Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision Lab

Approximate Second Order Optimization for Fast Few-Shot Learning

Master's Thesis

Yihang She
Department of Civil, Environmental, and Geomatic Engineering

Advisors: Dr. Martin Danelljan, Goutam Bhat
Supervisor: Prof. Dr. Fisher Yu, Prof. Dr. Konrad Schindler

July 5, 2021

Abstract

Few-shot object detection refers to the problem of detecting objects with only a few examples. It is a challenging but real setting as many datasets arising from nature have long-tailed distributions. Currently, mainstream research in few-shot object detection employs traditional gradient descent to find a solution to optimization problems. Unfortunately gradient descent's non-competitive convergence rate results in high latency and consequently restricts real-time applications. Another issue is catastrophic forgetting where previously learned knowledge is lost as the model overfits on few-shot data.

In this thesis, we first revisited the optimization problem in few-shot object detection and designed a steep gradient descent strategy based on Conjugate Gradients (CG). In addition, we applied a meta learned weight generator and a feature-wise regularizer to make CG even faster and more stable on existing benchmark. Next, we proposed a simple yet effective hierarchical detection approach (HDA) to address catastrophic forgetting. We ran extensive experiments on the COCO dataset to measure model performance and converging speed. Experiments show that our optimization strategy CG is about 30 times faster than stochastic gradient descent when applied to the same two-stage tuning approach (TFA). Consequently, HDA achieves state-of-the-art in base class performance without any knowledge forgetting and has a competitive performance on novel classes. HDA converges 200 to 250 times faster than SGD in TFA, and as such shows potential for real-time applications.

Acknowledgements

This master thesis would be impossible without the guidance of my esteemed colleagues. In particular, I wish to express my appreciation towards Prof. Dr. Fisher Yu and Prof. Dr. Konrad Schindler for advising me on the choice of topic and supervision throughout this journey. I also wish to express my enormous gratitude to Dr. Martin Danelljan and Goutam Bhat for their generous support and timely advice. I would also like to thank my friends Nando Metzger and Bryan Yu sincerely for proofreading the thesis and giving valuable feedback.

Last but not least, I would like to thank my parents for their unconditional love and support. I dedicate my thesis to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Humans have the ability to quickly learn new concepts after experiencing similar tasks. This ability is commonly demonstrated as they go about their daily lives. For example, a pianist can readily learn to play the violin while exerting less effort compared to a novice with no background in music. If this could be simply be explained by age, consider how a toddler can learn new words after being given only a few instructions. Such an ability to effectively adapt to a new task with only a few examples is an ability also desired in machine learning. In technical terms, we define this challenge as the few-shot learning problem. Few-shot learning is significant for real-world applications of machine learning because on the one hand, real-world data from many domains naturally have a long-tailed distribution with imbalances in sample sizes across different classes. On the other hand, constraints such as computational and memory resources, physical labor costs for human annotations, as well as data privacy restricts the ability to apply the traditional machine learning paradigm of training a model on an abundance data. Real world applications require making the most of limited data. The general paradigm of few-shot learning is to have a data-abundant set of base classes and a data-scarce set of novel classes and try to generalize the knowledge learned from the abundant dataset of base classes to a few-shot dataset of novel classes [13, 26, 20, 22, 37, 33, 6, 40].

Currently, in few-shot learning, the instance-level task of object detection is less well studied compared to the image-level task of classification. As one of the fundamental tasks in computer vision, object detection aims at localizing object instances in an image, which is usually represented by bounding box, and determining the class label for each instance. Few-shot object detection is challenging given that it requires both classification and localization of the detected objects but with only a few examples. Besides, it is important for few-shot object detection to retain the performance on base classes while learning novel classes because in real applications, a scene could contain instances of both base and novel classes. The requirement to achieve high performance on both base and novel classes is also known as generalized few-shot object detection [38, 23, 12]. Currently, there are two major paradigms for few-shot object detection including meta-learning and transfer learning. The meta-learning-based approaches first learns task-level knowledge from the data-abundant base set and then applies it to a few-shot task. Benchmarks [21, 27, 11] for few-shot object detection were first developed with meta-learning as its idea to first learn task-level knowledge fits the setting of few-shot learning well. Subsequent to the meta learning-based approach, the two-stage fine-tuning approach TFA [38] and its following works [23, 12] developed the transfer learning-based approach, where the model is first trained on the data-abundant base dataset and then only a few layers are fine-tuned while the other layers remain fixed. Besides, for few-shot object detection, researchers noticed the necessity of keeping the performance on base classes while learning novel classes, as in real-world object detection one scene could contain the instances of both base and novel classes [12].

While existing literature have developed a few methods to tackle few-shot object detection, there is still room for improvement in the space of ”fast adaptation” and the avoidance of ”catastrophic forgetting”. **Fast adaptation**: Currently, Stochastic Gradient Descent (SGD) is commonly used for model updates in few-shot object detection. On the one hand, SGD is computationally cheap and stable at every single iteration and it has been well implemented in popular machine learning libraries. On the other hand, it endures slow convergence with a vast number of iterations, which fails to meet the demand of fast adaption and therefore hinders the potential of few-shot object detection in real-world applications. **Catastrophic forgetting**: As aforementioned, retaining the performance on base classes is also important. However, in few-shot learning, it is often the case that the knowledge learned on old classes is largely forgotten when the model adapts to new classes, which is referred as catastrophic forgetting. Even in the transfer learning-based approach where the model performance on base classes is largely refined than its previous meta learning-based benchmarks, forgetting still happens compared to the pre-trained base model. To mitigate the forgetting, various techniques such as knowledge distillation have been applied [12]. However, even in the work [12] that the model performance on base class has been successfully retained, complexity of the model is increased and loss in knowledge distillation is still inevitable. If we want a robot to quickly detect objects in a novel scene but with a few examples, current approaches are unsatisfactory and more works need to be done to ensure a real-time performance without catastrophic forgetting.

1.1 Focus of this Work

Given the issues analyzed above, the focus of the thesis can be captured with three words: **Faster**, **Higher**, **Stronger**. We propose a steep gradient descent technique and a novel hierarchical detection framework to achieve faster convergence in adaptation, higher performance on base classes without catastrophic forgetting, and stronger performance in real-world settings.

To achieve fast adaptation, we tailored an optimization strategy based on Conjugate Gradient (CG) [8]. We started by interpreting the optimization problem in the transfer learning-based approach as a shallow learning problem, where only a few layers are optimized. We then analyzed how to apply Newton’s method properly for such a problem by approximating the Hessian matrix at every step using the Gauss-Newton’s method. Then we applied CG to efficiently approximate the second-order optimization at each step. We first show that our optimization method is significantly faster than the traditional SGD by applying CG to the popular transfer learning approach TFA [38]. During subsequent analysis, we investigated how task-level knowledge could benefit transfer learning approaches with CG, wherein we determined that a good initial guess of weights can speed up CG while knowledge of the sensitivity of weights to different classes can regularize the CG to minimize overfitting to few-shot data. Therefore, we meta-learned a weight generator with feature embedding and a feature-wise regularizer to boost the performance of the current transfer learning approach when CG is deployed. Furthermore, to have higher performance, we proposed a simple yet effective hierarchical detection approach (HDA) that completely retains base class performance, thus negating catastrophic forgetting. The faster convergence and higher performance of HDA together make it stronger in real world challenges such as online learning of robots.

To summarize, the contributions of the thesis are as follows:

- We designed a CG-based steepest gradient descent strategy for few-shot object detection, which achieves mush faster convergence than traditional gradient descent and thus ensures a seamless performance in real-time applications.
- We integrated the meta-learning strategy with the transfer learning approach and utilized task-level

knowledge to make CG even faster and minimize overfitting when applied to current transfer learning approach.

- We proposed a simple yet effective hierarchical detection approach to completely alleviate the catastrophic forgetting on base classes without any increase of model complexity.

1.2 Thesis Organization

The remaining chapters of the thesis are organized as follows:

Chapter 2 will introduce the recent literature that are related to our work, including researches in few-shot learning, object detection, and few-shot object detection.

Chapter 3 will detail the methodology used in the work, including Conjugate Gradient-based optimization strategy, meta learning of weight generator and feature-wise regularization, and hierarchical detection approach.

Chapter 4 will present the dataset, evaluation metrics, implementation details of our experiments and results.

Chapter 5 will compare the results with existing benchmarks and discuss the implications.

Chapter 6 will summarize the work done in the thesis and give an outlook for future researches.

Chapter 2

Related Work

2.1 Few-shot Learning

Few-shot learning aims at learning a model that can generalize well to new knowledge provided considerably small datasets. Currently, several paradigms have been developed for the few-shot learning issues: meta-learning [13, 26, 20, 32, 14, 39, 34], metric learning [22, 37, 33, 35, 7], and transfer learning [7, 9]. The goal of meta-learning is to first learn task-level knowledge before learning a specific task. This task-level knowledge helps the model quickly adapt to new tasks with limited training data, which makes it suitable for the problem of few-shot learning. Optimization-based meta-learning is a prevalent paradigm and one way is to learn a good weight initialization. MAML [13] and its variants [26, 20] try to learn a good initial weight by optimizing it over simulated tasks in the meta-learning phase, and with the meta-learned weights, the model can generalize to the few-shot data with only a few steps of gradient update. Instead of learning the initial weights directly, another way is to meta-learn weight generation. Rusu *et al.* [32] meta learned a low dimensional embedding space to sample the model weights by conditioning on the few-shot data, which is easier to optimize compared to the high dimension space of model parameters. Gidaris *et al.* [14] adopted an attention-based weight generator for few-shot classification. Wang *et al.* [39] applied a meta-learned network module to generate the model weights by feature embedding. Sun *et al.* [34] meta-learned the scaling and shifting of pre-trained weights of a deep neural network to leverage the advantages of both meta-learning and transfer learning. Compared to learning the model weights directly, generating the weights conditioned on inputs is usually more task-specific, which was also applied in the thesis work.

Another paradigm to tackle the few-shot learning issue is metric learning, which also achieved competitive results. Metric learning aims at measuring the distance of the input data in a learned space. There are different variants to measure the distance. Siamese Neural Network [22] compare the similarity of the inputs for one-shot learning. Matching Nets [37] maps a query example to its label by matching it with the labeled support set. The Prototypical Network [33] computes the distance based on the prototype representation of input classes. Relation Network [35], which is also meta-learned, computes the relation scores between the query image and the few-shot examples of novel classes. Researchers also found that distance-based classifiers, which compute the cosine similarities between the feature and weight vectors, can reduce the intra-class variation and thus benefit the joint prediction of base and novel classes [14, 7, 38].

On another note regarding transfer learning, Chen *et al.* [7] fixed the pre-trained weights of base representation and fine-tuned the classifier for few-shot novel classes, which achieved results comparable with state-of-the-art. Dhillon *et al.* propose a transfer learning-based approach for image classification and consider it as a strong baseline for few-shot learning problem.

2.2 Optimization in Few-shot Learning

Bertinetto *et al.* [1] noticed that in the few-shot learning setting, updating only the parameters sensitive to specific classes actually formulate a shallow learning problem, and adaptation strategies that are more efficient than gradient descent exist for such a problem. They tested ridge and sigmoid regressions with closed-form solutions to achieve fast convergence for the meta-learning-based few-shot learning. Similarly, FIML [36] proposed a steepest gradient optimization algorithm that can be generalized to various learning objectives. A series of works [8, 2] applied the conjugate gradient method to update the few-shot learner in online tracking, where only the annotation of the initial frame is available and the model needs to be able to quickly adapt the following frames. A similar approach with conjugate gradient was also applied to other online settings such as video object segmentation [3] and has been proved to be efficient. While in current researches of few-shot object detection the gradient descent method is still dominant, we applied the conjugate gradient method for fast convergence that is preferable in real-time applications.

2.3 Object Detection

Currently, the paradigms of object detection are dominated by the one-stage [28, 25, 29, 30] and two-stage detection approaches [16, 15, 31, 18, 17]. YOLO series [28, 29, 30] is one of the representative works in one-stage detection, which applied a single neural network to predicting both the bounding box and classification score based on the anchors densely placed on an image. For two-stage detection, R-CNN [16] and its following variants (e.g. Fast R-CNN [15], Faster-RCNN [31], Mask R-CNN [17]) are popular, which firstly samples class-agnostic proposals from image features in the first stage, and makes region-wise predictions given the sampled proposals in the second stage. Compared to the one-stage approach, the two-stage approach is usually slower but more accurate. Recently, the attention-based transformer was also introduced to vision task [10] and Carion *et al.* [5] proposed DETR with encoder-decoder architecture to detect objects directly from global context with less heuristic design, which also achieve competitive results. Currently, the two-stage approach Faster-RCNN is widely used as the base detector to build benchmarks in existing researches for few-shot object detection [38, 19, 23, 12]. In the thesis, we applied the two-stage Faster RCNN as our basic model architecture.

2.4 Few-shot Object Detection

Few-shot object detection aims to detect the novel objects with only a few training examples, which is less studied compared to few-shot classification but is more complicated as an instance-level task that requires both localization and classification. Existing literature mainly adopted two paradigms to tackle the issue: meta-learning-based approach [21, 27, 11, 41, 19] and transfer learning based approach [6, 40, 38, 23, 12]. Similar to the idea of meta-learning introduced before, researchers leverage the meta-learned task-level knowledge to the detection task with limited training data. MetaYOLO [21] meta learned a feature learner module to extract the generic features of novel objects and a reweighting module to make predictions provided these features. ONCE [27] studied the few-shot object detection in the incremental setting by extracting class-agnostic features and meta-learning a code generator that is class-specific. Fan *et al.* [11] proposed Attention-RPN and Multi-Relation Detector to learn a metric space to measure the similarity of object pairs for detection. Following the DETR [41] introduced before, Meta-DETR meta leaned an encoder-decoder transformer for the few-shot object detection. DCNet [19] meta learned a Dense Relation Distillation module and Context-aware Aggregation module to fully leverage the features of novel objects.

For the transfer learning approach, LSTD [6] is one of the early works that adapted the detector learned on data-abundant objects to the target domain of few-shot novel objects. Wang *et al.* [38] proposed the two-stage fine-tuning approach TFA, where in the first stage a base predictor was trained in data-abundant base objects, and in the second stage the final layers for classification and bounding box localization were tuned on a balanced few-shot dataset containing both base and novel classes. This tuning-based approach is simple yet effective, which outperforms previous benchmarks using meta-learning. Additionally, they measured the model performance on both base and novel classes, as in real case one scenario can contain objects of both base and novel classes. Following TFA, LEAST [23] attributed the layers of the base model into the class agnostic layers extracting features at the image level and the class sensitive layers extracting features at the object level. Compared to TFA, more layers considered to be class sensitive were tuned, which boosts the performance on novel classes accordingly, however, deteriorates the performance on base classes. To mitigate this knowledge forgetting, they further applied knowledge distillation and the clustered exemplars of base objects. Fan *et al.* [12] noticed that even though TFA has achieved competitive results, it is still inferior to the performance of the pre-trained detector on base classes. They proposed Retentive R-CNN, which inherited the tuning approach of TFA with an auxiliary consistency loss to distill the knowledge of the base detector. Retentive R-CNN achieved performance on base classes that are on par with the pre-trained detector. However, even with distillation, loss of pre-learned knowledge is still inevitable. In this thesis, we proposed the hierarchical detection approach, which is simpler, more intuitive, and achieved the performance on base classes exactly as the pre-trained detector without any losses.

Chapter 3

Methods

Our work follows the transfer-learning approach [6, 40, 38, 23, 12] for few-shot object detection, which has a problem setting as follows: given data-abundant base categories C_b and data-scarce novel categories C_n with corresponding datasets D_b and D_n, we first train a base detector M_b on D_b, which is supposed to have strong ability to detect objects in C_b. Then we freeze the modules in the base detector M_b that are considered as class-agnostic and fine-tune only final layers on D_n or a combination of D_n on a small subset of D_b to obtain the novel detector M_n that is able to detect the novel categories C_n.

In the following sections, we will first revisit the optimization problem in one of the representative transfer-learning approaches TFA [38], and introduce our faster convergence algorithm using Conjugate Gradient (CG). Next, we will describe a dedicated strategy of meta-learning to make CG even faster and more stable based on the TFA framework. Then, we will introduce a novel hierarchical detection framework to address the catastrophic forgetting issues on base classes C_b with the ability of fast adaptation, and a new setting for few-shot object detection.

3.1 Fast Convergence with Conjugate Gradient

3.1.1 Baseline Approach: TFA

The two-stage fine-tuning approach TFA [38] deployed the two-stage detector Faster-RCNN as the base model architecture. Faster-RCNN [31] consists of a convolutional neural network (CNN) module for extracting generic image features, a Regional Proposal Network (RPN) to generate proposals for potential objects and a Region of Interest (ROI) head for making detections. Specifically for the ROI head, it first extracts the box features from the sampled proposals with two fully connected linear layers, and then the extracted box features will be passed two separate linear layers: one as a classifier $\mathcal{C}$ to classify each proposal and another as bounding box regressor $\mathcal{R}$ to localize each proposal, wherein the classification and the localization are made simultaneously.

TFA proposes to first train Faster-RCNN on the data-abundant D_b to have a base detector M_b, and in the second stage, it freezes all the modules of M_b except for the two final linear layers, namely the classifier $\mathcal{C}$ and $\mathcal{R}$, which will be fine-tuned on a balanced few-shot dataset containing both base categories C_b and novel categories C_n. Therefore, the loss $\mathcal{L}$ being optimized is actually composed of two components:

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{loc} \tag{3.1}$$

where $\mathcal{L}_{cls}$ is a cross entropy loss for classification in $\mathcal{C}$ and $\mathcal{L}_{loc}$ is a smoothed L_1 loss for bounding box regression in $\mathcal{R}$. Stochastic Gradient Descent (SGD) [4], as a widely used optimization technique in the machine learning, was also applied to optimize $\mathcal{L}$ in TFA and many other researches that tackle few-shot object detection. SGD updates the model weights w by simply computing the first-order gradient of $\mathcal{L}$:

$$w^{i+1} = w^i - \Delta w^i \tag{3.2}$$

$$\Delta w^i = \alpha \cdot \Delta \mathcal{L}(w^i) \tag{3.3}$$

where Δw^i is the gradient update at iteration i. α is the step length, and $\Delta \mathcal{L}$ is the first-order gradient of $\mathcal{L}$.

SGD is computationally cheap for each iteration and is stable to reach convergence in various learning settings. However, optimization with SGD can usually suffer a slow convergence, which harms the potential of few-shot object detection in an online setting. To address this issue, we propose an optimization strategy based on Conjugate Gradient to ensure the seamless performance of the few-shot detector in real-time applications, which we will introduce in the following part.

3.1.2 Conjugate Gradient-based Optimization

. In the second stage of TFA, only the parameters of two separate linear layers each for $\mathcal{C}$ and $\mathcal{R}$ are optimized. This actually formulates a shallow learning problem. As mentioned before, besides gradient descent, other optimization strategies could be used to optimize such a non-deep model for a faster convergence [1, 36]. To make the loss $\mathcal{L}$ fully convex and differentiable, we replace the smoothed L_1 loss with a L_2 loss for $\mathcal{L}_{loc}$. Thus each term of $\mathcal{L}$ is now formulated as:

$$\mathcal{L}_{cls}(pred_{cls}, gt_{cls}) = -\log \frac{\exp(pred_{cls}[gt_{cls}])}{\Sigma_j \exp(pred_{cls}[j])} \tag{3.4}$$

$$\mathcal{L}_{loc}(pred_{loc}, gt_{cls}, gt_{loc}) = \frac{1}{n}(pred_{loc}[gt_{cls}] - gt_{loc})^2 \tag{3.5}$$

where $pred_{loc}$ is the classification score from $\mathcal{C}$, and gt_{cls} is the ground truth label. j indexes the other predictions scores other than gt_{cls}. $pred_{loc}$ is the prediction of bounding box regression from $\mathcal{R}$, and gt_{cls} is the ground truth localization. n of the number of detected proposals.

These two terms together formulate a convex, differentiable and positive-definite loss function for $\mathcal{L}$. One well-known steep gradient technique to optimize such a formulation is Newton's method, which leverages the second-order information to search for a steep converging direction. To find the steepest gradient Δw^i for every iteration i, Newton's method approximate the loss $\mathcal{L}(w^{i+1})$ at next parameter estimation w^{i+1} using second-order Taylor expansion at current estimation w^i:

$$\mathcal{L}(w^{i+1}) = \mathcal{L}(w^i + \Delta w^i) \approx \mathcal{L}(w^i) + (\Delta w^i)^T \Delta \mathcal{L}(w^i) + \frac{1}{2}(\Delta w^i)^T H_{w^i} \Delta w^i \tag{3.6}$$

where H_{w^i} is the Hessian matrix of second-order derivatives.

A solution can be found by setting the the derivative of $\mathcal{L}(w^i + \Delta w^i)$ to Δw^i as zero:

$$\frac{\partial \mathcal{L}(w^i + \Delta w^i)}{\partial \Delta w^i} = \frac{\partial}{\partial \Delta w^i}[\mathcal{L}(w^i) + (\Delta w^i)^T \Delta \mathcal{L}(w^i) + \frac{1}{2}(\Delta w^i)^T H_{w^i} \Delta w^i] = 0 \tag{3.7}$$

$$\Delta w^i = -H_{w^i}^{-1} \cdot \Delta \mathcal{L}(w^i) \tag{3.8}$$

However, calculating the inverse of Hessian matrix H^{-1} is computationally expensive and can be numerically unstable. As analyzed before that we are actually optimizing a shallow learning problem where only the parameters of two separate linear layers are tuned, it is viable to approximate $\mathcal{L}$ at every step as a simpler positive-definite quadratic loss. Gauss-Newton algorithm can be applied to problems with such a formulation, which avoids the calculation of second-order Hessian by approximating it with the first-order Jacobian. Gauss-Newton algorithm defines a residual r and optimizes $\mathcal{L}$ by formulating it as a non-linear least square of r:

$$r(w) = w(x) - y \tag{3.9}$$

$$\mathcal{L}(w) = \|r(w)\|^2 = \|(w(x) - y)\|^2 \tag{3.10}$$

where $r(w)$ is the residual vector of at weight estimate w, (x, y) is a pair of input and ground truth data, $w(x)$ is the model prediction given the input x.

Then it approximates the residual $r(w^{i+1})$ at next parameter estimation w^{i+1} with the first-order Taylor expansion of $r(w^i)$ at current estimation w^i, and then approximate the loss $\mathcal{L}(w^{i+1})$ by replacing $r(w^{i+1})$ with its first-order expansion:

$$r(w^{i+1}) = r(w^i + \Delta w^i) = r(w^i) + J_{w^i}\Delta w^i \tag{3.11}$$

$$\mathcal{L}(w^{i+1}) = ||r(w^{i+1})||^2 \approx (\Delta w^i)^T J_{w^i}^T J_{w^i}\Delta w^i + 2(\Delta w^i)^T J_{w^i}^T r(w^i) + r(w^i)^T r(w^i) \tag{3.12}$$

where $J_w = \frac{\partial r(w^i)}{\partial w^i}$ is the Jacobian of first-order derivative of r at w.

A solution can be found by setting the the derivative of $\mathcal{L}(w^{i+1})$ to Δw^i as zero:

$$\frac{\partial \mathcal{L}(w^{i+1})}{\partial \Delta w^i)} = \frac{\partial}{\partial \Delta w^i}[(\Delta w^i)^T J_{w^i}^T J_{w^i}\Delta w^i + 2(\Delta w^i)^T J_{w^i}^T r(w^i) + r(w^i)^T r(w^i)] = 0 \tag{3.13}$$

$$\Delta w^i = -(J_{w^i}^T J_{w^i})^{-1} J_{w^i} r(w^i) \tag{3.14}$$

If we set $H_{w^i} \approx J_{w^i}^T J_{w^i}$, and $B_{w^i} = J_{w^i} r(w^i)$, a numerical solution of Δw^i can then be found by solving the following system of linear equations:

$$H_{w^i}\Delta w^i = B_{w^i} \tag{3.15}$$

Provided that H_{w^i} is positive-definite, one efficient way to find an approximated solution of Δw^i is the iterative Conjugate Gradient (CG) algorithm, which has been introduced and successfully applied to the online tracking [8, 2] that shares a similar setting of few-shot learning. CG approximate the solution of Δw^i at every iteration i by iteratively calculating a series of conjugate gradients β_j and corresponding coefficients Δc_j:

$$\Delta w^i = \Sigma_j \beta_j \Delta c_j \tag{3.16}$$

Intuitively, it finds the optimal step length and direction for each step of Newton iteration, which results in a steep gradient for fast convergence. To summarize, we adopt Newton's method as the foundation to derive the steepest gradient for every iteration. Given the non-deep model to optimize in the transfer-learning-based approach, we further approximate the loss at every Newton iteration as a quadratic loss where the Gauss-Newton algorithm can be applied to find the gradient update by solving a linear system. Then the solution of this linear system, namely the steep gradient, can be efficiently approximated by a series of CG iterations.

3.2 Refined CG in TFA with Meta-Learning

Task-level knowledge to learn. While previously the transfer-learning and meta-learning based approaches are two explicit streamlines, we argue that the task-level knowledge can also benefit the transfer-learning based approach if it is learned properly. We consider two kinds of task-level knowledge that can especially benefit the TFA framework to which CG is deployed. One of task-level knowledge is a good initialization of novel weights $w_n^0 = \left\{w_{n,i}^0, i = 1, \cdots, N\right\}$ for detecting C_n (N is the number of weight matrix or bias vector in each linear layer of $\mathcal{C}$ and $\mathcal{B}$), which can help CG achieve even faster convergence. For this goal, we meta learn a network module $\mathcal{N} = \{\mathcal{N}_i, i = 1, \cdots, N\}$ to generate the initial weights by embedding the task-specific box features, where $\mathcal{N}_i$ is the sub-network to predict specific $w_{n,i}^0$. Given the input features $\overline{x}_n \in \mathbb{R}^{c_n \times d}$ (each row of $\overline{x}_n$ is the average of box features belonging to the same class in C_n. c_n is the number of novel classes. d is the dimension of flattened box features), each sub-network $\mathcal{N}_i$ generates the corresponding weight or bias $w_{n,i}^0 \in \mathbb{R}^{out_{n,i} \times feat_i}$ for each linear layer of classifier $\mathcal{C}$ and box regressor $\mathcal{B}$ ($out_{n,i}$ is the output dimension of $w_{n,i}^0$. $feat_i$ is the dimension of input feature. $feat_i = d$ for weight and $feat_i = 1$ for bias). The predicted novel weights w_n^0 will be concatenated with the pretrained base weights w_b^0 to have the initial weights $w^0 = \left\{w_b^0, w_n^0\right\}$ for fine-tuning. $\mathcal{N}_i$ consists of two fully connected linear layers and a function $\mathcal{F}$ to rearrange the dimension of the outputs and scale the predicted weights so that it is at the same magnitude as w_b^0:

$$w_{n,i}^0 = \mathcal{N}_i(\overline{x}) = \mathcal{F}(\texttt{Linear2}_i(\texttt{ReLU}(\texttt{Linear1}_i(\overline{x}_n)))) \tag{3.17}$$

Another is the feature-wise sensitivity of base weights w_b when making detection of different classes. As in the fine-tuning stage, model performance on the base classes C_b tends to drop when the well pretrained weights w_b^0 get updated, which is referred as 'catastrophic forgetting' [42, 12]. This is especially an issue for CG, as it can easily overfit the few-shot dataset due to fast convergence, which leads to the drop of model performance on C_b. Task-level knowledge of the sensitivity of the base weights w_b for different classes can be used to regularize the update of w_b so that the decline in model performance on C_b can be mitigated during fine-tuning. For this goal, we meta learn a set of regularizing vectors $\lambda = \{\lambda_i, i = 1, \cdots, N\}$ to regularize the base weights $w_b = \{w_{b,i}, i = 1, \cdots, N\}$ feature-wise by comparing it with the pretrained base weights w_b^0, where $w_{b,i} \in \mathbb{R}^{out_{b,i} \times feat_i}$ and $\lambda_i \in \mathbb{R}^{feat_i}$. This feature-wise regularization term $\mathcal{L}_{reg}$ will added to $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{loc} + \mathcal{L}_{reg} \tag{3.18}$$

$$\mathcal{L}_{reg} = \Sigma_i^N \|\lambda_i \odot (w_{b,i} - w_{b,i}^0)\|^2 \tag{3.19}$$

where $\odot$ is element-wise multiplication between λ_i and each row of the difference matrix between ($w_{b,i}$ and $w_{b,i}^0$.

Meta learning phase. We follow the idea of optimization-based mata learning [13] to meta learn the weight generator $\mathcal{N}$ and feature-wise regularizing factor λ. The core of the meta learning phase is to simulate the few-shot object detection task and optimize the modules over sampled tasks to learn the task-level knowledge. To simulate the few-shot dataset, we randomly sample a stream of tasks from the base dataset D_b. Each task contains a support set D_s, a query set D_q, and a random split of pesudo base classes $C_{meta,b}$ and pesudo novel classes $C_{meta,n}$ from base classes C_b. For each sampled task, the meta learning phase jointly optimize $\mathcal{N}$ and λ with an inner loop and an outer loop. The inner loop is used to optimize the model weights w_{meta} with $\mathcal{N}$ and λ being integrated in the optimization. Specifically, for each sampled task at the inner loop, w_{meta} is initialized with $w_{meta}^0 = \left\{w_{meta,b}^0, w_{meta,n}^0\right\}$, where $w_{meta,b}^0$ is a subset of pretrained weights w_b^0 corresponding to $C_{meta,b}$, and $w_{pseudo,n}^0$ is generated by $\mathcal{N}$ via embedding the box features of

$C_{meta,n}$ in D_s. Then w_{meta} get updated over the support set D_s with a few iterations, where the feature-wise regularizing factor λ is applied to the loss. With the updated model weights w_{meta} passed from the inner loop, the outer loop will update $\mathcal{N}$ and λ with a step of gradient descent by evaluating w_{meta} on the query set D_q. Note that in this phase the parameters of $\mathcal{N}$ and λ are set as learnable, and the computational process though out both loops are tacked so that the loss evaluated on D_q can be back propagated to update $\mathcal{N}$ and λ.

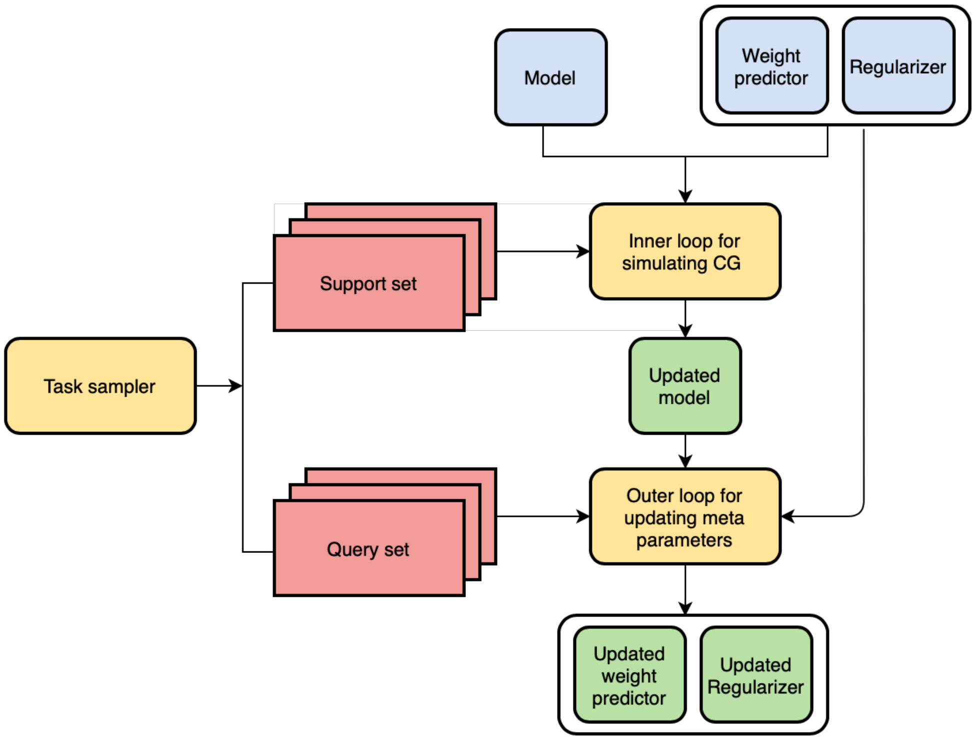


Figure 3.1: Work flow of meta learning weight generator and regularizer

Meta testing phase. In this phase, we fix the meta learned $\mathcal{N}$ and λ and applied it to refine the CG optimization deployed in the TFA framework. $\mathcal{N}$ predicts the novel weight w_n^0 for C_n by embedding $\overline{x}_n$, which will be concatenated with w_b^0 to have the initial weights w^0. λ will be applied to regularize the update of base weights w_b feature-wise to alleviate the knowledge forgetting on C_b.

3.3 Hierarchical Detection Approach

In this section we proposed a novel Hierarchical Detection Approach (HDA) for few-shot object detection to address the catastrophic forgetting on base classes C_b. It is based on transfer learning similar as TFA, but different from TFA and its variants that make joint detection on C_b and C_n by fine-tuning both w_b and w_n, HDA detect the objects according to their label hierarchy. Given the sampled box proposals, at the first hierarhical level it applied the pretrained base detector M_b to extracting the box proposals and detecting the object classes $C = \{C_b, C_{bg,1}\}$, where $C_{bg,1}$ is a super-class of all proposals that are considered as background by M_b. Then at the second hierarchical level, we train a non-deep novel detector $M_n = \{\mathcal{C}_n, \mathcal{B}_n\}$, which only contains a classifier $\mathcal{C}_n$ and a box regressor $\mathcal{B}_n$. From the background proposals corresponding to $C_{bg,1}$ and applied it to detecting $C_{bg,1} = \{C_n, C_{bg,2}\}$, where $C_{bg,2}$ is another super class of background proposals detected by M_n. HDA was proposed based on two assumptions:

- **Assumption 1**. The pretrained detector M_b is good enough to detect C_b.
- **Assumption 2**. M_b is robust enough to distinguish $C_{bg,1}$ from C_b.

The first assumption is based on the fact that M_b was pre-trained on a large dataset D_b with abundant annotations of classes C_b. Further fine-tuning the well-trained parameters w_b on a few-dataset usually does not improve the model performance on C_b. On the contrary, it results in 'catastrophic forgetting' of prel-learned knowledge, which requires additional tricks such as knowledge distillation to alleviate this forgetting. Therefore, to make sure the performance on C_b, one intuitive approach would be to fix M_b and apply it directly to detect C_b. However, in the few-shot object detection task, the model needs to detect not only C_b, but also C_n. This requires the second assumption regarding the robustness M_b against background proposals, which is supposed to contain most of the box proposals of C_n so that it is reliable for the further detection on C_n. In the following sections, we will detail the workflow of the hierarchical detection.

Detection on base classes. Same as TFA, HDA applied Faster-RCNN as base detector and the pretrained detector M_b was decomposed into three parts $M_b = \{\mathcal{M}_{base}, \mathcal{C}_b, \mathcal{B}_b\}$, where $\mathcal{M}_{base}$ is used to extract the class-agnostic proposals and box features, $\mathcal{C}_b$ and $\mathcal{B}_b$ are two separate linear layers for classification and localization on C_b. We first applied the class-agnostic $\mathcal{M}_{base}$ to sample a set of box proposals $P = \{P_i, i = 1, \cdots, K\}$ for detection of all classes $C = \{C_b, C_n\}$. Each P_i here represents the bounding box, box features and associated data for training or inference. To make sure that the performance on base classes can be completely retained while not hindering the detection on novel classes, we proposed a dedicated workflow to filter the proposals P_b and $P_{bg,1}$ for detecting C_b and C_n. Provided the box proposals P and the classification score $S \in \mathbb{R}^{K \times (\|C_b\|_0 + 1)}$ from $\mathcal{C}_b$ (K is the number of box proposals, $\|C_b\|_0 + 1$ is the number of output channels of $\mathcal{C}_b$ including background class), the filtering of P_b is actually to simulate how inference on S will be done to output the detection for C_b. As shown in Algorithm 1, step 1 and step 2 filter the indices I_b of proposals which have a classification score in any of the output channels for C_b greater than the `score_thresh`. This makes sure that the proposals with very likely objectness are selected. However, the selected proposals with high objectness could also contain many proposals belonging to C_n. Thus step 3 to step 4 further filter the proposal indices using the Non-maximum Suppression (NMS) with the given threshold `NMS_thresh}` and selects the top-k (`top_k`) proposals from the NMS outputs according to the scores in descending order. Step 1 to 5 together select a subset of proposals P_b that are most likely to contain positive objects in C_b and will leave the remaining proposals P_{rest} for further detection.

Detection on novel classes. From the remaining proposals P_{rest} and scores S_{rest} filtered out by step 1 to step 5 in the Algorithm 1, we filter the proposals $P_{bg,1}$ belonging to the super-class of background $C_{bg,1}$ at the first hierarchical level, which is done by selecting proposals that have a background score greater than scores of any other class in C_b (step 6 to step 8). Then the background proposals $P_{bg,1}$ will be passed to the novel detector $M_n = \{\mathcal{C}_n, \mathcal{B}_n\}$ for detecting objects in $C = \{C_n, C_{bg,2}\}$, where C_n is the novel class and $C_{bg,2}$ is another super-class of background at the second hierarchical level.

Training and inference. For training, we fix the pretrained M_b and only need to train the novel detector M_n. The dataset is same as the one used in TFA, namely a balanced few-shot dataset D_{train} containing both base and novel classes. D_{train} will be first passed to the base detector to generate proposals P, which will be split into P_b and $P_{bg,1}$ according to the procedure described before. Then $P_{bg,1}$ will be used to train the classifier $\mathcal{C}_n$ and box regressor $\mathcal{B}_n$ of M_n. The inference process follows a similar logic, where we passed the test set D_{test} first to M_b to extract the proposals P and then filter it into P_b and $P_{bg,1}$. Then we passed P_b to the $\mathcal{C}_b$ and $\mathcal{B}_b$ of M_b for detecting C_b, and passed $P_{bg,1}$ to the trained M_n for detecting C_n.

Algorithm 1 Procedure of Proposal Filtering

Input:
Extracted proposals $P = \{P_i, i = 1, \cdots, K\}$, classification scores $S \in \mathbb{R}^{K \times (\|C_b\|_0 + 1)}$
Require: `score_thresh`, `nms_thresh`, `top_k`
Output: Proposals P_b for detecting C_b, proposals $P_{bg,1}$ for detecting C_n

```
1: $S_b \leftarrow S[:, : -1]$    ▷ $S_b$ is the classification score for $C_b$
2: $I_b \leftarrow S_b > \texttt{score\_thresh}$    ▷ $I_b$ are indices of selected base proposals
3: $I_b \leftarrow \texttt{NMS}(P, I_b, \texttt{nms\_thresh})$
4: $I_b \leftarrow I_b[: \texttt{topk}]$
5: $P_b \leftarrow P[I_b]$    ▷ $P_b$ is the set of filtered proposals for detecting $C_b$
6: $P_{rest} \leftarrow P \setminus P_b,\ S_{rest} \leftarrow S \setminus S[I_b]$
7: $I_{bg,1} \leftarrow S_{rest}[:, -1] > \texttt{Max}(S_{rest}[:, : -1], \texttt{dim} = 1)$    ▷ $I_{bg,1}$ are indices of background proposals
8: $P_{bg,1} \leftarrow P_{rest}[I_{bg,1}]$    ▷ $P_{bg,1}$ is the set of proposals for detecting $C_n$
```

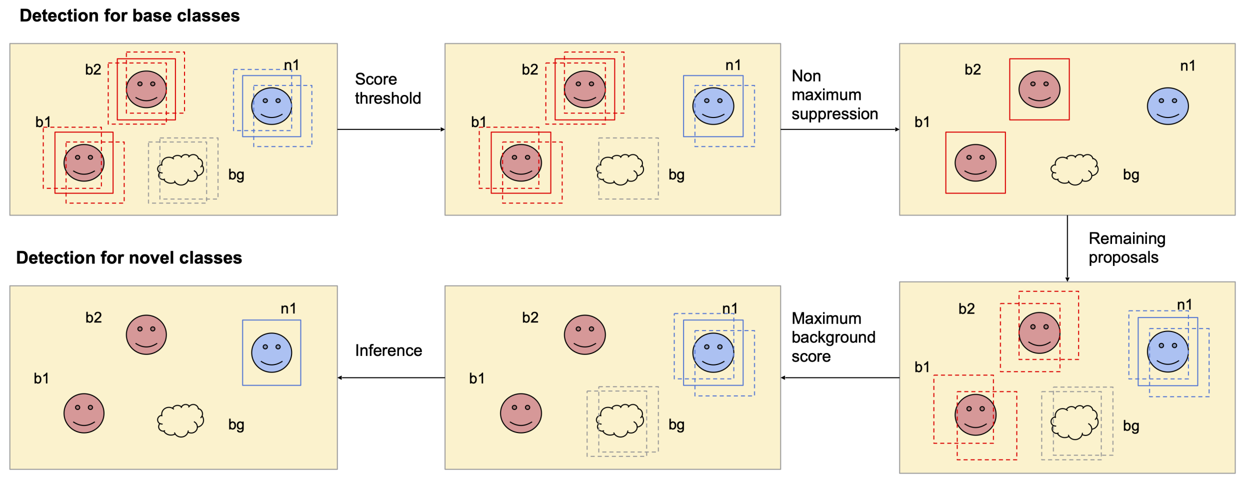


Figure 3.2: Workflow of hierarchical detection approach

Chapter 4

Experiments and Results

4.1 Dataset

We used the Microsoft COCO dataset [24] for our experiments, which has abundant images and annotations and is widely used for researches in few-shot object detection. COCO dataset has 80 categories that can be used for training. Following previous works [38, 23, 12], we split the 80 categories into 60 base categories and 20 novel categories. The test set has 5000 images that contain all categories, and the remaining images were used for training. A base dataset containing 60 base categories is used for training the base model and also for meta-learning. For few-shot learning of novel categories, we have a $K \in \{1, 2, 3, 5, 10\}$ -shot datasets sampled over 10 random seeds following [38], which is a balanced small dataset containing all categories.

4.2 Evaluation Metrics

Precision aspect. Following the setting of generalized few-shot object detection [38, 23, 12], we apply the average precision to measure the model performance in all classes (AP), base classes (bAP), and novel classes (nAP), respectively. Average precision is defined as the area under the precision-recall curve at a certain IOU threshold. Same as previous benchmarks [38], here AP refers to the mean of average precisions of all classes with an IOU threshold from 0.5 to 0.95. The definition for bAP and nAP is similar to AP, but they are calculated for base and novel classes, respectively. Besides, AP at a certain IOU threshold is also used for measuring the model performance. For instance, AP50 is the AP calculated at an IOU threshold of 0.5.

Time aspect. To measure the speed of convergence with different optimization strategies, we measure the following metrics: time used for extracting features and proposals from the frozen layers T', time per iteration t, number of measured iterations n, and time needed for reaching convergence $T = t \cdot n$. The unit for time measurement is seconds (s).

4.3 Implementation Details

We use Faster-RCNN as the basic model architecture and ResNet-101 as the backbone. The base detector M_b pre-trained by Wang *et al.* [38] is directly applied in the thesis, wherein M_b was trained using SGD with 110000 iterations, momentum 0.9, weight decay 0.0001, and base learning rate of 0.02 on the dataset containing 60 base categories. After the base detector being trained, all its layers except for the layers of $\mathcal{C}$ and $\mathcal{B}$ are fixed, and $\mathcal{C}$ and $\mathcal{B}$ will be tuned in TFA or adapted for hierarchical detection.

When applying CG, we first extract all proposals and associated box features from the fixed layers of the detector, and then optimize CG on all the extracted proposals and features. To have a fair comparison regarding the running speed of SGD with CG, we adapt the implementation of SGD in TFA, where the proposals and features for each image are first extracted, and then the extracted proposals and features are batched to fine-tune the model with SGD.

All experiments are run on the $K \in \{1, 2, 3, 5, 10\}$ -shot datasets sampled over 10 random seeds, and the results of average precision are aggregated over the 10 groups of different random seeds to measure the model performance more accurately [38]. For the time aspect, we report the time metrics from experiments that were run on seed 0.

To measure the time metrics, we run all experiments on a single GPU with the same type 'TITXAN Xp'.

4.4 Experiments

4.4.1 Naive CG in TFA

Baseline. To have a fair comparison for the running time with CG, we adapt the original implementation of SGD in `TFA w/cos` to run it on the extracted proposals and features. The remaining setting for running SGD is as same as the one in TFA. Specifically, we first train the weights corresponding to detecting C_n on the novel dataset and then concatenate the weights with the pre-trained weights corresponding to C_b, and then fine-tune the model weights on the few-shot dataset. Same as the setting in TFA, we apply the SGD with momentum 0.9 and weight decay 0.0001. The learning rate for training novel weights is 0.01 and the one for fine-tuning is 0.001. The number of iterations increases with the size of the few-shot dataset. The model run in this experiment is noted as `TFA+SGD` for short. Table 4.1 displays the model performance of `TFA+SGD`.

Table 4.1: Average precision of the model `TFA+SGD`

Shots	All			Base			Novel		
	AP	AP50	AP75	AP	AP50	AP75	AP	AP50	AP75
1	24.1	40.3	25.4	31.5	52.3	33.2	2.3	4.4	2.1
2	24.7	40.5	26.4	31.7	51.5	34.1	3.8	7.5	3.5
3	25.0	40.6	27.0	31.6	51.0	34.5	4.9	9.4	4.7
5	25.7	41.3	27.6	31.8	50.9	34.8	6.4	12.6	6.0
10	26.1	42.3	27.8	31.6	51.0	34.5	8.3	16.4	7.7

Naive CG. We replace the SGD with CG in the two-stage tuning approach TFA with cosine output layers (`TFA w/cos`) to observe how the model performance and converging speed will change. As aforementioned, for the few-shot dataset, we first extract the proposals and box features from the fixed layers of the pre-trained detector and then run CG directly on these extracted proposals and features. For weights in $\mathcal{C}$ and $\mathcal{B}$, we initialize the parameters corresponding to C_n with randomly initialized weights. For all the datasets with different shots, we run CG with 100 Newton iterations and each Newton iteration is approximated with 2 CG iterations. The model run in this experiment is noted as `TFA+CG` for short. Table 4.2 displays the model performance of `TFA+CG`.

Table 4.2: Average precision of the model `TFA+CG`

Shots	All			Base			Novel		
	AP	AP50	AP75	AP	AP50	AP75	AP	AP50	AP75
1	24.2	40.5	25.3	31.4	52.2	33.0	2.6	5.4	2.3
2	25.3	41.8	27.1	32.4	53.0	34.9	4.0	8.1	3.5
3	26.2	42.7	28.3	33.3	53.7	36.5	4.9	9.8	4.5
5	27.3	44.1	29.7	34.3	54.6	37.6	6.4	12.5	6.0
10	28.5	45.6	31.0	35.2	55.4	38.6	8.5	16.1	8.1

4.4.2 Meta-Learning to Refine CG in TFA

Each element of the regularizer λ is randomly initialized from the Gaussian distribution $\mathcal{G} \sim (0, 0.1)$, and the predicted weights from $\mathcal{N}$ are scaled with the factor 0.1 to make sure that the magnitude of the predicted weights is same as the one of the pre-trained weights. For each sampled task, we randomly split the 60 categories in C_b into 40 pseudo base categories $C_{meta,b}$ and 20 pseudo novel categories $C_{meta,n}$. Both the support set and the query set are 60-way-1-shot datasets. Each meta-learning loop contains an inner loop and an outer loop. For the inner loop of meta-learning, we run 5 Newton iterations and 2 CG iterations for each Newton iteration to update the model weights. For the outer loop of meta-learning, we run a single SGD step to update $\mathcal{N}$ and λ with the learning rate 0.01, momentum 0.09, and weight decay 0.0001. We run 1000 meta-learning loops to meta learn $\mathcal{N}$ and λ. When applying these meta-learned modules to TFA with CG, we run 30 Newton iterations, each with 2 CG iterations. The model run in this experiment is noted as `TFA+CG+meta` for short. Table 4.3 displays the model performance of `TFA+CG+meta`.

Table 4.3: Average precision of the model `TFA+CG+meta`

Shots	All			Base			Novel		
	AP	AP50	AP75	AP	AP50	AP75	AP	AP50	AP75
1	24.3	40.6	25.7	31.6	52.8	33.4	2.3	4.2	2.3
2	25.3	41.4	27.4	32.6	53.3	35.4	3.3	6.0	3.3
3	25.9	41.9	28.3	33.3	53.5	36.4	3.8	6.9	3.9
5	26.4	42.2	28.9	33.8	53.6	37.0	4.5	8.0	4.6
10	26.9	42.3	29.4	34.2	53.6	37.5	4.8	8.5	5.0

4.4.3 HDA with CG

Naive CG. For the proposal filtering in HDA, we set `score_thresh` = 0.05, `nms_thresh` = 0.5 and `top_k` = 100 following the common setting of Faster R-RCNN during inference in previous benchmarks [38, 23, 12]. Given the filtered proposals, we run 30 Newton iterations and each with 2 CG iterations to train the novel detector head, which is randomly initialized. The model is noted as `HDA+CG` for short. Table 4.4 displays the model performance of `HDA+CG`.

HDA with feature augmentation. To boost model performance on novel classes and make CG less overfitting, we further augment the filtered proposals that have the ground truth label belonging to the novel

Table 4.4: Average precision of the model `HDA+CG`

Shots	All			Base			Novel		
	AP	AP50	AP75	AP	AP50	AP75	AP	AP50	AP75
1	30.0	45.8	32.6	39.2	59.3	42.8	2.4	5.2	1.9
2	30.4	46.6	32.9	39.2	59.3	42.8	4.1	8.7	3.4
3	30.7	47.1	33.3	39.2	59.3	42.8	5.1	10.3	4.5
5	31.0	47.7	33.5	39.2	59.3	42.8	6.4	12.8	5.8
10	31.4	48.3	34.0	39.2	59.3	42.8	8.0	15.2	7.5

categories C_n. For each Newton iteration, we first duplicate every single shot of the proposal and corresponding box features into 5 shots, and apply Gaussian noise sampled from $\mathcal{G} \sim (0, 0.1)$ to all the box features and then apply dropout with a rate of 0.5. The model is noted as `HDA+CG+aug` for short. Table 4.5 displays the model performance of `HDA+CG+aug`.

Table 4.5: Average precision of the model `HDA+CG+aug`

Shots	All			Base			Novel		
	AP	AP50	AP75	AP	AP50	AP75	AP	AP50	AP75
1	30.2	46.2	32.7	39.2	59.3	42.8	3.0	6.8	2.3
2	30.6	47.1	33.0	39.2	59.3	42.8	4.7	10.4	3.6
3	31.0	47.5	33.2	39.2	59.3	42.8	5.6	12.0	4.6
5	31.2	48.2	33.6	39.2	59.3	42.8	7.1	14.9	6.0
10	31.8	49.4	34.1	39.2	59.3	42.8	9.1	18.4	7.9

4.4.4 Running Time

For each model, we measure the converging speed by counting the time per iteration t and number of iterations n, based on which we calculate the time T needed for optimization. Besides, we measure the time needed for extracting proposals and box features before the optimization starts, in order to have an overall view regarding the time needed for model training. The time measurement is based on the experiments run on datasets sampled from random seed 0. As `TFA+SGD` has a stage for pre-training the novel weights and a stage for fine-tuning, we counted the running time for each stage, which are noted as `TFA+SGD` (novel) and `TFA+SGD` (tuning), respectively. The results can be referred in Table 4.6 and Table 4.7.

Table 4.6: Time per iteration and number of iterations in optimization

Model	t (s)					n					T (s)				
	1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
`TFA+SGD` (novel)	0.0055	0.0059	0.0053	0.0055	0.0058	500	1500	1500	1500	2000	2.75	8.85	7.95	8.25	11.60
`TFA+SGD` (tuning)	0.0067	0.0067	0.0068	0.0070	0.0071	16000	32000	48000	80000	160000	107.20	214.40	326.40	560.00	1136.00
`TFA+CG`	0.0378	0.0687	0.1024	0.1683	0.3339	100	100	100	100	100	3.78	6.87	10.24	16.83	33.39
`TFA+CG+meta`	0.0464	0.0787	0.1129	0.1808	0.3401	100	100	100	100	100	4.64	7.87	11.29	18.08	34.01
`HDA+CG`	0.0191	0.0316	0.0458	0.0737	0.1407	30	30	30	30	30	0.57	0.95	1.37	2.21	4.22
`HDA+CG+aug`	0.0211	0.0359	0.0513	0.0822	0.1589	30	30	30	30	30	0.63	1.08	1.54	2.47	4.77

Table 4.7: Running time for feature extraction and optimization

Model	T' (s)					T (s)					$T' + T$ (s)				
	1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
`TFA+SGD` (novel)	3.03	5.51	6.75	10.38	19.58	2.75	8.85	7.95	8.25	11.60	5.78	14.36	14.70	18.63	31.18
`TFA+SGD` (tuning)	8.55	16.70	23.73	39.20	78.76	107.20	214.40	326.40	560.00	1136.00	115.75	231.10	350.13	599.20	1214.76
`TFA+CG`	9.36	17.43	23.20	37.53	74.42	3.78	6.87	10.24	16.83	33.39	13.14	24.30	33.44	54.36	107.81
`TFA+CG+meta`	8.82	18.77	24.84	37.72	76.87	4.64	7.87	11.29	18.08	34.01	13.46	26.64	36.13	55.80	110.88
`HDA+CG`	16.22	26.59	38.00	61.80	122.55	0.57	0.95	1.37	2.21	4.22	16.79	27.54	39.37	64.01	126.77
`HDA+CG+aug`	13.79	26.84	40.74	59.69	132.79	0.63	1.08	1.54	2.47	4.77	14.42	27.92	42.28	62.16	137.56

Chapter 5

Discussion

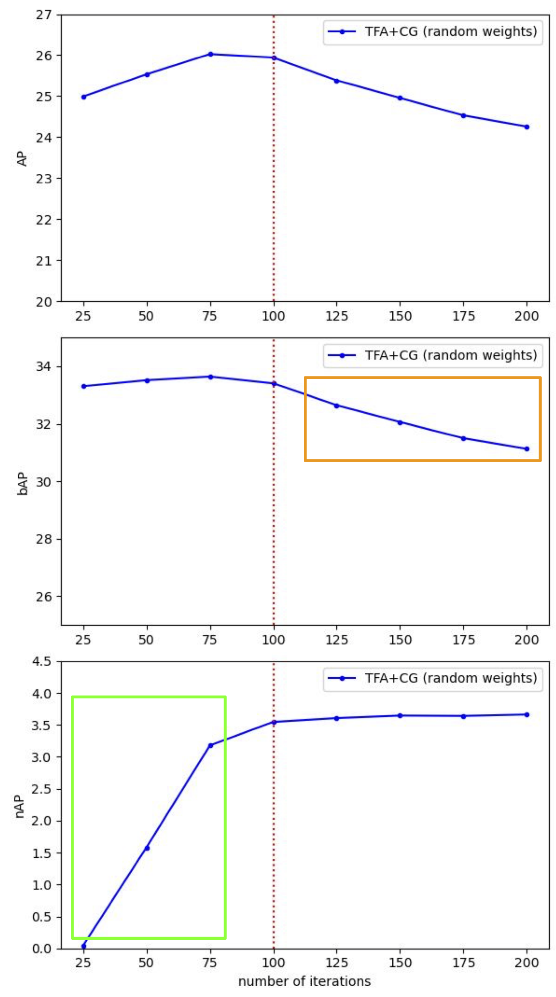
5.1 Benchmarks

For comparison, we mainly cite the previous benchmarks [38, 23, 12] that apply the setting of generalized few-shot object detection (Table 5.1), where AP and bAP are given besides nAP to have a comprehensive view of model performance and the results are run on multiple groups of random seeds were given to reduce the variation of results on single seed. Specifically, we compare the two TFA models developed by Wang *et al.* [38]: `TFA w/fc` and `TFA w/cos`, where the former applies the fully connected layer in the classification layer $\mathcal{C}$ and the later applies the cosine similarity in $\mathcal{C}$. Besides, they provide the benchmark that fully fine-tune the Faster-RCNN detector, noted as `FRCN+ft-full`. We also compare with the metrics reported in `LEAST` [23] and `Retentive R-CNN` [12]. For converging speed, we mainly compare our models with `TFA+SGD`, as both `LEAST` and `Retentive R-CNN` follow the tuning approach of TFA.

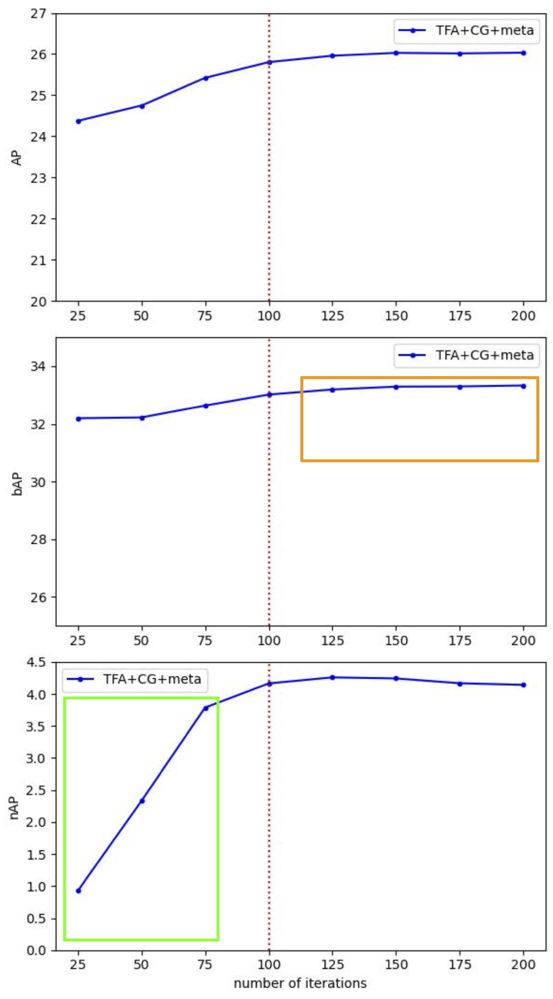
5.2 Model Performance in Average Precision

For models in TFA family, our implementation of SGD in TFA (`TFA+SGD`) is roughly on par with the original implementation of `TFA w/cos`. Its metrics are slightly lower than the one of `TFA w/cos` with less than 1 point, probably due to the fact that we extracted all the proposals at one time while in the original implementation, different groups of images were batched for every iteration to extract the proposals, the randomness of which could make the model less overfitting. Our naive CG implementation in TFA, `TFA+CG`, achieves performance on par with `TFA w/cos` with much less iterations. Especially for the fewer-shot case (e.g. 1-shot), `TFA+CG` outperforms the original `TFA w/cos` much (2.6 vs. 1.9 on 1-shot dataset).

For `TFA+CG+meta` where the meta learned modules are applied, we observe that for the 1-shot case, it has a performance on par with `TFA+CG`. To further verify that whether the meta learned modules are effective in the 1-shot case, we run the `TFA+CG+meta` for 200 iterations on the dataset sampled in seed 0 and compared it with `TFA+CG` initialized with random weights (Figure 5.1). Compared to the `TFA+CG` with randomly initialized weights, `TFA+CG+meta` with the generated weights from $\mathcal{N}$ reaches higher nAP in the beginning iterations (highlighted by the green box), which suggests that the $\mathcal{N}$ has learned a good guess of the initial weights that can speed up the convergence. With more iterations (e.g. from 100 to 200 iterations), we observe that bAP of `TFA+CG` without regularization has a drop of about 2 points while the bAP of `TFA+CG+meta` is steady (highlighted by the orange box). This indicates that the regularizer λ has effectively learned the sensitivity of features to different classes and regularizes the weight updates accordingly. However, in the case with more shots, the meta-learned modules harm the model performance on novel classes although the bAPs are well retained. As the weight generator and regularizer were meta learned in the 1-shot case, the results imply that these modules are not feasible for the case of higher shots, for which we might need to meta learn the modules again.



(a) TFA+CG



(b) TFA+CG+meta

Figure 5.1: Influence of meta learned modules on TFA+CG for 1-shot COCO dataset (seed 0)

Figure 5.2 displays the evaluation of convergence of other models (TFA+SGD, HDA+CG, HDA+CG+aug) on COCO 1-shot dataset sampled from seed 0. Compared to the baseline TFA+SGD, it can be observed that the HDA models achieve significantly higher nAP with much less iterations. To further evaluate the influence the feature augmentation on HDA, we run the experiments of HDA with 40 iterations, which have 10 additional iterations than the one we use to run the benchmarks. With more iterations, we observe that there is a slight decrease on nAP in HDA+CG, while the nAP of HDA+CG+aug converges to a higher value and is more steady after convergence, indicating that feature augmentation can not only boost the performance of HDA on novel classes, but also make it less overfitting.

Our HDA model with feature augmentation, HDA+CG+aug, achieves the overall AP at the top among all benchmarks (Table 5.1). With the base detector being fixed, it has a bAP exactly the same as the pre-trained model without any forgetting and it also achieves competitive metrics on nAP. Specifically, its performance on base classes is on par with the competitive `Retentive R-CNN`. Compared to `Retentive R-CNN`, our HDA approach, however, is much simpler without any auxiliary modules for knowledge distillation. It also has much fewer parameters to optimize, where only a detector head for novel classes needs to be trained. This also makes the HDA extremely fast in optimization, which we will discuss in the following section. For performance on novel classes, the benchmark `LEAST` is the best and nAP of our `HDA+CG+aug` is comparable with the second-place `Retentive R-CNN`. But in `LEAST`, more layers are fine-tuned, which can be expected to boost the nAP, however, results in more overfitting and have the bAP dropped accordingly. As can be observed, even with the examplar being applied to retain the performance on base classes, bAP of `LEAST` is just on par with TFA models and is still inferior to HDA models.

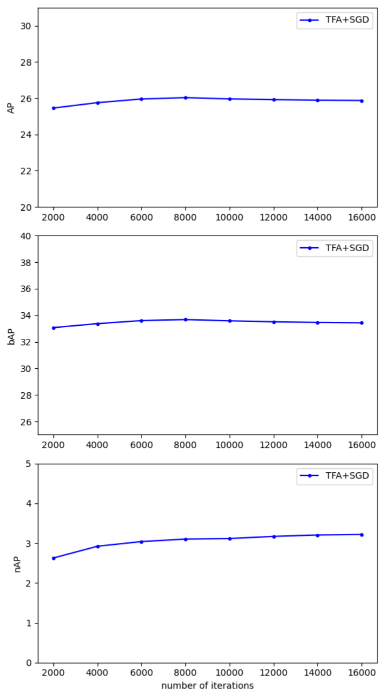
Table 5.1: Performance of different models in generalized few-shot object detection

Model	AP					bAP					nAP				
	1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
`FRCN+ft-full`	16.2	15.8	15.0	14.4	13.4	21.0	20.0	18.8	17.6	16.1	1.7	3.1	3.7	4.6	5.5
`TFA w/fc`	24.0	24.5	24.9	25.6	26.2	31.5	31.4	31.5	31.8	32.0	1.6	3.8	5.0	6.9	9.1
`TFA w/cos`	24.4	24.9	25.3	25.9	26.6	31.9	31.9	32.0	32.3	32.4	1.9	3.9	5.1	7.0	9.1
`TFA+SGD` (Our Impl.)	24.1	24.7	25.0	25.7	26.1	31.5	31.7	31.6	31.8	31.6	2.3	3.8	4.9	6.4	8.3
`LEAST`	-	-	-	-	-	29.5	-	-	31.3	31.3	4.2	-	-	9.3	12.8
`Retentive R-CNN`	-	-	-	31.4	31.8	-	-	-	39.3	39.2	-	-	-	7.7	9.5
`TFA+CG` (Ours)	24.2	25.3	26.2	27.3	28.5	31.4	32.4	33.3	34.3	35.2	2.6	4.0	4.9	6.4	8.5
`TFA+CG+meta` (Ours)	24.3	25.3	25.9	26.4	26.9	31.6	32.6	33.3	33.8	34.2	2.3	3.3	3.8	4.5	4.8
`HDA+CG` (Ours)	30.0	30.4	30.7	31.0	31.4	39.2	39.2	39.2	39.2	39.2	2.4	4.1	5.1	6.4	8.0
`HDA+CG+aug` (Ours)	30.2	30.6	31.0	31.2	31.8	39.2	39.2	39.2	39.2	39.2	3.0	4.7	5.6	7.1	9.1

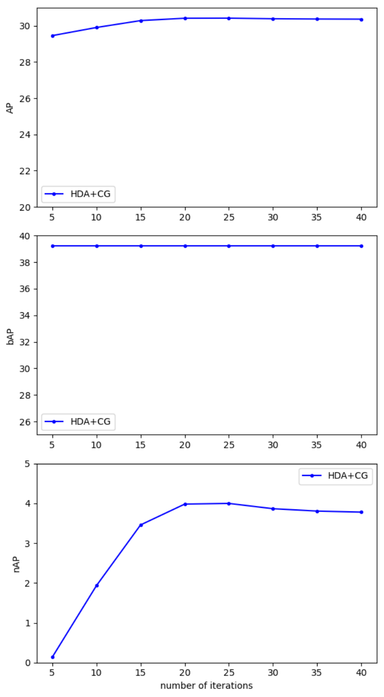
5.3 Model Performance in Convergence Speed

Regarding the time needed for convergence (Table 4.6 and Table 4.7), compared to `TFA+SGD`, the running time for a single iteration of `TFA+CG` is longer than SGD, as CG iteratively approximates the second-order optimization at every step. However, benefiting from the steep gradient, `TFA+CG` can reach convergence for all datasets within 100 iterations, while for SGD the number of iterations is much larger and it significantly increases with the size of the dataset. Overall, `TFA+CG` is about 30 times faster compared to `TFA+SGD` regarding time needed for convergence (T), and the time needed for feature extraction are close (T'). Additionally, the original TFA with SGD first pre-trains the weights for novel classes, which brings additional time costs, while CG with the steepest gradient optimizes the model weights from scratch and can delineate comparable results with much less time.

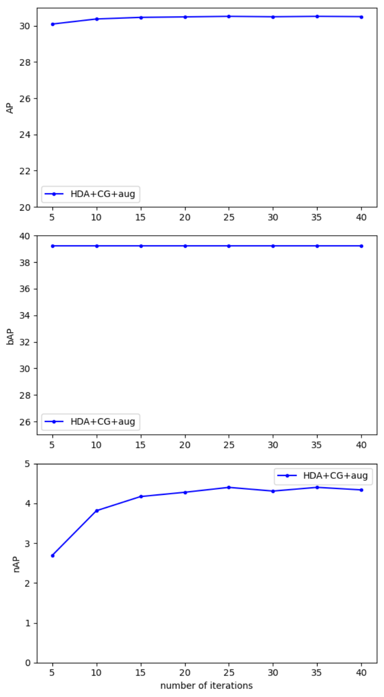
Overall, the models in the HDA family are fastest, compared to `TFA+SGD`, `HDA+CG` is about 200 to 250 times faster regarding the time needed for convergence (T). Compared to TFA, HDA has much fewer parameters to optimize as aforementioned, which also helps to speed up the optimization. For instance, the time per iteration of `HDA+CG` is almost halved compared to the one of `TFA+CG`, and it can reach convergence for datasets with different shots with only 30 iterations. The converging speed of `HDA+CG+aug` is slightly slower than `HDA+CG` due to the feature augmentation at every iteration. Regarding the time needed for feature extraction, HDA takes a longer time than TFA due to the process of proposal filtering. Table 5.2



(a) `TFA+SGD`



(b) `HDA+CG`



(c) `HDA+CG+aug`

Figure 5.2: Convergence of different models on 1-shot COCO dataset (seed 0)

compares the overall time needed by HDA and TFA. For `TFA+SGD`, we summarize the time in both stages of pretraining novel weights and fine-tuning. With the time for feature extraction taken into account, `HDA+CG` is about 7 to 10 times faster than `TFA+SGD`.

Table 5.2: Overall running time of different models

Model	$T' + T$ (s)				
	1	2	3	5	10
`TFA+SGD`	121.53	245.46	364.83	617.83	1245.94
`HDA+CG`	16.79	27.54	39.37	64.01	126.77
`HDA+CG+aug`	14.42	27.92	42.28	62.16	137.56

Chapter 6

Conclusion

6.1 Summary

In this work, we tailor a steep gradient optimization for few-shot object detection based on Conjugate Gradient. Compared to traditional gradient descent, CG performs 30 times faster in optimization when applied to the previous benchmark of the two-stage tuning approach TFA. Additionally, we consider task-level knowledge that can refine the CG in TFA to make it even faster while minimizing overfitting. Experiment shows that in the 1-shot case for which the meta-learning was conducted, the predicted weight can make CG faster in beginning iterations and the feature-wise regularizer can effectively retain model performance on base classes. Furthermore, we propose a novel hierarchical detection approach (HDA) to address the issue of catastrophic forgetting. We observe that HDA can completely retain the model performance on base classes while achieving competitive results on novel classes. HDA's optimization is 200 to 250 times faster than SGD in TFA and overall run-time is 7 to 10 times faster than the previous benchmarks. We conducted extensive experiments on datasets sampled from different groups of random seeds and measure the model performance comprehensively. Our approach helps release the potential of few-shot object detection in real-time applications while without the worries of catastrophic forgetting.

6.2 Outlook

Future works that are well worth exploring include:

Hierarchical detection setting. Following the hierarchical detection approach proposed in the thesis, we observe a novel experimental setting of hierarchical detection that has not been explored in the existing literature. Perceiving objects hierarchically is natural for humans. For instance, when annotating the images in the COCO dataset, the researchers asked the labors to first decide the super-category of the object to annotate as it is more time-efficient [24]. With the HDA, we could further experiment the setting to first detect the super-category of an object and then decide its child class.

Incremental learning. Incremental learning is a common setting in robotics, where only the novel data is available and the robot needs to learn it without seriously forgetting learned knowledge [42]. Given the fast convergence of CG and the merit of HDA without catastrophic forgetting, it would be interesting to test our model in this setting.

Efficient feature extraction. In `TFA+SGD`, the time needed for convergence is dominant compared to the time needed for feature extraction. But with CG being applied, in `HDA+CG`, the time needed for convergence is largely shortened while the time needed for feature extraction becomes dominant. Researchers can look into a more efficient way to extract the proposals and box features to further speed up the overall time needed to run the model.

Bibliography

[1] Luca Bertinetto, Joao F Henriques, Philip HS Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. *arXiv preprint arXiv:1805.08136*, 2018.

[2] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. Learning discriminative model prediction for tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6182–6191, 2019.

[3] Goutam Bhat, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. Learning what to learn for video object segmentation. In *European Conference on Computer Vision*, pages 777–794. Springer, 2020.

[4] Léon Bottou. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer, 2012.

[5] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision*, pages 213–229. Springer, 2020.

[6] Hao Chen, Yali Wang, Guoyou Wang, and Yu Qiao. Lstd: A low-shot transfer detector for object detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[7] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[8] Martin Danelljan, Goutam Bhat, Fahad Shahbaz Khan, and Michael Felsberg. Atom: Accurate tracking by overlap maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4660–4669, 2019.

[9] Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*, 2019.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4013–4022, 2020.

[12] Zhibo Fan, Yuchen Ma, Zeming Li, and Jian Sun. Generalized few-shot object detection without forgetting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4527–4536, 2021.

[13] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017.

[14] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018.

[15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1904–1916, 2015.

[19] Hanzhe Hu, Shuai Bai, Aoxue Li, Jinshi Cui, and Liwei Wang. Dense relation distillation with context-aware aggregation for few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10185–10194, 2021.

[20] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11719–11727, 2019.

[21] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8420–8429, 2019.

[22] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[23] Pengyang Li, Yanan Li, and Donghui Wang. Class-incremental few-shot object detection. *arXiv preprint arXiv:2105.07637*, 2021.

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[26] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018.

[27] Juan-Manuel Perez-Rua, Xiatian Zhu, Timothy M Hospedales, and Tao Xiang. Incremental few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13846–13855, 2020.

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[29] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[30] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015.

[32] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[33] Jake Snell, Kevin Swersky, and Richard S Zemel. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*, 2017.

[34] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

[35] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1199–1208, 2018.

[36] Ardhendu Shekhar Tripathi, Martin Danelljan, Luc Van Gool, and Radu Timofte. Few-shot classification by few-iteration meta-learning. *arXiv preprint arXiv:2010.00511*, 2020.

[37] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638, 2016.

[38] Xin Wang, Thomas E Huang, Trevor Darrell, Joseph E Gonzalez, and Fisher Yu. Frustratingly simple few-shot object detection. *arXiv preprint arXiv:2003.06957*, 2020.

[39] Xin Wang, Fisher Yu, Ruth Wang, Trevor Darrell, and Joseph E Gonzalez. Tafe-net: Task-aware feature embeddings for low shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1831–1840, 2019.

[40] Jiaxi Wu, Songtao Liu, Di Huang, and Yunhong Wang. Multi-scale positive sample refinement for few-shot object detection. In *European Conference on Computer Vision*, pages 456–472. Springer, 2020.

[41] Gongjie Zhang, Zhipeng Luo, Kaiwen Cui, and Shijian Lu. Meta-detr: Few-shot object detection via unified image-level meta-learning. *arXiv preprint arXiv:2103.11731*, 2021.

[42] Wang Zhou, Shiyu Chang, Norma Sosa, Hendrik Hamann, and David Cox. Lifelong object detection. *arXiv preprint arXiv:2009.01129*, 2020.

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
	Yihang She

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.