr})\}_{i=1}^{N^{tr}}$, and the model is then evaluated on the testing data $\mathscr{D}^{te} = \{(\mathbf{x}_i^{te}, y_i^{te})\}_{i=1}^{N^{te}}$. In the following, we discuss the challenges of long-tail distributions for modern classifiers from various perspectives.

Distribution shift: The learning framework above is effective under the assumption that the training and testing distributions are well aligned, i.e. $P_{X^{tr},Y^{tr}}(\mathbf{x},y) = P_{X^{te},Y^{te}}(\mathbf{x},y)$. However, this is commonly violated as long as either $P_{X^{tr}|Y^{tr}}(\mathbf{x}|y) = P_{X^{te}|Y^{te}}(\mathbf{x}|y)$ or $P_{Y^{tr}}(y) = P_{Y^{te}}(y)$ does not comply. The former corresponds to the shift of the data distribution typically results from the difference in the data collection pipelines, imaging devices, or the change in lighting conditions or viewpoints, studied as the domain adaptation/generalization problem [33, 138]. The latter is associated with the label distribution shift, which is inevitable since training data are naturally long-tailed distributed while the label distribution at testing is usually uniform [53]. In addition, there can even be a compounded problem where both conditions are violated, and so is the assumption, which makes it even more difficult to learn a reliable machine learning classifier. This thesis concentrates on the problem of label distribution shift.

Unequal training: When the training distribution is long-tailed, a randomly sampled mini-batch $\mathscr{B} \sim \mathscr{D}^{tr}$ from the dataset will be dominated by the samples from the head classes. Since the typical training objective for the classifier of (1.1) is to minimize the cross-entropy loss of a mini-batch $\mathscr{B}$, i.e.

$$-\frac{1}{|\mathscr{B}|}\sum_{(\mathbf{x}_i,y_i)\in\mathscr{B}}\left[\mathbf{w}_{y_i}^T h(\mathbf{x}_i;\Phi) - \log\left(\sum_{l=1}^{C}\exp\left(\mathbf{w}_l^T h(\mathbf{x}_i;\Phi)\right)\right)\right], \tag{1.2}$$

which usually causes the classifier weight $\mathbf{w}_j$ of popular classes to have larger norms [61] and increases the tendency to predict the head classes [121]. On the other hand, even when the weights are normalized, the model still performs poorly in tail classes because the samples from the tail classes are sparsely sampled, introducing weak supervision to these classes.

Data scarcity: While data resampling or class-aware optimization objectives can alleviate unequal training, data scarcity is still a key essence of the long-tail problem that cannot be ignored. If a tail class only has a few training samples, it is unlikely that the model can learn generalized features for that class when training from scratch. While fine-tuning from a large-scale pretrained model can lead to some improvements, there is still a risk of overfitting on those tail samples.

1.1.3 Existing methods

Several strategies have been developed for addressing the performance degradation problem in long-tail recognition. An intuitive method for reducing the class bias is to create a mini-batch that is balanced across the classes. This can be achieved by down-sampling the head classes and up-sampling the tail classes, either in the data space or in the feature space [94, 6]. While the number of samples in the rare classes is very limited, the data re-sampling can be integrated with pseudo sample synthesis [16, 107] and mix-up operations [136], where the main idea is to generate pseudo tail samples with the help of the head samples. The knowledge transfer across the class popularities is also explored with meta-learning techniques [112, 80]. Another way to re-balance the training is to design class-specific optimization objectives to compensate for the bias in the loss, treating tail classes as hard samples [24] and making the training objective adaptive to the class frequency [20, 8]. While the re-balancing techniques are effective in creating more balanced decision boundaries, they may not be optimal for feature learning in an end-to-end manner. As a result, later works adapted these ideas to model ensembling [118, 137] and multi-stage (decoupled) training [61, 136] to boost the performance.

1.2 Contributions of the Thesis

In this thesis, we seek a more general definition of long-tail recognition and focus on developing a classifier that is more *realistic* like humans, which is robust and adaptive to the distribution shift. This is what modern classifiers are lacking and is critical for building reliable AI systems that can work in the wild.

Beyond the conventional setting introduced in section 1.1, we first explore a factorial long-tail problem with visual relation learning. We verify that the devil is in the tails. By looking into the long-tail essence, we show that significant improvement in performance can be achieved with appropriate data sampling, even with compact architectures. However, this is not yet aligned with human cognition because humans adopt class taxonomies when making predictions.

To study this direction, we consider a hierarchical label space for an image recognition problem with a single long-tail distribution. We combine realism with taxonomic classification, letting the model decide at which hierarchical level to provide the predictions according to its competent level. This is shown to be a more effective solution for long-tail recognition since the model performance can be improved across different class popularities.

With the recent paradigm shift on large foundation models, the classification problem has been changed. Unlike traditional classifiers that can only support a fixed label set, large visual-language models (VLMs) like CLIP [89] support inference with any label set. While these VLMs are trained with image captions generated by humans, which still suffer from the long-tail problem [103, 134], we hypothesize that CLIP predictions may have some hierarchical consistency issues because some concepts (class names) appear more frequently than others in captions. We verify this hypothesis and propose a plug-and-play solution to fix this inconsistency.

1.2.1 Learning of Visual Relations: The Devil is in the Tails

Significant effort has been recently devoted to modeling visual relations. This has mostly addressed the design of architectures, typically by adding parameters and increasing model com-

plexity. However, visual relation learning is a long-tailed problem, due to the combinatorial nature of joint reasoning about groups of objects. Increasing model complexity is, in general, ill-suited for long-tailed problems due to their tendency to overfit. In this paper, we explore an alternative hypothesis, denoted *the Devil is in the Tails*. Under this hypothesis, better performance is achieved by keeping the model simple but improving its ability to cope with long-tailed distributions. To test this hypothesis, we devise a new approach for training visual relationship models, which is inspired by state-of-the-art long-tailed recognition literature. This is based on an iterative decoupled training scheme, denoted Decoupled Training for Devil in the Tails (DT2). DT2 employs a novel sampling approach, Alternating Class-Balanced Sampling (ACBS), to capture the interplay between the long-tailed entity and predicate distributions of visual relations. Results show that, with an extremely simple architecture, DT2-ACBS significantly outperforms much more complex state-of-the-art methods on scene graph generation tasks. This suggests that the development of sophisticated models must be considered in tandem with the long-tailed nature of the problem.

1.2.2 Deep Realistic Taxonomic Classifier for Long-tail Recognition

Long-tail recognition tackles the naturally non-uniform distributions in real-world scenarios. While modern classifiers perform well on populated classes, their performance degrades significantly on tail classes. Humans, however, are less affected by this since, when confronted with uncertain examples, they simply opt to provide coarser predictions. Motivated by this, a *deep realistic taxonomic classifier* (Deep-RTC) is proposed as a new solution to the long-tail problem, combining realism with hierarchical predictions. The model has the option to reject classifying samples at different levels of the taxonomy, once it cannot guarantee the desired performance. Deep-RTC is implemented with a stochastic tree sampling during training to simulate all possible classification conditions at finer or coarser levels and a rejection mechanism at inference time. Experiments on the long-tailed version of four datasets, CIFAR100, AWA2, Imagenet, and iNaturalist, demonstrate that the proposed approach preserves more information on all classes with different popularity levels. Deep-RTC also outperforms the state-of-the-art

methods in long-tail recognition, hierarchical classification, and learning with rejection literature using the proposed *correctly predicted bits* (CPB) metric.

1.2.3 Taxonomic Open Set Classification

Visual-language foundation models, like CLIP, learn generalized representations with large-scale, web-crawled, image-caption pairs that enable zero-shot open-set classification. Since web-crawled data are long-tail distributed by nature, the model does not fare well in the *taxonomic open set* (TOS) setting, where the classifier is asked to make a prediction from a label set across different levels of semantic granularity. Frequently, they infer incorrect labels at coarser taxonomic class levels, even when the inference at the leaf level (original class labels) is correct. To address this problem, we propose a prompt tuning technique that calibrates the hierarchical consistency of model predictions. A set of metrics of hierarchical consistency, the Hierarchical Consistent Accuracy (HCA) and the Mean Treecut Accuracy (MTA), are first proposed to evaluate TOS model performance. A new *Prompt Tuning for Hierarchical Consistency* (ProTeCt) technique is then proposed to calibrate classification across label set granularities. Results show that ProTeCt can be combined with existing prompt tuning methods to significantly improve TOS classification without degrading the leaf-level classification performance.

1.3 Organization of the Thesis

The rest of the thesis is structured as follows. Chapter 2 explores the combinatorial long-tail distributions under the scenario of visual relation predictions, where relation triplet *subject - predicate - object* is a combination of a predicate and two entities (i.e. subject, object), which are both long-tailed distributed. We show that performance can be largely improved by looking into the long-tail essence, even with a compact architecture. In Chapter 3, we study a hierarchical label space for an image classification task that involved single long-tail distribution. We propose a taxonomic classifier for *realistic* recognition of long-tail datasets. The approach learns a hierarchical label set to guide the classifier to make predictions at a competent level,

generating coarse-grained but accurate information for uncertain samples. This strategy can improve the prediction accuracy across the class popularity without compromising the head class performance. In Chapter 4, we study the long-tail label space distribution in the large visual-language foundation models. These models are trained with large image-caption pairs, where different concepts may appear more frequently than others. User-defined label space, however, can be very different and across different granularities. We introduce the *taxonomic open set* (TOS) setting to study this problem and propose ProTeCt to improve the model's robustness on the label space shift within the class hierarchy. Chapter 5, we summarize and conclude the thesis.

Chapter 2

Learning of Visual Relations: The Devil is in the Tails

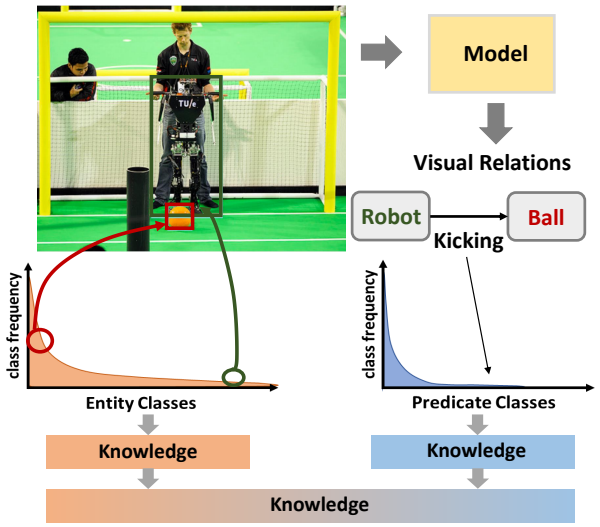


Figure 2.1. The devil is in the tails: Architecture design and learning process of visual relations need to consider the long-tailed nature of both entity and predicate class distributions.

2.1 Introduction

Scene graphs provide a compact structured description of complex scenes and the semantic relationships between objects/entities. Modeling and learning such visual relations benefit several high-level Vision-and-Language tasks such as caption generation [126, 125], visual question answering [51], image retrieval [57, 109], image generation [58, 72, 102] and robotic manipulation planning [87]. Scene graph generation requires the understanding of the locations and the class associated with the entity as well as the relationship between a pair of entities. The relationship between a pair of entities is usually formulated as a $< subject - predicate - object >$ tuple, where subject and object are two entities. Scene graph generation (SGG) faces the challenges from both the long-tailed entity recognition problem and visual relation recognition problem.

While long-tailed entity recognition has been addressed in the literature [80, 8, 20, 61], the imbalance becomes more prevalent for the SGG tasks, owing to the severe long-tailed nature of the predicate distribution. Take Figure 2.1 for example. While the class of the subject ("ball") is popular, the class of the object ("robot") and the predicate ("kicking") can be infrequent, leading to the rare occurrence of the tuple "robot-kicking-ball". This shows that even when the

entity class distribution is balanced, the imbalanced predicate class distribution can lead to a more imbalanced tuple distribution. Of course, such imbalance issues can be exacerbated if both entity classes and predicate classes are skewed (e.g. "tripod-mounted-on-donkey"). The combination of long-tailed entity and predicate classes makes SGG a more challenging problem.

While the long-tailed problem poses a great challenge to SGG tasks, it has not been well addressed in the SGG literature. Existing works [130, 124, 10, 101, 131] instead focused on designing more complex models, primarily by adding architectural enhancements that increase model size. While this has enabled encouraging performance under the Recall@k (R@k) metric, this metric is biased toward the highly populated classes. This suggests that prior works may be overfitting on popular predicate classes (e.g. *on/has*), but their performances could degrade on the less frequent classes (e.g. *eating/riding*). Such a bias towards the populated classes is problematic, because predicates lying in the tails often provide more informative depictions of scene content. The failure to predict tail classes could lead to a less informative scene graph , limiting the effectiveness of scene graphs for intended applications. In this paper, we explore the hypothesis that the Devil is in the tails. Under this hypothesis, visual relation learning is better addressed by a simple model of improved ability to cope with long-tailed distributions.

To investigate this hypothesis, we first analyze the distribution of entity and predicate classes in the Visual Genome dataset. As shown in Figure 2.2, both distributions are heavily skewed, but with different magnitude. The imbalance in the predicate distribution is more severe than that in the entity distribution. To the best of our knowledge, none of the existing SGG methods considered the jointly long-tailed distributions of entity and predicate classes. To address this, we propose a new approach to visual relationship learning, based on a simpler architecture than those in the literature but a more sophisticated training procedure, denoted *Decoupled Training for Devil in the Tails (DT2).*

DT2 is a generalization of the decoupled training procedures that have recently become popular for long-tailed recognition [61]. It consists of an alternative sampling scheme that produces distributions balanced for entities and predicates. This is accompanied by a novel

sampling scheme, *Alternating Class-Balanced Sampling (ACBS)*, which captures the interplay between the two different long-tailed distributions through an implementation of learning without forgetting [74] based on a mechanism that introduces memory between the sampling iterations, using knowledge distillation. With DT2, we show that a simple architecture with **10**$\times$ fewer parameters significantly outperforms prior, and more sophisticated, architectures designed for SGG, under the mRecall@K metric, which is suited for measuring the performance of a long-tailed dataset. Ablation studies of different sampling schemes as well as analysis of performance on classes of different popularity further validate our hypothesis.

Overall, the paper makes three contributions. 1) We devise a simple model architecture with the decoupled training scheme, namely **DT2**, suited for the long-tailed SGG tasks. 2) We propose a novel sampling strategy, **Alternating Class-Balanced Sampling (ACBS)**, to capture the interplay between different long-tailed distributions of entities and relations. 3) The combined **DT2-ACBS** significantly outperforms state-of-the-art methods of more complex architectures on all SGG tasks on the Visual Genome benchmark. The code is available on the project website[1].

2.2 Related work

2.2.1 Scene graph generation

Several works have addressed the generation of scene graphs for images [128, 122, 129, 47, 116, 120, 130, 124, 73, 39, 10, 101, 52, 131, 25]. Most approaches focus on either sophisticated architecture design or contextual feature fusion strategies, such as message passing and recurrent neural networks [130, 101], to optimize SGG performance on the Visual Genome dataset [64] under the Recall@K metric. While these approaches achieved gains for highly populated classes, underrepresented classes tend to have much poorer performance. Recently, [10, 100, 122, 115, 70] started to address the learning bias induced by the dataset statistics, by using a more suitable evaluation metric, mRecall@K, which averages recall values across

[1] http://www.svcl.ucsd.edu/projects/DT2-ACBS

classes. To address the dataset bias, TDE [100] employed causal inference in the prediction stage , whereas [115] used a pseudo-siamese network to extract balanced visual features, and PCPL [122] harnessed implicit correlations among predicate classes and used a complex graph encoding module consisting of a number of stacked encoders and attention heads. A concurrent work [70] introduces confidence-based gating with bi-level data resampling to mitigate the training bias. These methods considered, at most, the long-tailed distribution of either predicates or entities and do not disentangle the gains of sampling from those of complex architectures. For example, [122] proposed a contextual feature generator via graph encoding with 6 stacked encoders, each with 12 attention heads and a feed-forward network. We argue that long-tailed distributions should be considered for both entities and predicates and show that, when this is done, better results can be achieved with a much simpler architecture.

2.2.2 Long-tailed recognition

Prior work addresses the long-tailed issue in 3 directions: data re-sampling, cost-sensitive loss and transfer learning.

Data resampling [43, 41, 142, 42, 28, 9] is a popular strategy to oversample tail (underrepresented) classes and undersample head (populated) classes. Oversampling is achieved either by duplicating samples or by synthesizing data [41, 142, 9]. While producing a more uniform training distribution, recent works [61, 137] argue that this strategy is unsuitable for deep representation learning like CNN. [61] decouples the representation learning from the classifier learning, adopting different sampling strategies in the two stages, whereas [137] proposes a two-stream model with a mixed sampling strategy. The proposed method lies in this direction, since we consider different distributions of entity and predicate classes, and adopt different sampling strategies for training different model components.

Cost-sensitive losses [24, 20, 8, 75] assign different costs to the incorrect prediction of different samples, according to class frequency [20, 8] or difficulty [24, 75]. This is implemented by assigning higher weights or enforcing larger margins for classes with fewer samples. Weights

can be proportional to inverse class frequency or effective number [20] and can be estimated by meta-learning [53]. This re-weighting strategy was recently applied to the scene graph literature [122] to overcome long-tailed distributions.

Transfer learning methods transfer information from head to tail classes. [112, 113] learns to predict few-shot model parameters from many-shot model parameters, and [80] proposes a meta-memory for knowledge sharing. [117] leverages a hierarchical classifier to share knowledge among classes. [118] learn an expert model for each class popularity, and combine them by knowledge distillation.

2.3 Formulation and data statistics

In this section, we review the problem of learning visual relations and discuss its long-tailed nature.

2.3.1 Definitions

The inference of the visual relationships in a scene is usually formulated as a three stage process. The objects/entities in the scene are detected, classified, and the relationships between each pair of entities, in the form of predicates, are finally inferred. [57] formulated these stages with a *Scene Graph*. Let C and P be the set of entity and predicate classes, respectively. Each entity $e=(e^b,e^c)\in\mathscr{E}$ is composed by a bounding box $e^b\in\mathbb{R}^4$ and a class label $e^c\in C$. A relation $r=(s,p,o)$ is a three-tuple, connecting a subject s and an object o identities ($s,o\in\mathscr{E}$), through a predicate $p\in P$. For example, *person-riding-bike*. The scene graph $G=(E,R)$ of an image I contains a set of entities $E=\{e_i\}_{i=1}^m$ and a set of relations $R=\{r_j\}_{j=1}^n$ extracted from the image. This can be further decomposed into a set of bounding boxes $B=\{e_i^b\}_{i=1}^m$, a set of class labels $Y=\{e_i^c\}_{i=1}^m$, and a set of relations R.

The generation of a scene graph G from an image I is naturally mapped into the prob-

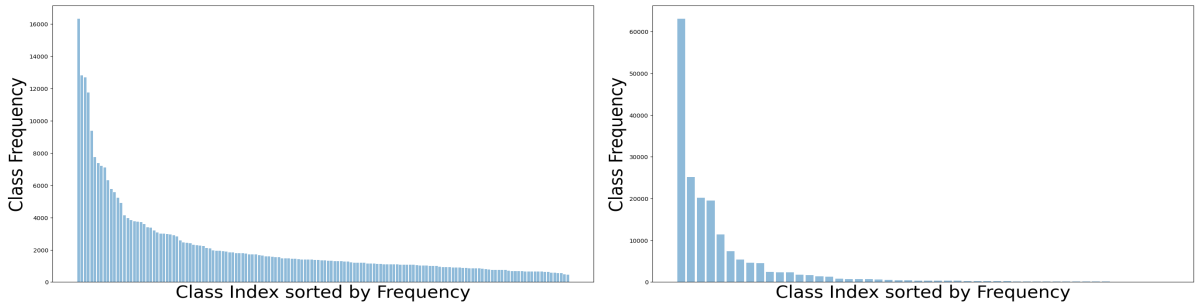


Figure 2.2. Object classes (left) and predicate classes (right) are both long-tailed distributed in Visual Genome (VG150).

abilistic model

$$Pr(G|I) = Pr(B|I)Pr(Y|B,I)Pr(R|B,Y,I), \tag{2.1}$$

where $Pr(B|I)$ is a bounding box prediction model, $Pr(Y|B,I)$ an entity class model and $Pr(R|B,Y,I)$ is a predicate class model. Joint inference of the three tasks is referred to as **Scene Graph Detection (SGDet)**. However, because bounding box prediction has been widely studied in object detection [91], it is possible to simply adopt an off-the-shelf detector. This motivates two other tasks: **Predicate classification (PredCls)**, where both bounding boxes and entity classes are given, and **Scene Graph Classification (SGCls)**, where only bounding boxes are known.

2.3.2 Long-tailed visual relations

Long-tailed distributions are a staple of the natural world, where different classes occur with very different frequencies. For example, while some entity classes (e.g. chair) occur very frequently, others (e.g donkey) are much less frequent. Long tails are problematic because, under standard loss functions and evaluation metrics, they encourage machine learning systems to over-fit on a few head classes and ignore a large number of tail classes. Recent works [80, 20, 137, 61] have shown that sampling techniques which de-emphasize popular classes, giving more weight to rare ones, can induce very large recognition gains when distributions are long-tailed. However, the issue has not been thoroughly considered in the visual relations literature.

This is somewhat surprising, given the combinatorial dependence of visual relationships on entities and predicates. Since entities are long-tailed, relationships between pairs of entities have even more skewed distributions. For example, because the entity classes "donkey" and "cliff" are less frequent than "chair" and "leg", the relation "donkey-on-cliff" is much less frequent than "chair-has-leg". This, however, is not the only source of skew, since predicates can be rare even when associated entity classes are popular, e.g. *playing* is much less popular than *has*. Finally, relationships can be rare even when involving frequent entities and predicates, e.g. the relation "car-has-wheel" is much more likely than "car-has-camera". For all these reasons, very long tails are unavoidable for visual relations. This is quite visible in the widely used Visual Genome [64] dataset. As shown in Figure 2.2, both the distribution of entity and predicate classes are long-tailed. For entities, the most populated class is 35$\times$ larger than the least populated. For predicates, the former is 12,000$\times$ larger than the latter (5,000$\times$ if the least frequent predicate class is discarded). Note that this is much larger than the square of the ratio between entity classes (1,225) suggested by the factorial nature of relationships.

The long-tailed problem is exacerbated by the evaluation protocol, based on the Recall@K (R@K) measure, adopted in most of the scene graphs literature. This measures the average percentage of ground truth relation triplets that appear in the top K predictions and, like any average, is dominated by the most frequent relationship classes. Hence, it does not penalize solutions that simply ignore infrequent relationship classes. Since most works, e.g. [101, 25, 10], focus on designing ever more complex network architectures to optimize R@K performance, it is unclear whether all that is being accomplished is stronger overfitting to a few dominant classes (e.g. "on"). This is undesirable for two reasons. First, the number of infrequent relations is much larger than that of dominant relationships. Second, while dominant relations include many obvious contextual relationships (e.g."car-has-wheels"), infrequent ones are potentially more informative (e.g. "monkey-playing-ball") of the scene content. In summary, the focus on optimizing R@K could lead to systems that are only capable of detecting a few relationships of relatively low information content.

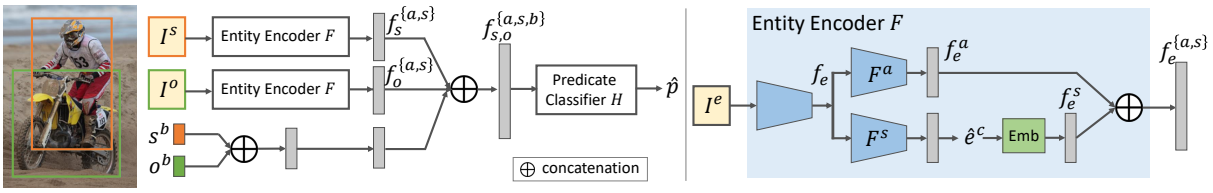


Figure 2.3. The model architecture of DT2 is composed of an entity encoder F (right) and a predicate classifier H.

This problem has been recognized in the recent literature, where some works [10, 100] have started to adopt the mRecall@K (mR@K) metric, which first averages the recall of triplets within the same predicate class and then averages the class recalls over all the predicate classes. While this is a step in the right direction, it is not sufficient to account for class imbalance *only* at the evaluation stage. Instead, the learning algorithm should explicitly address this imbalance. This leads to an alternative hypothesis that we explore in this work: *Is the devil in the tails?* Or, in other words, can a simple model designed explicitly to cope with the long-tailed nature of visual relations outperform existing models, which are much more complex but ignore this property? To investigate this hypothesis, we introduce a solution that uses a model much simpler than recently proposed architectures, but is much more sophisticated in its use of sampling techniques that target the long-tailed nature of visual relationship.

2.4 Method

In this section, we introduce the proposed network architecture, losses, and the training procedure.

2.4.1 Notations

For a relation tuple $r_j = (s_j, p_j, o_j)$ in image I, p_j is the ground truth predicate class, while $s_j = (s_j^b, s_j^c)$ and $o_j = (o_j^b, o_j^c)$ are the subject and object entities, composed of its associated bounding box coordinates (e.g. s_j^b) and ground truth entity class (e.g. s_j^c). The bounding boxes of an entity can be either the ground truth coordinates or the predictions from a detection

model, depending on the task of interest (i.e. SGCls or SGDet). With the bounding boxes, the corresponding image patch I_j^s and I_j^o for the subject and object can be cropped from the image I.

In addition, we define ρ as a probability vector at the output of the softmax function with temperature τ, and its i^{th} entry is formulated as

$$\rho_i(f,\mathbf{W},\tau) = \frac{\exp(\mathbf{w}_i^T f/\tau)}{\sum_k \exp(\mathbf{w}_k^T f/\tau)} , \tag{2.2}$$

where $f \in \mathscr{R}^d$ is a feature vector, $\mathbf{W} \in \mathscr{R}^{d\times k}$ is the matrix of k weight parameters $\mathbf{w}_k \in \mathscr{R}^d$.

2.4.2 Model architecture

Figure 2.3 summarizes the architecture of the *Decoupled Training for Devil in the Tails* (DT2) model. This combines an entity encoder F, as shown in the right part of Figure 2.3, and a predicate classifier H. DT2 takes the bounding box coordinates s_j^b, o_j^b [12] and the corresponding cropped image patches I_j^s and I_j^o as input. The entity encoder F is then applied to both I_j^s and I_j^o, to extract a pair of subject-object feature vectors $f_s^{\{a,s\}}$,$f_o^{\{a,s\}}$ that represent both the *appearance* and *semantics* of entities s_j and o_j. These are then concatenated with an embedding of the bounding box coordinates s_j^b and o_j^b, and fed to a predicate classifier H. Implementation details of the entity encoder and the predicate classifier are elaborated below.

Entity encoder F first maps image patch I^e of entity e through a feature extractor, implemented with the first three convolutional blocks of a pretrained ResNet101 [44]. We use a faster R-CNN pre-trained for object detection on Visual Genome under regular sampling (all images are sampled uniformly). The resulting feature vector f_e is then mapped to two feature vectors, f_e^s and f_e^a, that encode semantics and appearance information respectively, through two different branches sharing identical architecture. The semantic branch $F^s(\cdot;\theta)$ of parameter θ is implemented with a stack of convolution layers (the last convolutional block of ResNet101). Its output

is then fed to a softmax layer that predicts the probability $\bar{e}^c \in [0,1]^C$ of the class of the entity e, i.e.

$$\bar{e}^c = \rho(F^s(f_e;\theta), \mathbf{W}^e, \tau = 1) \ , \tag{2.3}$$

where $\mathbf{W}^e$ is the matrix of the entity classifier weights and τ of ρ in (2.2) is set to 1. The one-hot encoding $\widehat{e}^c$ can be generated by taking the *argmax* of $\bar{e}^c$, which is then mapped into a semantic feature vector $f_e^s \in \mathbb{R}^{128}$ with a single fully connected layer.

While the semantic branch would be, in principle, sufficient to convey the entity identity to the remainder of the network, this does not suffice to infer visual relationships. For example, the detection of the "people" and "bike" entities in Figure 2.3 is not enough to infer whether the relationship is "person-standing by-bike" or "person-riding-bike". This problem is addressed by introducing the appearance branch $F^a(\cdot;\phi)$ of parameter ϕ, which outputs a feature vector $f_e^a \in \mathbb{R}^{128}$ with no pre-defined semantics, simply encoding entity appearance. Finally, the feature vectors f_e^a and f_e^s are concatenated into a vector $f_e^{\{a,s\}} \in \mathbb{R}^{256}$ that represents both the appearance and semantics of entity e.

Predicate classifier takes the subject $f_s^{\{a,s\}}$ and object $f_o^{\{a,s\}}$ feature vectors as input. These vectors are then concatenated with an embedding of subject s^b and object o^b bounding boxes produced by a fully-connected layer, to create a joint encoding $f_{\{s,o\}}^{\{a,s,b\}} \in \mathbb{R}^{520}$ of the semantics, appearance, and location of the subject-object patches I^s and I^o. The predicate classifier H is implemented with a small feature extractor $H(.,\psi)$, consisting of three layers that perform dimension reduction. The input $f_{\{s,o\}}^{\{a,s,b\}} \in \mathbb{R}^{520}$ is first transformed into a 256-dimension vector with a fully connected layer, followed by a batch normalization and a ReLU layer, the output of which is finally passed through a fully connected layer with a tanh non-linearity, to produce a final feature vector $f_{s,o} \in \mathbb{R}^{128}$. This is fed to a softmax layer to produce the probability of the predicate class

$$\bar{p} = \rho(f_{s,o}, \mathbf{W}^p, \tau = 1) \tag{2.4}$$

where $\mathbf{W}^p$ is the weight matrix of the predicate classifier.

2.4.3 Training

DT2 is trained with standard cross-entropy losses targeted on entity and predicate classification. The former is defined as

$$L_{ent} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|E_i|}\sum_{e_k \in E_i} L_{ce}(e_k^c, \bar{e}_k^c) \tag{2.5}$$

where L_{ce} denotes the cross-entropy loss, $\bar{e}_k^c$ is the output probability prediction of (2.3) and e_k^c is the ground truth one-hot code of the k^{th} entity in the set E_i from image I_i. This is complemented by a predicate classification loss

$$L_{pred} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|R_i|}\sum_{r_k=(s_k,p_k,o_k)\in R_i} L_{ce}(p_k, \bar{p}_k) \tag{2.6}$$

where $\bar{p}_k$ is the output probability of (2.4) and p_k the ground truth one-hot code for the k^{th} predicate in the set R_i of visual relations in image I_i. Both (2.5) and (2.6) are important to guarantee that the network can learn from both entities and predicate relationships.

2.4.4 Sampling strategies

While encapsulating both semantics and appearance information, the proposed training loss in Sec. 2.4.3 requires a complementary sampling strategy tailored for long-tailed data. This long-tailed problem has been studied mostly in the object recognition literature, where an image patch is fed to a feature extractor with the parameter φ and the softmax layer ρ of (2.2) with weight matrix $\mathbf{W}$. A popular training strategy is to use different sampling strategies to train the two network components [61]. The intuition is that, because the bulk of the network parameters are in the feature extractor (φ), this should be learned with the largest possible amount of data. Hence, the entire network is first trained with **Standard Random Sampling (SRS)**, which samples images uniformly, independent of their class labels.

While this produces a good feature extractor, the resulting classifier usually overfits to the head classes, which are represented by many more images and have a larger weight on the cost function. The problem is addressed by fine-tuning the network on a balanced distribution, obtained with **Class Balanced Sampling (CBS)**. This consists of sampling uniformly over classes, rather than images, and guarantees that all classes are represented with equal frequencies. However, because images from tail classes are resampled more frequently than those of head classes, it carries some risk of overfitting to the former. To avoid overfitting, the fine-tuning is restricted to the weights $\mathbf{W}$ of the softmax layer. In summary, the network is trained in two stages. First, the parameters φ and $\mathbf{W}$ are jointly learned with SRS. Second, the feature extractor (φ) is fixed and the softmax layer parameters $\mathbf{W}$ are relearned with CBS.

2.4.5 Sampling for visual relationships

Similar to long-tailed object recognition, it is sensible to train a model for visual relations in two stages. In the first stage, the goal is to learn the parameters θ, ϕ, ψ of the feature extractors (see Sec. 2.4.2), which are the overwhelming majority of the network parameters. As in object recognition, the network should be trained with SRS. In the second stage, the goal is to fine-tune the softmax parameters $\mathbf{W}^e$ and $\mathbf{W}^p$ to avoid overfitting to head classes. However, unlike long-tailed object recognition, Figure 2.2 shows that predicates and entities can have very different distributions, which makes the learning of long-tailed visual relations a distinct problem. This indicates that two class-balanced sampling strategies are required to accommodate the distribution difference between predicate and entity classes.

A straightforward solution is to introduce a 2-step iterative training procedure, namely *entity-optimization step* (E-step) and *predicate-optimization step* (P-step), to optimize the weight of $\mathbf{W}^e$ and $\mathbf{W}^p$ respectively. In E-step, images are sampled from a distribution $\mathscr{P}_e$ that is uniform with respect to entity classes, which is denoted as Entity-CBS. While in P-step, they are sampled from a distribution $\mathscr{P}_p$ uniform with respect to predicate classes, denoted as Predicate-CBS. However, since the uniform sampling of $\mathscr{P}_p$ is not class-balanced for entity classes, P-step would

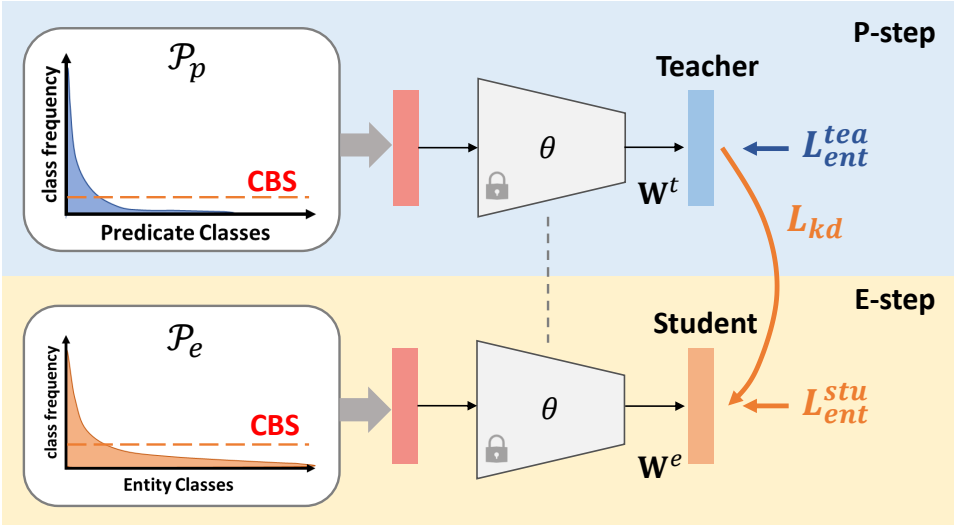


Figure 2.4. ACBS captures the interplay between the long-tailed distributions of entities and relations by implementing the knowledge distillation between P-step and E-step.

lead to the overfitting of the entity classification parameters $\mathbf{W}^e$.

To address this problem, we propose a novel sampling strategy, **Alternating CBS** (ACBS), tailored for long-tailed visual relations. ACBS contains a memory mechanism to maintain the entity predictions of the P-step, making sure that what was learned is not forgotten in the E-step. It is implemented with distillation [48] between the P-step and E-step and an auxiliary *teacher* entity classifier of weight matrix $\mathbf{W}^t$. The *teacher* entity classifier is inserted in parallel with the entity classifier of weight matrix $\mathbf{W}^e$ in (2.3), which is its *student*, and produces a second set of entity prediction probabilities as

$$\vec{e}^t = \rho(F^s(f_e;\theta),\mathbf{W}^t,\tau=1). \tag{2.7}$$

With the introduction of the teacher entity classifier, we rewrite (2.5) into L_{ent}^{stu} and L_{ent}^{tea}, where the former operates on $\vec{e}^c$ of (2.3) and the latter operates on $\vec{e}^t$. Furthermore, to distill knowledge

Algorithm 1. Training procedure of ACBS

Input: Training dataset $\mathscr{D}$, predicate distribution $\mathscr{P}_p$, entity distribution $\mathscr{P}_e$, ACBS hyperparameters (α,β,τ_s), and model parameters (θ,ϕ,ψ).
Output: Model parameters ($\mathbf{W}^p$, $\mathbf{W}^e$).

```
while Not convergence do
    // P-Step
    D_p ← BalancedSample(D, P_p)  while batch in D_p do
        L_total ← L_pred (2.6) + β L_ent^tea (2.5)
        Minimize L_total with respect to (W^p, W^t)
    end
    // E-Step
    D_e ← BalancedSample(D, P_e)  while batch in D_e do
        L_total ← L_ent^stu (2.5) + α L_kd (2.8)
        Minimize L_total with respect to W^e
    end
end
```

from the teacher entity classifier, a Kullback-Leibler divergence (KL) loss (L_{kd}) is defined as

$$\mathrm{KL}(\rho(F^s(f_e;\theta),\mathbf{W}^e,\tau=\tau_s)||\rho(F^s(f_e;\theta),\mathbf{W}^t,\tau=\tau_s)), \tag{2.8}$$

where the two inputs to L_{kd} are the smooth version of (2.3) and (2.7) with temperature τ_s.

In summary, the P-step updates parameters $\mathbf{W}^p$ of the predicate classifier and $\mathbf{W}^t$ of the teacher with (2.6) and L_{ent}^{tea} respectively, while the student parameters $\mathbf{W}^e$ are kept fixed. In the E-step, $\mathbf{W}^p$ and $\mathbf{W}^t$ (teacher) are kept fixed, and $\mathbf{W}^e$ (student) is optimized with L_{ent}^{stu} and (2.8). This implements learning without forgetting [74] between the two steps, encouraging the student classifier to mimic the predictions of the teacher classifier, and enabling the network to learn the new parameters for one distribution, e.g. $\mathbf{W}^e$, without forgetting the one, e.g. $\mathbf{W}^t$, previously learned for the other. The training procedure is detailed in Algorithm 1.

2.5 Experiments

In this section, several experiments are performed to validate the effectiveness of DT2-ACBS.

Table 2.1. The result (mRecall@K) of SGG tasks (PredCls, SGCls, SGDet) compared to SOTA in scene graphs. Results for other methods are reported from the corresponding paper in general. † denotes our reproduced model with ResNet101-FPN backbone.

Method	Predicate Classification			Scene Graph Classification			Scene Graph Detection		
	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100	mR@20	mR@50	mR@100
IMP+ [120]	-	9.8	10.5	-	5.8	6.0	-	3.8	4.4
FREQ [130]	8.3	13.0	16.0	5.1	7.2	8.5	4.5	6.1	7.1
MOTIFS [130]	10.8	14.0	15.3	6.3	7.7	8.2	4.2	5.7	6.6
MOTIFS [130]†	13.2	16.3	17.5	7.1	8.8	9.3	4.9	6.7	8.2
KERN [10]	-	17.7	19.2	-	9.4	10.0	-	6.4	7.3
VCTree [101]	14.0	17.9	19.4	8.2	10.1	10.8	5.2	6.9	8.0
GBNet [128]	-	22.1	24.0	-	12.7	13.4	-	7.1	8.5
TDE-MOTIFS-SUM [100]	18.5	25.5	29.1	9.8	13.1	14.9	5.8	8.2	9.8
TDE-MOTIFS-SUM [100]†	17.9	24.8	28.6	9.6	13.0	14.7	5.6	7.7	9.1
TDE-VCTree-SUM [100]	18.4	25.4	28.7	8.9	12.2	14.0	6.9	9.3	11.1
TDE-VCTree-GATE [100]	17.2	23.3	26.6	8.9	11.8	13.4	6.3	8.6	10.3
PCPL [122]	-	35.2	37.8	-	18.6	19.6	-	9.5	11.7
DT2-ACBS (ours)	**27.4**	**35.9**	**39.7**	**18.7**	**24.8**	**27.5**	**16.7**	**22.0**	**24.4**

Table 2.2. mR@100 on SGG tasks for head, middle, tail classes. † denotes our reproduced models with ResNet101-FPN backbone.

Method	Predicate Classification			Scene Graph Classification			Scene Graph Detection		
	Head (16)	Middle (17)	Tail (17)	Head (16)	Middle (17)	Tail (17)	Head (16)	Middle (17)	Tail (17)
MOTIFS [130]†	42.3	9.8	0.6	24.6	4.0	0.1	20.2	4.6	0.4
TDE-MOTIFS-SUM [100]†	**44.9**	35.8	6.1	**25.6**	15.8	3.3	22.2	5.6	0.1
DT2-ACBS (ours)	35.1	**45.2**	**38.6**	24.6	**29.1**	**28.6**	**22.3**	**26.7**	**24.0**

2.5.1 Dataset

Visual Genome (VG) [64] is composed of 108k images across 75k object categories and 37k predicate categories, but 92% of the predicates have less than 10 instances. Following prior works, we use the original splits of the popular subset (i.e. VG150) for training and evaluation. It contains the most frequent 150 object classes and 50 predicate classes. The distribution remains highly long-tailed. To perform balanced sampling during training, predicate classes with less than 5 instances, e.g. "flying in," are ignored.

2.5.2 Comparison to SOTA

To validate our hypothesis, we compare DT2-ACBS with the state-of-the-art methods on PredCls, SGCls and SGDet task on the popular subset VG150 of VG [64], under the mRecall@K metric. As shown in Table 2.1, compared baselines include 1) simple frequency-based method [130], 2) sophisticated architecture design for contextual representation learn-

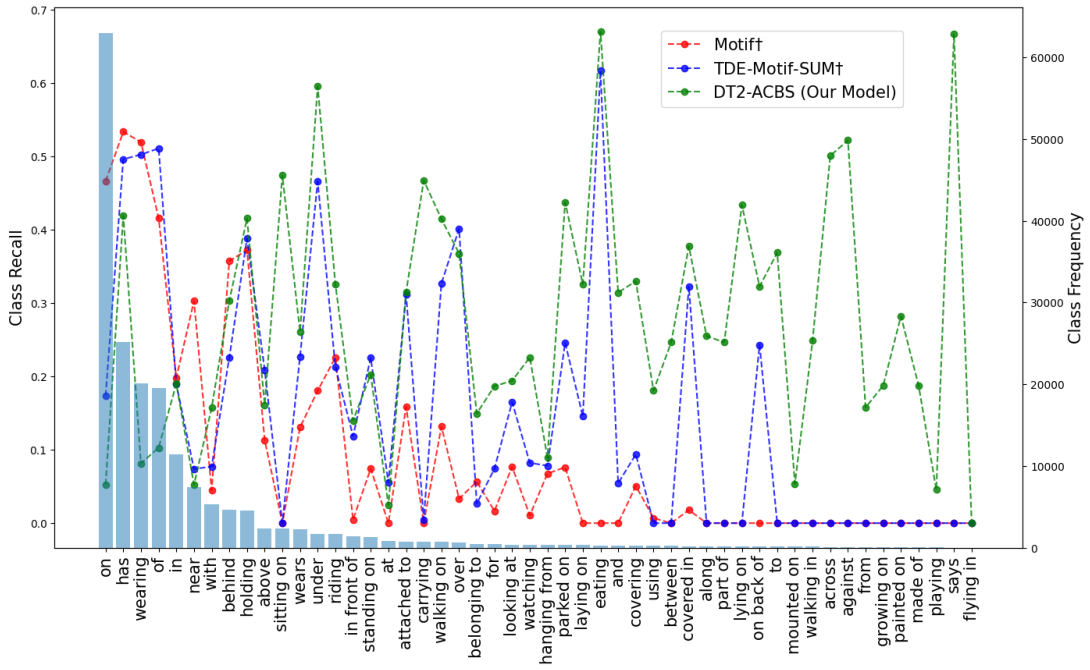


Figure 2.5. Comparisons of per class Recall@100 on SGCls. Classes are sorted in decreasing order of the number of samples.

ing [120, 10, 101, 128] and 3) recent works that tackle the long-tailed bias of predicate classes [100, 122]. Several observations can be made. First, DT2-ACBS outperforms all baselines in the first two groups by a large margin (mR@100 gain larger than 15.7%) on the PredCls task, where entity bounding boxes and categories are given. The baselines in the third group [100, 122], which address the long-tailed bias of the predicate distribution, are similar in spirit to DT2-ACBS. However, the latter relies on a simpler model design and a more sophisticated decoupled training scheme to overcome overfitting. This enables a 1.9% improvement on mR@100 (5% relative improvement), showing the efficacy of the proposed sampling mechanism for tackling the long-tailed problem in predicates distribution.

Next, when predicting both predicate and entity class given the ground truth bounding boxes (SGCls task), DT2-ACBS outperforms all existing methods by a larger mR@100 margin (1.9% on PredCls vs 7.9% on SGCls, equivalently relative improvement of 5% in PredCls vs 40% in SGCls). This significant improvement in SGCls performance can be ascribed to the decoupled training of ACBS, which better captures the interplay between the different distributions of

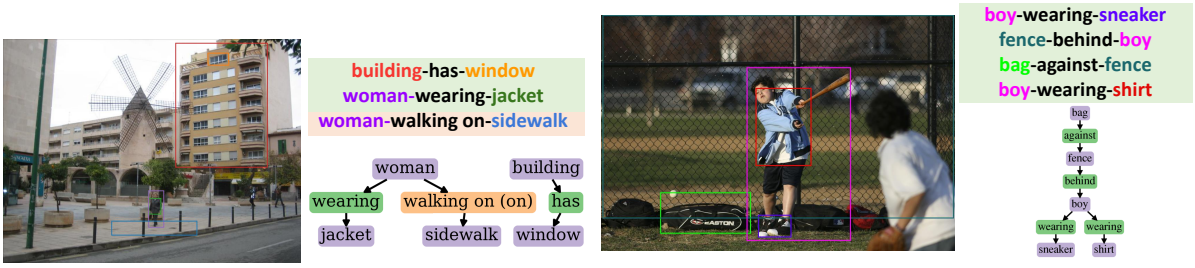


Figure 2.6. Qualitative results of PredCls (left) and SGCls (right). In each sub-figure, colors of bounding boxes in the image (left) are corresponding to the entities in the triplets (upper-right) with the background color green/orange for correct/incorrect predicate predictions. In the generated graphs (lower-right), correct/incorrect predictions of entities and predicates are shown in purple/blue and green/orange respectively, with the ground truth noted in the bracket (best viewed in color). More examples are shown in the supplemental.

entities and predicates.

Finally, we also ran DT2-ACBS on proposal boxes generated by a pre-trained Faster-RCNN for the SGDet task.Table 2.1 shows that DT2-ACBS outperforms existing methods by a significantly larger mR@100 margin of 12.7% ($>100\%$ relative improvement) on the SGDet task.

Class-wise performance analysis: To study the performance of classes with different popularity, we sort the 50 relation classes by their frequencies and divide them into 3 equal parts, head (16), middle (17) and tail (17). Table 2.2 presents the mR@100 performance on these partitions for each SGG task. As observed in prior long-tailed recognition work [80, 61], a performance drop in head classes is hard to avoid while improving tail class performance. The goal, instead, is to achieve the best balance among all the classes, which DT2-ACBS clearly does with notable improvements in the middle and tail classes. It should also be noted that the drop in head performance can be deceiving, due to dataset construction problems like "wearing" and "wears" appearing as two different relationship classes. Most importantly, many VG150 tail categories (e.g. "standing on", "sitting on") are fine-grained versions of a head category ("on"). Some of the degradation in head class performance is just due to the predicates being pushed to the fine-grained classes, which is more informative. We notice that one of the high-frequency predicate classes *On* has a low recall value (Figure 2.5) and observe that DT2-ACBS often instead predicts its fine-

Table 2.3. Ablations on different sampling strategies for SGCls.

Method	mR@20	mR@50	mR@100
Single Stage-SRS	6.4	9.6	11.2
Single Stage-Indep. CBS	8.5	11.2	12.4
DT2-Predicate-CBS	10.0	13.0	14.3
DT2-Indep. CBS	17.3	23.9	26.7
DT2-ACBS (ours)	**18.7**	**24.8**	**27.5**

grained sub-categories, such as *standing on*, *sitting on*, *mounted on*. In particular, there are 41,620 ground truth instances of *On* predicate in the test set, and DT2-ACBS predicts *On*-subcategories 14,317 times on PredCls, which constitutes 34% incorrect predictions as per the metric. Overall, DT2-ACBS performs significantly better in middle and tail classes on SGG tasks, and performs comparably on head classes for SGCls and SGDet, reaching the best balance across all the classes.

2.5.3 Ablations on sampling strategies

SGCls performance is affected by the intertwined entity and predicate distributions. In this section, we conduct ablation studies in Table 2.3 on 1) single-stage vs two-stage training and 2) different sampling schemes. The first half of the table shows the performances of single-stage training, where the representation and the classifier are learned together. This clearly underperforms the two-stage training, which is listed in the second half of the table, where we compare different sampling strategies in the second stage of DT2. For the predicate classifier, it can be trained based on either SRS or class-balanced sampling for predicates (Predicate-CBS). Since each relation comes with a subject and an object, it is possible to train the entity classifier with respect to Predicate-CBS, indicating the entity classifier can be trained based on SRS, Predicate-CBS or class-balanced sampling for entities (Entity-CBS). Note that the predicate classifier can not be trained with Entity-CBS, since an entity does not always belong to a visual relation tuple. From the second half of the table, we find that considering the distribution differences in predicates and entities is important, because DT2-Predicate CBS (i.e. Predicate-CBS for both entity and predicate classifier) does not perform as well as DT2-Indep. CBS (i.e. Entity-CBS

for the entity classifier and Predicate-CBS for the predicate classifier). The observation that DT2-Indep. CBS already performs better than existing methods (Table 2.1) supports our claim that visual relations can be effectively modeled with a simple architecture if the long-tailed aspect of the problem is considered. Nevertheless, the proposed ACBS further improves the SGCls performance by distilling the knowledge between P-step and E-step (see Algorithm 1).

2.5.4 Qualitative results

Figure 2.6 presents qualitative results of DT2-ACBS. In PredCls task, DT2-ACBS can correctly predict populated predicate classes (*has* & *wearing*) as well as non-populated predicate classes (*walking on*). Not only robust to long-tailed predicate classes, DT2-ACBS is also able to classify entities ranging from more populated classes (*boy*) to tail classes (*sneaker*). We can observe that while the predicted predicates can be different from the ground truth, the relation can still be reasonable (e.g. a *subclass* or a *synonym* of the ground truth). For example, the predicted predicate "walking on" is actually a subclass of the ground truth predicate "on". These examples show that DT2-ACBS is able to predict more fine-grained predicates in tail classes and provide more exciting descriptions of the scene.

2.6 Conclusions

Learning visual relations is inherently a long-tailed problem. Existing approaches have mostly proposed complex models to learn visual relations. However, complex models are ill-suited for long-tailed problems, due to their tendency to overfit. In this paper, we consider the uniqueness of visual relations, where entities and relations have skewed distributions. We propose a simple model, namely DT2, along with an alternating sampling strategy (ACBS) to tackle the long-tailed visual relation problem. Extensive experiments on the benchmark VG150 dataset show that DT2-ACBS significantly outperforms the state-of-the-art methods of more complex architectures.

Chapter 2 is, in full, based on the material as it appears in the publication of "Learning

of Visual Relations: The Devil is in the Tails", Tz-Ying Wu*, Alakh Desai*, Subarna Tripathi, and Nuno Vasconcelos, In Proceedings of International Conference on Computer Vision (ICCV), 2021. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Solving Long-tailed Recognition with Deep Realistic Taxonomic Classifier

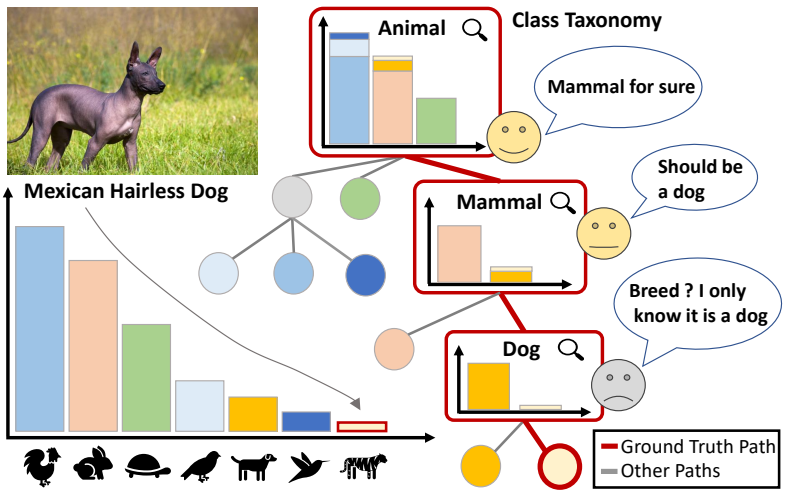


Figure 3.1. Real-world datasets have class imbalance and long tails (left). Humans deal with these problems by combining class taxonomies and self-awareness (right). When faced with rare objects, like a "Mexican Hairless Dog", they push the decision to a coarser taxonomic level, e.g., simply recognizing a "Dog", of which they feel confident. This is denoted as *realistic taxonomic classification* to guarantee that all samples are processed with a high level of confidence.

3.1 Introduction

Recent advances in computer vision can be attributed to large datasets [22] and deep convolutional neural networks (CNN) [67, 95, 44]. While these models have achieved great success on balanced datasets, with approximately the same number of images per class, real world data tends to be highly imbalanced, with a very long-tailed class distribution. In this case, classes are frequently split into many-shot, medium-shot and few-shot, based on the number of examples [80]. Since deep CNNs tend to overfit in the small data regime, they frequently underperform for medium and few-shot classes. Popular attempts to overcome this limitation include data resampling [42, 6, 28, 9], cost-sensitive losses [20], knowledge transfer from high to low population classes [80, 113], normalization [61], or margin-based methods [8]. All these approaches seek to improve the classification performance of the standard softmax CNN architecture.

There is, however, little evidence that this architecture is optimally suited to deal with long-tailed recognition. For example, humans do not use this model. Rather than striving for discrimination between all objects in the world, they adopt class *taxonomies* [99, 59, 5, 60, 4],

where classes are organized hierarchically at different levels of granularity, e.g. ranging from coarse domains to fine-grained 'species' or 'breeds,' as shown in Figure 3.1. Classification with taxonomies is broadly denoted as *hierarchical*. The standard softmax, also known as the *flat,* classifier is a hierarchical classifier of a single taxonomic level. The use of deeper taxonomies has been shown advantageous for classification by allowing feature sharing [123, 38, 141, 77, 2, 63] and information transfer across classes [84, 135, 21, 92, 93]. While most previous works on either flat or hierarchical classification attempt to classify all images at the leaves of the taxonomic tree, independently of how difficult this is, the introduction of a taxonomy enables alternate strategies.

In this work, we explore a strategy inspired by human cognition and suited for long-tailed recognition. When humans feel insufficiently trained to answer a question at a certain level of granularity, they simply provide an answer to a coarser level, for which they feel *confident.* For example, most people do not recognize the animal of Figure 3.1 as a "Mexican Hairless Dog". Instead, they change the problem from classifying dog breeds into classifying mammals and simply say it is a "Dog". Hence, a long-tailed recognition strategy more consistent with human cognition is to adopt hierarchical classification and allow decisions at *intermediate* tree levels, to achieve two goals: 1) classify all examples with high confidence, and 2) classify each example as deep in the tree as possible without violating the first goal. Since examples from low-shot classes are harder to classify confidently than those of popular classes, they tend to be classified at earlier tree levels. This can be seen as a soft version of *realistic classification* [19, 108] where a classifier refuses to process examples of low-classification confidence and is denoted *realistic taxonomic classification* (RTC). The taxonomic extension enables multiple "exit levels" for the classification, at different taxonomic levels.

RTC recognizes that, while classification at the leaves uncovers full label information, *partial* label information can still be recovered when this is not feasible, by performing the classification at intermediate taxonomic stages. The goal is then to maximize the *average* information recovered per sample, favoring correct decisions of intermediate level over incorrect decisions at the leaves. We introduce a new measure of classifier performance, denoted *correctly predicted*

bits (CPB), to capture this average information and propose it as a new performance measure for long-tailed recognition. Rather than simply optimizing classification accuracy at the leaves, high CPB scores require learning algorithms that produce calibrated estimates of class probabilities at *all* tree levels. This is critical to enable accurate determination of when examples should leave the tree. For long-tailed recognition, where different images can be classified at different taxonomic levels, this calibration is particularly challenging.

We address this problem with two new contributions to the training of deep CNNs for RTC. The first is a new regularization procedure based on *stochastic tree sampling,* (STS) which allows the consideration of all possible cuts of the taxonomic tree during training. RTC is then trained with a procedure similar to dropout [98], which considers the CNNs consistent with all these cuts. The second contribution addresses the challenge that RTC requires a *dynamic* CNN, capable of generating predictions at different taxonomic levels for each input example. This is addressed with a novel *dynamic predictor synthesis* procedure inspired by parameter inheritance, a regularization strategy commonly used in hierarchical classification [92, 93]. To the best of our knowledge, these contributions enable the first implementation of RTC with deep CNNs and dynamic predictors. This is denoted as *Deep-RTC*, which achieves leaf classification accuracy comparable to state of the art long-tail recognition methods, but recovers much more average label information per sample.

Overall, the paper makes three contributions. 1) we propose RTC as a new solution to the long-tailed problem. 2) the *Deep-RTC* architecture, which implements a combination of stochastic taxonomic regularization and dynamic taxonomic prediction, for implementation of RTC with deep CNNs. 3) an alternative setup for the evaluation of long-tailed recognition, based on CPB scores, that accounts for the amount of information in class predictions.

3.2 Related Work

This work is related to several previously explored topics.

Long-Tailed Recognition: Several strategies have been proposed to address class unbalance in recognition. One possibility is to perform data resampling [43], by undersampling head and oversampling tail classes [42, 6, 28, 9]. Sample synthesis [41, 142] has also been proposed to increase the population of tail classes. Unlike Deep-RTC, these methods do not seek improved classification architectures for long-tailed recognition. An alternative is to transfer knowledge from head to tail classes. Inspired by meta-learning [111, 112], these methods learn how to leverage knowledge from head classes to improve the generalization of tail classes [113]. [80] introduces memory features that encapsulate knowledge from head classes and uses an attention mechanism to discriminate between head and tail classes. This has some similarity with Deep-RTC, which also transfers knowledge from head to tail classes, but does so by leveraging hierarchical relations between them. Long-tailed recognition has also been addressed with cost-sensitive losses, which assign different weights to different samples. A typical approach is to weight classes by their frequency [50, 82] or treat tail classes as hard examples [24]. [20] proposed a class balanced loss that can be directly applied to a softmax layer and focal loss [75]. These approaches can underperform for very low-frequency classes. [8] addressed this problem by enforcing large margins for few-shot classes, where the margin is inversely proportional to the number of class samples. While effective losses for long-tailed recognition are a goal of this work, we seek losses for calibration of taxonomic classifiers, which cost-sensitive losses do not address. Finally, inspired by the correlation between the weight norm of a class and its number of samples, [61] proposed to adjust the former after classifier training. All these approaches use the flat softmax classifier architecture and do not address the design of RTC.

Hierarchical Classification: Hierarchical classification has received substantial attention in computer vision. For example, sharing information across classes has been used for object recognition on large and unbalanced datasets [84, 135, 21], and defining a common hierarchical semantic space across classes has been explored for zero-shot learning [86, 3]. Some of the ideas used in this work, e.g. parameter inheritance, are from this literature [92, 93, 23]. However, most of them precede deep learning and cannot be directly applied to modern CNNs. More

recently, the ideas of sharing parameters or features hierarchically have inspired the design of CNN architectures [123, 38, 141, 77, 2, 63]. Some of these do not support class taxonomies, e.g. learning hierarchical feature structures for flat classification [86, 63]. Others are only applicable to a somewhat rigid two-level hierarchy [2, 123]. Closer to this work are architectures that complement a flat classifier with convolutional branches that regularize its features to enforce hierarchical structure [38, 141, 77]. These branches can be based on hierarchies of feature pooling operators [38], or classification layers [141, 77] supervised with labels from intermediate taxonomic levels. However, the use of additional layers makes the comparison to flat classifier unfair, which would undermine an important goal of the paper: to investigate the benefit of hierarchical (over flat) classification for long-tailed recognition. Hence, we avoid hierarchical architectures that add parameters to the backbone network. These methods also fail to address a central challenge of RTC, namely the need for simultaneous optimization with respect to many label sets, associated with the different levels of the class taxonomy. This requires a dynamic network, whose architecture can change on-the-fly to enable 1) the use of different label sets to classify different samples, and 2) optimization with respect to many label sets.

Learning with Rejection The idea of learning with rejection dates back to at least [14]. Subsequent works derive theoretical results on the error-rejection trade-off [15, 30], and explore alternative rejection criteria that avoid computation of class posterior probabilities [31, 18, 19]. Since the introduction of deep learning has made the estimation of the posterior distribution central to classification, most recent rejection functions consist of thresholding posteriors or derived quantities, such as the posterior entropy [35, 36, 108]. Alternative rejection methods have also been proposed, including the use of relative distances between samples [55], Monte-Carlo dropout [32], or classification model with a routing or rejection network [108, 36, 17]. We adopt the simple threshold based rejection rule of [35, 36, 108] in our implementation of RTC. However, rejection is applied to each level of a hierarchical classifier, instead of once for a flat classifier. This resembles the hedge your bets strategy of [23, 56], in that it aims to maximize the average label information recovered per sample. However, while [23, 56] accumulate the

class probabilities of a flat classifier, our Deep-RTC addresses the calibration of probabilities *throughout* the tree. Our experiments show that this significantly outperforms the accumulation of flat classifier probabilities. [56] further calibrates class probabilities before rejection, but calibration is only conducted a posteriori (at test time). Instead, we propose STS for training hierarchical classifiers whose predictions are inherently calibrated at all taxonomic levels.

3.3 Long-tailed recognition and RTC

This section motivates the need for RTC as a solution to long-tailed recognition.

Long-tailed Recognition Existing approaches formulate long-tailed recognition as flat classification, solved by some variant of the softmax classifier. This combines a feature extractor $h(\boldsymbol{x};\boldsymbol{\Phi}) \in \mathbb{R}^k$, implemented by a CNN of parameters $\boldsymbol{\Phi}$, and a softmax regression layer composed by a linear transformation $\boldsymbol{W}$ and a softmax function $\sigma(\cdot)$

$$f(\boldsymbol{x};\boldsymbol{W},\boldsymbol{\Phi}) = \sigma(z(\boldsymbol{x};\boldsymbol{W},\boldsymbol{\Phi})) \qquad z(\boldsymbol{x};\boldsymbol{W},\boldsymbol{\Phi}) = \boldsymbol{W}^T h(\boldsymbol{x};\boldsymbol{\Phi}). \tag{3.1}$$

These networks are trained to minimize classification errors. Since samples are limited for mid and low-shot classes, performance can be weak. Long-tailed recognition approaches address the problem with example resampling, cost-sensitive losses, parameter sharing across classes, or post-processing. These strategies are not free of drawbacks. For example, cost-sensitive or resampling methods face a "whack-a-mole" dilemma, where performance improvements in low-shot classes (e.g. by giving them more weight) imply decreased performance in more populated ones (less weight). They are also very different from the recognition strategies of human cognition, which relies extensively on class taxonomies.

Many cognitive science studies have attempted to determine taxonomic levels at which humans categorize objects [99, 59, 5, 60, 4]. This has shown that most object classes have a default level, which is used by most humans to label the object (e.g. "dog" or "cat"). However, this so-called basic level is known to vary from person to person, depending on the person's

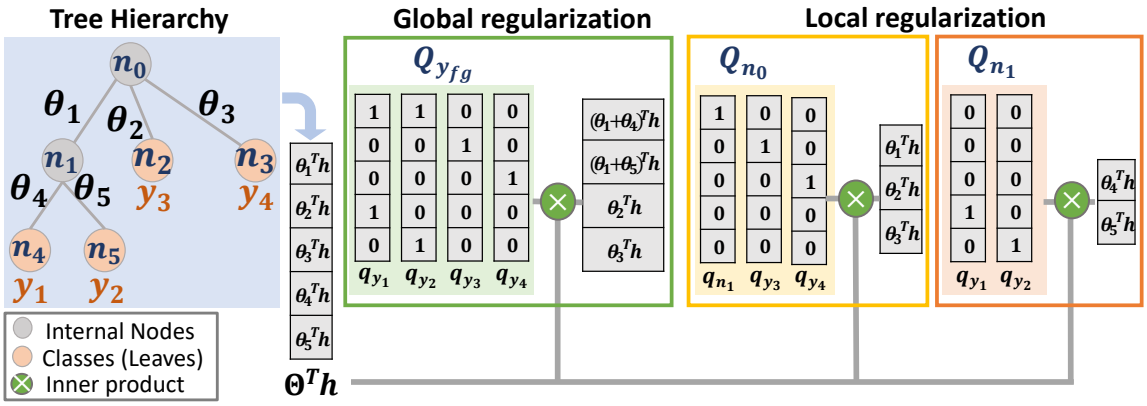


Figure 3.2. Parameter sharing based on the tree hierarchy are implemented through the codeword matrices Q. The training is regularized globally from the stochastically selected label set and locally from the node-conditional consistency loss.

training, also known as *expertise*, on the object class [99, 60, 4]. For example, a dog owner naturally refers to his/her pet as a "labrador" instead of as "dog." This suggests that even humans are not great long-tail recognizers. Unless they are experts (i.e. have been extensively trained in a class), they instead perform the classification at a higher taxonomic level. From a machine learning point of view, this is sensible in two ways. First, by moving up the taxonomic tree, it is always possible to find a node with sufficient training examples for accurate classification. Second, while not providing full label information for all examples, this is likely to produce a higher average label information per sample than the all-or-nothing strategy of the flat classifier [23, 56]. In summary, when faced with low-shot classes, humans *trade-off classification granularity for class popularity,* choosing a classification level where their training has enough examples to guarantee accurate recognition. This does not mean that they cannot do fine-grained recognition, only that this is reserved for classes where they are experts. For example, because all humans are extensively trained on face recognition, they excel in this very fine-grained task. These observations motivate the RTC approach to long-tailed recognition.

Realistic Taxonomic Classification A taxonomic classifier maps images $\boldsymbol{x} \in \mathscr{X}$ into a set of C classes $y \in \mathscr{Y} \in \{1,...,C\}$, organized into a taxonomic structure where classes are recursively grouped into parent nodes according to a tree-type hierarchy $\mathscr{T}$. It is defined by a set of

classification nodes $\mathcal{N} = \{n_1, \cdots, n_N\}$ and a set of taxonomic relations $\mathcal{A} = \{\mathcal{A}(n_1), \cdots, \mathcal{A}(n_N)\}$, where $\mathcal{A}(n)$ is the set of ancestor nodes of n. The finest-grained classification decisions admitted by the taxonomy occur at the leaves. We denote this set of fine-grained classes $\mathcal{Y}_{fg} = \text{Leaves}(\mathcal{T})$. Figure 3.2 gives an example for a classification problem with $|\mathcal{Y}_{fg}| = 4, |\mathcal{N}| = 5$, $\mathcal{A}(n_4) = \mathcal{A}(n_5) = \{n_1\}$ and $\mathcal{A}(n_i) = \emptyset, i \in \{1,2,3\}$. Classes y_1, y_2 belong to parent class n_1 and the root n_0 is a dummy node containing all classes. Note that we use n to represent nodes and y to represent leaf labels.In RTC, *different samples can be classified at different hierarchy levels.* For example, a sample of class y_2 can be rejected at the root, classified at node n_1, or classified into one of the leaf classes. These options assign successively finer-grained labels to the sample. Samples rejected at the root can belong to any of the four classes, while those classified at node n_1 belong to classes y_1 or y_2. Classification at the leaves assigns the sample to a single class. Hence, RTC can predict any sub-class in the taxonomy $\mathcal{T}$. Given a training set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{M}$ of images and class labels, and a class taxonomy $\mathcal{T}$, the *goal* is to learn a pair of classifier $f(\mathbf{x})$ and rejection function $g(\mathbf{x})$ that work together to assign each input image $\boldsymbol{x}$ to the finest grained class $\widehat{y}$ possible, while guaranteeing certain confidence in this assignment.

The depth at which the class prediction $\widehat{y}$ is made depends on the sample difficulty and the *competence-level* γ of the classification. This is a lower bound for the confidence with which $\boldsymbol{x}$ can be classified. A confidence score $s(f(\mathbf{x}))$ is defined for $f(\mathbf{x})$, which is declared competent (at the γ level) for classification of $\boldsymbol{x}$, if $s(f(\mathbf{x})) \geq \gamma$. RTC has competence level γ if all its intermediate node decisions have this competence level. While this may be impossible to guarantee for classification with the leaf label set $\mathcal{Y}_{fg}$, it can always be guaranteed by rejecting samples at intermediate nodes of the hierarchy, i.e. defining

$$g_v(\boldsymbol{x}; \gamma) = 1_{[s(f_v(\boldsymbol{x})) \geq \gamma]} \tag{3.2}$$

per classification node v, where $1_{[.]}$ is the Kroneker delta. This prunes the hierarchy $\mathcal{T}$ *dynamically* per sample $\mathbf{x}$, producing a customized cut $\mathcal{T}_p$ for which the hierarchical classifier is

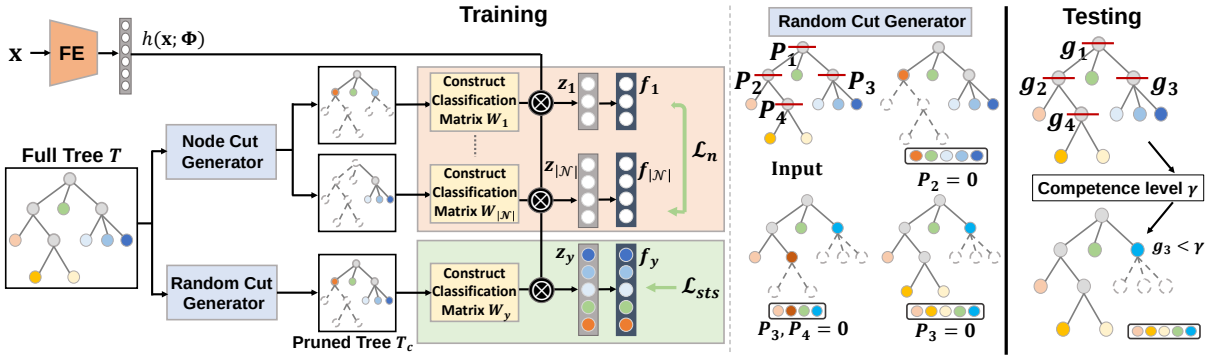


Figure 3.3. Left: Deep-RTC is composed of a feature extractor, a node cut generator producing $\mathscr{Y}_n = \mathscr{C}(n)$ for all internal nodes and a random cut generator producing a potential label sets $\mathscr{Y}_c$ from $\mathscr{T}_c$. Classification matrix $\boldsymbol{W}_{\mathscr{Y}_c}$ is constructed for each label set and loss of (3.12) is imposed. Right: Rejecting samples at certain level during inference time.

competent at a competence level γ. This pruning is illustrated on the right of Figure 3.3. Samples that are hard to classify, e.g. from few-shot classes, induce low confidence scores and are rejected earlier in the hierarchy. Samples that match the classifier expertise, e.g. from highly populated classes, progress until the leaves. This is a generalization of flat realistic classifiers [108], which simply accept or reject samples. RTC mimics human behavior in that, while $\boldsymbol{x}$ may not be classified at the finest-grained level, confident predictions can usually be made at intermediate or coarse levels. The competence level γ offers a guarantee for the quality of these decisions. Since larger values of γ require decisions of higher confidence, they encourage sample classification early in the hierarchy, avoiding the harder decisions that are more error-prone. The trade-off between accuracy and fine-grained labeling is controlled by adjusting γ. The confidence score $s(\cdot)$ can be implemented in various ways [108, 36, 17]. While RTC is compatible with any of these, we adopt the popular maximum posterior probability criterion, i.e. $s(f(\mathbf{x})) = \max_i f^i(\mathbf{x})$, where $f^i(\cdot)$ is the i^{th} entry of $f(.)$. In our experience, the calibration of the node predictors $f_v(\mathbf{x})$ is more important than the particular implementation of the confidence score function.

3.4 Taxonomic probability calibration

In this section, we introduce the architecture of Deep-RTC.

Taxonomic calibration Since RTC requires decisions at all levels of the taxonomic tree, samples can be classified into any potential label set $\mathscr{Y}$ containing leaf nodes of any cut of $\mathscr{T}$. For example, the taxonomy of Figure 3.2 admits two label sets, namely, $\mathscr{Y}_{fg} = \{y_1, y_2, y_3, y_4\}$ containing all classes and $\mathscr{Y} = \{n_1, y_3, y_4\}$ obtained by pruning the children of node n_1. For long-tailed recognition, where different images can be classified at very different taxonomic levels, it is important to calibrate the posterior probability distributions of *all* these label sets. We address this problem by optimizing the ensemble of *all* classifiers implementable with the hierarchy, i.e., minimize the loss

$$\mathscr{L}_{ens} = \frac{1}{|\Omega|} \sum_{\mathscr{Y} \in \Omega} L_{\mathscr{Y}} \ , \tag{3.3}$$

where Ω is the set of all target label sets $\mathscr{Y}$ that can be derived from $\mathscr{T}$ by pruning the tree and $L_{\mathscr{Y}}$ is a loss function associated with label set $\mathscr{Y}$. While feasible for small taxonomies, this approach does not scale with taxonomy size, since the set Ω increases exponentially with $|\mathscr{T}|$. Instead, we introduce a mechanism, inspired by dropout [98], for *stochastic tree sampling (STS)* during training. At each training iteration, a random cut $\mathscr{T}_c$ of the taxonomy $\mathscr{T}$ is sampled, and the predictor $f_{\mathscr{Y}_c}(\boldsymbol{x}; \boldsymbol{W}_{\mathscr{Y}_c}, \boldsymbol{\Phi})$ associated with the corresponding label set $\mathscr{Y}_c$ is optimized. For this, random cuts are generated by sampling a Bernoulli random variable $P_v \sim Bernoulli(p)$ for each internal node v with a given dropout rate p. The subtree rooted at v is pruned if $P_v = 0$. Examples of these taxonomy cuts are shown in Figure 3.3. The predictor $f_{\mathscr{Y}_c}$ of (3.1) consistent with the target label set $\mathscr{Y}_c$ associated with the cut $\mathscr{T}_c$ is then synthesized, and the loss computed as

$$\mathscr{L}_{sts} = \frac{1}{M} \sum_{i=1}^{M} L_{\mathscr{Y}_c}(\boldsymbol{x}_i, y_i). \tag{3.4}$$

By considering different cuts at different iterations, the learning algorithm forces the hierarchical classifier to produce well calibrated decisions for all label sets.

Parameter sharing The procedure above requires *on-the-fly* synthesis of predictors $f_{\mathscr{Y}_c}$ for all possible label sets $\mathscr{Y}_c$ that can be derived from taxonomy $\mathscr{T}$. This implies a *dynamic* CNN architecture, where (3.1) changes with the sample $\boldsymbol{x}$. Deep-RTC is one such architecture, inspired by the fact that, for long-tailed recognition, the predictors $f_{\mathscr{Y}_c}$ should share parameters, so as to enable information transfer from head to tail classes [113, 80]. This is implemented with a combination of two parameter sharing mechanisms. First, the backbone feature extractor $h(\boldsymbol{x};\boldsymbol{\Phi})$ is shared across all label sets. Since this enables the implementation of Deep-RTC with a single network and no additional parameters, it is also critical for fair comparisons with the flat classifier. More complex hierarchical network architectures [38, 141, 77] would compromise these comparisons and are not investigated. Second, the predictor of (3.1) should reflect the hierarchical structure of each label set $\mathscr{Y}_c$. A popular implementation of this constraint, denoted *parameter inheritance (PI),* reuses parameters of ancestors nodes $\mathscr{A}(n)$ in the predictor of node n. The column vector $\mathbf{w}_n$ of $\mathbf{W}_{\mathscr{Y}}$ is then defined as

$$\mathbf{w}_n = \theta_n + \textstyle\sum_{p\in\mathscr{A}(n)} \theta_p \ , \quad \forall n \in \mathscr{Y} \tag{3.5}$$

where θ_n are non-hierarchical node parameters. This compositional structure has two advantages. First, it leverages the parameters of parent nodes (more training data) to regularize the parameters of their low-level descendants (less training data). Second, the parameter vector θ_n of node n only needs to model the residuals between n and its parent, in order to be discriminative of its siblings. In summary, low-level decisions are simultaneously simplified and robustified.

Dynamic predictor synthesis Deep-RTC is a novel architecture to enable the *dynamic* synthesis of predictors $f_{\mathscr{Y}_c}$ that comply with (3.5). This is achieved by introducing a codeword vector $\mathbf{q}_n \in \{0,1\}^{|\mathscr{N}|}$ per node n, containing binary flags that identify the ancestors $\mathscr{A}(n)$ of n

$$\boldsymbol{q}_n(v) = 1_{[v\in\mathscr{A}(n)\cup\{n\}]}. \tag{3.6}$$

For example, in the taxonomy of Figure 3.2, $\mathbf{q}_{n_1} = (1,0,0,0,0)$ since $\mathscr{A}(n_1) = \varnothing$, and $\mathbf{q}_{n_4} = (1,0,0,1,0)$ since $\mathscr{A}(n_4) = \{n_1\}$. Codeword $\mathbf{q}_n$ encodes which nodes of $\mathscr{T}$ contribute to the prediction of node n under the PI strategy, thus providing a recipe for composing predictors for any label set $\mathscr{Y}$. A matrix of node-specific parameters $\mathbf{\Theta} = [\theta_1,...,\theta_{|\mathscr{N}|}]$ where $\theta_n \in \mathbb{R}^k$ for all $n \in \mathscr{N}$ is then introduced, and $\mathbf{w}_n$ can be reformulated as

$$\mathbf{w}_n = \mathbf{\Theta}\mathbf{q}_n. \tag{3.7}$$

The codeword vectors of all nodes $n \in \mathscr{Y}$ are then written into the columns of a codeword matrix $\mathbf{Q}_{\mathscr{Y}} \in \{0,1\}^{|\mathscr{N}|\times|\mathscr{Y}|}$, to define a predictor as in (3.1),

$$f_{\mathscr{Y}}(\boldsymbol{x};\mathbf{\Theta},\mathbf{\Phi}) = \sigma(z_{\mathscr{Y}}(\boldsymbol{x};\mathbf{\Theta},\mathbf{\Phi})) \qquad z_{\mathscr{Y}}(\boldsymbol{x};\mathbf{\Theta},\mathbf{\Phi}) = \boldsymbol{W}_{\mathscr{Y}}^T h(\mathbf{x};\blacksquare), \tag{3.8}$$

where $\boldsymbol{W}_{\mathscr{Y}} = \mathbf{\Theta}\mathbf{Q}_{\mathscr{Y}}$. This enables the classification of sample $\boldsymbol{x}$ with respect to *any* label set $\mathscr{Y}_c$ by simply making $\boldsymbol{Q}_{\mathscr{Y}}$ a dynamic matrix $\boldsymbol{Q}_{\mathscr{Y}}(\boldsymbol{x}) = \boldsymbol{Q}_{\mathscr{Y}_c}$, as illustrated in Figure 3.3.

Loss function Deep-RTC is trained with a cross-entropy loss

$$L_{\mathscr{Y}}(\boldsymbol{x}_i, y_i) = -\boldsymbol{y}_i^T \log f_{\mathscr{Y}}(\boldsymbol{x};\mathbf{\Theta},\mathbf{\Phi}) \ , \tag{3.9}$$

where $\boldsymbol{y}_i$ is the one-hot encoding of $y_i \in \mathscr{Y}$. When this is used in (3.4), the CNN is globally optimized with respect to the label set $\mathscr{Y}_c$ associated with taxonomic cut $\mathscr{T}_c$. The regularization of the many classifiers associated with different cuts of $\mathscr{T}$ is a *global* regularization, guaranteeing that all classifiers are well calibrated. Beyond this, it is also possible to calibrate the internal node-conditional decisions. Given that a sample $\boldsymbol{x}$ has been assigned to node n, the node-conditional decisions are *local* and determine which of the children $\mathscr{C}(n)$ the sample should be assigned to. They consider only the target label set $\mathscr{Y}_n = \mathscr{C}(n)$ defined by the children of n. For these label sets, all nodes $v \in \mathscr{C}(n)$ share the same ancestor set $\mathscr{A}_v$ and thus the second term of (3.5). Hence, after softmax normalization, (3.5) is equivalent to $\mathbf{w}_v = \theta_v$ and the node-conditional classifier

$f_n(\cdot)$ reduces to

$$f_n(\boldsymbol{x};\boldsymbol{\Theta},\boldsymbol{\Phi}) = \sigma(\boldsymbol{Q}_n^T \boldsymbol{\Theta}^T h(\boldsymbol{x};\boldsymbol{\Phi})), \tag{3.10}$$

where, as illustrated in Figure 3.2, the codeword matrix $\boldsymbol{Q}_n$ contains zeros for all ancestor nodes. Internal node decisions can thus be calibrated by noting that sample $\mathbf{x}_i$ provides supervision for all node-conditional classifiers in its ground-truth ancestor path $\mathscr{A}(y_i)$. This allows the definition of a node-conditional consistency loss per node n of the form

$$\mathscr{L}_n = \frac{1}{M}\sum_{i=1}^{M} \frac{1}{|\mathscr{A}(y_i)|} \sum_{n \in \mathscr{A}(y_i)} L_{\mathscr{Y}_n}(\boldsymbol{x}_i, y_{n,i}) \tag{3.11}$$

where $L_{\mathscr{Y}_n}$ is the loss of (3.9) for the label set $\mathscr{Y}_n$ and $y_{n,i}$ the label of $\mathbf{x}_i$ for the decision at node n. Deep-RTC is trained by minimizing a combination of these local node-conditional consistency losses and the global ensemble loss of (3.4)

$$\mathscr{L}_{cls} = \mathscr{L}_n + \lambda \mathscr{L}_{sts}, \tag{3.12}$$

where λ weights the contribution of the two terms.

Performance Evaluation Due to the universal adoption of the flat classifier, previous long-tailed recognition works equate performance to recognition accuracy. Under the taxonomic setting, this is identical to measuring leaf node accuracy $\mathbb{E}\{1_{[\widehat{y}_i = y_i]}\}$ and fails to reward trade-offs between classification granularity and accuracy. In the example of Figure 3.1, it only rewards the "Mexican Hairless Dog" label, making no distinction between the labels "Dog" or "Tarantula," which are both considered errors. A taxonomic alternative is to rely on hierarchical accuracy $\mathbb{E}\{1_{[\widehat{y}_i \in \mathscr{A}(y_i)]}\}$ [23]. This has the limitation of rewarding "Dog" and "Mexican Hairless Dog" equally, i.e. does not encourage finer-grained decisions. In this work, we propose that a better performance measure should capture the amount of class label information captured by the classification. While a correct classification at the leaves captures all the information, a rejection

at an intermediate node can only capture partial information. To measure this, we propose to use the number of *correctly predicted bits* (CPB) by the classifier, under the assumption that each class at the leaves of the taxonomy contributes one bit of information. This is defined as

$$\text{CPB} = \frac{1}{M}\sum_{i=1}^{M} 1_{[\widehat{y}_i \in \mathscr{A}(y_i)]} \left(1 - \frac{|\text{Leaves}(\mathscr{T}_{\widehat{y}_i})|}{|\text{Leaves}(\mathscr{T})|}\right) \tag{3.13}$$

where $\mathscr{T}_{\widehat{y}_i}$ is the sub-tree rooted at $\widehat{y}_i$. This assigns a score of 1 to correct classification at the leaves, and smaller scores to correct classification at higher tree levels. Note that any correct prediction of intermediate level is preferred to an incorrect prediction at the leaves, but scores less than a correct prediction of finer-grain. Finally, for flat classifiers, CPB is equal to classification accuracy.

3.5 Experiments

This section presents the long-tailed recognition performance of Deep-RTC.

3.5.1 Experimental Setup

Datasets. We consider 4 datasets. **CIFAR100-LT**[20] is a long-tailed version of [66] with "imbalance factor" 0.01 (i.e. most populated class 100× larger than rarest class). **AWA2-LT** is a long-tailed version, curated by ourselves, of [65]. It contains 30475 images from 50 animal classes and hierarchical relations extracted from WordNet [85], leading to a 7-level imbalanced tree. The training set has an imbalance factor of 0.01, the testing set is balanced. **ImageNet-LT**[80] is a long-tailed version of [22], with 1000 classes of more than 5 and less than 1280 images per class, and a balanced test set. **iNaturalist (2018)** [49, 1] is a large-scale dataset of 8142 classes with the class imbalance factor of 0.001, and a balanced test set. While the full iNaturalist dataset is used for comparisons to previous work, a more manageable subset, iNaturalist-sub, containing 55929 images for training and 8142 for testing, is used for ablation studies. Please refer to supplementary material for more details.

Data partitions for long-tail evaluation The evaluation protocol of [80] is adopted by splitting the classes into many-shot, medium-shot, and few-shot. The splitting rule of [80] is used on iNaturalist. On CIFAR100-LT and AWA2-LT, the top and bottom $1/3$ populated classes belong to many-shot and few-shot respectively, and the remaining to medium-shot.

Backbone architectures CIFAR100-LT and iNaturalist use the setup of [20], where ResNet32 [44] and ImageNet pre-trained ResNet50 are used respectively. For ImageNet-LT, ResNet10 is chosen as in [80]. For AWA2-LT, we use ResNet18.

Competence level Unless otherwise noted, the value of γ is cross-validated, i.e. the value of best performance on the validation set is applied to the test set.

3.5.2 Ablations

We started by evaluating how the different components of Deep-RTC - parameter inheritance (PI) regularization of (3.5), node-consistency loss (NCL) of (3.11), and stochastic tree sampling (STS) of (3.9) - affect the performance. Two baselines were used in this experiment. The first is a flat classifier, implemented with the standard softmax architecture, and trained to optimize classification accuracy. This is a representative baseline for the architectures used in the long-tailed recognition literature. The second is a hierarchical classifier derived from this flat classifier, by recursively adding class probabilities as dictated by the class taxonomy. This is denoted as the *recursive hierarchical classifier* (RHC). We refer to this computation of probabilities as *bottom-up* (BU) inference. This is opposite to the *top-down* (TD) inference used by most hierarchical approaches, where probabilities are sequentially computed from the root (top) to leaves (bottom) of the tree. The performance of the different classifiers was measured in multiple ways. CPB is the metric of (3.13). Leaf acc. is the classification accuracy at the leaves of the taxonomy. For a flat classifier, this is the standard performance measure. For a hierarchical classifier, it is the accuracy when intermediate rejections are not allowed. Hier acc. is the accuracy of a classifier that supports rejections, measured at the point where the decision is taken. In the example of Figure 3.1 a decision of "dog" is considered accurate under this metric.

Table 3.1. Ablations on iNaturalist-sub.

Method	leaf acc.	depth	hier. acc.	CPB	inference
Flat classifier	.163	1	.163	.163	-
RHC	.163	.58	.754	.537	BU
PI+STS	.174	.46	.913	.601	TD
PI+NCL	.185	.48	.904	.563	TD
PI+STS+NCL (Deep-RTC)	.181	.50	.899	.619	TD

Table 3.2. Comparisons to hierarchical classifiers.

Method	CIFAR100-LT	AWA2-LT	ImageNet-LT	inference
CNN-RNN [40]	.379	.882	.514	TD
B-CNN [141]	.366	.805	.511	TD/BU
HND [68]	.374	-	-	TD
NofE [2]	.373	.770	.463	BU
Deep-RTC	**.397**	**.894**	**.529**	TD

Finally, depth is the average depth at which images are rejected, normalized by tree depth (e.g. 1 when no intermediate rejections are allowed).

CPB Performance: Table 3.1 shows that the flat classifier has very poor CPB performance because prediction at leaves requires the classifier to make decisions on tail classes where it is poorly trained. The result is a very large number of errors, i.e. images for which no label information is preserved. RHC, its bottom-up hierarchical extension, is a much better solution to long-tailed recognition. While most images are not classified at the leaves, both hierarchical accuracy and CPB increases dramatically. Nevertheless, RHC has weaker CPB performance than the combination of the PI architecture of Figure 3.2 with either STS or NCL. Among these, the global regularization of STS is more effective than the local regularization of L_n. However, by combining two regularizations, they lead to the classifier (Deep-RTC) that preserves most information about the class label.

Performance measures: The long-tailed recognition literature has focused on maximizing the accuracy of flat classifiers. Table 3.1 shows some limitations of this approach. First, all classifiers have very poor performance under this metric, with leaf acc. between 16% and 18%. Furthermore, as shown in Figure 3.4, the labels can be totally uninformative of the object class.

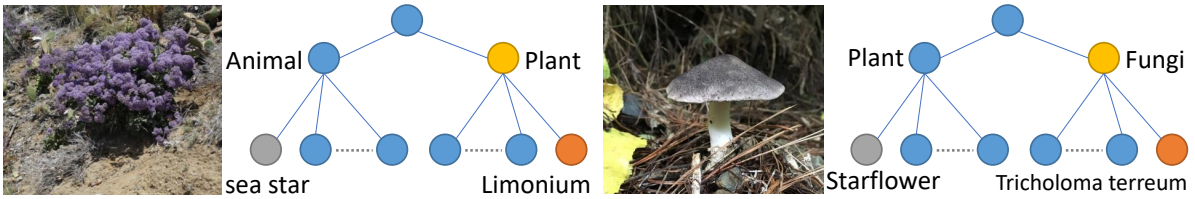


Figure 3.4. Prediction of Deep-RTC (yellow) and flat classifer (gray) on two iNaturalist-sub images (orange: ground truth).

In the example, the flat classifier assigns the label of "sea star" ("star flower") to the image of the plant (mushroom) shown on the top (at the bottom). We are aware of *no* application that would find such labels useful. Second, all classifiers perform dramatically better in terms of hier. acc. For the practitioner, this means that they are accurate classifiers. Not expert enough to always carry the decision to the bottom of the tree, but reliable in their decisions. In the same example, Deep-RTC instead correctly assigns the images to the broader classes of "Plant" (left) and "Fungi" (right). Furthermore, Deep-RTC classifies 90% of the images correctly at this level! This could make it useful for many applications. For example, it could be used to automatically route the images to experts in these particular classes, for further labeling. Third, among TD classifiers, Deep-RTC pushes decisions furthest down the tree (e.g. 4% deeper than PI+STS). This makes it a better *expert* on iNaturalist-sub than its two variants, a fact captured by the proposed CPB measure. Given all this, we believe that CPB optimality is much more meaningful than leaf acc. as a performance measure for long-tailed recognition.

3.5.3 Comparisons to hierarchical classifiers

We next performed a comparison to prior works in hierarchical classification with CPB in Table 3.2. These experiments show that prior methods have similar performance, without discernible advantage for TD or BU inference; however, they all underperform Deep-RTC. This is particularly interesting because these methods use networks more complex than Deep-RTC, adding branches (and parameters) to the backbone in order to regularize features according to the taxonomy. Deep-RTC simply implements a dynamic softmax classifier with the label encoding

Table 3.3. Results on iNaturalist. Classes are discussed with popularity classes (many, medium and few-shot).

Method	metric	Many	Medium	Few	All
Softmax	CPB	0.76	0.67	0.62	0.66
CBLoss [20]		0.61	0.62	0.61	0.61
LDAM-SGD [8]		-	-	-	0.65
LDAM-DRW [8]		-	-	-	0.68
NCM [61]		0.61	0.64	0.63	0.63
cRT [61]		0.73	0.69	0.66	0.68
τ-norm [61]		0.71	0.69	0.69	0.69
Deep-RTC	CPB	**0.84**	**0.79**	**0.75**	**0.78**
	hier. acc.	0.92	0.91	0.89	0.90
	leaf freq.	0.71	0.56	0.48	0.54
	leaf acc.	0.76	0.67	0.60	0.64

Table 3.4. Results on ImageNet-LT.

Method	CPB
FSLwF [37]	0.28
Focal Loss [75]	0.31
Range Loss [133]	0.31
Lifted Loss [97]	0.31
OLTR [80]	0.36
Softmax	0.35
NCM [61]	0.36
cRT [61]	0.42
τ-norm [61]	0.41
Deep-RTC	**0.53**

Table 3.5. Comparisons to learning with rejection under different rejection rates (CPB).

		CIFAR100-LT				AWA2-LT			
Rej. Rate	Method	Many	Medium	Few	All	Many	Medium	Few	All
5%	RP [108]	**.779**	**.722**	.306	.404	**.977**	.963	.887	.914
	Deep-RTC	.773	.719	**.335**	**.416**	.975	**.978**	**.907**	**.931**
10%	RP [108]	**.793**	**.734**	.315	.416	**.980**	.966	.900	.924
	Deep-RTC	.789	.7314	**.344**	**.439**	.975	**.984**	**.929**	**.947**
20%	RP [108]	.816	.751	.328	.433	**.985**	.970	.916	.939
	Deep-RTC	**.833**	**.770**	**.393**	**.491**	.969	**.975**	**.943**	**.954**

of Figure 3.2. Instead, it leverages its dynamic ability and stochastic sampling to simultaneously optimize decisions for many tree cuts. The results suggest that this optimization over label sets

is more important than shaping the network architecture according to the taxonomy. This is sensible since, under the Deep-RTC strategy, feature regularization is learned end-to-end, instead of hard-coded. Details of the compared methods are in the supplementary material.

3.5.4 Comparisons to long-tail recognizers

A comparison to the state of the art methods from the long-tailed recognition is presented in Tables 3.3-3.4 for iNaturalist and ImageNet-LT respectively. More comparisons for other datasets are provided in the supplementary material. In all cases, Deep-RTC predicts *more bits* correctly (i.e. higher CPB), which beats the state of the art flat classifier by 9% on iNaturalist and 11% on ImageNet-LT. For iNaturalist, we also discuss other metrics by class popularity, where leaf freq. represents the frequency that samples are classified to leaves. A comparison to the standard softmax classifier shows that prior long-tailed methods improve performance CPB on few-shot classes but *degrade* for popular classes. Deep-RTC is the only method to consistently improve CPB performance for all levels of class popularity. It is also noted that, unlike the state of the art flat classifier, Deep-RTC does not have to sacrifice leaf acc. for the many-shot classes in order to accommodate few-shot classes where its performance will not be great anyway. Instead, it exits early for about half of the images of the few-shot classes and guarantees highly accurate answers for all classes (around 90% hier. acc.). This is similar to how humans treat the long-tail recognition problem.

3.5.5 Comparisons to learning with rejection

While the classifiers of the previous sections were allowed to reject examples at intermediate nodes, whenever feasible, they were not explicitly optimized for such rejection. Table 3.5 shows a comparison to a state-of-the-art flat realistic predictor (RP) [108], on CIFAR100-LT and AWA2-LT. In these comparisons, the percentage of rejected examples (rejection rate) is kept the same. The rejection rate of Deep-RTC is the percent of examples rejected at the root node. Deep-RTC achieves the best performance for all rejection rates on both datasets, because

it has the option of soft-rejecting, i.e. letting examples propagate until some intermediate tree node. This is not possible for the flat RP, which always faces an all or nothing decision. In terms of class popularity, Deep-RTC always has higher CPB for few-shot classes, and frequently considerable gains. For many and medium-shot classes, the two methods have the comparable performance on CIFAR100-LT. On AWA2-LT, RP has an advantage for many and Deep-RTC for medium-shot classes. This shows that the gains of Deep-RTC are mostly due to its ability to push images of low-shot classes as far down the tree as possible without forcing decisions for which the classifier is poorly trained.

3.6 Conclusion

In this work, a *realistic taxonomic classifier* (RTC) is proposed to address the long-tail recognition problem. Instead of seeking the finest-grained classification for each sample, we propose to classify each sample up to the level that the classifier is competent. Deep-RTC architecture is then introduced for implementing RTC with deep CNN and is able to 1) share knowledge between head and tail classes 2) align data hierarchy with model design in order to predict at all levels in the taxonomy, and 3) guarantee high prediction performance by opting to provide coarser predictions when samples are too hard. Extensive experiments validate the effectiveness of the proposed method on 4 long-tailed datasets using the proposed tree metric. This indicates that RTC is well suited for solving long-tail problem. We believe this opens up a new direction for long-tailed literature.

Chapter 3 is, in full, based on the material as it appears in the publication of "Solving Long-tailed Recognition with Deep Realistic Taxonomic Classifier", Tz-Ying Wu, Pedro Morgado, Pei Wang, Chih-Hui Ho, and Nuno Vasconcelos, In Proceedings of European Conference on Computer Vision (ECCV), 2020. The dissertation author was the primary investigator and author of this paper.

Chapter 4

ProTeCt: Prompt Tuning for Taxonomic Open Set Classification

4.1 Introduction

Vision-language foundation models (FMs) have opened up new possibilities for image classification. They are large models, trained on large corpora, to learn aligned representations of images and text. For example, CLIP [89] combines text and image encoders trained with 400M image-text pairs in an open vocabulary fashion, using a contrastive loss [11, 13, 96, 104]. Zero-shot classification can then proceed by leveraging the alignment of image and text features. Each class name is first converted to a text prompt, e.g., "a photo of [CLASS]," which is fed to the text encoder. The resulting text feature is then used as the parameter vector of a softmax classifier of image feature vectors. Since the training does not emphasize any particular classes, CLIP supports open set classification. Several works [140, 139, 62, 127] have shown that classification accuracy can be enhanced by fine-tuning the FM on the few-shot setting (i.e. few examples per class). To adapt the model and maintain the alignment between text and images, these works augment the FM with a few learnable prompts [140, 139, 127, 62]. The model parameters are then frozen and only the prompts are optimized. This process is known as **prompt tuning** and can achieve significant improvement over the zero-shot performance, on the dataset of interest.

While prompting enables classifiers to be designed for virtually any classes with minimal dataset curation effort, it should not compromise the open set nature and generality of the FM representation. In this work, we consider the setting where "open set" means the ability to refer to concepts at different levels of granularity. Consider, for example, an educational application in biology. While at grade school level it will teach students to classify animals into ("cat","dog","lizard"), at the high-school level the **exact same images** should be classified into much more detailed classes, e.g. ("iguana","anole", "komodo", etc.) for lizards. A classifier that classifies an image as a "komodo" lizard for high schoolers but "dog" for gradeschoolers is not useful and trustworthy. Advanced biology students should even learn about the taxonomic relations between the different species. This requires a representation that supports hierarchical classification [123, 117, 68, 84], where the classifier understands the relations between the

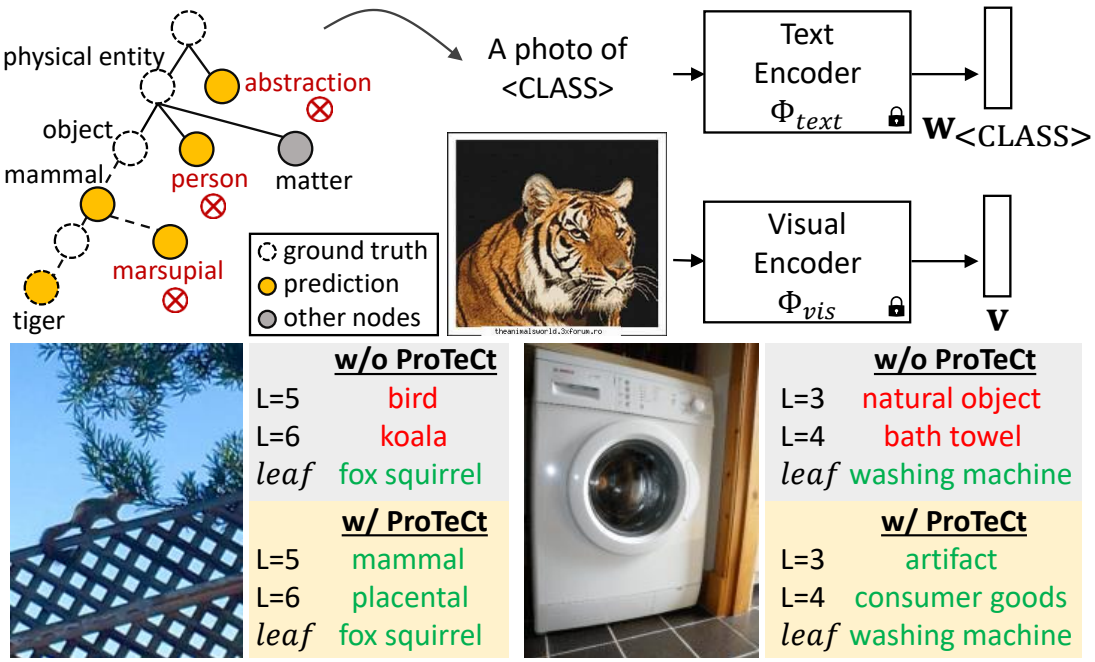


Figure 4.1. (Top) An example of class hierarchy, where CLIP predicts the tiger image as "person" at the internal hierarchy level. (Bottom) Correct/incorrect model predictions (green/red) of CoOp w/ and w/o ProTeCt, respectively. L denotes the tree level.

superclasses and subclasses that compose a class hierarchy, and provide correct predictions *across* hierarchy levels.

Fig. 4.1 shows an example of a class hierarchy built from ImageNet [22] classes, according to the WordNet [85]. When faced with the image of a tiger, the classifier should provide a correct prediction under the label sets $\mathscr{Y}_1$ =("dog", "cat", "**tiger**"), $\mathscr{Y}_2$ = ("person", "**animal**", " insect") or $\mathscr{Y}_3$ =("**physical entity**", "abstraction"), where the correct prediction is shown in bold. Note that, given a classifier with this property, teachers have the ability to define many different classification problems, for many levels of granularity, tailoring the same app to different uses. We refer to this setting as **taxonomic open set** (TOS) classification. In many real-world applications, support for this restricted form of open set classification is much more important than support for unbounded open set classification. In the example above, biology teachers do not really care if the classifier can still discriminate between cars and trucks, or soda cans and wine bottles. Hence, these classes are irrelevant for the app developer.

Table 4.1. TOS classification performance of CLIP-based classifiers.

Method	Acc_{leaf}	HCA	MTA
CLIP [89]	68.36	3.32	48.21
CoOp [140]	71.23	2.99	46.98
MaPLe [62]	70.70	4.15	48.29

In principle, TOS should be trivially supported by FMs. Even at zero-shot level, it should suffice to specify [CLASS] names at the desired levels of granularity. However, our experiments show that this does not work because the representation of most FMs fails to capture taxonmic relations. This is illustrated for CLIP in Fig. 2.1. While the model knows that the object is a tiger, it fails to know that it is "a physical entity" and not an "abstraction" or that it is a "placental mammal" and not a "marsupial," indicating that it only understands class relations locally. It can perform well for the leaf class label set $\mathscr{Y}_1$, but cannot reason across abstraction levels, and can thus not support TOS classification. To enable TOS, we introduce the notion of **hierarchical consistency**, and a new *hierarchical consistency accuracy* (HCA) metric, where classification is defined with respect to a taxonomic tree and its success requires the correct prediction of all superclasses (e.g., mammal, object and physical entity) of each ground truth leave class (e.g., tiger). This is complemented by the notion of **TOS classification**, where classifiers can have any set of nodes in the class hierarchy as the label set, and a new *mean treecut accuracy* (MTA) metric, which estimates classification accuracy in this setting.

Our experiments show that neither CLIP nor existing prompt tuning methods [140, 139, 62] perform well under the HCA and MTA metrics of the TOS setting. Fig. 2.1 illustrates the problem and the *inconsistent* CLIP class predictions (orange dots) across hierarchy levels. Table 4.1 compares the standard (leaf) accuracy of the model with HCA/MTA, under both the zero-shot and two prompt-tuning settings. While the leaf accuracy is quite reasonable, hierarchical consistency is very poor. To address this problem, we propose a novel prompt-tuning procedure, denoted *Prompt Tuning for Hierarchical Consistency* (ProTeCt), that explicitly targets the TOS setting. Given a dataset of interest, a class hierarchy is extracted from the associated metadata, a generic public taxonomy (e.g. WordNet [85]), or a special purpose taxonomy

related to the application (e.g. scientific taxonomies). Since FMs support classification with open vocabulary, any node in the hierarchy can be used in the label set of the classifier. Prompts are then learned with the help of two new regularization losses that encourage hierarchical consistency. A *dynamic treecut loss* (DTL) encourages correct classification at all tree levels by sampling random tree cuts during training. A *node-centric loss* (NCL) contributes additional supervision to each internal tree node to increase classification robustness for all granularities of the hierarchy.

Experiments show that ProTeCt significantly improves the performance of prompt tuning methods, like CoOp [140] and MaPLe [62], under TOS setting. Fig. 2.1 shows the predictions of CoOp at different hierarchy levels before/after adding ProTeCt. Under the HCA/MTA metrics, the improvement can be more than 15/25 points on Cifar100, SUN and ImageNet datasets. Following [140, 139, 62], we show that these gains hold for zero-shot domain generalization to several variants of ImageNet [90, 106, 46, 45], showing that hierarchical consistency transfers across datasets. Furthermore, ablations show that ProTeCt can be used with different CLIP architectures, parameter tuning methods and taxonomies.

Overall, this work makes four contributions. First, we introduce the TOS setting, including two novel metrics (HCA and MTA) that evaluate the consistency of hierarchical classification. Second, we show that neither zero-shot CLIP nor existing prompting methods fare well in this setting. Third, we propose a novel prompt-tuning method for the TOS setting, ProTeCt, which improves hierarchical consistency by combining DTL and NCL losses. The former relies on a dynamic stochastic sampling of label sets involving multiple levels of the hierarchy, while the latter regularizes the classification of every node in the hierarchy. Finally, ProTeCt is shown to outperform vanilla prompt tuning methods on three datasets with different hierarchies. Extensive ablations demonstrate that ProTeCt is applicable to different parameter tuning methods, CLIP architectures, taxonomies and the learned hierarchical consistency transfers to unseen datasets from different image domains.

4.2 Related Work

Prompt Tuning of Vision-Language Models: Many large vision-language FMs have been proposed in recent years [132, 29, 110]. Despite their promising zero-shot performance, several works [139, 140, 54, 62] have shown that their few-shot finetuning with a dataset from the target application can further improve performance. Unlike conventional finetuning methods that optimize the entire model, these methods are designed to (a) be parameter efficient and (b) maintain the general purpose feature representation of the FM. Several such tuning methods have been proposed for CLIP [89]. Inspired by prompt tuning techniques from the language literature [69, 71, 76], CoOp [140] inserts learnable prompts at the CLIP text input. CoCoOp [139] further learns a meta-network to generate an image-conditioned prompt. The idea of connecting image and text prompts is further extended by UPT [127] and MaPLe [62]. The former learns a unified transformer for generating an image and text prompt, the latter learns a coupling function to generate image prompts from text prompts. LASP [7] proposed a text-to-text cross-entropy loss to regularize the distribution shift when different prompts are used. Unlike these works, we investigate the TOS problem, where labels can be drawn from any level in a class taxonomy, and propose prompting techniques to improve hierarchical classification consistency. This is shown to be compatible with several of the above prompt-tuning methods without degrading their leaf classification accuracy.

Hierachical Classifiers: Hierarchical classification aims to predict labels at different levels of a class hierarchy. Early works [84, 135, 21, 92, 93, 23] date back to the era before deep learning and are not directly applicable to deep learning-based models. Several works [123, 38, 141, 77, 2, 63] propose hierarchical classifiers for CNN-based deep models. For example, [38, 141, 77] use additional convolutional modules to learn a hierarchical feature space. It is unclear how these approaches generalize to the recent transformer-based architectures [26, 79, 78]. Furthermore, prior works [123, 38, 141, 77, 2, 117] finetune the entire model, which requires substantial data and computation, especially at the FM scale. In this work, we study the problem of hierarchical

consistency for foundational vision-language models (e.g., CLIP). While CLIP-based classifiers [89, 139, 140] have outstanding zero/few-shot performance, we show that they produce inconsistent predictions for label sets of different granularity and cannot be used in the TOS setting. We propose an efficient prompt tuning method to address this.

4.3 Preliminaries

Foundation Models (FMs): Visual-language FMs are composed by a text Φ_{text} and a visual Φ_{vis} encoder, which extract features from text and images, respectively. The two encoders are optimized by contrastive training [104, 96, 13, 11] to create a joint representation for the two modalities. Since the encoders are learned from a large-scale corpus of image-text pairs, the features are general and support various downstream tasks, e.g., image classification [140, 139, 62, 127] and segmentation [114, 81]. While in this work we use the CLIP [89], ProTeCt should generalize to other FMs.

Image Classification with FMs: Given a label set $\mathscr{Y} = \{t_y\}_{y=1}^{C}$, a zero-shot classifier can be designed in the FM representation space by introducing a weight vector $\mathbf{w}_y$ per class y. These weight vectors are obtained by simply using the class name t_y (e.g., "dog") as a text encoder prompt, i.e., $\mathbf{w}_y = \Phi_{text}(Emb_t(t_y)) \in \mathbb{R}^k$, where $Emb_t(\cdot)$ is a word embedding. Given these weight vectors, an image classifier of label set $\mathscr{Y}$ can be implemented by computing class posterior probabilities with

$$p(t_y|\mathbf{x};\mathscr{Y}) = \frac{\exp(\cos(\mathbf{w}_y,\mathbf{v})/\tau)}{\sum_{t_j\in\mathscr{Y}}\exp(\cos(\mathbf{w}_j,\mathbf{v})/\tau)}\ , \tag{4.1}$$

where $p(t_y|\mathbf{x};\mathscr{Y})$ is the probability of class label t_y given image $\mathbf{x}$, $\mathbf{v} = \Phi_{vis}(Emb_v(\mathbf{x})) \in \mathbb{R}^k$ the visual feature vector, $Emb_v(\cdot)$ an image embedding, $cos(\cdot,\cdot)$ the cosine similarity metric, and τ a temperature hyperparameter. Classification performance can usually be improved by inferring the classifier parameters $\mathbf{w}_y$ from multiple text prompts, e.g. by including context words such as a prompt prefix $p =$"a photo of", or $p =$"a drawing of", computing $\mathbf{w}_y = \Phi_{text}(Emb_t(\{p,t_y\}))$, and

ensembling the vectors $\mathbf{w}_y$ obtained from multiple prompts [89, 140]. This, however, requires multiple forward passes through Φ_{text} during inference and can be undesirable for downstream applications.

More efficient inference can be achieved with prompt tuning [140, 139, 62, 127], which leverages a set of learnable parameters $\{\mathbf{c}_m^t\}_{m=1}^M$ as context features. These are prepended to each class name embedding $Emb_t(t_y)$ as text prompts, to produce the weight vectors $\mathbf{w}_y = \Phi_{text}(\{\mathbf{c}_1^t,...\mathbf{c}_M^t,Emb_t(t_y)\})$. Note that each $\mathbf{c}_i^t$ has the same dimension as the word embedding. Given a training dataset $\mathscr{D}=\{(\mathbf{x}_i,y_i)\}_{i=1}^N$, context features can be end-to-end optimized with the cross-entropy loss

$$L_{\mathscr{Y}}(\mathbf{C}^t)=\frac{1}{N}\sum_{i=1}^{N}\sum_{t_j\in\mathscr{Y}}-\mathbb{1}(t_j=t_{y_i})\log p(t_j|\mathbf{x}_i;\mathscr{Y},\mathbf{C}^t) \tag{4.2}$$

for the classifier of (4.1), where $\mathbb{1}(\cdot)$ is the indicator function, and $\mathbf{C}^t$ the matrix of context features. Similarly, learnable prompts $\mathbf{c}_i^v$ can be inserted into the image branch, i.e. $\mathbf{v} = \Phi_{vis}(\{\mathbf{c}_1^v,\dots,\mathbf{c}_M^v,Emb_v(\mathbf{x})\})$, for better visual adaptation [54, 127, 62]. To prevent compromising the generalization of the FM embeddings, the parameters of the two encoders (i.e., Φ_{text},Φ_{vis}) are frozen in the few-shot setting. In this paper, we consider two prompt tuning variants, CoOp [140] and MaPLe [62], the former using learnable prompts in the text branch, and the latter on both branches.

Class Taxonomy: A class taxonomy $\mathscr{Y}^{tax}$ organizes classes into a tree where classes of similar semantics are recursively assembled into superclasses, at each graph node (e.g. "dog" is a superclass of "Chihuahua" and "Corgi"). For a tree hierarchy, $\mathscr{T}$, each node $n\in\mathscr{N}$ has a single parent and multiple child nodes $Chd(n)$, where $\mathscr{N}$ is the set of tree nodes. Given a set of classes $\{t_y\}_{y=1}^C$, a tree hierarchy $\mathscr{T}$ can be built by treating $\{t_y\}_{y=1}^C$ as leaf nodes (where $Chd(t_y)=\emptyset$), i.e., $Leaf(\mathscr{T})=\{t_y\}_{y=1}^C$, and recursively grouping classes in a bottom-up manner until a single root node is created, according to the similarity relationships defined by the taxonomy $\mathscr{Y}^{tax}$. For example, ImageNet [22] classes are organized into a tree of 1,000 leaf nodes derived from

the WordNet [85] taxonomy. Nodes that are not at the leaves are denoted as internal nodes $\mathscr{N}^{int} = \mathscr{N} \setminus Leaf(\mathscr{T})$.

4.4 Taxonomic Open Set Classification

Definition: A significant advantage of FMs for practical applications is their support for open set classification. Since the classifier of (4.1) can be implemented with any class names t_y, and the FM is trained with an open vocabulary, it is possible to perform classification for virtually any classes. Prompting methods improve the classification of the classes defined by the label set $\mathscr{Y}$, but attempt to maintain this generality. However, for most applications "open set" does not mean the ability to recognize "any possible word." On the contrary, the whole point of prompt tuning is to enhance the FM performance for a given application *context*. This context defines what "open set" truly means for the application. In practice, it frequently means "all the possible ways" to refer the classes in $\mathscr{Y}$.

One important component of this requirement is the ability to describe classes at different levels of granularity. For example, while user A (a car mechanic) may need to know if an image depicts a "Fan Clutch Wrench" or a "Box-Ended Wrench," user B (a retail store worker) may need to know if the exact same image depicts a "a mechanic's tool" or a "plumber's tool." A FM-based classification app should be deployable in both the car garage or the retail store. However, because the app is a tool classification app, the prompted model does not need to be good at recognizing "lollipops," which are beyond the context of the app. On the other hand, it is undesirable to have to prompt-tune the app for every specific use or user group. Ideally, it should be possible to prompt tune the FM *once,* with respect to the *entire* class taxonomy $\mathscr{Y}^{tax}$ of tools. The app can then be deployed to each user base without any retraining, by simply drawing the most suitable class names t_y from $\mathscr{Y}^{tax}$. We refer to this problem as **Taxonomic Open Set** (TOS) classification and introduce a formal definition in the remainder of this section.

Datasets: Most existing classification dataset can be used to study the TOS problem, since the very nature of taxonomies is to group objects or concepts into semantic classes of different levels

of granularity. Hence, most vision datasets are already labeled taxonomically or adopt classes defined by a public taxonomy, usually WordNet [85]. We consider three popular datasets: Cifar100 [66], SUN [119] and ImageNet [22]. ImageNet is complemented by the ImageNetv2 [90], ImageNet-S [106], ImageNet-A [46] and ImageNet-R [45] to enable the study of generalization across image domains. For each dataset, the K-shot setting is considered, where K images per class are sampled for training. We consider $K = \{1,2,4,8,16\}$.

Label sets: Given a dataset $\mathscr{D}$ and class hierarchy $\mathscr{Y}^{tax}$ a label set $\mathscr{Y}$ is defined at each level of granularity, according the latter. The leaf label set $\mathscr{Y}_{leaf}$ is defined as the set of classes of $\mathscr{D}$ and the class hierarchy $\mathscr{T}$ is build recursively, denoting by $\mathscr{Y}_n = Chd(n)$ the set of class labels for the children of node n. In our experiments, we adopt the default hierarchy of the SUN dataset and use WordNet [85] to build the hierarchy for Cifar100 and ImageNet. The resulting class hierarchies are as follows. Cifar100 [66] contains 100 leaf nodes and 48 internal nodes. SUN contains 324 leaf nodes and 19 internal nodes (after pruning 73 leaf classes that have confusing superclasses). ImageNet [22], ImageNetv2 [90] and ImageNet-S [106] share a class hierarchy of 1,000 leaf nodes and 368 internal nodes. ImageNet-A [46] and ImageNet-R [45] only contain 200 subclasses and the corresponding internal nodes from the ImageNet hierarchy.

Metrics: Given a classifier

$$\widehat{y}(\mathbf{x};\mathscr{Y}) = \operatorname*{argmax}_{t_y \in \mathscr{Y}} p(t_y|\mathbf{x};\mathscr{Y}) \tag{4.3}$$

using a label set $\mathscr{Y}$, several metrics are proposed to evaluate TOS performance.

Leaf Accuracy is defined as

$$Acc_{leaf} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[\widehat{y}(\mathbf{x}_i;\mathscr{Y}_{leaf}) = t_{y_i}] \tag{4.4}$$

and measures the classification accuracy at the leaves of the taxonomic tree. These are usually defined as the "dataset classes". This metric enables comparison of hierarchical classifiers to

standard, or *flat*, classifiers which only consider the leaf classes.

Hierarchical Consistent Accuracy (HCA) is defined as

$$HCA = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{1}[\widehat{y}(\mathbf{x}_i;\mathscr{Y}_{leaf}) = t_{y_i}] \prod_{n\in\mathscr{A}(t_{y_i})} \mathbb{1}[\widehat{y}(\mathbf{x}_i;\mathscr{Y}_n) \in \mathscr{A}(t_{y_i}) \cup \{t_{y_i}\}]), \tag{4.5}$$

where $\mathscr{A}(n)$ denotes all the ancestors of node n, and t_{y_i} is the leaf node corresponding to class label y_i. While Acc_{leaf} considers successful any correct classification at the leaf level of the tree, the *HCA* is stricter. It declares a success only when all the ancestors of the leaf node are correctly classified. In other words, each sample needs to be classified correctly at each tree level to be viewed as correctly classified under the *HCA*. Acc_{leaf} is an upper bound for the *HCA*.

Mean Treecut Accuracy (MTA) estimates the expected accuracy under the TOS classification setting. It computes the average accuracy over a set of treecuts $\mathscr{T}_c \in \Omega$,

$$MTA = \frac{1}{|\Omega|}\sum_{\mathscr{T}_c\in\Omega}\frac{1}{N}\sum_{i=1}^{N}\mathbb{1}[\widehat{y}(\mathbf{x}_i;\mathscr{Y}_{\mathscr{T}_c}) = t_{y_i}] \ , \tag{4.6}$$

where $\mathscr{Y}_{\mathscr{T}_c} = Leaf(\mathscr{T}_c)$. However, as shown by the following lemma the set of all possible tree cuts in the hierarchy $\mathscr{T}$ is usually very large.

Lemma 4.4.1. *For a balanced M-ary tree with depth L (root node is excluded and is at depth 0), the number of all valid treecut is* $L + \sum_{l=2}^{L}\sum_{k=1}^{N-1}\frac{N!}{k!(N-k)!}|_{N=M^{l-1}}$.

For example, a tree with $M = 2$ and $L = 6$ has more than 4 billion treecuts. For a dataset like ImageNet ($L = 15$) this number is monumental. Thus, we randomly sampled $|\Omega| = 25$ treecuts from $\mathscr{T}$ in all experiments. Our preliminary experiments have shown that the MTA is already fairly stable for this sample size.

State-of-the-art: To test TOS performance of the CLIP model with existing prompting techniques, we performed an experiment on ImageNet. Table 4.1 summarizes the performance of the

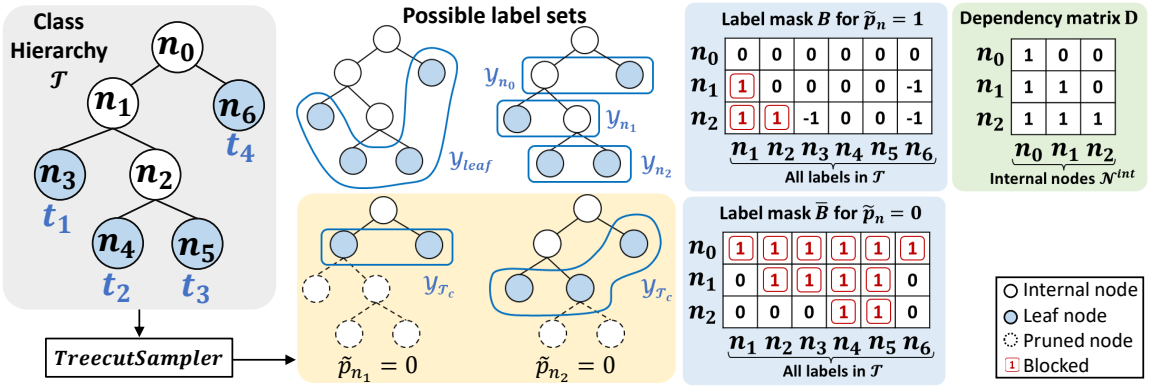


Figure 4.2. (Left) Multiple possible label sets are available in a class hierarchy. The label set can cover nodes at same level or across different hierarchy levels. (Right) Predefined matrices for efficient treecut sampling used in Algorithm 2.

different methods under the three metrics. Two conclusions are possible. First, the sharp drop from Acc_{leaf} to HCA shows that none of the methods make consistent predictions across the class hierarchy. Second, the low MTAs show that the expected accuracy of TOS classification is dramatically smaller than that of flat classification (leaf classes).

4.5 Prompt Tuning for Hierarchical Consistency

To enhance TOS performance of FMs, we propose *Prompt Tuning for Hierarchical Consistency* (ProTeCt). ProTeCt can be implemented with many existing prompt tuning methods (e.g., CoOp, MaPLe). These methods optimize context prompts using the cross-entropy loss of (4.2) with leaf label set $\mathcal{Y}_{leaf}$. While this optimizes leaf accuracy Acc_{leaf}, it is not robust to label set changes, even for label sets comprised of superclasses of $\mathcal{Y}_{leaf}$. A simple generalization would be to replace (4.2) with $\mathscr{L}(\mathbf{C}^t) = \sum_{\mathcal{Y}_p \in \mathcal{T}} L_{\mathcal{Y}_p}(\mathbf{C}^t)$, i.e., to consider all the partial label sets $\mathcal{Y}_p$ of the tree $\mathcal{T}$. However, for sizeable taxonomies, this involves a very large number of label sets and is not feasible. ProTeCt avoids the problem by dynamically sampling label sets from $\mathcal{T}$ during training, with a combination of two learning objectives, a *node-centric loss* (NCL) and a *dynamic tree-cut loss* (DTL).

Node-Centric Loss (NCL): NCL is the aggregate cross-entropy loss of (4.2) over all node-centric label sets $\mathscr{Y}_n = Chd(n)$ defined by each internal node $n \in \mathscr{N}^{int}$ of the hierarchy, i.e.,

$$\mathscr{L}_{NCL}(\mathbf{C}^t) = \frac{1}{|\mathscr{N}^{int}|} \sum_{n \in \mathscr{N}^{int}} L_{\mathscr{Y}_n}(\mathbf{C}^t) \ . \tag{4.7}$$

NCL optimization encourages prompts that robustify the classification at the different granularities. For example, "Corgi" should be classified as "mammal" within the animal label set $\mathscr{Y}_{n_1} = \{$mammal, reptile, bird$\}$, as a "dog" within the mammal label set $\mathscr{Y}_{n_2} = \{$dog, cat, elephant, tiger$\}$, and so forth.

Dynamic Treecut Loss (DTL): While NCL calibrates node classification, guaranteeing consistency within each node, the label sets of TOS classification can also span different sub-trees of the hierarchy, including nodes at different levels, e.g., $\mathscr{Y} = \{$dog, cat, elephant, tiger, reptile, bird$\}$. DTL seeks to calibrate such label sets, by aggregating the cross-entropy loss of (4.2) dynamically, i.e., on an example basis, over randomly sampled label sets $\mathscr{Y}_{\mathscr{T}_c} = Leaf(\mathscr{T}_c)$ comprised of the leaves of the tree cuts $\mathscr{T}_c$ (sub-trees) of $\mathscr{T}$. At each training iteration, a random tree cut $\mathscr{T}_c$ is sampled with the *TreeCutSampler* procedure of Algorithm 2, as illustrated on the middle of Fig. 4.2, to define the loss

$$\mathscr{L}_{DTL}(\mathbf{C}^t) = L_{\mathscr{Y}_{\mathscr{T}_c}}(\mathbf{C}^t) \qquad \mathscr{T}_c \sim TreecutSampler(\mathscr{T}, \beta), \tag{4.8}$$

where $\beta \in [0,1]$ is a rate of tree dropout. For this, a Bernoulli random variable $P_n \sim Bernoulli(\beta)$ of dropout rate β is defined for each internal node $n \in \mathscr{N}^{int} \setminus n_0$. The algorithm descends the tree $\mathscr{T}$, sampling a binary drop-out variable p_n at each node. If $p_n = 1$, node n is kept in the pruned tree $\mathscr{T}_c$. Otherwise, the sub-tree of $\mathscr{T}$ rooted with n is dropped from $\mathscr{T}_c$. The parameter β controls the degree of pruning. Larger β induces the pruning of more tree nodes, while $\beta = 0$ guarantess that $\mathscr{Y}_{\mathscr{T}_c} = \mathscr{Y}_{leaf}$. The root node n_0 is excluded, as $p_{n_0} = 0$ would imply discarding the whole $\mathscr{T}$.

The *TreeCutSampler* algorithm is an efficient procedure to sample tree cuts $\mathscr{T}_c$ from

Algorithm 2. Treecut Sampler

Input: The tree hierarchy $\mathscr{T}$ of the dataset, tree dropout rate β
Output: The treecut label set $\mathscr{Y}_{\mathscr{T}_c}$

```
// sampling p for internal nodes; prune the sub-tree rooted at
   n if p_n = 0
```
$p_{n_0} \leftarrow 1$; // always keep the root node
for $n \in \mathscr{N}^{int} \setminus n_0$ **do**
  $p_n \leftarrow Bernoulli(\beta)$
end
$\mathbf{p} \leftarrow (p_{n_1^{int}}, ..., p_{n_K^{int}})$
```
// correct p based on the dependency between internal nodes
```
$\widetilde{\mathbf{p}} \leftarrow \mathbf{p} \otimes \mathbb{1}[\mathbf{Dp} = \mathbf{D1}]$
```
// obtain blocked labels with predefined masks and the sampled
   p̃
```
$\mathbf{b} \leftarrow min(\mathbf{B}, 0)^T \widetilde{\mathbf{p}} + \bar{\mathbf{B}}^T (\mathbf{1} - \widetilde{\mathbf{p}})$
```
// gather available (unblocked) labels as the sampled label set
```
$\mathscr{Y}_{\mathscr{T}_c} \leftarrow \{n_j : n_j \in \mathscr{N} \setminus n_0, \mathbf{b}_j = 0\}$
return $\mathscr{Y}_{\mathscr{T}_c}$

$\mathscr{T}$. It starts by sampling a vector $\mathbf{p} = (p_{n_1^{int}}, ..., p_{n_K^{int}})$, where n_i^{int} denotes the i-th internal node and $K = |\mathscr{N}^{int}|$, containing pruning flags p_n for all internal nodes $n \in \mathscr{N}^{int}$. The next step is to enforce consistency between these flags, according to the tree structure. If any node in $\mathscr{A}(n)$ is pruned, then node n should be pruned even if $p_n = 1$. This is efficiently enforced across all the flags by defining a dependency matrix $\mathbf{D} \in \{0,1\}^{K \times K}$ where $\mathbf{D}_{ij} = \mathbb{1}[n_j^{int} \in \mathscr{A}(n_i^{int}) \cup \{n_i^{int}\}]$ indicates whether the i-th internal node n_i^{int} is a child of the j-th internal node n_j^{int}. An example is provided on the right of Fig. 4.2 for the tree on the left. The sampled flags are then corrected by computing $\widetilde{\mathbf{p}} = \mathbf{p} \otimes \mathbb{1}[\mathbf{Dp} = \mathbf{D1}]$, where $\mathbf{1}$ is the vector of K ones and $\otimes$ the Hadamard product. Note that both $\mathbf{D}$ and $\mathbf{D1}$ are pre-computed, making the complexity of this step roughly that of one matrix-vector multiplication.

To identify the leaves of the sampled treecut ($\mathscr{Y}_{\mathscr{T}_c} = Leaf(\mathscr{T}_c)$) efficiently, a mask $\mathbf{B} \in \{0, 1, -1\}^{K \times |\mathscr{N} \setminus \{n_0\}|}$ is defined, where each row corresponds to an internal node, and the columns contain all possible labels in $\mathscr{T}$, i.e., all nodes except the root n_0. Entry B_{ij} flags that n_j cannot

appear in the sampled label set, given that $n_i \in \mathscr{N}^{int}$ has not been pruned (i.e., $\widetilde{p}_{n_i^{int}} = 1$), as follows

$$B_{ij} = \begin{cases} 1, \text{ if } n_j \in \mathscr{A}(n_i^{int}) \cup \{n_i^{int}\} \quad (n_j \textit{ is an ancestor of } n_i^{int}) \\ 0, \text{ if } n_i^{int} \in \mathscr{A}(n_j) \quad (n_j \textit{ is a descendant of } n_i^{int}) \\ -1, \text{ otherwise} \quad (n_j \textit{ is outside of the sub-tree rooted at } n_i^{int}) \end{cases} . \tag{4.9}$$

Similarly, a matrix $\bar{\mathbf{B}}$, of entries $\bar{B}_{ij} = 1 - |B_{ij}|$, is defined to flag that n_j cannot appear in the label set, given that $n_i \in \mathscr{N}^{int}$ has been pruned, i.e. $\widetilde{p}_{n_i^{int}} = 0$. A mask of the nodes unavailable to the label set is then computed by accumulating the masks corresponding to the values of $\widetilde{\mathbf{p}}$,

$$\mathbf{b} = min(\mathbf{B},0)^T \widetilde{\mathbf{p}} + \bar{\mathbf{B}}^T (\mathbf{1} - \widetilde{\mathbf{p}}) , \tag{4.10}$$

where the mask in $min(\mathbf{B},0)$ is selected if $\widetilde{p}_n = 1$, and that in $\bar{\mathbf{B}}$ if $\widetilde{p}_n = 0$. Note that $min(\mathbf{B},0)$ clips $B_{ij} = -1$ to 0. The mask $\mathbf{b}$ can then be used to obtain $\mathscr{Y}_{\mathscr{T}_c} = Leaf(\mathscr{T}_c) = \{n_j : n_j \in \mathscr{N} \setminus n_0, b_j = 0\}$. Fig. 4.2 gives an example. When $\widetilde{\mathbf{p}} = (\widetilde{p}_{n_0}, \widetilde{p}_{n_1}, \widetilde{p}_{n_2}) = (1,0,0)$, then $\mathbf{b} = min(\mathbf{B}_1,0) + \bar{\mathbf{B}}_2 + \bar{\mathbf{B}}_3 = (0,1,1,2,2,0)$, signaling that only n_1 and n_6 are available to the label set (as $b_1, b_6 = 0$), resulting in $\mathscr{Y}_{\mathscr{T}_c} = \{n_1, n_6\}$.

Optimization: The overall loss used for prompt tuning is a combination of the two losses

$$\mathscr{L}(\mathbf{C}^t) = \mathscr{L}_{DTL}(\mathbf{C}^t) + \lambda \mathscr{L}_{NCL}(\mathbf{C}^t) \tag{4.11}$$

where λ is a hyperparameter. Note that, like previous prompting approaches, ProTeCt optimizes the learnable prompts $\{\mathbf{c}_m\}_{m=1}^M$ while keeping the parameters of Φ_{text}, Φ_{vis} frozen.

4.6 Experiments

In this section, we discuss experiments for evaluating the effectiveness of ProTeCt. To demonstrate that ProTeCt is a plug-an-play method, it was applied to two SOTA prompt tuning

Table 4.2. TOS performance with/without ProTeCt on Cifar100 ($\lambda = 0.5$), SUN ($\lambda = 0.5$) and ImageNet ($\lambda = 1$) dataset. $\beta = 0.1$ for all datasets.

Method	K-Shot	w/ ProTeCt	Cifar100				SUN			ImageNet		
			Acc_{leaf}	HCA	MTA (25)	MTA (100)	Acc_{leaf}	HCA	MTA (25)	Acc_{leaf}	HCA	MTA (25)
CoOp	16		72.88	10.04	50.64	51.14	73.82	38.28	52.99	71.23	2.99	46.98
	16	✓	72.94	56.85	87.69	87.30	74.59	62.94	83.51	69.92	37.74	88.61
			(+0.06)	(+46.81)	(+37.05)	(+36.16)	(+0.77)	(+24.66)	(+30.52)	(-1.31)	(+34.75)	(+41.63)
	1		65.03	7.81	41.78	44.17	63.65	33.36	51.20	63.67	1.59	40.52
	1	✓	66.88	41.01	81.64	81.01	63.79	49.62	76.25	66.11	25.79	86.14
			(+1.85)	(+33.2)	(+39.86)	(+36.84)	(+0.14)	(+16.26)	(+25.05)	(+2.44)	(+24.2)	(+45.62)
MaPLe	16		75.01	17.54	52.21	50.82	71.86	33.25	54.29	70.70	4.15	48.29
	16	✓	75.34	61.15	88.04	88.33	72.17	59.71	82.27	69.52	31.24	87.87
			(+0.33)	(+43.61)	(+35.83)	(+37.51)	(+0.31)	(+26.46)	(+27.98)	(-1.18)	(+27.09)	(+39.58)
	1		68.75	4.65	50.60	54.99	63.98	25.15	50.31	68.91	2.97	48.16
	1	✓	69.33	48.10	83.36	83.78	64.29	50.45	76.73	66.16	20.44	85.18
			(+0.58)	(+43.45)	(+32.76)	(+28.79)	(+0.31)	(+25.30)	(+26.42)	(-2.75)	(+17.47)	(+37.02)

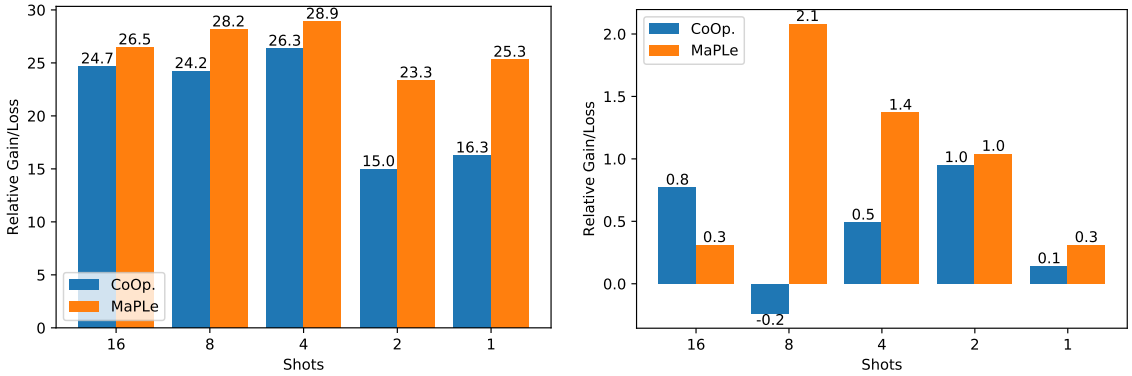
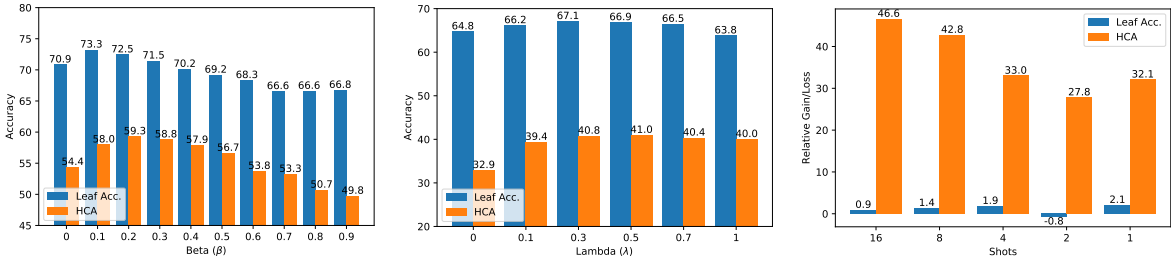


Figure 4.3. Relative gain/loss after adding ProTeCt. (Left) HCA ; (Right) Acc_{leaf}.



(a) (b) (c)

Figure 4.4. Ablation of (a) tree dropout rate β, (b) NCL strength λ and (c) CLIP ViT B32 architecture.

methods: CoOp [140] and MaPLe [62]. All experiments were conducted on a single Nvidia A10 GPU, using Pytorch [88]. ProTeCt code builds on the publicly available codebases for CoOp and MaPLe and will be released upon publication.

Metrics: Acc_{leaf} of (4.4), HCA of (4.5) and MTA of (4.6) are considered. MTA uses 5 tree dropout rates ($\beta \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$) to sample treecuts of various granularities. For each

Table 4.3. The gain of hierarchical consistency after adding ProTeCt generalizes across datasets in unseen domains. All methods are fine-tuned on ImageNet and evaluated on its 4 variants.

Method	K-Shot	w/ ProTeCt	ImageNetv2 [90]			ImageNet-S [106]			ImageNet-A [46]			ImageNet-R [45]		
			Acc_{leaf}	HCA	MTA (25)	Acc_{leaf}	HCA	MTA (25)	Acc_{leaf}	HCA	MTA (25)	Acc_{leaf}	HCA	MTA (25)
CoOp	16		64.01	2.31	43.74	47.82	1.39	38.58	50.28	2.97	52.56	75.83	18.49	64.13
	16	✓	62.60	32.84	86.66	46.80	20.73	82.60	49.08	22.45	78.21	74.94	31.18	75.59
			(-1.41)	(+30.53)	(+42.92)	(-1.02)	(+19.34)	(+44.02)	(-1.20)	(+19.48)	(+25.65)	(-0.89)	(+12.69)	(+11.40)
	1		56.43	1.51	38.27	41.38	1.11	33.61	45.92	1.76	47.54	69.84	11.74	55.31
	1	✓	60.16	22.95	84.38	44.75	13.88	80.64	48.95	20.52	76.95	74.26	27.46	76.48
			(+3.73)	(+21.44)	(+46.11)	(+3.37)	(+12.77)	(+47.03)	(3.03)	(+18.76)	(+29.41)	(+4.42)	(+15.72)	(+21.17)
MaPLe	16		64.15	1.97	45.93	48.97	1.58	43.37	50.61	2.31	54.88	76.61	20.67	63.06
	16	✓	62.77	27.86	86.14	47.47	17.77	82.52	47.41	19.75	77.46	75.70	32.58	77.99
			(-1.38)	(+25.89)	(+40.21)	(-1.50)	(+16.19)	(+39.15)	(-3.20)	(+17.44)	(+22.58)	(-0.91)	(+11.91)	(+14.93)
	1		61.78	2.18	45.50	46.79	1.70	45.26	47.55	3.52	55.48	74.55	18.85	62.48
	1	✓	59.14	17.89	83.27	44.92	11.24	79.94	47.15	16.03	76.81	74.60	25.20	75.72
			(-2.64)	(+15.71)	(+37.77)	(-1.87)	(+9.54)	(+34.68)	(-0.40)	(+12.51)	(+21.33)	(+0.05)	(+6.35)	(+13.24)

(a): [Boxer, Person]

(b): [Trolleybus, Animal]

(c):[Lion, Koala]

(d): [Cheeseburger, Ice cream]

Figure 4.5. ProTeCt correctly predicts examples from ImageNet (a,b) and its variants (c,d) at all levels. [GT, Prediction] shows the groundtruth and incorrect prediction by vanilla prompt tuning.

β, T treecuts are sampled without repetition to obtain a total of $5T$ treecuts. MTA($5T$) indicates the result is averaged over these $5T$ treecuts. We ablate $T=5$ and $T=20$ on Cifar100 and use $T=5$ for all datasets by default.

Training Details: All vanilla prompt tuning and their ProTeCt counterparts are trained under the same setting. The following configuration is used unless noted. All experiments use SGD optimizer and the learning rate is set to 0.02 with a cosine learning rate scheduler. By default, a pretrained ViT-B/16 CLIP model is used as initialization. For Cifar100 and SUN, we train both CoOp and MaPLe prompts for 200 epochs, using a batch size of 128 and 32, respectively. For ImageNet, CoOp is trained for 30 epochs with a batch size of 8, while MaPLe is trained for 10 epochs with a batch size of 2. Note that the setting is slightly different from the original paper due to our GPU availability.

4.6.1 TOS Classification Performance

Table 4.2 shows that vanilla CoOp and MaPLe have reasonable leaf accuracy for both 1-shot and 16-shot classification on Cifar100, SUN, and ImageNet. However, their very low HCA shows that their predictions are not consistent over the class hierarchy. As a result, their TOS classification performance (MTA) is much weaker than their leaf accuracy. For example, 16-shot classification with CoOp on ImageNet has a leaf accuracy of 71.23, but expected TOS accuracy of 46.98. This is explained by the very low HCA of 2.99. Similar observations hold for different few-shot configurations. In all cases, ProTeCt (results on rows with a checkmark) significantly improves HCA and MTA(25). For example, it boosts the HCA of 16-shot classification with CoOp on ImageNet by 34.75 (2.99 vs 37.74), leading to an increase of MTA(25) of 41.63 (46.98 to 88.61).

Note that, in all cases, MTA(25) after ProTeCt training is *higher* than leaf accuracy. This is expected for a well-calibrated classifier, since decisions at intermediate levels of the tree are coarser-grained than those at the leaves, which can require very fine class distinctions. These results show that ProTeCt robustifies the model for use in the TOS classification setting. The table also shows that ProTeCt maintains leaf accuracies comparable to those of the vanilla methods. Furthermore, the MTA results when 25 and 100 treecuts are sampled (corresponding to $T = 5$ and $T = 20$), are compared on Cifar100. It can be seen that the performances are similar, showing that sampling 25 treecuts is sufficient to achieve good estimation. Fig. 4.3 compares the **relative** gains in HCA and leaf accuracy of training with ProTeCt, as compared to vanilla prompt tuning. These gains are shown for both CoOp and MaPLe, under several few shot configurations, on SUN dataset. In all cases, ProTeCt increases HCA by more than 15 points, while maintaining a leaf accuracy comparable to that of vanilla CoOp/MaPLe.

Table 4.4. Comparison of CoOp with/without ProTeCt on FGVC Aircraft [83] dataset.

K-shot	w/ ProTeCt	Acc_{leaf}	HCA	MTA (25)
16		41.88	17.82	21.11
16	✓	42.00 (+0.12)	29.94 (+12.12)	32.95 (+11.84)
1		23.61	11.55	16.77
1	✓	27.30 (+3.69)	16.47 (+4.92)	24.67 (+7.90)

4.6.2 Domain Generalization of TOS Classification

Following the domain generalization setting of [140, 139, 62, 127], we investigate whether TOS classification performance generalizes across datasets. The CLIP model with ProTeCt prompts trained on ImageNet (source) is applied to 4 ImageNet variants (target) with domain shift: ImageNetv2 [90], ImageNet-Sketch [106], ImageNet-A [46] and ImageNet-R [45]. Table 4.3 summarizes the three metrics on these datasets for CoOp and MaPLe. Similarly to Table 4.2, ProTeCt enables significant gains in HCA and MTA(25) over the baselines for all datasets. Note that since ImageNet-A and ImageNet-R only contain 200 ImageNet subclasses, their hierarchy is different from that of ImageNet. These results demonstrate the flexibility and robustness of ProTeCt, even when transferring the model to a target domain whose class hierarchy is different from that of the source domain.

4.6.3 Ablation Study and Visualization

In this section, we discuss ablations of ProTeCt components and visualize the predictions of different models.

Tree Dropout Rate β: Fig. 4.4 (a) plots Cifar100 Acc_{leaf} and HCA as a function of the drop-out rate β, for 16-shot CoOp + ProTeCt training ($\lambda = 1$). Larger values of β reduce the likelihood of sampling the leaf nodes of the tree, resulting in shorter trees and weaker regularization. Hence, both leaf accuracy and HCA degrade for large β. However, always using the full tree ($\beta = 0$) also achieves sub-optimal results. The two metrics peak at $\beta = 0.1$ and $\beta = 0.2$, respectively. $\beta = 0.1$ is selected for all experiments.

Table 4.5. CoOp ablation on Cifar100 dataset. Both DTL and NCL loss improve the hierarchical consistency.

		16-shot			1-shot		
DTL	NCL	Acc_{Leaf}	HCA	MTA (25)	Acc_{Leaf}	HCA	MTA (25)
		72.88	10.04	50.64	65.03	7.81	41.78
✓		72.81	47.97	87.32	64.77	32.93	81.38
	✓	64.20	51.69	79.44	61.22	38.02	62.16
✓	✓	**72.94**	**56.85**	**87.69**	**66.88**	**41.01**	**81.64**

NCL strength λ: Fig. 4.4(b) summarizes the Cifar100 performance of 1-shot classification with CoOp and $\beta = 0.1$, as a function of the hyperparameter λ that balances the two losses of (4.11). The introduction of NCL improves leaf accuracy/HCA from 64.8/32.9 ($\lambda = 0$) to 66.9/41 ($\lambda = 0.5$). We adopt $\lambda = 0.5$ for CIFAR100 and SUN. For ImageNet, $\lambda = 0.5$ and $\lambda = 1$ have comparable performance.

Architecture: Fig. 4.4 (c) shows the plug-and-play properties of ProTeCt, by showing that the gains obtained for CoOp+ProTeCt in Fig. 4.3 with CLIP ViT B16 also hold for ViT B32 features.

Taxonomies: To investigate the robustness of ProTeCt across hierachies, we consider the FGVC Aircraft [83] dataset. This has its built-in hierarchy, which beyond differing from those of SUN [119] and WordNet [85], is a technical hierarchy of fine-grained aircraft classes. Table 4.4 summarizes the CoOp results for these experiments, showing that ProTeCt improves performance under all metrics. This illustrates its taxonomy robustness.

Loss: Table 4.5 ablates the model performance with/without NCL and DTL loss, for two few-shot settings, using CoOp on Cifar100. Both losses improve TOS performance individually and there is a significant additional gain when they are combined. Using NCL alone can degrade leaf performance, due to the lack of regularization across different levels of the hierarchy. The combination of the two losses overcomes this problem.

Adapter-based tuning methods: To investigate how ProTeCt affect the TOS performance of adapter-based tuning method on FM, we use the ProTeCt losses to train the CLIP adapter of [34] and the CLIP+LORA method of [27]. Table 4.6 shows that this again produces large consistency gains, indicating that ProTeCt losses generalize to both prompt-based and adapter-based methods.

Table 4.6. Improving other adapter-based tuning methods, including CLIP-Adapter and CLIP+LORA with ProTeCt on Cifar100.

K-Shot	w/ ProTeCt	CLIP-Adapter [34] Acc_{leaf}	CLIP-Adapter [34] HCA	CLIP-Adapter [34] MTA (25)	CLIP+LORA [27] Acc_{leaf}	CLIP+LORA [27] HCA	CLIP+LORA [27] MTA (25)
16		71.96	5.59	42.93	70.45	4.57	47.19
16	✓	72.47	57.15	87.67	70.64	51.06	77.29
		(+0.51)	(+51.56)	(+44.83)	(+0.19)	(+46.49)	(+30.10)
1		65.35	8.35	48.25	63.57	2.89	38.63
1	✓	67.29	36.21	78.49	63.62	24.66	56.42
		(+1.94)	(+27.86)	(+30.24)	(+0.05)	(+21.8)	(+17.79)

Visualization: Fig. 4.5 shows examples from ImageNet (a,b), ImageNet-A (c) and ImageNet-R (d). While ProTeCt can correctly classify these examples at all hierarchy levels, vanilla prompt tuning fails at certain levels.

4.7 Conclusion

In this work, we formulated the TOS classification setting, including datasets, performance metrics, and experiments. For a given dataset, a class hierarchy is built by assigning the dataset classes to leaf nodes and superclasses to internal nodes. The TOS classifier is then expected to support classification with label sets drawn throughout the taxonomy. We have shown that existing prompting methods fail to address this setting and proposed ProTeCt training for enhancing the TOS performance of FMs, like CLIP, with existing prompt tuning methods. ProTeCt includes two losses. A dynamic treecut loss, based on an efficient treecut sampler, dynamically regularizes labels of varying granularity. A node-centric loss encourages correct predictions at all hierarchy levels. Experiments show that ProTeCt enhances the TOS performance of existing prompt tuning techniques, both in the adaptation datasets and across unseen domains. Finally, it was shown that ProTeCt is successful for various architectures, hierarchies and parameter tuning methods.

Chapter 4 is, in full, based on the material as it appears in the publication of "ProTeCt: Prompt Tuning for Taxonomic Open Set Classification", Tz-Ying Wu*, Chih-Hui Ho*, and Nuno

Vasconcelos, In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition xiii (CVPR), 2024. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Discussion and Conclusion

In this thesis, we expand upon the scheme of the current long-tail research and delve into the nuanced challenges posed by long-tail distributions in real-world settings.

We started by exploring the long-tailed distributions with visual relation learning, which is a combinatorial long-tail problem. Through an iterative decoupled training scheme, we deftly navigate the interplay between multiple long-tail distributions, avoiding overfitting while maintaining model compactness. This demonstrates the sensibility of using compact structures to solve the complex long-tail problem.

Moreover, inspired by the adaptability of human decision-making, we pioneered the development of a taxonomic classifier tailored for realistic recognition of long-tail datasets. This is a long-tail recognition strategy more aligned with human cognition. This innovative approach, leveraging hierarchical label sets, enables reliable predictions across various levels of class popularity and improves the model performance for all the class popularities.

Furthermore, our exploration into the dynamics of label space distribution shift, particularly in the context of Visual-Language Models (VLMs), illuminates the inadequacies of existing methodologies in handling taxonomic open-set classification scenarios. By introducing novel metrics for hierarchical consistency evaluation and pioneering the Prompt Tuning for Hierarchical Consistency (ProTeCt) technique, we not only identify the shortcomings but also provide tangible pathways for improvement. This significantly enhances the VLM performance across varying levels of semantic granularity and underscores our contributions in advancing the capabilities of modern machine learning systems.

In summary, these works show that the long-tail distributions pose significant challenges to modern classifiers and the network design should delve into this problem, as model performance can be substantially improved with proper data sampling and model regularization. We also demonstrate the importance of bridging the gap between machine learning models and human-like adaptability through the provision of practical solutions and insightful analyses.

Bibliography

[1] inaturalist 2018 competition. https://github.com/visipedia/inat_comp.

[2] Karim Ahmed, Mohammad Haris Baig, and Lorenzo Torresani. Network of experts for large-scale image categorization. In *European Conference on Computer Vision (ECCV)*, 2016.

[3] Zeynep Akata, Scott Reed, Daniel Walter, Honglak Lee, and Bernt Schiele. Evaluation of output embeddings for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[4] D Anaki and S Bentin. Familiarity effects on categorization levels of faces and objects. *Cognition*, 2009.

[5] J Anderson. The adaptive nature of human categorization. *Psychological Review*, 1991.

[6] Mateusz Buda, Atsuto Maki, and Maciej Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106, 10 2017.

[7] Adrian Bulat and Georgios Tzimiropoulos. Lasp: Text-to-text optimization for language-aware soft prompting of vision & language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23232–23241, 2023.

[8] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[9] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, 16(1):321–357, June 2002.

[10] Tianshui Chen, Weihao Yu, Riquan Chen, and Liang Lin. Knowledge-embedded routing network for scene graph generation. In *Conference on Computer Vision and Pattern Recognition*, 2019.

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[12] Vincent S. Chen, Paroma Varma, Ranjay Krishna, Michael Bernstein, Christopher Re, and Li Fei-Fei. Scene graph prediction with limited labels. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[13] Xinlei Chen, Haoqi Fan, Ross B. Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *ArXiv*, abs/2003.04297, 2020.

[14] C. K. Chow. An optimum character recognition system using decision functions. *IRE Transactions on Electronic Computers*, EC-6:247 – 254, 12 1957.

[15] C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16:41–46, 1 1970.

[16] Peng Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 694–710. Springer, 2020.

[17] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Boosting with abstention. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[19] Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *International Conference on Algorithmic Learning Theory (ALT)*, 2016.

[20] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[21] Jia Deng, Nan Ding, Yangqing Jia, Andrea Frome, Kevin Murphy, Samy Bengio, Yuan Li, Hartmut Neven, and Hartwig Adam. Large-scale object classification using label relation graphs. In *European Conference on Computer Vision (ECCV)*, 2014.

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[23] Jia Deng, Jonathan Krause, Alexander C. Berg, and Li Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[24] Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *International Conference on Computer Vision (ICCV)*, 10 2017.

[25] Apoorva Dornadula, Austin Narcomey, Ranjay Krishna, Michael Bernstein, and Li Fei-Fei. Visual relationships as functions: Enabling few-shot scene graph prediction. *CoRR*, abs/1906.04876, 2019.

[26] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[27] Sivan Doveh, Assaf Arbelle, Sivan Harary, Rameswar Panda, Roei Herzig, Eli Schwartz, Donghyun Kim, Raja Giryes, Rogério Schmidt Feris, Shimon Ullman, and Leonid Karlinsky. Teaching structured vision & language concepts to vision & language models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2657–2668, 2022.

[28] Chris Drummond and Robert Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats oversampling. *Proceedings of the ICML'03 Workshop on Learning from Imbalanced Datasets*, 01 2003.

[29] Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. A survey of vision-language pre-trained models. In *International Joint Conference on Artificial Intelligence*, 2022.

[30] Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *Journal of Machine Learning Research*, 11:1605–1641, 5 2010.

[31] Giorgio Fumera and Fabio Roli. Support vector machines with embedded reject option. *Pattern recognition with support vector machines*, 2388:68–82, 7 2002.

[32] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, 2016.

[33] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of machine learning research*, 17(59):1–35, 2016.

[34] Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Jiao Qiao. Clip-adapter: Better vision-language models with feature adapters. *ArXiv*, abs/2110.04544, 2021.

[35] Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[36] Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International Conference on Machine Learning (ICML)*, 2019.

[37] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 06 2018.

[38] Wonjoon Goo, Juyong Kim, Gunhee Kim, and Sung Ju Hwang. Taxonomy-regularized semantic deep convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[39] Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. Scene graph generation with external knowledge and image reconstruction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[40] Yanming Guo, Yu Liu, Erwin M. Bakker, Yuanhao Guo, and Michael S. Lew. Cnn-rnn: a large-scale hierarchical image classification framework. *Multimedia Tools and Applications*, 77:10251–10271, 2018.

[41] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 1322–1328, 2008.

[42] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. *Advances in Intelligent Computing*, 3644:878–887, 09 2005.

[43] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, Sep. 2009.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[45] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Lixuan Zhu, Samyak Parajuli, Mike Guo, Dawn Xiaodong Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8320–8329, 2020.

[46] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. *CVPR*, 2021.

[47] Roei Herzig, Moshiko Raboh, Gal Chechik, Jonathan Berant, and Amir Globerson. Mapping images to scene graphs with permutation-invariant structured prediction. *CoRR*, abs/1802.05451, 2018.

[48] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[49] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam abd Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[50] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[51] Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[52] Zih-Siou Hung, Arun Mallya, and Svetlana Lazebnik. Union visual translation embedding for visual relationship detection and scene graph generation. *CoRR*, abs/1905.11624, 2019.

[53] Muhammad Abdullah Jamal, Matthew Brown, Ming-Hsuan Yang, Liqiang Wang, and Boqing Gong. Rethinking class-balanced methods for long-tailed visual recognition from a domain adaptation perspective. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[54] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision (ECCV)*, 2022.

[55] Heinrich Jiang, Been Kim, Melody Guan, and Maya Gupta. To trust or not to trust a classifier. *Advances in neural information processing systems*, 31, 2018.

[56] Davis Jim, Liang Tong, Enouen James, and Ilin Roman. Hierarchical semantic labeling with adaptive confidence. In *International Symposium on Visual Computing*, 2019.

[57] J. Johnson, R. Krishna, M. Stark, L. Li, D. A. Shamma, M. S. Bernstein, and L. Fei-Fei. Image retrieval using scene graphs. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 00, pages 3668–3678, June 2015.

[58] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[59] K Johnson. Impact of varying levels of expertise on decisions of category typicality. *Memory & Cognition*, 2001.

[60] K Johnson and C Mervis. Effects of varying levels of expertise on the basic level of categorization. *J Experimental Psychology: General*, 1997.

[61] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations (ICLR)*, 2020.

[62] Muhammad Uzair khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

[63] Hyo Jin Kim and Jan-Michael Frahm. Hierarchy of alternating specialists for scene recognition. In *European Conference on Computer Vision (ECCV)*, 2018.

[64] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *CoRR*, abs/1602.07332, 2016.

[65] Alex Krizhevsky and Geoffrey Hinton. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:2251 – 2265, 9 2019.

[66] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[68] Kibok Lee, Kimin Lee, Kyle Min, Yuting Zhang, Jinwoo Shin, and Honglak Lee. Hierarchical novelty detection for visual object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[69] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[70] Rongjie Li, Songyang Zhang, Bo Wan, and Xuming He. Bipartite graph network with adaptive message passing for unbiased scene graph generation. In *CVPR*, 2021.

[71] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics.

[72] Yikang Li, Tao Ma, Yeqi Bai, Nan Duan, Sining Wei, and Xiaogang Wang. Pastegan: A semi-parametric method to generate image from scene graph. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14, December 2019, Vancouver, BC, Canada*, pages 3950–3960, 2019.

[73] Yikang Li, Wanli Ouyang, Bolei Zhou, Jianping Shi, Chao Zhang, and Xiaogang Wang. Factorizable net: An efficient subgraph-based framework for scene graph generation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[74] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

[75] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP:1–1, 07 2018.

[76] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[77] Yuntao Liu, Yong Dou, Ruochun Jin, and Peng Qiao. Visual tree convolutional neural network in image classification. In *International Conference on Pattern Recognition (ICPR)*, 2018.

[78] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[79] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[80] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu. Large-scale long-tailed recognition in an open world. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[81] Timo Lüddecke and Alexander Ecker. Image segmentation using text and image prompts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7086–7096, June 2022.

[82] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens van der Maaten. Exploring the limits of weakly supervised pretraining. In *European Conference on Computer Vision (ECCV)*, 2018.

[83] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew B. Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *ArXiv*, abs/1306.5151, 2013.

[84] Marcin Marszałek and Cordelia Schmid. Semantic hierarchies for visual object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

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

[86] Pedro Morgado and Nuno Vasconcelos. Semantically consistent regularization for zero-shot recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[87] Son T. Nguyen, Ozgur S. Oguz, Valentin N. Hartmann, and Marc Toussaint. Self-supervised learning of scene-graph representations for robotic sequential manipulation planning. In *4rd Annual Conference on Robot Learning, CoRL 2020, Proceedings*, Proceedings of Machine Learning Research. PMLR, 2020.

[88] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[89] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[90] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning*, 2019.

[91] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28, pages 91–99. Curran Associates, Inc., 2015.

[92] Ruslan Salakhutdinov, Antonio Torralba, and Josh Tenenbaum. Learning to share visual appearance for multiclass object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[93] Babak Shahbaba and Radford M. Neal. Improving classification when a class hierarchy is available using a hierarchy-based prior. *Bayesian Analysis, 2(1):221–238*, 2007.

[94] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*, pages 467–482. Springer, 2016.

[95] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[96] Kihyuk Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, 2016.

[97] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016.

[98] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[99] J Tanaka and M Taylor. Object categories and expertise: Is the basic level in the eye of the beholder. *Cognitive Psychology*, 1991.

[100] Kaihua Tang, Yulei Niu, Jianqiang Huang, Jiaxin Shi, and Hanwang Zhang. Unbiased scene graph generation from biased training. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 3713–3722. IEEE, 2020.

[101] Kaihua Tang, Hanwang Zhang, Baoyuan Wu, Wenhan Luo, and Wei Liu. Learning to compose dynamic tree structures for visual contexts. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[102] Hung-Yu Tseng, Hsin ying Lee, Lu Jiang, Ming-Hsuan Yang, and Weilong Yang. RetrieveGAN: Image synthesis via differentiable patch retrieval. In *ECCV*, 2020.

[103] Vishaal Udandarao, Ameya Prabhu, Adhiraj Ghosh, Yash Sharma, Philip HS Torr, Adel Bibi, Samuel Albanie, and Matthias Bethge. No" zero-shot" without exponential data: Pretraining concept frequency determines multimodal model performance. *arXiv preprint arXiv:2404.04125*, 2024.

[104] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[106] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

[107] Jianfeng Wang, Thomas Lukasiewicz, Xiaolin Hu, Jianfei Cai, and Zhenghua Xu. Rsg: A simple but effective module for learning imbalanced datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3784–3793, 2021.

[108] Pei Wang and Nuno Vasconcelos. Towards realistic predictors. In *European Conference on Computer Vision (ECCV)*, 2018.

[109] Sijin Wang, Ruiping Wang, Ziwei Yao, Shiguang Shan, and Xilin Chen. Cross-modal scene graph matching for relationship-aware image-text retrieval. In *IEEE Winter Conference on Applications of Computer Vision, WACV 2020, Snowmass Village, CO, USA, March 1-5, 2020*, pages 1497–1506. IEEE, 2020.

[110] Xiao Wang, Guangyao Chen, Guangwu Qian, Pengcheng Gao, Xiaoyong Wei, Yaowei Wang, Yonghong Tian, and Wen Gao. Large-scale multi-modal pre-trained models: A comprehensive survey. *ArXiv*, abs/2302.10035, 2023.

[111] Yu-Xiong Wang and Martial Hebert. Learning from small sample sets by combining unsupervised meta-training with cnns. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[112] Yu-Xiong Wang and Martial Hebert. Learning to learn: Model regression networks for easy small sample learning. In *European Conference on Computer Vision (ECCV)*, 2016.

[113] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[114] Zhaoqing Wang, Yu Lu, Qiang Li, Xunqiang Tao, Yandong Guo, Mingming Gong, and Tongliang Liu. Cris: Clip-driven referring image segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022.

[115] Bin Wen, Jie Luo, Xianglong Liu, and Lei Huang. Unbiased scene graph generation via rich and fair semantic extraction, 2020.

[116] Sanghyun Woo, Dahun Kim, Donghyeon Cho, and In So Kweon. Linknet: Relational embedding for scene graph. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 560–570. Curran Associates, Inc., 2018.

[117] Tz-Ying Wu, Pedro Morgado, Pei Wang, Chih-Hui Ho, and Nuno Vasconcelos. Solving long-tailed recognition with deep realistic taxonomic classifier. In *European Conference on Computer Vision (ECCV)*, 2020.

[118] Liuyu Xiang and G. Ding. Learning from multiple experts: Self-paced knowledge distillation for long-tailed classification. In *European Conference on Computer Vision (ECCV)*, 2020.

[119] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010.

[120] Danfei Xu, Yuke Zhu, Christopher Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[121] Yue Xu, Yong-Lu Li, Jiefeng Li, and Cewu Lu. Constructing balance from imbalance for long-tailed image recognition. In *European Conference on Computer Vision*, pages 38–56. Springer, 2022.

[122] Shaotian Yan, Chen Shen, Zhongming Jin, Jianqiang Huang, Rongxin Jiang, Yaowu Chen, and Xian-Sheng Hua. PCPL: predicate-correlation perception learning for unbiased scene graph generation. In *MM '20: The 28th ACM International Conference on Multimedia, Virtual Event / Seattle, WA, USA, October 12-16, 2020*, pages 265–273, 2020.

[123] Zhicheng Yan, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. Hd-cnn: Hierarchical deep convolutional neural networks for large scale visual recognition. In *International Conference on Computer Vision (ICCV)*, 2015.

[124] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[125] Xu Yang, Kaihua Tang, Hanwang Zhang, and Jianfei Cai. Auto-encoding scene graphs for image captioning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[126] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[127] Yuhang Zang, Wei Li, Kaiyang Zhou, Chen Huang, and Chen Change Loy. Unified vision and language prompt learning. *ArXiv*, abs/2210.07225, 2022.

[128] Alireza Zareian, Svebor Karaman, and Shih-Fu Chang. Bridging knowledge graphs to generate scene graphs. In *Proceedings of the European conference on computer vision (ECCV)*, August 2020.

[129] Alireza Zareian, Svebor Karaman, and Shih-Fu Chang. Weakly supervised visual semantic parsing. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[130] Rowan Zellers, Mark Yatskar, Sam Thomson, and Yejin Choi. Neural motifs: Scene graph parsing with global context. *CoRR*, abs/1711.06640, 2017.

[131] Ji Zhang, Kevin J. Shih, Ahmed Elgammal, Andrew Tao, and Bryan Catanzaro. Graphical contrastive losses for scene graph generation. *CoRR*, abs/1903.02728, 2019.

[132] Jingyi Zhang, Jiaxing Huang, Sheng Jin, and Shijian Lu. Vision-language models for vision tasks: A survey. *ArXiv*, abs/2304.00685, 2023.

[133] Xiao Zhang, Zhiyuan Fang, Yandong Wen, Zhifeng Li, and Yu Qiao. Range loss for deep face recognition with long-tailed training data. In *International Conference on Computer Vision (ICCV)*, 2017.

[134] Yuhui Zhang, Alyssa Unell, Xiaohan Wang, Dhruba Ghosh, Yuchang Su, Ludwig Schmidt, and Serena Yeung-Levy. Why are visually-grounded language models bad at image classification? *arXiv preprint arXiv:2405.18415*, 2024.

[135] Bin Zhao, Li Fei-Fei, and Eric P. Xing. Large-scale category structure aware image categorization. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

[136] Zhisheng Zhong, Jiequan Cui, Shu Liu, and Jiaya Jia. Improving calibration for long-tailed recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16489–16498, 2021.

[137] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. BBN: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, pages 1–8, 2020.

[138] Kaiyang Zhou, Ziwei Liu, Yu Qiao, Tao Xiang, and Chen Change Loy. Domain generalization: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4396–4415, 2022.

[139] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[140] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision (IJCV)*, 2022.

[141] Xinqi Zhu and Michael Bain. B-cnn: Branch convolutional neural network for hierarchical classification. *CoRR*, abs/1709.09890, 2017.

[142] Yang Zou, Zhiding Yu, B.V.K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. In *European Conference on Computer Vision (ECCV)*, 2018.