Hawkeye: A PyTorch-based Library for Fine-Grained Image Recognition with Deep Learning

Jiabei He
Nanjing University of Science and Technology
Nanjing, China
hejb@njust.edu.cn

Yang Shen
Nanjing University of Science and Technology
Nanjing, China
shenyang_98@njust.edu.cn

Xiu-Shen Wei
Southeast University
Nanjing, China
weixs.gm@gmail.com

Ye Wu
Nanjing University of Science and Technology
Nanjing, China
wuye@njust.edu.cn

ABSTRACT

Fine-Grained Image Recognition (FGIR) is a fundamental and challenging task in computer vision and multimedia that plays a crucial role in Intellectual Economy and Industrial Internet applications. However, the absence of a unified open-source software library covering various paradigms in FGIR poses a significant challenge for researchers and practitioners in the field. To address this gap, we present *Hawkeye*, a PyTorch-based library for FGIR with deep learning. *Hawkeye* is designed with a modular architecture, emphasizing high-quality code and human-readable configuration, providing a comprehensive solution for FGIR tasks. In *Hawkeye*, we have implemented 16 state-of-the-art fine-grained methods, covering 6 different paradigms, enabling users to explore various approaches for FGIR. To the best of our knowledge, *Hawkeye* represents the first open-source PyTorch-based library dedicated to FGIR. It is publicly available at https://github.com/Hawkeye-FineGrained/Hawkeye/, providing researchers and practitioners with a powerful tool to advance their research and development in the field of FGIR.

CCS CONCEPTS

• **Software and its engineering** → **Software libraries and repositories**; • **Computing methodologies** → **Computer vision**.

KEYWORDS

Open-Source, Fine-Grained Image Recognition, Library, Deep Learning, Convolutional Neural Networks

1 INTRODUCTION

In recent years, significant advancements have been made in deep learning design and training, leading to substantial improvements in image recognition performance on large-scale datasets. Fine-Grained Image Recognition (FGIR) is a specialized area of research that focuses on the visual recognition of subcategories at a highly granular level within a broader semantic category. Despite significant progress with the help of deep learning [19], FGIR remains a highly challenging task. Also, it has significant scientific and practical applications in various scenarios within the Intellectual Economy and Industrial Internet, such as smart city, public safety, ecological protection, agricultural production and safety assurance, etc. The main challenge in FGIR is to understand the subtle visual differences that are necessary to distinguish objects with highly similar overall appearances but differing fine-grained features. The primary methods of FGIR can be roughly grouped into three paradigms [19]: (1) recognition by localization-classification subnetworks, (2) recognition by end-to-end feature encoding, and (3) recognition with external information.

Despite some methods from these paradigms being open-sourced, there is currently no unified open-source library available. New researchers in the field face a significant hindrance in replicating new approaches because different methods use distinct deep learning frameworks and design architectures, requiring the researchers to familiarize themselves with a new set of frameworks every time. Moreover, the absence of a unified library often necessitates researchers to develop the underlying code themselves, resulting in a waste of valuable time. Additionally, it is challenging to compare research results since each researcher/developer uses a distinct framework and base setup, leading to less reproducible results. Consequently, a unified open-source library is crucial for advancing the field of FGIR. To address this need, we developed a PyTorch-based library for FGIR, termed as *Hawkeye*.

The design of our work has the following advantages:

• To the best of our knowledge, this is the first dedicated codebase designed specifically for FGIR. Our library encompasses 16 representative methods spanning 6 paradigms in FGIR, providing researchers with a comprehensive understanding of the current
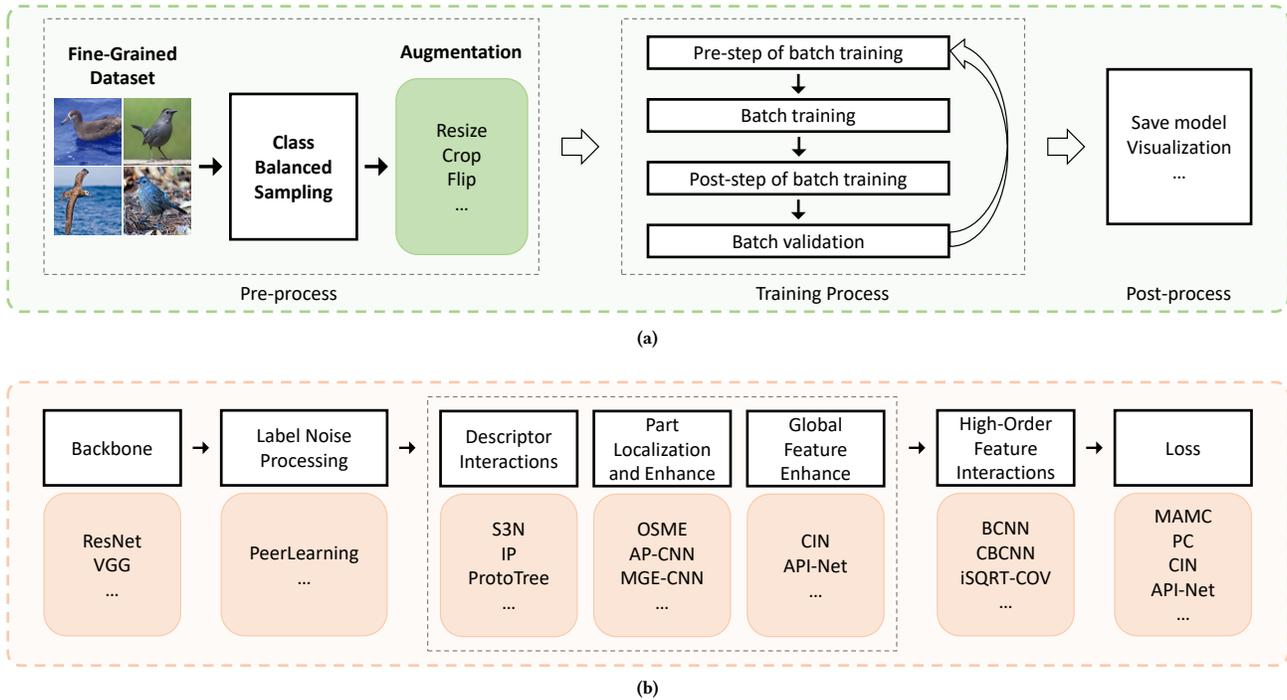
Figure 1: (a) Workflow of *Hawkeye,* including three stages: data pre-process, model training process and post-process. (b) Modules in *Hawkeye.* We present fine-grained methods involved in *Hawkeye* which mainly correspond to specific modules.

state-of-the-art techniques in the field. Furthermore, our modular design allows researchers to easily integrate their own methods or enhancements into the library, facilitating fair comparisons with other approaches. We are committed to maintaining and updating the open-source library to accommodate new advancements and emerging fine-grained methods in the future.

• Modular Design: The fine-grained recognition training pipeline is structured into multiple modules, which are subsequently integrated into a unified pipeline within the `Trainer` class. Users have the flexibility to override specific modules by inheriting the `Trainer` class, allowing for customization as per their requirements. The implementation of most methods does not necessitate extensive code modifications, ensuring that *Hawkeye* remains both flexible and user-friendly.

• High Code Quality: Our library prioritizes code readability in the pipeline implementation, emphasizing the simplification of each module and ensuring the complete comprehensibility of the pipeline process. This approach enables beginners to quickly familiarize themselves with the training process of fine-grained recognition and the functions of each part.

• Human-readable Configuration: Our library provides configuration files in the `YAML` format for each method. These files allow users to easily modify all the necessary hyperparameters for training, including those related to the dataset, model, optimizer, scheduler, and more. By focusing on a single configuration file, users can conveniently customize their experiments and adjust various settings to suit their specific needs.

2 DESIGN OVERVIEW

The workflow and composition modules of *Hawkeye* are illustrated in Figure 1. The pipeline of fine-grained image recognition is grouped into several modules, shown in black blocks of Figure 1b. Next, we will introduce the relationship of modules to paradigms in [19], essential modules, and the architecture of *Hawkeye*.

2.1 Correspondence between Modules and Learning Paradigms

Basically, the modules in *Hawkeye* correspond to the learning paradigms of fine-grained recognition [19]. Specifically, the label noise processing module corresponds to methods of the "recognition with web data" paradigm. The descriptor interactions module corresponds to methods of the "recognition by utilizing deep filters" paradigm. The part localization and enhancement module is mainly composed of methods from "recognition by leveraging attention mechanisms". The high-order feature interactions module is made up of methods from the "recognition by performing high-order feature interactions" paradigm. The loss function module corresponds to "recognition by designing specific loss functions".

2.2 Composition Modules in *Hawkeye*

We present the composition modules in *Hawkeye* as follows.

• **Class Balanced Sampling:** It samples data in the preprocess stage. This module is essential for methods that compare different classes of samples and require balanced sampling of multiple classes of samples in a single batch.

- **Backbone:** This module provides basic feature extraction networks, including ResNet and VGG.
- **Label Noise Processing:** It focuses on the process of handling label noise in webly fine-grained images, leaving clean data for subsequent modules.
- **Descriptor Interactions:** It leverages the locality and spatiality of descriptors to detect parts of fine-grained objects.
- **Part Localization and Enhancement:** This module detects the parts of a fine-grained object and constructs part-level representations corresponding to those parts, considering the small differences among fine-grained categories.
- **Global Feature Enhancement:** It explores interactions between deep channels or pairs of images using image-level representations.
- **High-Order Feature Interactions:** This module encodes the second-order statistics derived from convolutional activations.
- **Loss:** This module directly drives classifier learning and image representation learning through a loss function designed for fine-grained recognition.

2.3 Architecture

Each method has a configuration file in the YAML format that can be easily modified for specific parameters. The Trainer class implements the core functions for training, such as batch training methods, optimizers, hooks, and checkpoints. Users can implement their customized methods by inheriting the Trainer class, and a few lines of code require to be modified. A generic Dataset class is implemented for different fine-grained datasets. With the meta-data files provided in *Hawkeye*, users can easily apply and switch between the eight fine-grained benchmark datasets in experiments. The Model module includes the specific implementation of various methods, as well as the special Loss required by some methods. Users can easily add their own methods to *Hawkeye*. These modules are designed to be expandable, allowing users to implement customized designs without modifying unnecessary code.

3 SUPPORTED METHODS

In this paper, we provide the following representative fine-grained recognition methods of 6 different types according to [19], including utilizing deep filters, leveraging attention mechanisms, performing high-order feature interactions, designing specific loss functions, recognizing with web data, as well as miscellaneous. We have chosen 16 representative methods from these 6 types and implemented them in the library. We will briefly introduce these methods.

- S3N [3] leverages class peak responses, *i.e.*, local maximums, as the basis of part localization, based on class response maps [22].
- IP [7] provides an interpretation of classification results via the segmentation of object parts and the identification of their contributions.
- ProtoTree [14] combines prototype learning with decision trees, and thus results in an intrinsically interpretable model.
- MGE-CNN [21] promotes diversity among a mixture of experts by combing an expert gradually-enhanced learning strategy and a Kullback-Leibler divergence-based constraint.
- Sun *et al.* [15] incorporates channel attentions and metric learning to enforce the correlations among attended regions.

- APCNN [2] integrates low-level information to obtain enhanced feature representation and accurately located discriminative regions using a pyramidal hierarchy structure.
- Bilinear CNN [11] leverages bilinear pooling over the outputs of two CNNs to model local pairwise feature interactions in a translationally invariant manner.
- CBCNN [5] utilizes two compact bilinear representations with the same discriminative power as the full bilinear representation but with only a few thousand dimensions.
- Fast MPN-COV [10] (*i.e.*, iSQRT-COV) proposes an iterative matrix square root normalization method for fast end-to-end training of global covariance pooling networks.
- PC [4] reduces overfitting by intentionally introducing confusion in the activations.
- API-Net [23] attentively captures contrastive clues by pairwise interaction between two images.
- CIN [6] models the channel-wise interplay within and across images to exploit the rich relationships between channels.
- PeerLearning [16] trains two deep neural networks simultaneously, both of which mutually communicate proper knowledge from noisy web images.
- NTSNet [20] localizes informative regions with Navigator, Teacher and Scrutinizer cooperating and reinforcing each other.
- Cross-X [12] exploits the relationships between different images and between different network layers for robust multi-scale feature learning.
- DCL [1] destructs and then reconstructs the fine-grained image, for learning discriminative regions and features.

4 EXPERIMENTS

4.1 Benchmark Datasets

Eight representative fine-grained recognition benchmark datasets are provided. We provide the meta-data file of the datasets, and the *train* list and the *val* list are also provided according to the official splittings of the dataset. Researchers can easily utilize these datasets by following the examples provided in the library. Table 1 provides a summary of the year of publication, meta-category, number of images, and number ofcategories for each dataset.

4.2 Implementation Details

In our implementation, we use a NVIDIA GeForce 3060 GPU to train and infer the models of each method. We perform the training stage mainly using the image size of 448×448. The batch size, learning rate and epoch were set according to each method, using as many settings as possible from the method's corresponding paper, and these settings are detailed in individual config files for each method. We initializes the models with ResNet and VGG weights pre-trained on ImageNet, except for ProtoTree [14], which uses weights pre-trained on iNat2017 [17]. The optimisers are mainly Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam) or Adam with decoupled weight decay (AdamW). Most methods are trained using cosine learning rate scheduler with warm-up function to adjust the learning rate, while others use step learning rate or multiple step learning rate, etc. Image augmentation methods include random resized crop, random horizontal flip and random erasing with a probability of 0.1.

Table 1: Fine-grained benchmarks provided in *Hawkeye*.

Dataset	Year	Meta-class	# images	# categories
CUB-200 [18]	2011	Birds	11,788	200
Stanford Dog [8]	2011	Dogs	20,580	120
Stanford Car [9]	2013	Cars	16,185	196
FGVC Aircraft [13]	2013	Aircrafts	10,000	100
iNat2018 [17]	2018	Plants & Animals	461,939	8,142
WebFG-bird [16]	2021	Birds	18,388	200
WebFG-car [16]	2021	Cars	21,448	196
WebFG-aircraft [16]	2021	Aircrafts	13,503	100

Table 2: Performance of fine-grained recognition methods on the CUB-200 dataset. Except for the asterisked methods, 448×448 input images were used.

Methods	Original Acc.	Acc. in Hawkeye
Utilizing Deep Filters		
S3N [3]	88.50	88.29
IP [7]	87.30	86.65
ProtoTree [14]	82.20*	82.94*
Leveraging Attention Mechanisms		
MGE-CNN [21]	88.50	89.05
OSME+MAMC [15]	86.50	84.31*
APCNN [2]	88.40	87.84
Performing High-Order Feature Interactions		
BCNN [11]	84.10	83.80
CBCNN [5]	84.00	84.13
Fast MPN-COV [10]	88.10	88.81
Designing Specific Loss Functions		
Pairwise Confusion [4]	80.21	87.67
API-Net [23]	87.70	87.88
CIN [6]	87.50	85.34*
Recognition with Web Data		
Peer-Learning [16]	76.48	77.85
Miscellaneous		
NTS-Net [20]	87.50	88.19
CrossX [12]	87.70	87.65
DCL [1]	87.80	87.64

4.3 Results

We have conducted experiments on the methods implemented in *Hawkeye* using CUB-200 [18] to prove the effectiveness of our library. By integrating these methods with different implementations into a unified fine-grained recognition framework, some results show slight fluctuations, but they are still within acceptable limits.

We categorized the results based on the paradigm in [19] for easy observation and analysis, as presented in Table 2. Most of our experiments are performed on 448×448 input images, and the results marked with an asterisk use 224×224 input images.

5 AVAILABILITY

Hawkeye is released under the license of MIT and available at: https://github.com/Hawkeye-FineGrained/Hawkeye/. We also provide documentation and training samples. Contributions from the open-source community are welcome, via the GitHub issues/pull request mechanism.

6 CONCLUSIONS

We developed *Hawkeye*, the first open-source PyTorch-based library for fine-grained recognition with deep learning. Featuring a modular design, our library ensures simplicity and ease of extension. Each method is accompanied by training examples that require only minor code modifications, showcasing the user-friendly and highly adaptable nature of *Hawkeye*. We have implemented 16 fine-grained methods in a unified framework. It facilitates researchers in rapidly acquainting themselves with the cutting-edge advancements in fine-grained recognition, and expediting their exploration of novel ideas and enhancements We are dedicated to the ongoing maintenance and refinement of *Hawkeye* as an open-source project.

REFERENCES

[1] Yue Chen, Yalong Bai, Wei Zhang, and Tao Mei. 2019. Destruction and construction learning for fine-grained image recognition. In *CVPR*. 5157–5166.
[2] Yifeng Ding, Zhanyu Ma, Shaoguo Wen, Jiyang Xie, Dongliang Chang, Zhongwei Si, Ming Wu, and Haibin Ling. 2021. AP-CNN: Weakly supervised attention pyramid convolutional neural network for fine-grained visual classification. *IEEE TIP* 30 (2021), 2826–2836.
[3] Yao Ding, Yanzhao Zhou, Yi Zhu, Qixiang Ye, and Jianbin Jiao. 2019. Selective sparse sampling for fine-grained image recognition. In *ICCV*. 6599–6608.
[4] Abhimanyu Dubey, Otkrist Gupta, Pei Guo, Ramesh Raskar, Ryan Farrell, and Nikhil Naik. 2018. Pairwise confusion for fine-grained visual classification. In *ECCV*. 70–86.
[5] Yang Gao, Oscar Beijbom, Ning Zhang, and Trevor Darrell. 2016. Compact bilinear pooling. In *CVPR*. 317–326.
[6] Yu Gao, Xintong Han, Xun Wang, Weilin Huang, and Matthew Scott. 2020. Channel interaction networks for fine-grained image categorization. In *AAAI*. 10818–10825.
[7] Zixuan Huang and Yin Li. 2020. Interpretable and accurate fine-grained recognition via region grouping. In *CVPR*. 8662–8672.
[8] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Fei-Fei Li. 2011. Novel dataset for fine-grained image categorization: Stanford dogs. In *CVPR workshop*.
[9] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 2013. 3D object representations for fine-grained categorization. In *ICCV workshop*. 554–561.
[10] Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. 2018. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. In *CVPR*. 947–955.
[11] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. 2015. Bilinear CNN models for fine-grained visual recognition. In *ICCV*. 1449–1457.
[12] Wei Luo, Xitong Yang, Xianjie Mo, Yuheng Lu, Larry S Davis, Jun Li, Jian Yang, and Ser-Nam Lim. 2019. Cross-X learning for fine-grained visual categorization. In *ICCV*. 8242–8251.
[13] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151* (2013).
[14] Meike Nauta, Ron van Bree, and Christin Seifert. 2021. Neural prototype trees for interpretable fine-grained image recognition. In *CVPR*. 14933–14943.
[15] Ming Sun, Yuchen Yuan, Feng Zhou, and Errui Ding. 2018. Multi-attention multi-class constraint for fine-grained image recognition. In *ECCV*. 805–821.
[16] Zeren Sun, Yazhou Yao, Xiu-Shen Wei, Yongshun Zhang, Fumin Shen, Jianxin Wu, Jian Zhang, and Heng Tao Shen. 2021. Webly supervised fine-grained recognition: Benchmark datasets and an approach. In *ICCV*. 10602–10611.
[17] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. 2018. The inaturalist species classification and detection dataset. In *CVPR*. 8769–8778.
[18] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. 2011. The Caltech-UCSD birds-200-2011 dataset. (2011).
[19] Xiu-Shen Wei, Yi-Zhe Song, Oisin Mac Aodha, Jianxin Wu, Yuxin Peng, Jinhui Tang, Jian Yang, and Serge Belongie. 2022. Fine-grained image analysis with deep learning: A survey. *IEEE TPAMI* 44, 12 (2022), 8927–8948.
[20] Ze Yang, Tiange Luo, Dong Wang, Zhiqiang Hu, Jun Gao, and Liwei Wang. 2018. Learning to navigate for fine-grained classification. In *ECCV*. 420–435.
[21] Lianbo Zhang, Shaoli Huang, Wei Liu, and Dacheng Tao. 2019. Learning a mixture of granularity-specific experts for fine-grained categorization. In *ICCV*. 8331–8340.
[22] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *CVPR*. 2921–2929.
[23] Peiqin Zhuang, Yali Wang, and Yu Qiao. 2020. Learning attentive pairwise interaction for fine-grained classification. In *AAAI*. 13130–13137.