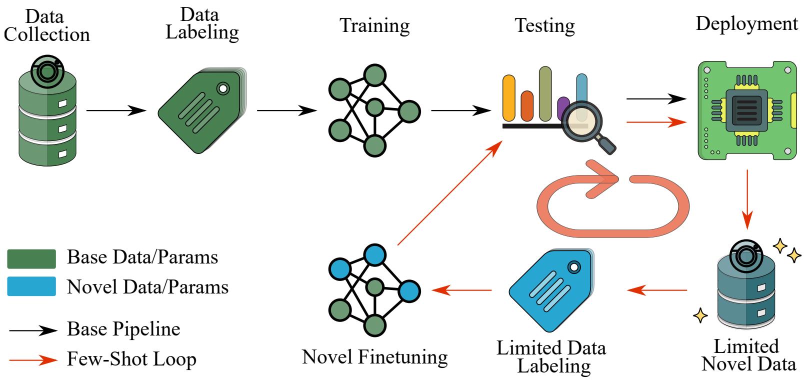
Karim Guirguis

Towards Learning Object Detectors with Limited Data for Industrial Applications



Schriftenreihe Automatische Sichtprüfung und Bildverarbeitung | Band 23

KIT Scientific Publishing

Karim Guirguis

Towards Learning Object Detectors with Limited Data for Industrial Applications

Schriftenreihe Automatische Sichtprüfung und Bildverarbeitung
Band 23

Herausgeber: Prof. Dr.-Ing. habil. Jürgen Beyerer

Lehrstuhl für Interaktive Echtzeitsysteme
am Karlsruher Institut für Technologie

Fraunhofer-Institut für Optronik, Systemtechnik
und Bildauswertung IOSB

Towards Learning Object Detectors with Limited Data for Industrial Applications

by
Karim Guirguis

Scientific Publishing

Karlsruher Institut für Technologie
Lehrstuhl für Interaktive Echtzeitsysteme

Towards Learning Object Detectors with Limited Data
for Industrial Applications

Zur Erlangung des akademischen Grades eines Doktors der Ingenieur-wissenschaen (Dr.-Ing.) von der KIT-Fakultät für Informatik des Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von Karim Guirguis, M.Sc.

Tag der mündlichen Prüfung: 28. Juni 2024
Erster Gutachter: Prof. Dr.-Ing. habil. Jürgen Beyerer
Zweiter Gutachter: Prof. Dr.-Ing. Bin Yang

Impressum

KIT Scientific Publishing

Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

Print on Demand 2024 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 1866-5934
ISBN 978-3-7315-1389-6
DOI 10.5445/KSP/1000174849

Abstract

Artificial Intelligence (AI) has been disrupting the manufacturing sector, completely reshaping how industries operate. These changes have resulted in cost reductions of approximately up to 20% and revenue increases of up to 10% in different industries. AI can empower computer vision systems to efficiently process and interpret vast volumes of data from production lines. This capability allows these systems to spot patterns, analyze and predict process behavior, and detect anomalies in real-time during production processes, among other functions. A pivotal component of these modern perception systems is an object detection model capable of accurately localizing and classifying known objects in diverse environmental settings. Nevertheless, scaling detection models to new products necessitates learning previously unseen classes during the training phase, calling for extensive data collection and labeling efforts. To address this challenge, Few-Shot Object Detection (FSOD) aims to learn new classes from limited data, offering a potential solution.

However, current FSOD algorithms suffer from two major drawbacks. First, they are vulnerable to catastrophic forgetting while learning new classes, as they tend to forget previously learned knowledge on base classes. This phenomenon can be highly detrimental in various situations, leading to system failures or even hazardous consequences. Second, they incur high computational overhead and slow performance, hindering their deployment in real-time systems. The complexity of inference paradigms and large model capacities contributes to these limitations.

This dissertation presents three novel Generalized FSOD (G-FSOD) approaches. These strategies are devised to tackle the challenge of forgetting previously acquired knowledge while learning new classes with limited data. The first two approaches are designed to alleviate forgetting on base classes when they are still available during training on novel data. For the first framework, a novel update rule is proposed to guide the model gradients, ensuring they remain aligned with the base gradients to facilitate effective knowledge transfer. The second framework introduces a progressive object proposal refinement network that leverages aleatoric and epistemic uncertainties to learn more robust representations for old and new classes. To address the base data-free scenario, the third framework introduces a knowledge distillation approach. The key novelty of this framework lies in the design of a lightweight standalone feature generator. This generator is employed to replicate base data within the high-level feature space. This approach stands in contrast to the commonly used but more expensive and less efficient model inversion technique, involving iterative optimization procedures to reconstruct the input data. All these three frameworks adopt the Decoupled Faster R-CNN (DeFRCN) model as a base framework. The choice of DeFRCN is driven by its superior performance in FSOD while having a simpler architecture compared to its transformer-based counterparts.

In pursuit of facilitating the deployment of FSOD models on embedded computing platforms, concerted efforts have been directed toward mitigating the existing bottlenecks. In alignment with this objective, a more resource-efficient FSOD framework has been introduced. Concretely, this framework comprises four novel components: a multi-scale feature fusion, a multi-way support training strategy, multi-scale data augmentation, and an adaptive class prototyping technique.

To validate the proposed approaches, extensive qualitative and quantitative experiments have been conducted on multiple detection datasets achieving the state-of-the-art on the challenging G-FSOD benchmarks, and shedding light on their practical applicability in real-world scenarios.

In summary, this dissertation offers several valuable industry benefits: Firstly, it enables the acquisition of new product knowledge with limited data rapidly. Secondly, it reduces labeling efforts and costs by efficiently leveraging a small subset of labeled samples to annotate the rest. Lastly, it enhances overall production quality by complementing human-based activities like manual product assembly, accounting for fatigue and distractions.

Kurzfassung

Künstliche Intelligenz (KI) hat die Produktionstechnik in der Industrie bereits revolutioniert und wird diese zukünftig weitgehend verändern. Diese Transformation hat bzw. wird abhängig von der jeweiligen Branche zu Kostensenkungen von bis zu ca. 20% und Umsatzsteigerungen von bis zu ca. 10% führen. Mithilfe von KI ist es IT-Systemen nun möglich, große Datenmengen aus Produktionslinien effizient zu verarbeiten, Schlussfolgerungen daraus zu ziehen und Optimierungen durchzuführen. Diese Fähigkeiten ermöglichen es diesen Systemen, Muster in den Daten zu erkennen, Prozessverhalten zu analysieren und vorherzusagen sowie Anomalien in Echtzeit während des Produktionsprozesses zu erkennen.

Eine wichtige Rolle nimmt hierbei die Perzeption von Objekten ein, um beispielweise zuvor bereits bekannte Objekte in unterschiedlichen Umgebungen präzise zu lokalisieren und zu erkennen. Für die Skalierung solcher Perzeptionsmodelle auf neue Produkte und unbekannte Objekte und dem damit verbundenem Training von neuen und zuvor nicht bekannten Klassen werden normalerweise sehr viele annotierte Daten benötigt. Um diese Menge von benötigten Trainingsdaten zu reduzieren, können Modelle auf Basis des Few-Shot-Object-Detection (FSOD)-Ansatzes eingesetzt werden, die diese neuen Objekte bzw. Klassen anhand einer begrenzten Anzahl von Daten erlernen können.

Allerdings haben aktuelle FSOD-Algorithmen zwei Hauptnachteile. Sie sind erstens anfällig für das sogenannte „katastrophale Vergessen“ beim Erlernen neuer Klassen, da sie dazu tendieren, zuvor bereits erlerntes Wissen über die Basis-Klassen zu vergessen. Dieses Phänomen kann in unterschiedlichen Situationen zu verschiedenen Fehlern führen. Zweitens sind sie im Vergleich zu anderen Ansätzen aufgrund ihrer Komplexität und der hohen Modellkapazität viel rechenaufwändiger und weisen eine verminderte Genauigkeit auf.

In dieser Dissertation werden drei neue „Generalized-Few-Shot-Object-Detection“ (G-FSOD)-Ansätze vorgestellt, die das katastrophale Vergessen in neuronalen Netzen untersuchen. In den ersten beiden Verfahren wird dieses Vergessen über das Wissen von Basis-Klassen verringert, in dem Informationen bzw. Daten über diese Klassen während des Trainings noch vorhanden sind. Hierfür wird in der ersten Methode eine neue Gradienten-Update-Regel für den Trainingsprozess vorgeschlagen, die zum einen die Gradienten in die richtige Richtung lenkt und zum anderen sicherstellt, dass diese Gradienten zu den Gradienten aus der Basis-Klasse ähnlich ausgerichtet sind. Dies ermöglicht gleichzeitig einen effektiven Informationsaustausch im Training. Das zweite Verfahren führt ein progressives Netzwerk zur Verfeinerung von Objektvorschlägen ein, das aleatorische und epistemische Unsicherheiten nutzt, um eine robustere Repräsentation für alte und neue Klassen zu erlernen. Im dritten Ansatz wird eine „Knowledge-Distillation“ eingeführt, damit auch ohne Informationen über die Basis-Daten neue Klassen gelernt werden können. Dies wird durch einen eigenständigen und leichtgewichtigen Feature-Generator sichergestellt, der zugleich die Hauptidee dieses Verfahren widerspiegelt und die Basis-Daten im hochdimensionalem Feature-Raum repliziert.

Alle drei entwickelten Verfahren basieren auf dem Decoupled Faster R-CNN (DeFRCN)-Modell. Das DeFRCN wurde verwendet aufgrund seiner hervorragenden Leistungsfähigkeit und seiner einfacheren Architektur im Vergleich zu den Transformer-basierten Modellen. Außerdem stehen diese drei Ansätze im Gegensatz zur häufig verwendeten, jedoch teureren und weniger effizienten „Modellinversionstechnik“, die iterative Optimierungsverfahren zur Rekonstruktion der Eingangsdaten nutzt.

Damit solche FSOD-Modelle auch auf eingebetteten Rechenplattformen eingesetzt werden können, wurden in dieser Arbeit zusätzlich Methoden untersucht, um vorhandene Bottlenecks in diesen FSOD-Architekturen zu reduzieren. Dazu wurde ein ressourceneffizienteres FSOD-Modell entwickelt, das aus folgenden vier Hauptkomponenten besteht: Eine Multiskalen-Feature-Fusion, eine Multi-Way-Support-Trainingstrategie, eine Multiskalen-Datenaugmentation und eine adaptive Klassen-Prototyping-Technik.

Am Schluss dieser Arbeit werden die vorgestellten Verfahren validiert und umfangreiche qualitative und quantitative Experimente an mehreren unterschiedlichen Datensätzen durchgeführt. In diesen Experimenten erzielen diese drei Verfahren state-of-the-art Ergebnisse in den FSOD-Benchmarks und können somit auch in vielen praktischen Anwendungen in realen Szenarien und im industriellen Umfeld eingesetzt werden.

Acknowledgements

I am filled with an overwhelming sense of gratitude as this life-changing journey is coming to an end. This Ph.D. would not have been possible without the guidance, support, and love of numerous individuals who have significantly shaped my academic and personal growth. I extend my heartfelt appreciation to each and every one of you.

First, I express my deepest gratitude to God for granting me strength, guidance, and support throughout this pursuit. Your blessings and grace have been my guiding light, giving me the perseverance to overcome challenges, keep moving forward, and find solace in moments of doubt.

To my family, Medhat Fahmy, Hoda Riad, Sameh Medhat, Riad Nasr, Ragaa Luka, who have been my unwavering support pillars, I cannot express how grateful I am for your love and sacrifices, which have paved the way for me to be who I am today. Also, to my nephew Mourad and his wonderful mother, Mariam Botros.

I extend my sincere appreciation to my friends: Mark Magdy, George Bassem, Kirolos Sami, Fady Aziz, Marina Hani, Sandra Henry, Clara Sameh, Botrous Malati, Thomas Zaki, Emanuella Medhat, Mina Alfred, John Philipp, Sara Ghattass, Daniel Riad, Mina Saleeb, Nardeen Michel, Nader Yosif, Carol Shahir, Mariam Riad, Mireille Nawar, Andrew Kamal, Nardine Sameh, Mina Nagui, Sandy Khouzam, Youssef Eskaros, Mirna Samir, Carla Wassef, Sherif Abdulatif, Youssef Ghaly, Monika Markos, Fady Farid, Laura Ashraf, Ramez Ramsis, Rana Mabrouk, Micheal Masloub, Mirna Magdi, and all who have stood by me throughout this journey. Your friendship, support, laughter, and encouragement have made this experience memorable. Thank you for understanding and sharing both my hardships and joys.

I am indebted to Prof. Jürgen Beyerer, Dr. Ing. Eduardo Monari, Dr. Ing. Matthias Kayser, Dr. Ing. Robin Gruna, and all my fellow PhDs and colleagues at Robert Bosch Corporate Research for their invaluable guidance and expertise. Your commitment to excellence and dedication to nurturing intellectual curiosity has impacted my academic and professional development. I am grateful for the countless hours spent providing feedback and pushing me to think critically, allowing me to refine my ideas and contribute meaningfully to my field.

To all who have contributed to this endeavor, whether big or small, know that your impact on my life has been immeasurable. Your kindness, encouragement, and belief in me have given me the strength to reach this milestone, and I aspire to pay it forward to those who set forth on their paths of exploration.

Contents

List of Symbols

General notation

Scalars	italic Roman and Greek lowercase letters	x, α
Vectors	bold Roman lowercase letters	$\boldsymbol{t}$
Matrices	bold Roman uppercase letters	$\boldsymbol{R}$
Random variables	italic Roman uppercase letters	E
Multi-dimensional random variables	bold italic Roman uppercase letters	$\boldsymbol{E}$

Distributions

$\mathcal{N}$	Gaussian normal distribution

Numbers and indexing

$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real numbers
$\mathbb{Z}$	set of integer numbers
$\mathbb{Z}^+$	set of positive integer numbers
W, H	width and height
t	discrete point in time

i, j, m, n, ℓ	indexing for images, feature maps and layers

Geometry Symbols

x, y, z	world coordinates
u, v	image coordinates
Δu	displacement in the image
l, w, h	length, width, height
$\boldsymbol{p}$	point in 2D and 3D space

Calligraphic Symbols Symbols

$\mathcal{X}$	image space
$\mathcal{Y}$	label space
$\mathcal{D}$	dataset
$\mathcal{D}_b$	base dataset
$\mathcal{D}_n$	novel dataset
$\mathcal{S}$	support set
$\mathcal{Q}$	query set
$\mathcal{C}$	set of classes
$\mathcal{C}_b$	set of base classes
$\mathcal{C}_n$	set of novel classes
$\mathcal{L}$	loss function
$\mathcal{L}_{cls}$	classification loss function
$\mathcal{L}_{loc}$	localization loss function
$\mathcal{L}_{\text{RCNN}}$	R-CNN multi-task loss function
$\mathcal{L}_{\text{RPN}}$	RPN loss function
$\mathcal{L}_{\text{det}}$	overall detection loss function

Greek Symbols

$\boldsymbol{x}$	perceptron input vector
$\boldsymbol{w}$	perceptron weight vector
$\boldsymbol{b}$	perceptron bias vector
$\hat{\boldsymbol{y}}$	neural network predictions
$\boldsymbol{y}$	ground-truth labels
$\boldsymbol{\beta}$	learnable shifting parameter
γ	learnable scaling parameter
$\boldsymbol{\mu}$	mean vector
$\boldsymbol{\mu}_c$	mean class prototype
$\boldsymbol{\sigma}$	variance vector
$\boldsymbol{W}$	neural network weight matrix
$\boldsymbol{X}$	input image
$\boldsymbol{F}$	intermediate feature map
$\boldsymbol{K}$	convolutional kernel
$\boldsymbol{\Theta}$	neural network parameters
p	convolution padding
s	convolution stride
P_H, P_W	pooling height and width
R	number of the regions of interest
C	total number of classes
$\boldsymbol{p}$	classification probabilities
φ	activation function
f	continuous function

Roman Symbols

$\mathbf{b}$	bounding boxes
$\mathrm{b}^x, \mathrm{b}^y, \mathrm{b}^w, \mathrm{b}^h$	bounding box top-left coordinates, width, and height

Operators

$\mathbb{1}_A$	$\mathbb{I}_A(x) = \begin{cases} 1 & \text{if } A(x) \text{ is true} \\ 0 & \text{if } A(x) \text{ is false} \end{cases}$
$\odot$	Hadamard Product

1 Intoduction

1.1 Motivation

Artificial Intelligence (AI) has advanced remarkably over the past decade, revolutionizing various industries and pushing the boundaries of modern machines' capabilities and learning potential. As a result, Machine Learning (ML) has become a widely embraced subfield of AI, enabling machines to learn from data instead of relying on explicit programming instructions. However, ML often involves manual feature engineering and shallow learning algorithms to detect patterns in input data, rendering this method inadequate in capturing intricate representations. To address this limitation, an AI subfield known as Deep Learning (DL) has arisen, which leverages Neural Networks (NNs) to automatically learn complex representations from input data without handcrafted features. However, it does come with the caveat of demanding greater computational resources and larger amounts of data than traditional ML approaches.

In recent years, DL has inspired more significant leaps in Computer Vision (CV) and Natural Language Processing (NLP), enabling what was previously thought impossible. For example, with the recent advances in Large Language Models (LLMs) and the release of GPT-3 [Bro20], AI has been a central topic of conversation on the potential prospects and capabilities in the near future. Another area that stands out for its significant impact is the development of autonomous robotics (e.g., a pick-and-place robot) and industrial applications (e.g., production parts detection). Integrating ML and DL techniques into robotics has facilitated automation, improved operational efficiency, increased productivity, enhanced manufacturing quality, and reduced costs. However, it is important to note that human involvement remains indispensable across various sectors, yet more prone to errors than machines. For instance, a worker can miss a part during a product assembly due to distractions (e.g., sudden loud noises and loss of concentration). This gives rise to a wide range of human-machine complementaries where machines can help alleviate as many human errors as possible. In the current era of the Industry 4.0 revolution, automation, connectivity, real-time data, and AI come together, forming a new ecosystem for smart and efficient automation.

Despite the many successes of DL in such systems, it faces several hurdles. The key challenges and limitations of the training and deployment of DL models can be summarized as follows:

- **Data-hungry:** Training DL models require abundant labeled training data to generalize to unseen examples effectively. However, data acquisition and labeling can be demanding in terms of time, cost, and labor. It becomes more challenging when dealing with niche domains (e.g., industrial products) or infrequent occurrences.
- **Computationally and memory intensive:** DL models typically require significant computational resources, such as high-performance Graphics Processintg Units (GPUs) and substantial memory, to compute and save the model weights for training and inference. This can present difficulties when deploying trained models on embedded hardware.
- **Catastrophic forgetting:** When a model is presented with new data to learn, the DL models can forget the previously trained classes, causing an overall deterioration in performance. This can be problematic for safety-critical systems. Moreover, new data may necessitate retraining the model, which may consume time and resources.
- **Societal and ethical concerns:** Integrating data-driven solutions raises ethical and societal concerns regarding bias and privacy. Data-driven models can potentially inherit biases in the training dataset, resulting in unfair or discriminatory results. Moreover, datasets containing various faces of people and their data give rise to notable privacy concerns.

This dissertation adopts Object Detection (OD) as its primary task while addressing the abovementioned challenges. OD is central to modern perception systems utilized in commercial and industrial autonomous robots (e.g., self-driving cars and pick-and-place robots). The main objective is to classify and localize objects of interest in the input data (e.g., images, videos, or point clouds). This research focuses mainly on the 2D OD task on Red-Green-Blue (RGB) images due to their widespread usage across diverse applications. More specifically, this work adopts a recent thriving field that learns detectors with limited data, namely Few-Shot Object Detection (FSOD).

FSOD strives to rapidly adapt detectors trained on base classes with abundant data to learn novel classes with scarce data. However, most FSOD approaches focus on improving the novel detection performance and tend to overlook the issue of catastrophic forgetting, which refers to the tendency of a DL model to forget how to detect objects from the base categories. This aspect holds significant importance in various practical applications, such as pick-and-place tasks, where a robot must be capable to operate new objects without losing its ability to handle previously known ones. To overcome this limitation, Generalized Few-Shot Object Detection (G-FSOD) emerges as a specialized branch within FSOD. G-FSOD focuses on training models to jointly detect both base and novel classes, enhancing their robustness and practicality for addressing real-world scenarios.

To this end, the following research questions are raised:

- How can existing detectors effectively adapt to learn new classes when presented with limited labeled data?
- To what extent can prior knowledge be harnessed to enhance the adaptation process of few-shot detectors?
- What strategies or techniques can be devised to enable detectors to learn and accommodate novel classes without causing significant forgetting or performance deterioration on previously learned base classes?

1.2 Contributions

The main focus of this dissertation is to tackle the three challenges mentioned above in the context of OD. Specifically, the main aim is to design and evaluate object detection frameworks that leverage prior knowledge from abundant data to rapidly learn new classes when presented with limited data. The main contributions of this dissertation can be summarized as follows:

- A Constraint-based Finetuning Approach (CFA) [Gui22b] for G-FSOD to alleviate base forgetting. CFA provides a new gradient update rule that adaptively reweights the base and novel gradients and guides them toward less forgetting and more effective knowledge transfer. CFA can be integrated with different frameworks plug-and-play without model capacity or inference time overhead.
- The Uncertainty-based Progressive Proposal Refinement (UPPR) approach leverages predictive uncertainties (i.e., aleatoric and epistemic uncertainties) to alleviate forgetting in G-FSOD while improving the detection of the novel classes. Specifically, UPPR progressively refines the object proposals via the estimated uncertainties in a stage-wise manner. Moreover, attention blocks are utilized in each stage to focus on discriminative features selectively. A new architecture is introduced to provide sufficient learning capacity, namely Decoupled Cascaded R-CNN (DeCRCN).
- Neural Instance Feature Forging (NIFF) [Gui23b] presents the first data-free G-FSOD framework that alleviates forgetting without using base data. This is particularly beneficial when sharing and storing data is problematic due to privacy or memory constraints. NIFF proposes a standalone feature generator with a negligible memory footprint and learns to generate base instance-level features by aligning class-specific statistics. During novel finetuning, these forged features are replayed, along with thoughtful design considerations in the training pipeline.
- Few-Shot RetinaNet (FSRN) [Gui23a] framework as a more embedded-friendly meta-learning-based one-stage detector for FSOD,

reducing the computational and memory requirements of the system. FSRN introduces four novel components, including a multi-way support training strategy, multi-scale feature fusion, multi-scale data augmentation, and a weighted class feature prototyping approach. Compared to the current state-of-the-art one-stage meta detectors, FSRN provides an embedded-friendly solution with significantly reduced parameters, Floating Point Operations Per Second (FLOPS), and inference time.

- The Zero-Shot Domain Adaptive FSOD (ZDA-FSOD) [Gui22a] presents a framework to tackle the proposed problem, identifying new objects in a target domain using only limited data from a source domain. ZDA-FSOD proposes a new contrastive loss function and feature-level augmentations to encourage the learning of domain-agnostic class-specific feature embeddings that are less sensitive to the potential domain shifts. This approach is presented in the appendix of this dissertation.

Extensive experiments on two well-established publicly available datasets and ablation studies are conducted for each approach to illustrate the improved detection performance and the different design choices. Different from most FSOD and G-FSOD works, the results over multiple runs with different seeds are provided to validate the robustness of the proposed model. The proposed methods achieve state-of-the-art results for G-FSOD and one-stage meta-learning based FSOD.

1.3 Dissertation Structure

The subsequent sections of the thesis are organized as follows. In Chapter 2, the fundamental theoretical principles underpinning this research are presented. This includes an exploration of the foundational concepts and models in deep learning. Additionally, a thorough review of deep learning-based object detection frameworks is conducted. This review encompasses the problem formulation, detection architectures, and evaluation metrics employed throughout this thesis.

Chapter 3 of the thesis delves into FSOD fundamentals. It begins by presenting the problem formulation of FSOD and proceeds to conduct a comprehensive analysis of two primary FSOD families of approaches: transfer learning and meta-learning. For each field, this encompasses a formal introduction and an exploration of the relevant works.

Chapter 4 establishes the concept of the proposed FSOD and G-FSOD pipelines. The implications of utilizing the different proposed frameworks in real-world scenarios are identified and discussed.

Chapter 5 addresses the challenge of catastrophic forgetting in the G-FSOD task while simultaneously improving the performance of detecting novel classes. The chapter begins with a literature review of relevant works from continual learning. Subsequently, three frameworks are introduced. The first framework is called Constraint-based Finetuning Approach (CFA). CFA establishes a gradient update rule that adaptively adjusts the contribution of base and novel gradients, achieving a better optimum for effective knowledge transfer and minimizing forgetting. The second framework, Uncertainty-based Progressive Proposal Refinement (UPPR), is then presented. UPPR utilizes predictive uncertainties, including aleatoric and epistemic uncertainties. A new architecture is introduced to provide enough learning capacity, namely Decoupled Cascade R-CNN (DeCRCN).

While the frameworks above assume the availability of base data, the third framework in Chapter 6, Neural Instance Feature Forging (NIFF), is the first data-free G-FSOD pipeline. NIFF incorporates a feature generator with a negligible memory footprint. It learns to generate base instance-level features by aligning class-specific statistics. During the novel finetuning stage, these forged features are replayed to maintain knowledge of the base classes.

Chapter 7 introduces a novel one-stage meta-learning-based framework called Few-Shot RetinaNet (FSRN) for the FSOD task. This chapter also covers the datasets and evaluation metrics that will be employed throughout the thesis.

Next, the primary performance bottlenecks in a two-stage meta detector are identified and discussed. To assess the generalization capability of the proposed framework, several experimental evaluations and ablation studies are conducted on the well-established FSOD benchmarks.

Chapter 8 concludes the dissertation by presenting the final remarks and summarizing potential avenues for future research.

Finally, a challenging new task, namely Zero-Shot Domain Adaptive FSOD (ZDA-FSOD), is introduced in Appendix B. Given a handful source domain data, ZDA-FSOD aims to detect novel objects in an unseen target domain. A ZDA-FSOD framework is accordingly proposed. ZDA-FSOD employs a novel contrastive loss function and feature-level augmentations to facilitate the learning of class-specific feature embeddings that are not influenced by domain-specific characteristics. By doing so, ZDA-FSOD mitigates the sensitivity to potential domain shifts to learn transferable domain-agnostic representations.

2 Theoretical Foundations

This chapter provides an overview of the rudimentary theoretical foundations utilized throughout this dissertation. Firstly, we present the background and theory of the two pillars of DL, artificial neural networks and fully connected neural networks. Secondly, we explain the Convolutional Neural Network (CNN), which is quintessential to designing neural architectures for various Computer Vision (CV) tasks. Next, we delve into the pillar computer vision task of this dissertation, the object detection task. The main architectural designs, training paradigms, limitations, and evaluation metrics will be discussed.

2.1 Deep Learning Foundations

2.1.1 Fully Connected Neural Networks

In the 1940s, researchers began exploring the possibility of developing AI that mimics the cognitive abilities of the human brain to approach non-linear tasks that a crude linear mathematical model cannot describe in a closed-form solution. This research was motivated by the remarkable capacity of the human brain to rapidly process information, identify patterns, and solve complex tasks (e.g., classify and localize a known object in an image).

The communication unit in the brain and nervous system is the *neuron*. It receives, processes, and transfers information to and from various body regions via electrical impulses and chemical signals. When a signal, in the form of an electrical impulse, reaches the neuron's cell body, the neuron begins to fire to send the signal to the neighboring neurons. Next, the cell body activates voltage-gated ion channels, which are proteins that control the movement of

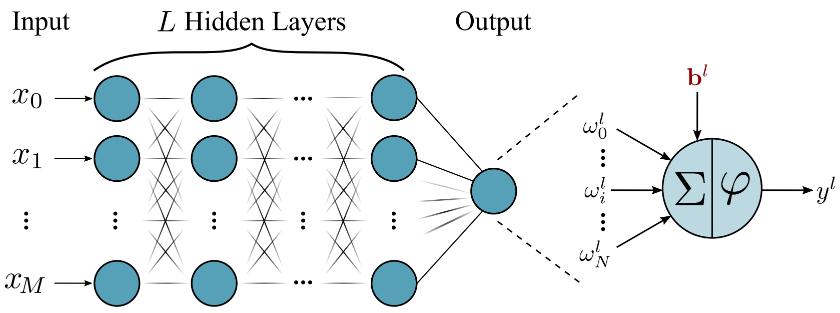


Figure 2.1: An illustration of the MLP architecture along with a zoom in on an artificial neuron.

ions (atoms or molecules with a positive or negative charge) across the cell membrane. As the voltage-gated ion channels open, positive ions flow into the cell, and negative ions flow out. The action potential, which is an electrical charge across the cell membrane, is produced by this ion movement. The release of neurotransmitters is then triggered by the action potential, which moves down the axon, an extension of the cell body. These neurotransmitters connect to receptors on the receiving end after passing through a tiny gap known as the *synapse*. This binding activates the receiving cell and initiates a new action potential in that cell, allowing the signal to be transmitted to the next cell in the chain [Day05].

In 1958, Frank Rosenblatt presented the *perceptron* as a rough representation of humans' biological neurons [Ros58]. This concept fueled interest in the AI field in creating an Artificial Neural Network (ANN) that resembles the interconnected neurons of the human brain. An ANN is a parallel computational model of interconnected perceptrons, where each link simulates the brain's synapses, transmitting signals between the artificial neurons. Each input is associated with a weight, which represents the importance of that input in determining the output. Formally, a perceptron is a non-linear mapping function that receives an input $\boldsymbol{x} \in \mathbb{R}^M$ and outputs a single scalar output $y \in \mathbb{R}$. The relative importance of the inputs to the output is represented by a weight vector $\boldsymbol{w} \in \mathbb{R}^M$. M denotes the length of the input vector. The perceptron computes the weighted sum of the inputs and passes them through a non-linear transformation function, or an activation function $\varphi : \mathbb{R} \mapsto \mathbb{R}$, which determines whether the output is a 0 or a 1 based on a threshold, or bias $b \in \mathbb{R}$, value. Formally, the output is denoted by

$$y = \varphi(\boldsymbol{w}^T \boldsymbol{x} + b). \tag{2.1}$$

However, a human brain consists of considerable interconnected perceptrons to tackle more complex tasks and problems. As a result, the Fully Connected Neural Network (FCNN) was developed to capture more complex non-linear relations between inputs and outputs. A FCNN comprises an input layer, an output layer, and intermediate hidden layers in a stacked manner, with all nodes in one layer connected to all nodes in the next. Consequently, the information flows through the network in a highly interconnected manner, allowing the network to interpret and learn from complex patterns and correlations in the incoming data. This architecture is most commonly referred to as a Multi Layer Perceptron (MLP), as presented in Figure 2.1.

Specifically, a FCNN is regarded as a parametric universal function approximator [Les93]. For a continuous function $f(\boldsymbol{x})$ on a compact area of $\mathbb{R}^M$, there exists a function, $f^*(\boldsymbol{x}; \Theta) \in \mathbb{R}^M$, with learnable parameters Θ controlling the function mapping, and non-polynomial activation functions (e.g., tanh function) that approximates $f(\boldsymbol{x})$ such that:

$$\mid f(\boldsymbol{x}) - f^*(\boldsymbol{x}; \Theta) \mid < \varepsilon, \tag{2.2}$$

where ε is an arbitrary small number of acceptable error. The output of the network can then be given as: ,

$$\hat{\boldsymbol{y}}_i = \varphi\left(\sum_{j=1}^{M} \boldsymbol{W}_{i,j} \boldsymbol{x}_j + \boldsymbol{b}_i\right), \tag{2.3}$$

where $\boldsymbol{W} \in \mathbb{R}^{N \times M}$ and $\boldsymbol{b} \in \mathbb{R}^N$ are the weight matrix and bias vector of the network, respectively. N denotes the length of the layer neurons. The network parameters are represented jointly by $\Theta = [\boldsymbol{W}, \boldsymbol{b}] \in \mathbb{R}^{N \times (M+1)}$.

Learning the parameter vectors Θ is the first step toward teaching a neural network to solve complex problems. For this, we first need a sufficiently labeled training dataset $\mathcal{D} = \{(\boldsymbol{x}_i,\boldsymbol{y}_i)\}_{i=1}^{T}$, where T is the total number of training examples. The ith example comprises an input sample $\boldsymbol{x}_i$ (e.g., an image) and the associated ground-truth target label $\boldsymbol{y}$ (e.g., class labels). Next, the *training* process is performed, where the network is presented with numerous examples of the task at hand and iteratively modifies the parameters to minimize some pre-defined cost function, or a loss function $\mathcal{L}(\cdot)$. The aim is to find the optimal parameters Θ^* via empirical risk minimization as follows:

$$\Theta^* = \arg\min_{\Theta} \frac{1}{T} \sum_{i=1}^{T} \mathcal{L}(\boldsymbol{x}_i,\boldsymbol{y}_i;\Theta). \tag{2.4}$$

Specifically, this is achieved via the *backpropagation* algorithm, which involves both a forward pass and a backward pass. The neural network is fed a training instance $(\boldsymbol{x},\boldsymbol{y})$ in the forward pass, and the intermediary outputs are stored using each layer's existing set of parameters. Next, the derivative of the utilized loss function with respect to the network's output $\hat{\boldsymbol{y}}$ is computed. On the other hand, during the backward pass, the derivative of the loss is computed with respect to the weights in all layers by leveraging the chain rule of differential calculus. Finally, these gradients traverse the network backward to update the weights at each layer. Note that this is done till the convergence criterion is satisfied or for a defined number of training epochs, with a single epoch meaning iterating once through the entire dataset.

The loss function is determined based on the learning task. For a classification task, the categorical cross-entropy (CE) loss is most commonly used:

$$\mathcal{L}_{\text{CE}} = -\sum_{n=1}^{N}\sum_{c=1}^{C} \boldsymbol{y}_{n,c} \log(\hat{\boldsymbol{y}}_{n,c}), \tag{2.5}$$

where N is the total number of predictions. C denotes the number of mutually exclusive classes. $\hat{\boldsymbol{y}}_{n,c}$ denotes the probability score of the n^{th} sample predicted by the network for the c^{th}-class. As for a regression task, Mean Squared Error

(MSE) loss function is typically utilized:

$$\mathcal{L}_{\text{MSE}} = -\sum_{n=1}^{N}\sum_{c=1}^{C}(\boldsymbol{y}_{c,n} - \hat{\boldsymbol{y}}_{c,n})^2, \tag{2.6}$$

where N is the total number of predictions. However, FCNNs do not make any prior assumptions about the structure of the features and how they relate to one another, which might impede learning from high-dimensional structured data. Since RGB images are utilized throughout this dissertation, they will be the main focus. Expressly, three main traits of images indicate the necessity for a more specialized architecture. First, images are high-dimensional data. Considering a shallow FCNN classifier network with an input of $256 \times 256 \times 3$, a single hidden layer of the same input size 196,608 already requires more than 38 billion parameters, rendering the scalability of such architectures intractable in terms of memory and computational power. Additionally, even with smaller input dimensions, unstable gradients are a risk as the network becomes deeper to extract more relevant features, tackling more challenging tasks. Either they significantly increase, which causes the gradients to explode, or they significantly decrease, which results in vanishing gradients and makes learning unfeasible since the parameters barely change with each update. Second, due to the structured nature of an image, statistical relations exist between neighboring image pixels. As mentioned above, FCNNs do not preserve any spatial structure ignoring any spatial relations between the image pixels. Third, understanding a given image should be consistent regardless of any geometric transformation (e.g., translation, rotation). However, for a FCNN, rearranging the input pixels will not yield any difference overlooking pixel patterns for a known object.

2.1.2 Convolutional Neural Networks

The late 1990s witnessed a technological leap in deep learning with the LeNet [LeC98] model recognizing handwritten digits $(0 - 9)$ in images. CNNs have thrived since then, enabling significant advancements in various computer vision applications, including image classification, object detection, and

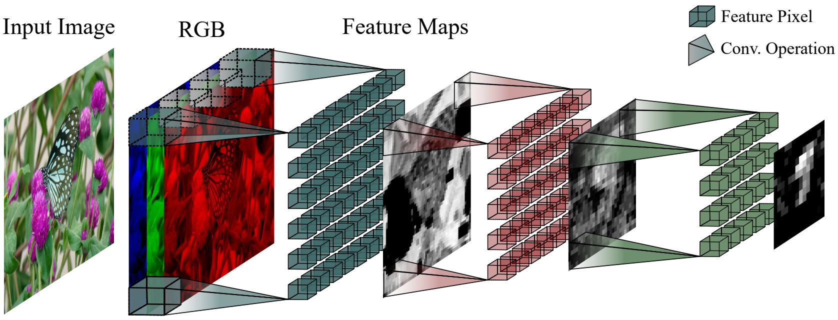


Figure 2.2: Example of a convolutional operation on an open source RGB input image.

semantic segmentation. Specifically, a CNN typically comprises a convolutional layer, a normalization layer, a pooling layer, and an activation function.

CNNs provide three essential advantages over the traditional FCNNs [Pri23]. First, they handle high dimensional data effectively because they use sparse connections rather than fully dense connections, which results in fewer parameters and computations. This is realized by using a kernel size smaller than the input. For instance, an image of size $256 \times 256 \times 3$, a 2D convolutional layer with 128 kernels of size 5×5 would require $((5 \times 5 \times 3) + 1) \times 128 = 9{,}728$ parameters only. Second, it shares parameters across the given input, promoting translation equivariance, which means that shifting or moving an object in the input will cause its representation to move in the output by the same amount. This means that for each pixel location, merely a single set of parameters is learned. While the computational runtime stays unaffected, the model's storage requirements are further reduced to the kernel size. Third, CNNs employ a vital subsampling in which the most prominent pixels are transferred to the next layer while the others are dropped. Hence, it outputs a fixed-size output matrix promoting rotation and translation invariance.

Convolutional Layer

The convolutional layer is the fundamental building block of a CNN that convolves the kernel with the given input image or feature map. Given a 2D input image $\boldsymbol{X} \in \mathbb{R}^{W \times H}$ and a 2D filter, or kernel $\boldsymbol{K} \in \mathbb{R}^{k \times k}$, the output feature map $\boldsymbol{F}$ for a given pixel (i, j) can be denoted as follows

$$\boldsymbol{F}_{i,j} = \boldsymbol{X}_{i,j} * \boldsymbol{K}_{i,j} = \sum_{m=1}^{k} \sum_{n=1}^{k} \boldsymbol{K}_{m,n} \cdot \boldsymbol{X}_{i-m,j-n} + \boldsymbol{b}, \tag{2.7}$$

where W and $H \in \mathbb{Z}^+$ are the width and height of the input image, respectively. $k \in \mathbb{Z}^+$ denotes the kernel size. However, building deep CNNs requires an essential adjustment to the convolutional operation, namely *padding*. Convolving a 224×224 image with a 3×3 kernel outputs a 222×222 feature map resulting in a loss of information, especially around the edges. This is because of the limited number of possible positions that the kernel can cover. To this end, padding appends additional p pixel borders around the image or feature map, usually with zero value. For instance, performing the padding of value $p = 1$ on the abovementioned example yields a 226×226 image resulting in a 224×224 feature map after the convolution operation, preserving the original input size. Furthermore, another fundamental concept in convolutional layers is the *stride*. It controls the amount by which the kernel shifts when convolving around the input volume. An illustration of a convolutional operation with the resulting intermediate features is depicted in Figure 2.2. Inherently, a convolution operation employs a stride $s = 1$, promoting more overlapping *receptive fields*. Note that the receptive field refers to all elements in previous layers that influence the computation of the output feature map. CNNs need broad enough receptive fields to cover objects of various shapes and sizes. Revisiting the previous example, performing a convolution with ($s = 2$, $p = 1$, $k = 3$) instead yields an output feature map of size 112×112, half the original image size. Formally, for a square input $W = H$, the output spatial dimension is governed by the following formula

$$\left\lfloor \frac{W + 2p - k}{s} + 1 \right\rfloor. \tag{2.8}$$

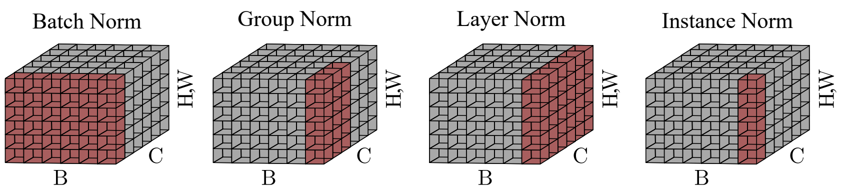


Figure 2.3: An overview of different normalization layers highlighting the main differences.

Normalization Layer

During training, the input distribution to each layer varies as the network parameters get updated, which is formally known as the *internal covariance shift* [Iof15]. With such volatility in the input distribution, the optimizer would then be more likely to hit a plateau, and the training would take longer to converge. To account for the internal covariance shift, Batch Normalization (BN) [Iof15] was introduced as a normalization method that fixes the first and second statistical moments of the hidden representations output before they are fed to the next input layer. Fixing the distribution of layer inputs allow for better gradient flow as they become less dependent on the initialization scheme or the scale of the layer parameters. A BN layer is typically placed between the convolutional and the non-linear activation function. During training, the BN computes the mean $\boldsymbol{\mu}$ and the variance $\boldsymbol{\sigma}^2$ of the input features across the current batch. Each input channel is then normalized. Finally, to acclimate the output distribution for each hidden layer, BN learns two parameters, $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$, responsible for scaling and shifting the normalized output, respectively. Intuitively, normalizing the input distribution via BN results in a less noisy learning signal and smoother activations, hence more stable training and faster convergence. Moreover, normalizing the activations across the model averts minor parameter changes from significantly altering the activation gradients. As a result, it is possible to utilize higher learning rates without risking training stagnation in saturated non-linearity regimes and gradients vanishing or exploding [Iof15]. Formally, the BN output can be denoted as

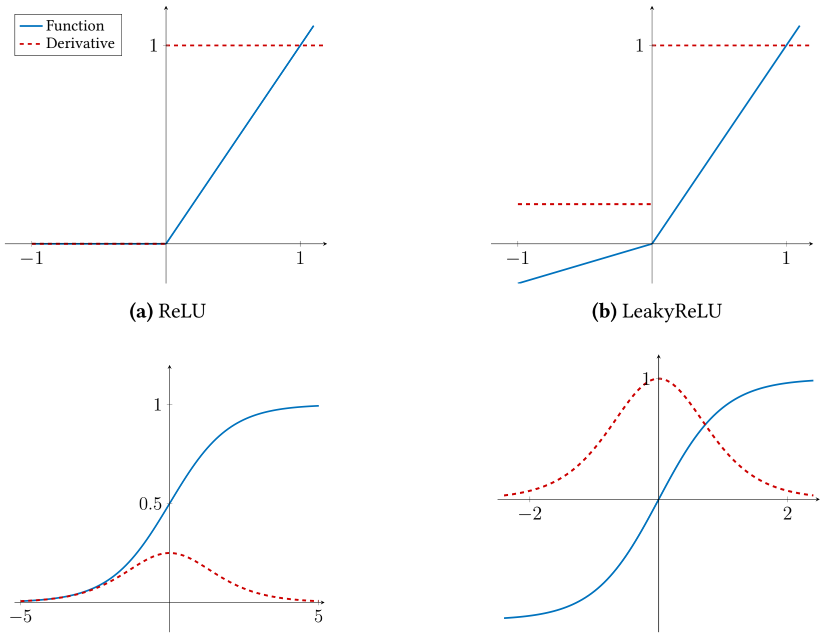
$$BN(\boldsymbol{x}_i) = \boldsymbol{\gamma}\frac{\boldsymbol{x}_i - \boldsymbol{\mu}_i}{\sqrt{\boldsymbol{\sigma}_i^2 + \varepsilon}} + \boldsymbol{\beta}. \tag{2.9}$$

ε is a significantly small value to avoid dividing by zero. During inference, BN is directly applied using the average mean and variance acquired during training.

However, a significant drawback of BN is that it becomes rapidly erroneous as the batch size decreases due to inaccurate batch statistics estimation. To overcome the inter-batch dependencies of the BN, Layer Normalization (LN) [Ba16] normalizes each mini-batch sample independently to zero mean and unit variance. Instance Normalization (IN) [Uly16], on the other hand, computes the mean and variance for each input sample across the spatial dimensions for each channel separately. As a result, instance-specific contrast information can be removed from the image, simplifying the task of stylized image generation, as was originally proposed [Uly16]. Unlike BN, both LN and IN perform the exact computations at test-time since they are independent of the batch size. Nevertheless, the channels of feature representations are not entirely independent. For example, early convolutional layers in the network learn low-level features like lines, corners, and edges. In contrast, the last layers learn more high-level features, such as complex shapes. Building upon these observations, Group Normalization (GN) [Wu18] divides the input channels into a pre-defined number of groups, divisible by the number of channels, more likely to share the same filter response. Next, GN computes the mean and variance along the spatial dimensions for each group separately. The advantages of GN are twofold: First, unlike LN, it can learn various distributions for each group of channels. Second, it scales better with smaller batch sizes than BN but may not perform as well with larger batch sizes. The differences between each normalization technique are highlighted in Figure 2.3.

Activation Functions

Without enforcing non-linear constraints, a neural network will only learn a linear transformation using each layer's weights and biases, resulting in a linear regression model. While this simplifies the neural network, it hinders extracting complex patterns from the data. To this end, activation functions are injected throughout the model to realize non-linear relations between the input and the output. In the following, the most commonly utilized activation



(a) ReLU **(b)** LeakyReLU **(c)** Sigmoid **(d)** Tanh

Figure 2.4: An illustration of the presented activation functions.

functions are presented, and a comprehensive representation of each function and its derivative are depicted in Figure 2.4.

Rectified Linear Unit. One of the most commonly used activation functions is the Rectified Linear Unit (ReLU) function, a piece-wise linear function that provides a simple non-linear transformation. For non-positive input values, the output and the function derivative are both consistently zero. Otherwise, it outputs the input value with a constant derivative of one. For an input x, a ReLU function is defined as

$$\mathrm{ReLU}(x) = \max(0,x). \tag{2.10}$$

However, the constant zero gradients for negative input values will prevent the update of relevant parameters during the gradient descent. Consequently, various neurons lose their sensitivity to input changes and vanish, a phenomenon known as the dying ReLU problem. In contrast, the output activations are unconstrained for positive values, resulting in exploding activations. Two variants of the ReLU function were proposed to address the abovementioned limitations. First, the Leaky ReLU introduces a constant slight negative slope for the non-positive region, allowing a small non-zero gradient, which overcomes the dying ReLU problem. This is realized by scaling negative activations via a constant scalar value $\alpha \ll 0$ as follows

$$\mathrm{LeakyReLU}(x) = \max(0,x) + \alpha \min(0,x). \tag{2.11}$$

Second, to allow for variable negative slope values, the Parametric Rectified Linear Unit (PReLU) extends the LeakyReLU by leveraging a learnable α instead.

Sigmoid. The sigmoid function, or a squashing function, transforms a real-valued input to a range of $(0,1)$.

$$\mathrm{sigmoid}(x) = \frac{1}{1+\exp(-x)}. \tag{2.12}$$

It is typically utilized for binary classification problems to map the output logits into probabilities. Due to the bell-shaped derivative function, the gradients moving toward the tail for large values in either direction hit a plateau, causing the gradients to vanish, posing a challenge to the optimizer.

Hyperbolic Tangent (tanh). A shifted version of the sigmoid function, namely the tanh function transforms the input into a range of $(-1,1)$.

$$\tanh(x) = \frac{1-\exp(-2x)}{1+\exp(-2x)}. \tag{2.13}$$

Compared to the sigmoid function, tanh yields higher gradients, providing a larger learning signal thus more significant parameters update. Moreover, the tanh function is symmetric around zero and hence faster convergence.

Pooling Layer

The final stage of a CNN is usually a pooling layer that spatially downsamples the output activation maps, reducing the parameters and computation needed to process the input. Analogous to the convolutional layer, the pooling operation involves a sliding window with a predefined window size and stride across the whole input region, computing a single output for each patch. However, the pooling layer has no learnable parameters calculating the average or maximum value of patch elements in the pooling window, referred to as average pooling or maximum pooling, respectively.

The benefits of a pooling layer are twofold. First, it enables learning translations invariant representations because most pooled activation map values remain unchanged even if the input marginally shifts. This means the network learns to identify the feature representation of the same object regardless of its location in the input region. Second, the resulting condensed activation map gives the subsequent layer fewer inputs by assembling a summary of statistics of each neighborhood area across the input region, which improves computational efficiency and expedites training. As a result, it becomes possible to build deeper CNNs by stacking additional layers. Expanding the receptive field allows combining earlier low-level features into increasingly higher-level features to learn robust hierarchical visual concepts.

2.1.3 Deep Feature Extractors

VGG Networks

In 2015, the VGG [Sim14] network was introduced as one of the first deep architectures for image classification and object detection tasks. The network utilizes small 3×3 convolution filters to build models with varying depths. The core idea is that by using smaller filters, the network could capture larger receptive fields while reducing the number of parameters and converging faster. The VGG architecture includes convolutional layers ranging from eight to sixteen, three fully connected layers, and a softmax layer. The most

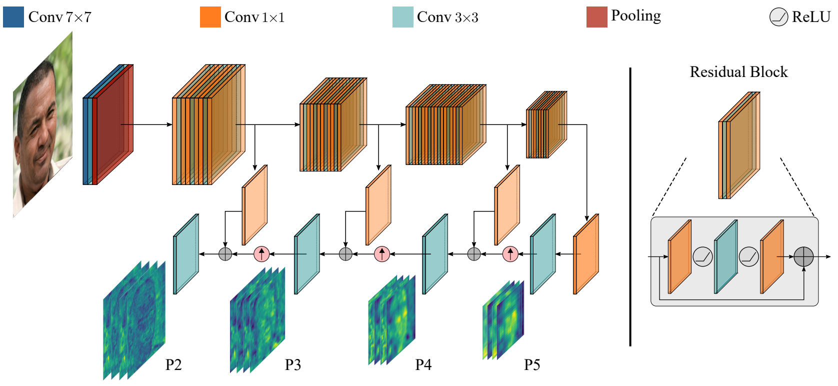


Figure 2.5: An illustration of a ResNet50 architecture with a FPN. The multiscale feature maps are visualized to show the extracted semantic information at each level.

commonly used VGG architectures are the VGG-16 and VGG-19, with 16 and 19 layers, respectively. A VGG model is normally trained in multiple iterations, where a small architecture is first trained with random initialization. Then, the weights of this network are used to train deeper networks to motivate more stable gradients.

Residual Neural Networks

However, adding more layers causes the backpropagated gradients to weaken and vanish, rendering the gradient descent excessively slow. Specifically, during backpropagation, the computed partial derivatives can get excessively small and decrease exponentially to zero, preventing weight parameters updates and thus hindering the overall training process.

To mitigate the vanishing gradient problem, He et al. proposed the Residual Neural Network (ResNet) [He16] featuring residual blocks stacked end-to-end. The key idea is to learn residual mapping by providing a shortcut for the gradients to skip over layers with non-useful information. A residual block comprises two 3×3 convolutional layers with the same number of output channels followed by a BN layer, and a ReLU activation function. Moreover, a skip connection adds the block input right before the final activation function, which results in learning a residual mapping. Figure 2.5 (right) depicts a residual block architecture. If the number of output channels differs, then a 1×1 convolutional layer is added in the shortcut connection to transform the input into the target output shape. It is important to highlight that ResNets are simpler to optimize because the deeper residual blocks learn to enhance the output of the preceding blocks rather than learning an underlying mapping from scratch.

Feature Pyramid Network

One of the main challenges that emerges is the ability to detect objects over a broad range of scales, particularly small objects. As a remedy, a Feature Pyramid Network (FPN) [Lin17] generates multi-scale feature maps from different feature extractor stages. The deeper the layer within the feature extractor, the more semantically rich the extracted feature maps, yet with a lower spatial resolution. Accordingly, FPN equips the feature extractor with a top-down pathway that learns to upscale the low-resolution feature maps by a factor of 2 and merges each with the corresponding bottom-up feature maps via element-wise addition. For instance, in a ResNet equipped with an FPN (ResNet-FPN), the bottom-up pathway is formed via the output of the last four residual blocks (R2, R3, R4, R5) featuring different strides (4,8,16,32) relative to the input image. Next, starting with the coarsest feature map, each feature map is upsampled by a factor of 2 and fed to a 1×1 convolutional layer, to reduce the channel dimension, to perform element-wise addition with the corresponding bottom-up map. Finally, each fused feature map undergoes a 3×3 convolutional layer to alleviate the upsampling aliasing effect and output the final multi-scale feature pyramid (P2, P3, P4, P5) with fixed feature dimension. An illustration of the pyramid feature maps is presented in Figure 2.5.

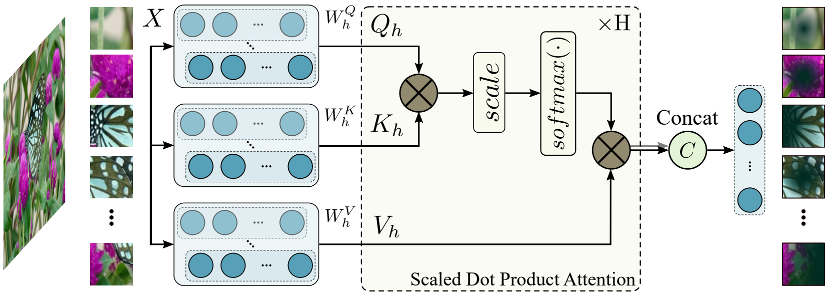


Figure 2.6: A depiction of the MHSA for a given input image. In vision transformers, the image is broken into patches of equal size. The MHSA outputs the attention maps.

Vision Transformer

In 2020, Dosovitskiy et al. introduced the Vision Transformer (ViT) [Dos21] architecture for computer vision tasks, drawing inspiration from the remarkable progress made by transformer models in natural language processing. The core idea behind the vision transformers is the self-attention [Vas17] mechanism on the input image, which is regarded as a sequence of smaller patches. Self-attention allows the model to dynamically adjust its focus to different parts of the input, capturing long-range dependencies and fine-grained details.

Multi-Head Self-Attention. ViTs employ a Multi-Head Self-Attention (MHSA) approach [Dos21], where the self-attention mechanism is applied numerous times simultaneously with distinct learned linear projections in separate heads. The outputs of all the heads are then combined and linearly transformed to form the final output of the multi-head attention layer. This enables the model to govern the fusion of information across segments of an input sequence, thereby learning more powerful global and local representations. Formally, each head $(h = 1,..,H)$ applies a learnable linear projection on the input $\boldsymbol{X}$ and computes three matrices:

$$\boldsymbol{Q}_h = \boldsymbol{X}\boldsymbol{W}_h^Q, \tag{2.14}$$

$$\boldsymbol{K}_h = \boldsymbol{X}\boldsymbol{W}_h^K, \tag{2.15}$$

$$\boldsymbol{V}_h = \boldsymbol{X}\boldsymbol{W}_h^V, \tag{2.16}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ denote query, key, and value, respectively .$\boldsymbol{W}$ denotes a learnable weight matrix. Next, for each head, the scaled dot-product attention $\boldsymbol{A}_h$ is computed as

$$\boldsymbol{A}_h = \text{softmax}\left(\frac{\boldsymbol{Q}_h\boldsymbol{K}_h^T}{\sqrt{d}}\right) \cdot \boldsymbol{V}_h. \tag{2.17}$$

d is the dimension of the features. The scaling by $\sqrt{d}$ aims to reduce the variance of the dot product. Finally, the attentions from all the heads are concatenated into a final matrix $\boldsymbol{A}$ and undergoes a linear transformation via a learnable weight matrix $\boldsymbol{W}^O$ to output the final attention map $\boldsymbol{A}\boldsymbol{W}^O$ of the same shape as the input.

However, adopting a general-purpose ViT architecture for object detection presents several challenges. First, unlike NLP tasks, where fixed scale word tokens are the basic processing elements, computer vision tasks use image pixels to describe visual elements that vary significantly in scale. Secondly, images contain many more pixels than words in a text passage, resulting in higher dimensional input sizes for the model to process. Consequently, the self-attention process can become computationally complex, increasing quadratically with the image size. As this dissertation primarily focuses on object detection with limited data, only the ViT-based backbones for object detection are examined.

Swin Transformer Backbone

To address the limitations above, Liu et al. proposed the Swin Transformer [Liu21], a recent adaptation of ViT. The Swin Transformer adopts a hierarchical approach for processing images, where the image is downsampled and split into smaller patches. This improves efficiency and scalability by reducing computational complexity compared to conventional methods that simultaneously process the entire image.

The Swin Transformer architecture comprises two fundamental components: the Path Merging and the Swin Transformer Block. First, the former is a downsampling method that does not use any convolutional layer and operates at a patch level. Patch merging groups $n \times n$ patches and concatenates them depth-wise, resulting in a downsampled feature map by a factor of n. This results in a transformation of the input from a shape of $H \times W \times C$ to $(H/n) \times (W/n) \times (n^2 \cdot C)$, where H, W, and C represent the height, width, and channel depth, respectively. Second, the Swin Transformer block comprises two sub-units, each containing a normalization layer, an attention module, another normalization layer, and an MLP layer.

Window MHSA (W-MHSA). The first sub-unit uses a W-MHSA module. The conventional MHSA utilizes global self-attention, whereby the correlation between each patch is computed against every other patch. This approach leads to a quadratic complexity concerning the number of patches, rendering it unsuitable for high-resolution images. The Swin Transformer uses a window-based multi-head self-attention method to overcome this challenge. In W-MHSA, the attention is computed only within each window (a set of patches). Since the window size remains constant throughout the network, the complexity is linear in relation to the number of patches.

Shifted Window MHSA (SW-MHSA). One disadvantage of W-MHSA is that it limits the modeling capacity of the network by constraining self-attention to each window. As a remedy, in the second sub-unit, a Shifted Window MHSA (SW-MHSA) module is utilized in conjunction with the W-MSA module. The SW-MHSA shifts the windows towards the bottom right corner by a factor of $M/2$ to introduce cross-window connections, where M is the window size. Nevertheless, this shift generates orphaned patches that do not belong to any window and windows with incomplete patches. Therefore, a cyclic shift technique moves the orphaned patches into the windows with incomplete patches. Finally, a mask restricts self-attention to adjacent patches in the original feature map.

Vision Transformer Detector Backbone

The Vision Transformer Detector (ViTD) [Li22] is a modified version of ViT that is specifically designed for object detection. The ViTD includes a backbone network that is based on a standard ViT architecture and is followed by a detection head that employs a set of learned linear projections to forecast the class and bounding box coordinates of objects present in the image. Specifically, ViTD generates a straightforward feature pyramid by utilizing only the final feature map of a standard ViT backbone. This approach eliminates the need for a hierarchical backbone and deviates from the FPN design. ViTD uses simple non-overlapping window attention to capture features from high-resolution images effectively. It also includes a few cross-window blocks, which can be either global attention or convolutions, to facilitate information propagation.

Formally, the ViTD backbone can be described as follows. First, the input image $\boldsymbol{X} \in \mathbb{R}^{H \times W \times C}$ is split into a sequence of flattened 2D patches $x_p \in \mathbb{R}^{N \times (P \cdot P \cdot C)}$, where $N = HW/P^2$ is the total number of patches or the effective input sequence length, and P denotes the patch size. Second, the patches are mapped to patch embeddings $E \in \mathbb{R}^{P^2 C \times D}$ via a linear projection layer. Positional embeddings are then added to the patch embeddings to capture the relative location of each patch in the image and fed into a regular ViT encoder, which comprises a sequence of L transformer layers. Each layer contains a MHSA module and a position-wise feed-forward network. Finally, the resulting feature map of the final transformer layer undergoes a group of convolutions or deconvolutions in parallel to generate multi-scale feature maps for the detection framework.

2.2 Deep Learning based Object Detection

When dealing with an image classification task, the general assumption is that only one significant object exists in the image, and the primary objective is to identify its class label. However, this assumption may not hold since the image often has multiple objects of interest. Object Detection (OD), on the other hand, aims to identify the category of each object and determine the corresponding precise location in terms of bounding box coordinates within the image.

Deep learning based OD approaches have demonstrated superior performance in light of recent advances in CNNs and ViTs, becoming a central building block in various modern perception systems. The deep learning based OD approaches can be classified into two main categories: two-stage (sparse) and one-stage (dense) detectors. The main difference between these methods is whether they generate object proposals. In two-stage detectors, the feature map generated by the feature extractor is used as input to a Region Proposal Network (RPN) [Ren15], which proposes a set of object candidate regions that may contain objects of interest. Next, a detection head is applied to classify and refine the location of each instance. Although two-stage detectors offer better accuracy by focusing on regions of interest, they are computationally expensive, memory intensive, and slow in training and inference.

On the other hand, one-stage detectors omit the proposal generation stage and directly predict the location and category of objects in a given image. Instead, an image is typically split into a grid of cells and predicts the category and location of each object in each cell. As a result, dense detectors are typically faster but less accurate than their sparse counterparts, particularly for small objects. Nevertheless, recent advances in one-stage and two-stage approaches have narrowed the performance gap, making the decision more nuanced and dependent on the application and available hardware resources [Zai22].

2.2.1 Problem Formulation

Mathematically, the object detection problem can be formulated as follows. Given a dataset $\mathcal{D}$ with abundant samples of $\mathcal{C}$ classes:

$$\mathcal{D} = \{ \, (\boldsymbol{X}, \boldsymbol{Y}) \mid \boldsymbol{Y} = \{(c_i, \mathbf{b}_i)\}, c_i \in \mathcal{C}, \mathbf{b}_i \in \mathbb{R}^4 \}, \tag{2.18}$$

where $\boldsymbol{X} \in \mathcal{X}$ is an input image, and $\boldsymbol{Y} \in \mathcal{Y}$ is the associated annotation. $\mathcal{X}$ and $\mathcal{Y}$ denote the image space and label space, respectively. For each object instance i in the image, c_i and $\mathbf{b}_i$ represent the corresponding object category and the bounding box coordinates, respectively. Each bounding box instance $\mathbf{b}_i = (\mathrm{b}_i^x, \mathrm{b}_i^y, \mathrm{b}_i^w, \mathrm{b}_i^h)$ is represented by the top-left coordinates (b_i^x, b_i^y) along with the width b_i^w and height b_i^h. Each entry belongs to $\mathbb{R}^+$. The objective is to learn a detector $h_\Theta : \boldsymbol{X} \to \boldsymbol{Y}$, parameterized by Θ, that minimizes the classification and localization errors between the network predictions and the ground-truth classes and bounding boxes.

2.2.2 Two-Stage Detection Networks

R-CNN

In 2014, Region Convolutional Neural Network (R-CNN) [Gir14] pioneered the first region-based family detector showing the ability of CNNs to boost detection performance significantly. R-CNN uses a class-agnostic region proposal module to break down the detection problem into instance-level classification and localization sub-problems. First, an input image is normalized using the mean and standard deviation of the pixel values. Second, the image is fed to a region proposal module, generating $R = 2000$ object candidates via the Selective Search (SS) algorithm. The generated proposals, commonly called Regions of Interest (RoIs), have a higher chance of containing an object of interest. Specifically, SS divides the image into regions based on color, texture, and intensity similarities. These regions are then merged using a hierarchical approach that combines smaller regions into larger ones using a graph-based clustering method to generate object proposals efficiently. Afterward, each of these possibilities is warped and propagated to a CNN network, extracting a 4096-dimension feature vector. Then, each class-specific Support Vector Machine (SVM) [Cor95] is given the extracted feature vectors to get confidence scores. Finally, Non-Maximum Suppression (NMS) eliminates redundant detections based on confidence and Intersection over Union (IoU) scores of neighboring bounding boxes. The less confident boxes with a lower IoU than a pre-defined threshold are discarded.

The training process for R-CNN involves two stages: pre-training the CNN on a large classification dataset and finetuning for detection. This is achieved by replacing the classification layer with a randomly initialized classifier and training a linear SVM and bounding box regressor for each class using Stochastic Gradient Descent (SGD) [Bot10].

Fast R-CNN

Due to the SS algorithm and multiple class-specific classifiers, R-CNN is computationally expensive and requires longer training times, up to a few days, even for small datasets. Later in 2015, Fast R-CNN [Gir15] was proposed as an end-to-end detection framework. Compared to R-CNN, Fast R-CNN has only a single network that leverages a multi-task loss. Specifically, Fast R-CNN utilizes a CNN-based backbone, such as the VGG [Sim14] network, to generate a single feature map for the image. Different than R-CNN, the set of object proposals feature maps are fed to a RoI pooling layer to extract fixed-length feature vectors. To obtain the final predictions, each RoI feature vector undergoes a sequence of MLPs that branches to classification scores and bounding box positions.

RoI Pooling Layer. The object proposals are fed to a RoI pooling layer, which extracts fixed-size instance-level features via max pooling operation. Specifically, each RoIs of size $\mathbb{R}^{H \times W}$ is divided into grid cells $\mathbb{R}^{H/P_H \times W/P_W}$, where P_H and P_W are the pooling height and width, respectively. Then, the maximum value for each grid cell is extracted. Then, the RoI pooled features undergo two MLPs, split into a softmax layer with $C+1$ classes (the additional class to account for the background) and a bounding box regressor layer with another MLP.

RCNN Loss. A multi-task (classification and localization) loss was introduced to enable end-to-end network training, eliminating the need for separate SVMs classifiers. The overall detection loss can be formulated as follows:

$$\mathcal{L}_{\text{RCNN}} = \frac{1}{R}\sum_{i=1}^{R} \mathcal{L}_{cls}^{\text{RCNN}}(\hat{\boldsymbol{p}}_i, c_i) + \lambda_{loc}\mathbb{1}_{c_i \geq 1}\mathcal{L}_{loc}^{\text{RCNN}}(\hat{\mathbf{b}}_i^{c_i}, \mathbf{b}_i), \tag{2.19}$$

where R is the total number of RoIs. $\mathcal{L}_{cls}^{\text{RCNN}}$ is the cross-entropy classification loss and L_{loc}^{RoI} is the smooth L1 localization loss. $\hat{\boldsymbol{p}}_i$ and c_i denote the predicted probability distribution and the ground-truth class of the i^{th} instance, respectively. Formally,

$$\mathcal{L}_{cls}^{\text{RCNN}}(\hat{\boldsymbol{p}},c) = -\log(\boldsymbol{p}_c). \tag{2.20}$$

For the localization task, the L2 loss function from R-CNN is replaced by a smooth L1 loss for two main reasons: First, it is less sensitive to outliers rendering it more robust to noisy detections. Second, it is continuously differentiable, allowing more efficient computation of gradients. For the i^{th} instance, $\hat{\boldsymbol{b}}_{c_i}$ is the predicted bound box for class c_i and $\mathbf{b}_i$ is the target bounding box. $\mathbb{1}_{c_i \geq 1}$ is an Iverson bracket indicator function that outputs 1 if a foreground class ($c_i \geq 1$) is encountered or 0 if background ($c_i = 0$). λ_{loc} serves as a balancing factor, influencing the weight of the localization loss within the overall training loss. The localization loss is computed as:

$$\mathcal{L}_{loc}^{\text{RCNN}}(\hat{\mathbf{b}}_c, \mathbf{b}) = \sum_{j \in (x,y,w,h)} \text{smooth}_{L1}(\hat{\mathbf{b}}_c^j - \mathbf{b}^j), \tag{2.21}$$

where

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1. \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{2.22}$$

Analogous to R-CNN, the bounding box regression is performed on box offsets $\boldsymbol{\delta}$ rather than absolute coordinates for the detector to be invariant to various scales and locations. For the i^{th} instance, given a ground-truth box $\mathbf{b}_i = (\mathrm{b}_i^x, \mathrm{b}_i^y, \mathrm{b}_i^w, \mathrm{b}_i^h)$ and a predicted bounding box $\hat{\boldsymbol{b}}_i = (\hat{\mathrm{b}}_i^x, \hat{\mathrm{b}}_i^y, \hat{\mathrm{b}}_i^w, \hat{\mathrm{b}}_i^h)$, the box

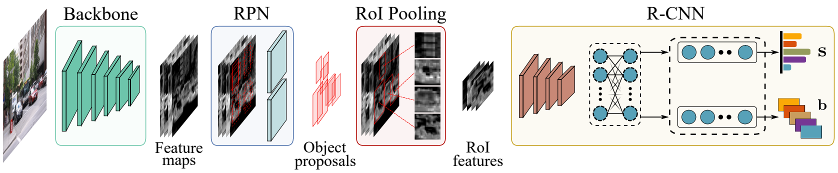


Figure 2.7: A detailed overview of the Faster R-CNN architecture. The image is sampled from the MS-COCO dataset [Lin14]. The image initially goes through the backbone, resulting in feature maps. These feature maps are then input into the RPN, which generates anchors and subsequently identifies object proposals (red boxes). The object proposals are then fed to the RoI pooling layer along with the input feature maps to produce RoI pooled features. Lastly, the R-CNN module performs further feature extraction at the instance level, yielding classification scores **s** and bounding boxes **b**.

offset $\boldsymbol{\delta}_i = (\delta_i^x, \delta_i^y, \delta_i^w, \delta_i^h)$ is parametrized as follows:

$$\delta_i^x = (\hat{\mathrm{b}}_i^x - \mathrm{b}_i^x)/\mathrm{b}_i^w, \tag{2.23}$$

$$\delta_i^y = (\hat{\mathrm{b}}_i^y - \mathrm{b}_i^y)/\mathrm{b}_i^h, \tag{2.24}$$

$$\delta_i^w = \log(\hat{\mathrm{b}}_i^w/\mathrm{b}_i^w), \tag{2.25}$$

$$\delta_i^h = \log(\hat{\mathrm{b}}_i^h/\mathrm{b}_i^h). \tag{2.26}$$

During training, Fast R-CNN samples 64 proposals for each input image, out of which 25% are foreground instances. Instances are considered to be foreground if their IoU ≥ 0.5 and background if $0.1 \leq$ IoU < 0.5.

Faster R-CNN

Despite the improvements made by Fast R-CNN, the SS algorithm remains a bottleneck. It is important to note that the SS algorithm runs on a Central Processing Unit (CPU) and not a GPU, increasing both the training and testing time.

Region Proposal Network (RPN). Faster R-CNN [Ren15] replaces the SS algorithm with a 3-layer CNN, namely the RPN. The RPN comprises a shared 3×3 convolutional layer followed by two 1×1 convolutions for the objectness scores, denoting the likelihood of an object being present, and the bounding box offsets refinement. To begin with, for each pixel in the input feature map, the RPN generates a fixed number of anchor boxes with varying sizes and aspect ratios. Next, the feature map undergoes a 3×3 convolution to produce a feature map for every pixel position. Following this, the two parallel 1×1 convolutional layers output the objectness score and the associated bounding box regression offsets. The RPN then filters the positive anchors with foreground objects by selecting the anchor box with an IoU > 0.7 and as negative in case of an IoU < 0.3 using NMS. The remaining anchors and boundary anchors are discarded during training. Finally, to avoid learning bias toward negative samples, a balanced batch of 128 positive and 128 negative anchors are randomly sampled, with supplemental negative samples if lacking positives. The RPN loss function is denoted by

$$\begin{aligned}\mathcal{L}_{\text{RPN}} = & \frac{1}{N_{cls}} \sum_{i=1}^{N_{cls}} \mathcal{L}_{cls}^{\text{RPN}}(\hat{\boldsymbol{p}}_i, l_i) \\ & + \lambda_{loc}^{\text{RPN}} \frac{1}{N_{reg}} \sum_{i=1}^{N_{reg}} l_i \cdot \mathcal{L}_{loc}^{RPN}(\hat{\boldsymbol{b}}_i, \mathbf{b}_i),\end{aligned} \tag{2.27}$$

where $\mathcal{L}_{cls}^{\text{RPN}}$ is a binary cross-entropy classification loss. $\hat{\boldsymbol{p}}_i$ denotes the the predicted probability of the i^{th} anchor. l_i is the associated ground-truth binary label denoting whether or not an object is present. N_{cls} is the total number of positive and negative anchors in the mini-batch for the RPN classification loss. $\mathcal{L}_{loc}^{\text{RPN}}$ is a smooth L1 loss. $\hat{\boldsymbol{b}}_i$ and $\mathbf{b}_i$ are the predicted and ground-truth bounding boxes, respectively. As previously mentioned, the regression is done on the parameterized offset coordinates. $\lambda_{loc}^{\text{RPN}}$ is a scaling factor that is typically set to 10. N_{reg} is the number of anchor locations (the area of the input feature map).

Alternate Training Paradigm. Faster R-CNN adopts a 4-step alternate training paradigm. First, the RPN is independently trained end-to-end by finetuning an ImageNet-pre-trained CNN backbone for the region proposal task. Second, a Faster R-CNN detection network is separately trained using the proposals generated by the trained RPN from the previous step. Third, the RPN is finetuned using the detector network, where the shared convolutional layers are fixed, and only the non-shared RPN layers are finetuned. The shared convolutional layers encourage the RPN to learn to generate higher-quality proposals. Finally, the non-shared layers in the detection network are finetuned while keeping the shared convolutional layers fixed. This enables the detector network to better classify and refine the RoIs generated by the RPN.

With both networks sharing the same convolutional layers, a unified network is formed, and the overall detection training loss combines the RCNN (Eq.2.19) and RPN (Eq.2.27) loss functions:

$$\mathcal{L}_{det} = \mathcal{L}_{RCNN} + \mathcal{L}_{RPN}. \tag{2.28}$$

An illustration of the Faster R-CNN model is presented in Figure 2.7.

Cascade R-CNN

Tuning the IoU threshold is a critical aspect of training a detection model. If the threshold is set too low, more samples would be considered as foreground resulting in more noisy detections. Conversely, if set too high, positive samples may vanish exponentially due to overfitting during training. This issue can worsen due to a misalignment between the IoU values during training between the RoIs and actual ground-truth boxes, and the IoU values during testing between the RoIs and the target boxes.

To overcome the limitations above, Cascade R-CNN [Cai18] was introduced, where several cascaded detectors are trained, each with increasing threshold values, making it more robust to false positives. In addition, the detectors are trained sequentially in stages, taking advantage of the output of one detector as a better prior distribution to training the next one.

The cascaded regression can be viewed as a resampling procedure that acclimates the distribution of the input hypotheses, where each specialized stage regressor is optimized for the resampled distribution. Moreover, this resampling is done during training and inference, ensuring consistency between the two and leading to more accurate localization. Additionally, the regression offsets are normalized using their mean and variance to facilitate multi-task learning.

Cascade R-CNN Loss. Each stage s, the R-CNN comprises a classifier h^s_{cls} and a bounding box regressor h^s_{reg}, which are optimized for an IoU threshold of t^s. The loss function for a single stage is given by

$$\mathcal{L}^s_{\text{RCNN}} = \mathcal{L}^{\text{RCNN}}_{cls}(\hat{\boldsymbol{p}}^s, c) + \lambda^s_{loc} \mathbb{1}_{c \geq 1} \mathcal{L}^{\text{RCNN}}_{loc}(\hat{\mathbf{b}}^c, \mathbf{b}_i), \tag{2.29}$$

where $\hat{\boldsymbol{p}}^s = h^s_{cls}(x^s)$ is the predicted probability for the current stage input x^s. $\hat{\mathbf{b}}^s = h^s_{reg}(x^{s-1}, \hat{\mathbf{b}}^{s-1})$ is the predicted bounding box based on the preceding bounding box from the previous stage. The classification and regression loss functions are computed as previously defined in 2.20 and 2.21, respectively.

The classification and regression losses are summed together at each stage and weighted by a loss balance parameter λ^s_{loc}. In the first R-CNN stage, $\lambda^{s=1}_{loc}$ is set to a large value to encourage accurate detection at lower IoU thresholds, and it is gradually lowered in succeeding stages to promote the network to focus on increasingly challenging objects. For S R-CNN stages, the overall training loss $\mathcal{L}_{\text{det}}$ is:

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{RPN}} + \sum_{s=1}^{S} \mathcal{L}^s_{\text{RCNN}}. \tag{2.30}$$

2.2.3 One-Stage Detection Networks

While two-stage frameworks yield high detection accuracy, particularly in cases where the objects are small or heavily occluded, they come at the expense of higher computational complexity and longer inference time. On the other hand, a one-stage framework works in a single-step process. It directly predicts the class labels and bounding boxes for each object without any intermediate region proposal step. Unlike the two-stage framework, it is faster and computationally efficient but may not achieve high detection accuracy, especially when objects are small or densely packed.

To predict the location and size of objects, one-stage frameworks frequently utilize anchor boxes, whereas two-stage frameworks use the RPN to suggest a set of candidate boxes for further refinement. Furthermore, one-stage frameworks often employ feature pyramids to handle objects of varying scales, while two-stage frameworks employ multi-scale feature maps and a spatial pyramid pooling layer. However, the most suitable framework depends on the application and available resources. For example, if high accuracy is essential and computational resources are not a concern, then a two-stage framework may be the optimal choice. However, a one-stage framework may be more suitable, if efficiency and speed are more critical.

YOLO Family

In 2016, Redmon et al. proposed You Only Look Once (YOLO) [Red16], the first real-time one-stage object detector among the YOLO family. YOLO views the object detection problem as an end-to-end regression problem, where the network directly predicts the class probabilities and the associated bounding boxes from the input image.

YOLO is a CNN-based architecture that divides the input image into $G \times G$ grid cells and predicts a fixed number of bounding boxes, associated class probabilities, and confidence scores in each grid cell. The confidence score is the probability that the predicted bounding box contains an object which overlaps with the ground-truth one. Specifically, the input image is resized to a fixed size. Next, the resized input image is then processed through a series of convolutional layers, beginning with a 1×1 convolution and a subsequent 3×3 convolutional layers. A ReLU activation function is then utilized after each convolution, except for the final layer.

The output is a tensor with dimensions $(G, G, B \times 5 + C)$, where G is the size of the grid. C is the total number of detection classes. B denotes the number of predicted bounding boxes for each grid cell, and 5 refers to the 4 bounding box coordinates (x, y, w, h) and confidence score cs. (x, y) corresponds to the center of the box, and (w, h) represents the width and height of the predicted box, respectively. These values are then normalized by the grid size and are relative to the grid cell in which the box is located.

YOLO Loss Function. The YOLO framework predicts multiple bounding boxes for each grid cell. However, when computing the loss function, only one bounding box is selected for the object, if it exists within that cell. The selection criterion is based on the highest IoU with the ground-truth object, making each prediction more accurate for different sizes and aspect ratios of objects. The loss function consists of three parts: classification loss, localization loss, and confidence loss. First, for each grid cell g, the classification loss computes the squared error between the predicted probability and target class:

$$\mathcal{L}_{\text{cls}} = \sum_{g=0}^{G^2} \mathbb{1}_{\text{obj}^g} \sum_{c \in C} (\hat{\boldsymbol{p}}_g(c) - \boldsymbol{y}_g(c))^2, \tag{2.31}$$

where $\mathbb{1}_{\text{obj}^g}$ is an indicator function that evaluates to 1 if an object is present in grid cell g. For class c, $\hat{\boldsymbol{p}}_g(c)$ and $\boldsymbol{y}_g(c)$ represent the predicted class probability and one-hot target label, respectively. Second, the localization loss measures the squared error between the predicted and target bounding boxes taking into consideration the bounding boxes responsible for the detection:

$$\begin{aligned}\mathcal{L}_{\text{loc}} = \lambda_{\text{loc}} \sum_{g=0}^{G^2} \sum_{b=0}^{B} \Big(\mathbb{1}_{\text{obj}^{g,b}} (\hat{x}_g - x_g)^2 + (\hat{y}_g - y_g)^2 \\ + (\sqrt{\hat{w}_g} - \sqrt{w_g})^2 + (\sqrt{\hat{h}_g} - \sqrt{h_g})^2 \Big),\end{aligned} \tag{2.32}$$

where $\mathbb{1}_{\text{obj}^{g,b}}$ evaluates to 1 if the b^{th} bounding box in the grid cell g is responsible for detecting the present object. λ_{loc} is the scaling factor to control the contribution of the localization loss. The width and height are square rooted to emphasize the error relative to the bounding box size. Third, the confidence loss measures the squared error difference between the predicted and target confidence scores for foreground and background boxes:

$$\mathcal{L}_{\text{conf}} = \sum_{g=0}^{G^2} \sum_{b=0}^{B} \left(\mathbb{1}_{\text{obj}^{g,b}} (\hat{cs}_g - cs_g)^2 + \lambda_{\text{noobj}} \mathbb{1}_{\text{noobj}^{g,b}} (\hat{cs}_g - cs_g)^2 \right), \quad (2.33)$$

where $\hat{cs}$ and cs denote the predicted and target confidence scores, respectively. The target confidence score is computed using the IoU between the predicted and ground-truth box. $\mathbb{1}_{\text{noobj}^{g,b}}$ evaluates to 1 if the b^{th} bounding box in the grid cell g is background. λ_{noobj} is the scaling factor to weight down the loss for background detection. The final training loss is a summation of the three losses.

The benefits of YOLO is twofold. Firstly, it tackles object detection as a single regression problem, making it faster than R-CNN detectors and able to work in real-time. Secondly, YOLO omits the RPN, reducing the chance of false positives as it processes the entire image, maintaining a strong sense of context. On the other hand, YOLO may perform poorly when dealing with small objects and crowded scenes due to the restricted spatial grid resolution and the fixed number of boxes per grid cell.

YOLO Variants. Since the first introduction of ground-truth, numerous variants have followed, proposing various architectural and data augmentation modifications. To improve upon the original ground-truth architecture, YOLOv2 [Red17] makes several modifications. Instead of VGG-16, it utilizes a Darknet-19 [Red17] network with fewer convolutional layers and higher-resolution inputs. YOLOv2 also incorporates anchor boxes and predicts offsets to each anchor box to enhance localization. To regularize and accelerate the training process, BN is employed. Moreover, YOLOv2 picks the anchor box dimensions with a K-means clustering algorithm applied to the training dataset bounding boxes. This provides better box priors with higher IoU scores across different classes. These changes have led to improved detection performance while reducing the inference time for YOLOv2. In order to improve the detection accuracy of small objects, YOLOv3 [Red18] employs a FPN. In addition, YOLOv3 utilizes a deeper backbone network, Darknet-53 [Red18], providing higher accuracy in object detection. However, this comes at the cost of a slower inference speed than YOLOv2.

RetinaNet

Without a RPN, the dense predictions of a one-stage detector can yield many empty anchor boxes hindering the learning signal. RetinaNet [Lin18] addresses the issue of imbalanced foreground and background class distribution in an anchor-based framework. The model introduces a novel loss function, referred to as focal loss, which prioritizes the contribution of hard examples during training while down-weighting the contribution of easier ones. Hard examples refer to samples the detector fails to classify and localize accurately. Focusing on more challenging examples helps mitigate the issue of easy negative examples dominating the loss function. Formally, the focal loss can be written as:

$$\mathcal{L}_{\mathrm{F}}(\hat{p}) = \begin{cases} -(1-\hat{p})^{\gamma}\log(\hat{p}), & \text{if } y = 1. \\ -\hat{p}^{\gamma}\log(1-\hat{p}), & \text{otherwise.} \end{cases} \tag{2.34}$$

where $\hat{p}$ is the predicted class probability. γ is a modulating parameter, which can be tuned to focus on the hard negative examples and reduce the loss contribution of easy ones. In Online Hard Example Mining (OHEM), the loss of each example is calculated, and then NMS is used to remove redundant detections. A mini-batch is then formed by selecting samples with the highest loss scores. OHEM focuses on misclassified examples, similar to focal loss, but it differs from focal loss in that it completely discards samples that are easy to classify.

To detect objects with various scales and sizes, RetinaNet uses ResNet-FPN to extract features from different levels of the input image. The model also employs a dense anchor coverage approach, utilizing nine anchors per pyramid level. RetinaNet uses separate classification and localization subnets to learn more class-specific and class-agnostic features, unlike other models that share a CNN for classification and regression tasks. These subnets have the same architecture: four consecutive 3×3 convolutional layers and ReLU activations. However, the output of the classification subnet is $C \times A$, where C is the total number of classes, and A is the number of anchors. Meanwhile, the localization subnet outputs $4A$ predicted offsets for each anchor.

CenterNet

Most previously reviewed object detection methods are anchor-based, relying on predefined anchor boxes with fixed sizes and aspect ratios. However, this dependency on predefined anchor sizes may impede the detection of objects with a broad range of sizes and shapes, especially for small objects and objects with uncommon aspect ratios. To this end, anchorless object detectors predict the bounding boxes of objects without any predefined assumptions regarding their size or shape. Instead, an anchorless detector directly regresses the bounding boxes from the feature maps of the input image. This enables them to improve the accuracy and efficiency of object detection for objects with different sizes and aspect ratios, especially for smaller objects.

CenterNet [Dua19] was a pioneer anchorless detector representing the object as a point, namely the keypoint, corresponding to the bounding box center. CenterNet comprises three heads: keypoint heatmap, localization offset, and object size. The keypoint heatmap head outputs C channels, one for each class, where a heatmap value evaluates to one at the center of the object and exponentially decreases as it moves further away from that center. The heatmap loss $\mathcal{L}_{\text{hm}}$ is defined as the MSE between the predicted heatmap $\hat{Y}^c_{xy}$ for class c and the ground-truth heatmap Y^c_{xy}:

$$\mathcal{L}_{\text{hm}} = -\frac{1}{N_{\text{kp}}} \sum_{xy} \sum_{c=1}^{C} \begin{cases} \left(1 - \hat{Y}^c_{xy}\right)^{\alpha} \log\left(\hat{Y}^c_{xy}\right), & \text{if } Y^c_{xy} = 1. \\ \left(1 - Y^c_{xy}\right)^{\beta} \left(\hat{Y}^c_{xy}\right)^{\alpha} \log\left(1 - \hat{Y}^c_{xy}\right), & \text{otherwise.} \end{cases} \tag{2.35}$$

N_{kp} is the total number of keypoints. x and y denote the keypoints coordinates. α and β are hyperparameters that balance the positive and negative samples. For a positive sample $Y^c_{xy} = 1$ or a negative sample, a focal loss is utilized with the only difference of a modulating factor $\left(1 - Y^c_{xy}\right)^{\beta}$ to reduce the contribution of negative examples. The generated heatmaps are in a lower spatial resolution, where the width and height of the input image are divided by the model output stride R and floored. When scaling up the heatmap to the original image size to get the final predictions, the precision errors can be up to a few pixels. To account for the pixels-mapping errors, the localization offset head predicts the pixels offset for each spatial dimension. For the N_{pos} positive samples, the localization offset loss is defined as the Mean Absolute Error (MAE) between the predicted offset $(\hat{O}^x, \hat{O}^y)$ and ground-truth offset values (O^x, O^y):

$$\mathcal{L}_{\text{off}} = \frac{1}{N_{\text{pos}}} \sum_{i=1}^{N_{\text{pos}}} \left|\hat{O}_i^x - O_i^x\right| + \left|\hat{O}_i^y - O_i^y\right|. \tag{2.36}$$

Third, the object size head predicts the width and height of the bounding box. Similar to the localization offset loss function, for all positive samples, the shape size loss functions compute the MAE between the predicted $(\hat{w}, \hat{h})$ and ground-truth (w, h) object shapes:

$$\mathcal{L}_{\text{shape}} = \frac{1}{N_{\text{pos}}} \sum_{i=1}^{N_{\text{pos}}} \frac{1}{w_i + h_i} \cdot \left(|\hat{w}_i - w_i| + |\hat{h}_i - h_i|\right), \tag{2.37}$$

where $1/(w_i + h_i)$ is a normalization factor that helps to prevent objects with larger shapes from dominating the loss compared to smaller ones. The overall training loss is then the summation of the three losses above.

2.2.4 Evaluation Metrics

The performance of a 2D OD model is typically measured by various evaluation metrics that measure its performance based on the prediction statistics comprising the following:

- True Positives (TPs): the count of correctly identified positive objects.
- False Positives (FPs): the count of incorrectly identified ones.
- True Negatives (TNs): the count of accurately identified negative objects.

- False Negatives (FNs): the count of mistakenly identified ones.

This section discusses the most commonly used evaluation metrics for 2D object detection and their significance.

Intersection over Union

Given a predicted and a ground-truth box, the Intersection over Union (IoU) computes the quality of the predicted bounding box by computing the intersection area and dividing it by the total area of both boxes. Not only is it a vital evaluation metric, but it is also often utilized during training, for example, to filter the object candidates generated by a RPN based on a predefined threshold.

$$IoU = \frac{A_{\text{IN}}}{A_{\text{UN}}} = \frac{\min(A_{\text{P}}, A_{\text{GT}})}{\max(A_{\text{P}}, A_{\text{GT}})}, \tag{2.38}$$

where A_{IN} is the intersection area between the predicted and ground-truth boxes. A_{UN} denotes the union area between the predicted and ground-truth boxes. A_{P} and A_{GT} denote the area of the predicted and ground-truth bounding boxes, respectively.

Average Precision and Recall

The Average Precision (AP) [Sal83] is the most commonly used evaluation metric for object detection. AP is the area under a precision-recall graph for different IoU thresholds. The Precision evaluates the effectiveness of the model in identifying the TPs within its positive predictions:

$$\text{Precision} = \frac{TPs}{TPs + FPs}. \tag{2.39}$$

The recall represents how much the model could recall from the provided ground-truth labels:

$$\text{Recall} = \frac{TPs}{TPs + FNs}. \tag{2.40}$$

Following Microsoft Common Objects in Context (MS-COCO) [Lin14], the AP is computed using 101-point interpolation across a range of IoU thresholds for each class, commonly $[0.5, 0.95]$ with a step of 0.05. For the c^{th} class, the AP is defined as:

$$AP_{c^{\text{th}}} = \int_{0}^{1} \text{Precision}(r)\ \mathrm{d}r, \tag{2.41}$$

where $\text{Precision}(r)$ represents the precision at a given recall level r. Then, the mean Average Precision (mAP) is computed by averaging across all C classes as follows:

$$\text{mAP} = \frac{1}{C} \sum_{c=1}^{C} \text{AP}_{c^{\text{th}}}. \tag{2.42}$$

The Average Recall (AR) is defined as the area under the recall-IoU graph. Specifically, for each class, the AR is computed by doubling the area under the graph with IoU threshold range of $[0.5, 1.0]$. For the c^{th} class, the AR is denoted by:

$$AR_{c^{\text{th}}} = 2 \int_{0.5}^{1} \text{Recall}(iou)\ \mathrm{d}iou, \tag{2.43}$$

where $\text{Recall}(iou)$ is the recall value for the IoU threshold iou. Analogous to mAP, the mean Average Recall (mAR) finally averages the AR for all classes.

The AP metric captures the trade-off between precision and recall as the IoU threshold varies. A high AP indicates that the model is accurate while being able to find most of the objects. On the other hand, a high AR suggests that the model can consistently detect a high proportion of objects, even with higher precision. This reflects the robustness of the model in detecting objects with different difficulty levels.

2.3 Discussion

This chapter established the foundational theoretical principles of DL that underpin the subsequent discussions in this dissertation. It provided a thorough overview of fundamental concepts such as neural networks, activation functions, and normalization techniques. Additionally, various fundamental deep feature extractors were introduced, such as ResNet, FPN, and VGG. Moreover, the chapter formally introduced the OD task within the context of CV and outlined the primary objectives and associated challenges. Furthermore, the chapter conducted an extensive survey of the most widely used OD architectures such as Faster R-CNN, and YOLO. Finally, the chapter introduced a variety of evaluation metrics employed to assess the performance of object detection models. The rationale behind metrics such as mAP and mAR was clarified. These metrics established the criteria against which the original methodologies proposed in the dissertation will be rigorously evaluated.

3 Few-Shot Object Detection Fundamentals

Naturally, humans possess an innate ability to learn new skills and concepts by building upon their previous experiences, even with only a few examples, rather than starting from scratch. An illustrative example is when a child, who has seen just a handful of various animals, can effortlessly recognize new animals with similar characteristics when presented with a few pictures, despite never having encountered them.

Few-Shot Learning (FSL) is a growing machine learning subfield striving to replicate this cognitive ability of humans. FSL leverages prior knowledge and experience to adapt to new situations with minimal additional training rapidly. More specifically, FSL involves training a model on a set of related tasks with abundant data and then adjusting the model parameters to generalize to new unseen tasks, even with just a few examples available.

Transitioning from the concept of humans' innate ability to learn new skills to the challenges in OD holds significant importance. Specifically, the considerable volume of labeled data required for training object detectors can be expensive, labor-intensive, and time-consuming. Additionally, recent models require extensive training periods and substantial computational resources, which poses a challenge for efficient OD on mobile and edge devices. Addressing this need for adaptability, Few-Shot Object Detection (FSOD) emerges as a specialized subdiscipline. FSOD aligns with the FSL concept, focusing on swiftly teaching detectors to identify novel classes using minimal data, thereby bridging the gap between human-like learning and machine-based detection capabilities.

In FSOD, the training dataset is scarce, resulting in an unreliable empirical risk minimizer (Section 2.1.1) where a significant gap exists between the expected and empirical risk. To address this issue, FSOD utilizes prior knowledge to find a more reliable risk minimizer based on the three main perspectives [Wan20b]. First, prior knowledge can be used to augment the scarce training dataset, increasing the number of training samples and creating a more accurate empirical minimizer. Second, the hypothesis space can be constrained using prior knowledge, eliminating the improbable space to contain the optimal hypothesis rather than navigating through a vast hypothesis space. Third, prior knowledge can influence the search strategy by providing a better initialization rather than starting from random initialization or guiding the search steps to find the parameterization of the best hypothesis.

This chapter thoroughly examines the foundational principles of FSOD, encompassing its two primary categories: transfer learning and meta-learning based approaches. The chapter begins by formally introducing the FSOD problem, providing a structured foundation for subsequent discussions. Following this, an in-depth review of transfer learning based approaches and meta-learning methods is conducted. Various strategies and methodologies are investigated, highlighting their strengths and limitations and contributing to a nuanced understanding of their effectiveness. Next, the well-established FSOD and G-FSOD benchmarks, the MS-COCO [Lin14] and PASCAL Visual Object Classes (PASCAL-VOC) [Eve10] datasets, are comprehensively introduced. Finally, the FSOD evaluation metrics for few-shot detectors are presented

3.1 Problem Formulation

FSOD divides the training dataset $\mathcal{D}_{\text{train}}$ into a base dataset $\mathcal{D}_{\text{base}}$ with abundant instances of base classes $\mathcal{C}_{\text{base}}$ and a novel dataset $\mathcal{D}_{\text{novel}}$ with limited number of instances of novel classes $\mathcal{C}_{\text{novel}}$. It is important to note that no overlap between the base and novel classes (i.e., $\mathcal{C}_{\text{base}} \cap \mathcal{C}_{\text{novel}} = \emptyset$). Each input image $\boldsymbol{X} \in \mathcal{X}$ comprises multiple object instances associated with an annotation $\boldsymbol{Y} \in \mathcal{Y}$ for each object. $\mathcal{X}$ and $\mathcal{Y}$ denote the image and label space, respectively. For each object instance i, the annotation set contains the class label c_i, and the corresponding bounding box coordinates $\mathbf{b}_i = (\mathrm{b}_i^x, \mathrm{b}_i^y, \mathrm{b}_i^w, \mathrm{b}_i^h)$ with image coordinates $(\mathrm{b}_i^x, \mathrm{b}_i^y)$ along with the box width b_i^w and height b_i^h. Formally,

$$\begin{aligned}\mathcal{D}_{\text{base}} &= \{(\boldsymbol{X}_b, \boldsymbol{Y}_b) \mid \boldsymbol{Y}_b = \{(c_i,\mathbf{b}_i)\}, c_i \in \mathcal{C}_{\text{base}}, \mathbf{b}_i \in \mathbb{R}^4\},\\ \mathcal{D}_{\text{novel}} &= \{(\boldsymbol{X}_n, \boldsymbol{Y}_n) \mid \boldsymbol{Y}_n = \{(c_i,\mathbf{b}_i)\}, c_i \in \mathcal{C}_{\text{novel}}, \mathbf{b}_i \in \mathbb{R}^4\},\end{aligned} \tag{3.1}$$

where the subscripts b and n denote the base and novel data, respectively.

As discussed earlier, object detectors require large amounts of data to perform well. Therefore, if trained only on the scarce novel dataset, they may easily overfit, leading to poor performance and generalization. On the other hand, if the training is conducted using the whole training dataset, the detector will be biased toward the base classes since the base categories outnumber the novel categories significantly. To tackle the abovementioned issues, FSOD adopts a two-stage training strategy: base and novel. In the base training stage, a detector with a pre-trained backbone is trained on the $\mathcal{D}_{\text{base}}$, yielding a base model. In the novel training stage, the model is finetuned on $\mathcal{D}_{\text{novel}}$ while keeping some network parameters frozen to preserve prior knowledge, resulting in the final model.

3.2 Transfer Learning based FSOD

3.2.1 Preliminaries

FSOD can be categorized into two main learning approaches: meta-learning and transfer learning. Transfer learning is a learning strategy that uses the knowledge gained from a previously trained model as a starting point for a new learning task rather than training a new model from scratch. Transfer learning is based on the idea that the model has learned features that can be transferred to new tasks. Formally, during the base training phase, a base model parameterized by Θ_b aims to optimize for the log-likelihood:

$$\Theta_b^* = \arg\max_{\Theta_b} \log p\,(\boldsymbol{Y}_b | \boldsymbol{X}_b;\, \Theta_b). \tag{3.2}$$

Next, the optimal base parameters $\varTheta_b^*$ can be leveraged as initialization parameters for novel training, encapsulating prior knowledge. Formally, the optimization objective during novel training can be denoted as follows:

$$\varTheta_n^* = \arg\max_{\varTheta_n} \log p\left(\boldsymbol{Y}_n | \boldsymbol{X}_n;\, \varTheta_n\right) \qquad s.t.\ \boldsymbol{\varTheta}_n^{(0)} = \boldsymbol{\varTheta}_b^*\,. \tag{3.3}$$

Moreover, transfer learning challenges the common assumption made by deep learning models that the distribution and feature space of the training and testing data are the same, which is not always valid in real-world settings. For example, if a model is trained on synthesized images and encounters real camera images during testing, it may perform poorly due to the distribution shift or domain gap. As such, transfer learning based FSOD methods primarily focus on the inductive setting, where a model trained on a source task is aimed to enhance the learning of a different target task. In FSOD, the base task has a significant amount of data to narrow down the range of possible hypotheses, while the novel task utilizes limited examples to explore within the restricted hypothesis space and induce a robust predictive model.

Transfer learning offers multiple benefits, such as reducing the time and resources required to train a model from scratch by utilizing the knowledge gained from pre-training on large datasets. It can also be useful in situations with limited data availability or a domain shift between training and testing data. Additionally, finetuning pre-trained models for new tasks often leads to better performance than training a model from scratch. However, transfer learning has several limitations. Firstly, it assumes that the pre-trained model has learned relevant features that can be transferred to the target task, which may not always be true. Secondly, the pre-trained model may have biases or limitations that could negatively impact performance on the target task. Finally, the transfer of features may not be optimal for the new task, which can lead to suboptimal performance.

3.2.2 Related Works

LSTD: Low-Shot Transfer for Object Detection (LSTD) [Che18a] is the first FSOD framework. The architecture is designed to perform classification similar to two-stage detectors and localization similar to one-stage detectors, all while integrating additional regularization modules. First, LSTD performs classification like Faster R-CNN [Ren15] with the only difference of replacing the fully connected layers with convolutional layers. By doing so, the classifier can focus on the candidate objects in a coarse-to-fine manner, which should share more common foreground features than those in the background. Second, since LSTD divides the image into smaller grid cells, a set of candidate boxes is generated for each spatial location in the backbone convolutional feature map. The bounding boxes are then regressed using a smooth L1 loss to penalize the offset error between predicted and ground-truth bounding boxes. Third, to enhance knowledge transfer and localization, LSTD employs two techniques: background suppression and transfer-knowledge regularization. The latter adds the L2-norm of the activations obtained by projecting groundtruth bounding boxes into the convolutional feature map to the global loss, allowing the model to focus on suppressing background regions in target objects. On the other hand, background suppression uses base domain knowledge as a regularizer to finetune the novel target domain. Additionally, LSTD introduces an extra classification head to the target domain model, which classifies the classes of the source domain. This encourages more effective incorporation of base domain knowledge when learning novel classes.

TFA: In 2020, Wang et al., proposed a pioneering FSOD work, namely Two-Stage Finetuning Approach (TFA) [Wan20a], based on the Faster R-CNN [Ren15] detector. Based on the assumption that the base and novel tasks are highly related, the learned base representations from the backbone and RPN are considered transferable to novel classes without finetuning. In contrast, the localization features learned by the box predictor are class-specific and thus require finetuning on novel classes. To this end, TFA proposes a two-stage finetuning scheme. In the first phase, the base training is conducted on the base training dataset using the Faster R-CNN losses. Then, to avoid overfitting on limited data, TFA freezes the whole network except for the final box predictor layers in the second stage while leveraging a cosine-based similarity classifier with a reduced learning rate. Scaling the similarity scores between the feature and class weight vectors using the cosine similarity decreases the intra-class variance, improving the detection performance during novel training.

MPSR: The problem of scale variation in a few-shot setting is not addressed by TFA, as just adding an FPN to the backbone is not enough to compensate for the sparsity of samples available for the novel categories at different scales. To address this, Multi-Scale Positive Sample Refinement (MPSR) [Wu20a] proposes using object pyramids, where each object is extracted and resized to multiple scales. However, this approach is not directly applicable to the detection pipeline since standard object pyramids contain only a single instance in each image. The MPSR method addresses the scale variation problem by creating object pyramids, which involve cropping objects using their ground truth bounding box and resizing them to various scales. A positive sample refinement branch is then added, which selects feature maps from the FPN to feed into both the RPN and detection heads for refinement. Moreover, the MPSR branch computes objectness and classification scores to augment the RPN and RoI loss functions, respectively.

FSCE: Few-Shot Contrastive Encoding (FSCE) [Sun21] revisits the TFA method and allows the RPN and RoI head to be unfrozen without negatively affecting performance. This is done to learn more semantically meaningful information for the novel classes. Unlike TFA, FSCE utilizes a Faster R-CNN [Ren15] with a FPN to allow multi-scale feature learning. The authors provide critical insight that the RPN assigns low objectness scores to positive novel anchors at the start of the novel training phase, causing them to be eliminated during NMS. Consequently, a shortage of foreground proposals leads to a dominance of background features during the learning process.

Unlike TFA, FSCE introduces a stronger baseline unfreezing the RPN and RoI head with two modifications. First, the number of post-NMS proposals doubles for more positive novel anchors. Second, the number of sampled proposals for the RoI head is decreased by half to discard background features maintaining a balance between the foreground and background proposals. To

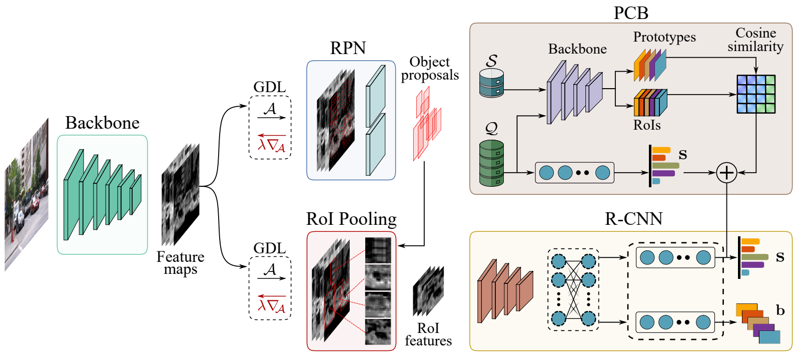


Figure 3.1: A detailed overview of the Decoupled Faster R-CNN (DeFRCN) framework. In the forward pass, the shared backbone features undergo affine transformations to different spaces. During the backward propagation, gradient scaling adjusts the degree of decoupling between the RPN and R-CNN. DeFRCN introduces the Prototypical Calibration Block (PCB), a metric-based scores refinement module to enhance the separation of classification and localization tasks during inference. PCB computes cosine similarity between predicted instance-level features and stored class prototypes to refine classification scores. The resulting similarity scores are used to perform a weighted average with the predicted classification score.

improve the learning of semantically rich information for novel classes in the FSCE framework, a contrastive branch is added in parallel to the classification and regression branches of the tunable RoI head. Specifically, the RoI features are projected to a lower dimensional feature space using a single MLP, and similarity scores are computed between the encoded object proposals. Subsequently, a contrastive objective is employed to increase the agreement between object proposals belonging to the same class while promoting the distinctiveness of proposals from different classes. This, in turn, allows the object proposal embeddings to form tighter clusters and have greater separation between different clusters in a projected hyper-sphere, leading to increased model generalizability in few-shot scenarios.

DeFRCN: Two-stage object detectors face a conflicting optimization problem between the class-agnostic RPN and the class-specific R-CNN through a shared backbone. While R-CNN requires translation-invariant features for the classification, translation-covariant features are required for the box regression, resulting in many low-quality IoU scores and reduced classification accuracy. These adversities are further exacerbated in few-shot learning scenarios due to the limited available samples. Moreover, the shared backbone in two-stage detectors seeks to extract robust and diverse features suitable for various downstream tasks since the RPN and R-CNN exchange knowledge through the shared backbone parameters [Ren15]. However, in FSOD, the RPN may confuse foreground and background during novel training. This is because proposals identified as background during the base training phase may become foreground during the novel finetuning phase. Although sharing convolutional layers improves the base performance, it overfits the base data, impairing its ability to transfer rapidly and effectively to the novel classes.

Decoupled Faster R-CNN (DeFRCN) [Qia21] exploits the insights mentioned above to improve the FSOD performance of a simple Faster R-CNN model. Concretely, DeFRCN decouples the learning tasks of the backbone, RPN, and R-CNN by modifying the backpropagated gradients. Central to the DeFRCN method is the introduction of the Gradient Decoupling Layer (GDL) module that modifies the gradients differently during the forward and backward passes. GDL uses an affine transformation layer with learnable channel-wise weights and bias during the forward pass to improve feature representations. During the backward pass, GDL multiplies the gradient from the subsequent layer by a positive constant lower than 1.0 to control the contribution of the backpropagated gradients to the backbone. DeFRCN inserts one GDL between the backbone and RPN and another between the backbone and R-CNN. Concretely, during the forward propagation, the feature from the shared backbone is transformed into different feature spaces through the affine transformation. Moreover, the decoupling degree is adjusted by rescaling the gradients during the backward propagation. It is important to note that the RPN gradients are killed during both the base and novel training phases since the RPN aims to learn class-agnostic features. Since the localization task gradients tend to force the backbone to learn translation-covariant features, it may negatively affect the translation-invariant classifier performance. The issue can be more severe in data-scarce scenarios due to the complexity of the model.

To improve the decoupling of classification and localization tasks during inference, DeFRCN introduces a metric-based scores refinement module called the Prototypical Calibration Block (PCB). This module comprises an ImageNet pre-trained classifier to extract a feature map for the input image. Additionally, the PCB includes a prototype bank consisting of class prototypes computed by averaging instance-level features. To refine the classification scores, the PCB calculates cosine similarity between the predicted instance-level features and the stored class prototypes. This similarity score is then used to perform a weighted average with the predicted classification score using a pre-defined hyperparameter. Because there are no shared parameters between the few-shot detector and the PCB module, the PCB preserves the quality of the translation-invariant features aimed at classification and better separates the classification and regression tasks within the R-CNN.

3.3 Meta-Learning-based

3.3.1 Preliminaries

Rather than improving the model predictions, meta-learning offers a peculiar learning paradigm to enhance the learning algorithm. Meta-learning gathers learning experiences throughout numerous episodes. Each episode $\boldsymbol{E}$ comprises an N-way-K-shot task $\boldsymbol{T} \in \mathcal{T}$, where $\mathcal{T}$ is the task space. A task $\boldsymbol{T} = \{\{\boldsymbol{S}^1, \dots, \boldsymbol{S}^{\mathrm{C}}\}, \boldsymbol{Q}\}$ is made up of C classes, featuring labeled support sets with K instances each. $\boldsymbol{Q}$ is a query image with objects belonging to the N classes. $\boldsymbol{S}^{\mathrm{c}} = \{\boldsymbol{S}^{\mathrm{c}}_1, \dots, \boldsymbol{S}^{\mathrm{c}}_K\}$ is a support set for the c$^{\text{th}}$ class. The k^{th} support image $\boldsymbol{S}^{\mathrm{c}}_k$ is a close-up of an object of class c cropped via the annotated bounding box.

To simulate the test-time scenario during training, meta-learning employs a two-stage training approach consisting of meta-training and meta-testing. During meta-training, episodes are generated with non-overlapping base tasks and objective functions. During meta-testing, episodes with novel tasks are utilized to update the inner base algorithm to enhance the outer objective. Formally, the meta-training phase is denoted as:

$$\omega^* = \arg\max_{\omega} \log p\left(\boldsymbol{Y}_b | \boldsymbol{X}_b, \omega\right). \tag{3.4}$$

The meta-parameters ω^* can be initial parameters, an optimization approach, or a learning model, depending on the utilized meta-learning technique. Meta-testing then leverages the acquired meta-knowledge ω^* to learn a novel task:

$$\Theta_n^* = \arg\max_{\Theta_n} \log p\left(\boldsymbol{Y}_n | \boldsymbol{X}_n; \Theta_n, \omega^*\right). \tag{3.5}$$

Meta-learning provides several advantages. Firstly, it enhances the generalization performance of the model, as it enables learning how to learn from a set of related tasks. Secondly, it can significantly reduce the amount of data required to train models for new tasks by enabling knowledge transfer between tasks. Thirdly, meta-learning proves beneficial in situations involving shifts in data distribution, such as domain shifts between training and testing data. This is due to the episodic training approach, which addresses distinct sub-tasks within each episode. This strategy drives the model to gather knowledge on how to solve various tasks rather than excessive overfitting to class-specific or domain-specific attributes. However, meta-learning also has limitations. First, it typically requires significant computational and memory resources. Secondly, identifying a suitable set of related tasks for the model to learn from can be difficult, and if the tasks are not sufficiently related, performance may degrade significantly. Thirdly, the hyperparameters and architecture design choices can highly influence the meta-learned knowledge, impeding effective generalization to new tasks.

3.3.2 Related Works

MetaYOLO: In 2018, Kang et al. introduced the first one-stage meta detector based on the YOLOv2 [Red17] architecture, MetaYOLO [Kan18]. The approach builds upon the assumption that the model has already been trained on adequate base data during the meta-training phase. In the meta-testing phase, with only a few support examples, the idea is to learn weight coefficients that will be used to perform a weighted average of the novel features. Given a query image, the backbone is a meta-feature learner extracting class-agnostic features. In parallel, a shallow CNN reweighting module accepts a single support image. It generates a class embedding and then performs channel-wise multiplication with the extracted meta-features to highlight the more relevant features. The features are then reweighted class-wise via a 1×1 depthwise convolutional layer and fed to the prediction layer to regress the classification scores, bounding box offsets, and objectness scores. More specifically, the model jointly trains the detection network and the reweighting module during the meta-training phase to ensure their coordination. In the meta-testing phase, the model is trained on base and novel classes while maintaining a balanced training by sampling base samples to match the number of available novel samples. The reweighting coefficients depend on the randomly sampled pairs during the training phase. To compute the final reweighting vector for a target class, the predicted K-shot samples are naively averaged. During inference, the reweighting module can be removed since it the model can perform detection without a support set.

MetaDet: As mentioned, CNNs start by learning low-level features (e.g., edges and corners) and more high-level features (e.g., faces) as the network gets deeper. Consequently, in meta-learning, the low-level base features are class-agnostic and thus more transferable to novel classes. In contrast, the high-level features are class-specific, meaning they are not directly transferable to novel classes and may need to be finetuned. Motivated by the insights above, Wang et al. proposed MetaDet [Wan19c], which leverages a few support examples to predict the class-specific parameters through a meta-learner model trained on sufficient database data.

Specifically, MetaDet adopts a Faster R-CNN detector where the RPN is considered the class-agnostic component with easily transferable features between base and novel classes. In contrast, the final fully-connected prediction layers learn class-specific features that require adaptation. MetaDet employs a weight prediction meta-model that inputs a few support images and learns a transformation needed for class-specific bounding box detection parameters. Concretely, the goal of MetaDet is to penalize the difference between the base-trained and predicted weights in addition to the detection loss. Moreover, MetaDet splits the meta-training into two phases for the class-agnostic and class-specific components, respectively. In the first phase, a base detector is trained on abundant base data and outputs the class-specific base parameters. In the subsequent phase, the base detector is finetuned in an episodic manner with base data samples by freezing the class-agnostic components and only training the class-specific ones. This phase is vital for learning the weight prediction meta-model. The meta-testing phase is normally conducted after initializing the class-agnostic components with the base parameters while randomly initializing the class-specific ones. The meta-model predicts the desired class-specific parameters for the novel classes during meta-testing. At inference time, the meta-model is totally omitted, and the model operates as a normal Faster R-CNN detector.

Meta R-CNN: Meta R-CNN [Yan19] modifies MetaDet by focusing the meta-learner on the R-CNN features rather than the whole image features. The Meta R-CNN model proposes a Predictor-head Remodelling Network (PRN) module that aims to meta-learn class-attentive vectors that can be used to exploit the RoI features. The PRN receives images with bounding boxes and learns class-attentive vectors to achieve this. These vectors use channel-wise soft attention on RoI features to adapt the R-CNN predictor heads, enabling them to detect objects belonging to the classes at hand. Specifically, Meta R-CNN inputs the few-shot support images to the PRN to compute the class attention vectors by averaging the computed class attention vectors. These vectors are then multiplied channel-wise with the RoI head features to attend specific classes. These attended feature maps are then passed to the final R-CNN predictor layers for object detection or segmentation. To supplement the Faster R-CNN loss, Meta R-CNN incorporates a meta-loss that employs a cross-entropy loss function to classify the predicted class-attentive vectors. By doing so, the model guarantees that the class attention vectors are diverse.

Attention-RPN: The metric-based FSOD is another line of work in meta-learning [Wan20b] that trains models to learn a feature space for comparing novel query images with a support set. This allows for object class identification. These methods use a similarity kernel to score the similarity between feature vectors. Higher scores indicate greater similarity, aiding in assigning labels without retraining. Similarity scores from comparing query and support set instances enable label assignment for query instances.

In 2020, Fan et al. proposed Attention-RPN [Fan20], a pioneering metric-based meta-learning FSOD approach. The Attention-RPN method, like previous two-stage methods, is built on the Faster R-CNN detector. Commonly, the RPN is trained as a class-agnostic component with the primary goal of generating candidate regions to facilitate the task of the subsequent R-CNN detector. In FSOD, the RPN should be able to eliminate further proposals not belonging to the given support set in each iteration. However, in most existing FSOD approaches, the RPN is not explicitly trained to consider the novel classes in the support set. As a result, the RPN may assign a high confidence score to a proposal that belongs to a class different from the ones in the support set, increasing false positives and thus impeding the overall detection performance.

To this end, Attention-RPN proposes an attention mechanism incorporating support information into the RPN. Specifically, the class-specific support instances features are average pooled and depth-wise cross-correlated with the query feature maps. The attended feature maps are then fed to the RPN to guide the proposals toward the relevant support classes. The Multi-Relation Head is the other main module in Attention-RPN, which, like the attention mechanism in the RPN, tries to fuse the support information with the instance query features. The goal is to improve the discriminative ability of the R-CNN and reduce confusion between classes. To achieve this, three attention heads are added to the detector: global-relation head, local-correlation head, and patch relation head. This helps the model learn robust feature embeddings at global, pixel, and patch levels.

Unlike previous FSOD works, Attention-RPN adopts a unique two-way contrastive training strategy. This strategy involves forming training triplets that consist of a query instance, a positive support instance, and a negative support instance. To avoid the background proposals from dominating the training, a predefined sampling scheme is used to balance the ratio of matching pairs between query proposals and support images. The network strives to minimize the distance between a positive support instance and a foreground proposal while pulling away from a background proposal. Moreover, it learns to pull away from a negative support instance while ignoring a background proposal from a different class.

FsDetView: Similar to Attention-RPN, Few-Shot Detection and Viewpoint Estimation (FSDetView) [Xia20] strives to incorporate support information but only before the final R-CNN predictor layers. FSDetView presents a unique feature aggregation module that combines query RoI instance features and support features using three techniques: concatenation of the query features by channel, channel-wise multiplication of the support and query features, and subtraction of the support features from the query features. The proposed aggregation method effectively minimizes the variance caused by the random selection of support data in FSOD.

ONCE: Open-ended CentreNet (ONCE) [Per20] is among the first works to tackle the FSOD in an incremental setting via meta-learning. In conventional batch training, the network parameters are updated after processing a batch of inputs. Instead, in incremental training, the parameters are updated as new data samples are presented to the model. ONCE uses a one-stage CenterNet [Dua19] detection model, providing a better balance between speed and accuracy than RetinaNet [Lin18] and YOLO-based [Red16, Red17] architectures. Moreover, CenterNet adopts a class-wise heatmap-based centroid prediction, making adding new classes straightforward and more suited for incremental learning. ONCE uses a two-stage meta-training approach. A feature extractor is trained on the abundant base data in the first stage. Then, in the second stage, a fixed feature extractor is used to train an object locator conditioned on a class-specific code, along with the meta detector, given a support set. During meta-testing, the ONCE model uses a few-shot support set of novel classes to generate object-specific weights. The object locator then uses these weights to detect objects in test images. This process occurs in a feed-forward manner without the need for further model training or adaptation. For a fair comparison with batch-based FSOD frameworks, ONCE can also operate in a non-incremental fashion.

CME: Class Margin Equilibrium (CME) [Li21] is a metric-based meta-learning FSOD approach. In the task of FSL, a performance trade-off exists between maximizing the distance between the representations of base classes and reducing the intra-class distance for novel representations. CME modifies upon

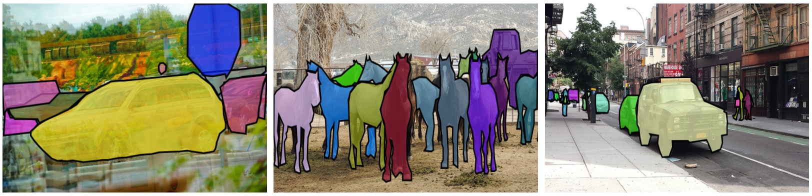


Figure 3.2: Examples from the MS-COCO dataset. Note that the segmentation mask is used for better visualization and is not utilized in this work.

the MetaYOLO framework to a support-query branch architecture similar to Attention-RPN. In base training, the architecture removes localization features to avoid interfering with class margins. It uses a fully connected layer to separate these features and maximize the class margin. During fine-tuning, the margin should be decreased to reconstruct novel classes using trained base features. To accomplish this, the discriminability of the prototypes is reduced via online data augmentation, which involves removing pixels that correspond to high gradient values. The model balances the class margin by maximizing margins during backpropagation and minimizing them during the forward pass.

3.4 Datasets

In FSOD and G-FSOD benchmarks, the two most commonly utilized datasets are: MS-COCO [Lin14] and PASCAL-VOC [Eve10].

3.4.1 MS-COCO Dataset

The MS-COCO [Lin14] dataset is a widely utilized and influential benchmark dataset in the field of computer vision. It offers a vast and diverse collection of images, encompassing 80 different categories, which serves as a valuable resource for advancing research in object recognition, detection, segmentation, and captioning. The dataset consists of an extensive set of 330k images, obtained from diverse sources like the internet and professional photographers, ensuring a wide range of visual contexts and scenarios. Various examples are

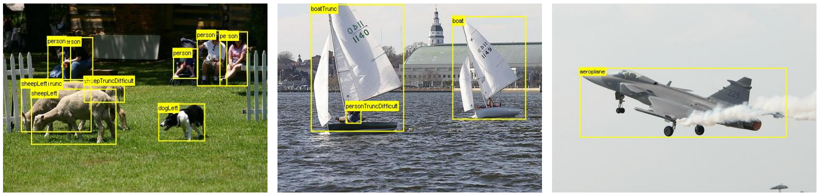


Figure 3.3: Examples from the PASCAL-VOC dataset.

presented in Figure 3.2. Most of the images in the dataset are annotated with: (1) class labels, indicating the corresponding object category, (2) bounding box coordinates that specify the spatial extent of the objects within the image, and (3) for a subset of images, the dataset also provides pixel-level segmentation masks, enabling more detailed research tasks like instance segmentation.

Due to the inconsistency of the utilized MS-COCO versions in the FSOD literature [Wan20a, Qia21, Fan20, Xia20], each proposed method has accordingly adopted the relevant version. Specifically, the three proposed transfer learning approaches utilize the 2014 version, while the introduced meta-learning approach in the last chapter employs the 2017 version. In FSOD, the dataset is divided into two parts: 60 base classes that do not overlap with the PASCAL-VOC categories, and 20 novel classes. The testing phase utilizes 5k images from the validation set, while the remaining images are used for training the base and novel classes. The MS-COCO benchmark employs three different settings based on the number of shots: 5, 10, and 30 shots. For instance, in a 10-shot setting, the model is finetuned using 10 labeled instances for each novel class. Therefore, considering the 20 novel classes, there would be a total of 200 novel training instances available for finetuning.

3.4.2 PASCAL-VOC Dataset

The PASCAL-VOC [Eve10] dataset is another well-known dataset that is extensively employed as a benchmark in various computer vision for tasks. Like the MS-COCO dataset, most images in the PASCAL-VOC dataset are annotated with class labels, bounding box coordinates, and, in some cases, pixel-level segmentation masks. PASCAL-VOC encompasses 20 object categories, including commonly encountered classes such as person, car, and dog. Figure 3.3 displays different data samples.

For FSOD tasks, the PASCAL-VOC dataset is divided into three distinct sets, each comprising 20 categories. These categories are then randomly split into 15 base and 5 novel classes. The novel classes are organized differently in each split:

- Novel Split 1: bird, bus, cow, motorbike, and sofa.
- Novel Split 2: aeroplane, bottle, cow, horse, and sofa.
- Novel Split 3: boat, cat, motorbike, sheep, and sofa.

Following the well-defined FSOD datasplits [Wan20a, Qia21], training, the base and novel data are sampled from both the 2007 and 2012 train/val sets during training. The testing is then performed on the 2007 test set. Different from MS-COCO, the PASCAL-VOC benchmark is provided for various numbers of shots per class, specifically 1, 2, 3, 5, 10, and 30 shots.

3.5 Evaluation Metrics

To analyze the performance of both base and novel classes, two evaluation metrics are employed: base Average Precision (bAP) and novel Average Precision (nAP). The bAP measures the AP (Section 2.2.4) only for the base classes, while the nAP reports the AP for the novel classes. The overall performance on both classes is denoted by AP. Furthermore, the AP metrics can be measured for different IoU thresholds. For example, bAP50 and bAP75 represent the bAP performance when considering predicted bounding boxes with an IoU score of at least 50% or 75% with the ground truth bounding boxes, respectively. These metrics provide a better insight into the localization accuracy of the detector model.

Similarly, the few-shot evaluation metrics also include the base Average Recall (bAR) and novel Average Recall (nAR). These metrics quantify the proportion of true positive detections among all the ground truth positive instances, considering the limited number of labeled instances per class during training. The bAR and nAR metrics provide insights into the ability of the model to recall relevant objects from base and novel classes within the few-shot setting, respectively.

3.6 Discussion

This chapter has explored FSOD, focusing on several key elements contributing to understanding this dynamic field. Firstly, the chapter began with an in-depth problem formulation of FSOD, outlining the challenges and nuances of training object detectors with limited examples. This formalization paved the way for a comprehensive investigation into two main paradigms: transfer learning-based and meta-learning based approaches.

A systematic literature review was conducted for transfer learning, dissecting various approaches and strategies to tackle the FSOD problem. One approach that has been extensively reviewed is the DeFRCN framework, as it serves as a cornerstone for multiple methodologies proposed in this dissertation. Parallel to the transfer learning paradigm, the chapter delved into meta-learning, which holds immense potential for rapid detector adaptation in FSOD scenarios. The formulation of the FSOD problem within a meta-learning framework was presented, paving the way for exploring various approaches and strategies that harness meta-learning principles. Subsequently, a comprehensive overview of the utilized FSOD datasets was provided, offering detailed insights into MS-COCO and PASCAL-VOC datasets. Finally, the FSOD evaluation metrics were introduced.

4 Concept

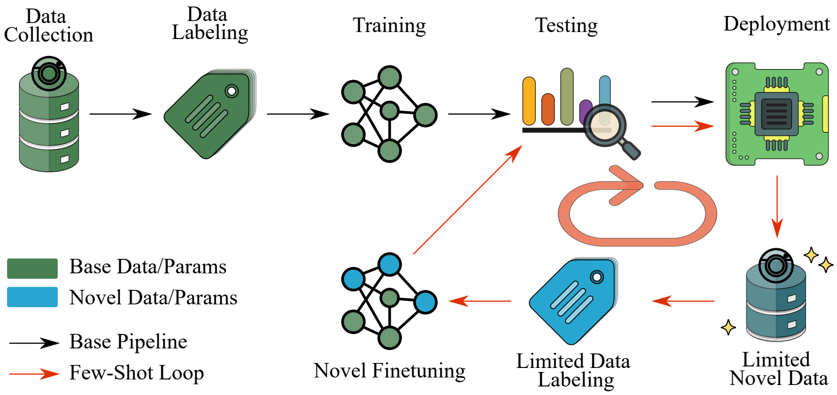


Figure 4.1: An illustration of the overall concept. When incorporating the few-shot detection paradigm into the ML production pipeline, overcoming the challenges posed by extensive data acquisition, labeling, and prolonged retraining periods becomes feasible. As a result, this not only facilitates faster deployment but also allows for the handling of new objects with limited data that are encountered after the initial deployment.

The main goal of this dissertation is to design various deep learning-based object detection frameworks that can effectively handle situations with limited data available for new classes. This circumvents the need for extensive labeling and retraining process from scratch. More specifically, the work involves tackling various challenges within this context: developing an embedded-friendly model, alleviating forgetting base data, considering scenarios where base data is unavailable due to privacy or memory constraints, and detecting objects in unseen domains using limited source data.

A typical ML production pipeline is presented in the upper part of Figure 4.1 comprising five main steps: data collection, labeling, model selection and training, model testing, and deployment on edge devices. However, challenges arise in data acquisition and labeling, particularly when detecting new classes that were not seen during the training phase, resulting in significant labor, time, and cost implications. FSOD promises a potential solution by leveraging prior knowledge from abundant base data to rapidly learn new classes. Then, when encountering new objects, only a handful of data needs to be acquired and labeled. Next, the model is rapidly finetuned, with only a subset of the parameters being updated while the rest remain fixed. Utilizing the few-shot detection paradigm significantly saves time, cost, and labor while enabling faster deployment. The discussed few-shot detection loop, highlighted in red, is depicted in Figure 4.1.

FSOD can be categorized into transfer learning and meta-learning approaches. In transfer learning, pre-trained models on abundant base classes are finetuned using limited labeled data for novel classes. On the other hand, meta-learning involves training the model on diverse few-shot tasks to acquire generalized knowledge, enabling rapid adaptation to new classes during inference. While traditional meta-learning-based approaches exhibit superior detection performance, they are more complex and resource-intensive. However, recent transfer learning approaches are closing the performance gap with simpler and lighter designs, thus bringing the two paradigms closer together.

Dealing with a limited number of samples for these new classes raises important learning questions: *how to efficiently learn these new classes with only a handful of labeled examples without overfitting*? *How to prevent forgetting the previously learned classes*? And finally, *how to design such an efficient system while abiding by low computational resources*? Addressing these questions can enhance the detection pipeline, making it more efficient, rapid, and adaptable.

To this end, the central concept behind this work is providing solutions to the existing challenges in FSOD pipelines. This dissertation explores multiple detection pipelines instead of just one, catering to various requirements, available resources, and application scenarios. The conceptual depiction in Figure 4.2 illustrates the contributions made in both learning paradigms. Specifically,

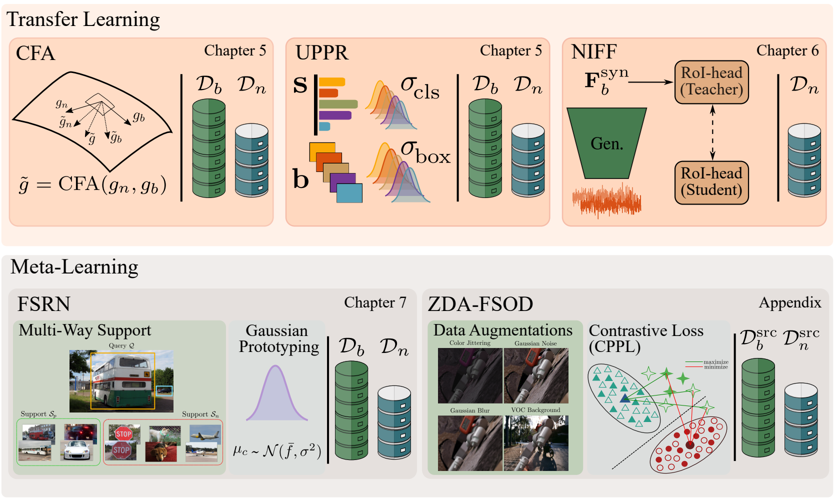


Figure 4.2: A conceptual representation of the primary contributions made in each chapter of this dissertation, all of which serve the overarching concept.

within transfer learning-based approaches, DeFRCN [Qia21] detector, known for its superior performance and reliability within this learning paradigm, has been utilized as a base framework, and several methods have been introduced to tackle the challenge of forgetting when learning new classes with limited data. Figure 4.2 (top) first illustrates the Constraint-based Finetuning Approach (CFA) [Gui22b], which introduces a new gradient update rule. This rule dynamically adjusts the weights of the base and novel gradients, ensuring less forgetting and a more effective knowledge transfer between base and novel classes. Next, the Uncertainty-based Progressive Proposal Refinement (UPPR) framework utilizes predictive uncertainties, such as aleatoric and epistemic uncertainties, to mitigate forgetting in G-FSOD and improve the overall detection performance. In cases where base data is unavailable during novel training, the Neural Instance Feature Forging (NIFF) [Gui23b] method adopts a separate feature generator with minimal memory usage. It learns to generate base instance-level features by aligning class-specific statistics and subsequently replays these forged features to maintain the base knowledge.

While transfer learning approaches demonstrate competitive detection performance, they necessitate finetuning novel data. In contrast, meta-learning few-shot detectors can directly infer new classes by utilizing a support set of new objects without the need for retraining. Figure 4.2 (bottom) provides an overview of the two meta-learning frameworks introduced in this thesis. The first framework, Few-Shot RetinaNet (FSRN) [Gui23a], is a meta-learning-based one-stage detector more compatible with embedded systems than the two-stage detectors. FSRN aims to reduce the computational and memory requirements of the system by various architectural considerations. It significantly improved performance compared to the current state-of-the-art one-stage meta detectors. This improvement can be attributed to several factors, including the introduction of a multi-way support training strategy that enhances the number of foreground samples for dense meta-detectors during training, early multi-level feature fusion that encompasses the entire anchor area, and the use of two augmentation techniques on the query and support images to enhance transferability. FSRN provides an embedded-friendly solution by significantly reducing the number of model parameters, FLOPS, and inference time.

The second meta-learning-based approach aims to detect new objects in a target domain when only limited data from a source domain is available. This approach is called Zero-Shot Domain Adaptive FSOD (ZDA-FSOD) [Gui22a]. It utilizes various domain randomization techniques, such as pixel-level domain randomization, on both the query and support data for the novel objects. Notably, it does not require the generation of additional data from a simulator and can be applied to different domain gaps. To encourage the learning of domain-agnostic, class-specific feature embeddings, a new contrastive loss is introduced in ZDA-FSOD. This loss maximizes the mutual information between foreground proposals from the query image and the associated class embedding. It achieves this by bringing the embeddings closer to each other in the feature space while simultaneously being further apart from a negative class embedding. Note that ZDA-FSOD is presented in the appendix of this thesis.

5 Replay-based G-FSOD

While FSOD focuses solely on the novel performance, Generalized Few-Shot Object Detection (G-FSOD) addresses the more challenging task of jointly learning to detect both base and novel classes, optimizing for both. However, when learning new classes, models often encounter a phenomenon called catastrophic forgetting. This phenomenon refers to the tendency of the model to forget the previously learned classes, resulting in degraded detection performance for those classes. Retaining knowledge of previous tasks is vital for the reliable operation of modern perception systems. For instance, a pick-and-place robot needs to remember how to grasp both base and novel objects to avoid hazardous failures.

This chapter proposes new methods to alleviate forgetting in G-FSOD models using limited base and novel data. It starts with a comprehensive literature review of existing G-FSOD approaches, their strategies to address forgetting, and relevant Continual Learning (CL) techniques that share a similar interest. Next, a constrained optimization method for G-FSOD method called Constraint-based Finetuning Approach (CFA) [Gui22b] is proposed. CFA introduces a new update rule that guides the model gradients toward a better optimum of less forgetting and improved detection performance.

Moreover, learning novel classes with limited data amplifies predictive uncertainties, contributing to a decline in detection performance and catastrophic forgetting. To this end, this chapter further introduces an uncertainty-based method called Uncertainty-based Progressive Proposal Refinement (UPPR). It leverages uncertainty estimation techniques to refine the object proposals generated by the detection model. To evaluate the effectiveness of the proposed methods, thorough comparison and ablation experiments are conducted.

5.1 Literature Review

5.1.1 Generalized Few-Shot Object Detection

G-FSOD, a sub-discipline of FSOD that seeks the detection of base and novel classes, has been gaining prominence in recent years [Wan20a, Per20, Fan21]. The first two approaches [Wan20a, Per20] have been reviewed in Section 3.3.2 and Section 3.2.2, respectively. While TFA [Wan20a] is considered to mitigate forgetting by performing finetuning on both base and novel classes, ONCE [Per20] addresses the issue in an incremental setting by employing a meta-learning approach with a CenterNet [Dua19] detector. The core concept of ONCE involves meta-learning a class code generator that progressively learns to synthesize a class code for the novel classes. However, none of the methods mentioned above explicitly tackle the catastrophic forgetting of base classes.

In contrast, Retentive R-CNN [Fan21] presents an alternative approach that employs transfer learning to explicitly address the issue of forgetting the base classes. It retains the learned knowledge by leveraging both the pre-trained base model and the novel model for detecting both classes in a student-teacher distillation fashion. Nonetheless, it is important to note that this approach comes with associated computational and memory expenses. Additionally, it increases both the novel training and inference time.

5.1.2 Replay-based Continual Learning Methods

A related field focusing similarly on reducing catastrophic forgetting is Continual Learning (CL). Recently, CL approaches have gained recognition in various computer vision tasks, such as image classification and object detection. The main goal of CL methods is to accumulate knowledge from previous problems and quickly learn new tasks without forgetting. There are three main approaches in CL [Lan21a]: replay-based, regularization-based, and parameter isolation methods. Replay-based methods [Reb17, Kam17, Rol19, Ise18, Cha19b, Lan21b, Atk18, Lav18, Ram20, Lop17, Cha19a, Rie19, Alj19] involve storing or generating samples from previous tasks to replay while learning a new task. Regularization-based methods [Kir16, Alj18, Lee17, Zen17, Liu18b, Cha18, Li16, Jun16, Tri17, Zha20a] introduce a regularization term to the objective function to incorporate knowledge from previous tasks without storing data. Parameter isolation methods [Mal18a, Rus16, Mal18b, Ser18, Alj17, Xu18, Ros20] assign separate model parameters to each task to avoid forgetting. However, CL methods have not yet been explored in the context of G-FSOD.

This chapter leverages replay-based methods to mitigate forgetting in G-FSOD. The replay-based continual learning approaches can be divided into three primary categories:

- **Rehearsal methods** [Reb17, Rol19, Ise18, Cha19b, Lan21b] store real samples from previous tasks and replay them while learning new tasks. When training on a new task, these stored samples are combined with the current task data to enhance the diversity of the training set and inject knowledge from past tasks. While rehearsal methods are simple and efficient, they require storing previous task samples, which can be memory-intensive.
- **Pseudo-rehearsal methods** [Kam17, Atk18, Lav18, Ram20] overcome the memory constraint of storing real samples by generating synthetic samples based on the previous knowledge of the model. Instead of saving real samples, these methods utilize generative models like GANs or VAEs to produce synthetic samples that exhibit similar characteristics to the original data distribution. These synthetic samples are then replayed while learning new tasks to reinforce past experiences. While pseudo-rehearsal methods are more memory-efficient than rehearsal methods, their effectiveness relies on the ability of the generative models to accurately capture the underlying data distribution.
- **Constrained optimization methods** [Lop17, Cha19a, Rie19, Alj19] offers an alternative solution to the rehearsal methods above, which may be prone to overfitting the stored sample subset and are constrained by joint training. In contrast, constrained optimization

provides more flexibility for backward/forward transfer in continual learning scenarios.

Due to the limited data in G-FSOD, constrained optimization methods are chosen to be further explored to alleviate forgetting in G-FSOD scenarios and avoid overfitting the limited replayed data.

Although in G-FSOD, the model has the advantage of being finetuned with base classes to avoid any performance drop, aligning with the fact that the few-shot samples cannot sufficiently represent the base data distribution, leading to overfitting on the limited examples. This has been demonstrated in previous works [Wan20a, Fan21, Qia21], where better performance is achieved on the base task but with a degradation in the novel task. In this chapter, the aim is to bridge the performance gap between G-FSOD and FSOD. Specifically, the goal is to develop a constraint optimization method to alleviate catastrophic forgetting without hindering the performance of novel classes.

5.1.3 Uncertainty Estimation for Object Detection

Predictive uncertainties, as described in the literature [Ken17, Gaw23], are commonly divided into aleatoric and epistemic uncertainties. Aleatoric uncertainty pertains to the inherent variability in the data, such as sensor noise, and is typically managed by incorporating it as learnable parameters associated with the model's predicted outputs. In the context of OD, these parameters can specifically account for aleatoric uncertainty related to class probabilities or bounding box coordinates, as demonstrated in previous research [Kra19, Har20].

Epistemic uncertainty, conversely, encompasses uncertainty arising from limited knowledge or a scarcity of training data. In OD, addressing epistemic uncertainty often involves integrating dropout techniques during the model training phase [Nit14], such as the approach employed in Bayesian YOLO and BayesOD [Kra19, Har20]. During training, a subset of neurons is randomly deactivated, effectively creating an ensemble of models. By analyzing the variation among predictions generated by these diverse models, one can approximate the level of epistemic uncertainty within the model. Monte

Carlo Dropout (MC-Dropout) [Gal16] extends this method during inference by performing multiple forward passes with dropout enabled and subsequently averaging the resulting predictions. It is important to note that epistemic uncertainty tends to be more pronounced, particularly when dealing with the challenge of learning novel classes with only limited labeled instances.

Nevertheless, it is worth highlighting that while predictive uncertainties have found application in conventional OD settings [Kra19, Har20, Fen18, Wir19], they have yet to be systematically addressed in the context of Few-Shot Object Detection (FSOD) or Generalized Few-Shot Object Detection (G-FSOD) scenarios.

5.2 Constraint-based Finetuning Approach

5.2.1 Revisiting GEM-based Algorithms

Drawing inspiration from gradient-based replay-based CL approaches [Lop17, Cha19a], the proposed approach, known as CFA, modifies the search strategy to discover optimal parameters that enhance generalization across the base and novel tasks. This method offers two advantages: it does not require specific data augmentations or model modifications and can be easily integrated into various detectors regardless of their architecture. Specifically, CFA regularizes the gradient update using a subset of the base dataset stored in the episodic memory, similar to the approach taken in A-GEM [Cha19a].

Episodic memory-based approaches [Lop17, Cha19a, Rie19, Alj19] mitigate catastrophic forgetting by maintaining an episodic memory, denoted as $\mathcal{M}$. These approaches prevent an increase in losses for previous tasks and facilitate a decrease in losses, resulting in positive backward transfer. Meaning that the performance on previous tasks can be improved while learning new ones. Instead of optimizing for all samples in the episodic memory as originally proposed [Lop17], A-GEM [Cha19a] ensures that the average episodic memory loss does not increase over a mini-batch of samples from the episodic memory.

To begin with, an episodic memory-based constraint optimization algorithm is formulated within the context of G-FSOD. The episodic memory, denoted as $\mathcal{M}_b$, is utilized to store a randomly sampled subset of K few-shot examples from each base class, which are drawn from the dataset $\mathcal{D}_b$. The number of few-shot examples K is selected to match the number of novel few-shot examples in $\mathcal{D}_n$. Unlike the conventional CL paradigms, the training is conducted in batches, allowing the data to be seen multiple times rather than being handled one by one on-the-fly. Additionally, the episodic memory remains static during the novel training phase, indicating that no further samples are added.

The finetuning process is executed as follows: Firstly, a mini-batch is randomly selected from the base episodic memory $\mathcal{M}_b$ to compute the base gradient $\boldsymbol{g}_b$. Then, another mini-batch is sampled from the novel dataset $\mathcal{D}_n$, and the novel gradient $\boldsymbol{g}_n$ is computed. Following the definition provided in previous works [Lop17, Cha19a], a positive knowledge transfer is achieved when the angle between $\boldsymbol{g}_b$ and $\boldsymbol{g}_n$ is acute. If this constraint is satisfied, $\boldsymbol{g}_n$ is directly backpropagated. However, if the constraint is not met, $\boldsymbol{g}_n$ is projected onto a region in the hypothesis space closer to the base task gradients, determined by $\boldsymbol{g}_b$, and then backpropagated through the model.

Formally, the constraint optimization problem for G-FSOD is denoted by:

$$\begin{aligned} \text{minimize}_\theta \quad & \mathcal{L}(h_\theta(\boldsymbol{x}), \boldsymbol{y}) \\ \text{subject to} \quad & \mathcal{L}(h_\theta, \mathcal{M}_b) \leq \mathcal{L}(h_\theta^b, \mathcal{M}_b), \end{aligned} \tag{5.1}$$

where $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{D}_n$. $\boldsymbol{x}$ is the input image with the associated label $\boldsymbol{y}$. h_θ^b is the pretrained model on the base dataset $\mathcal{D}_b$.

The loss function using the base episodic memory $\mathcal{M}_b$ is given by:

$$\mathcal{L}(h_\theta, \mathcal{M}_b) = \frac{1}{|\mathcal{M}_b|} \sum_{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \mathcal{M}_b} \mathcal{L}(h_\theta(\boldsymbol{x}_i), \boldsymbol{y}_i), \tag{5.2}$$

where $\mathcal{L}$ is the standard training loss function from Faster R-CNN [Ren15]:

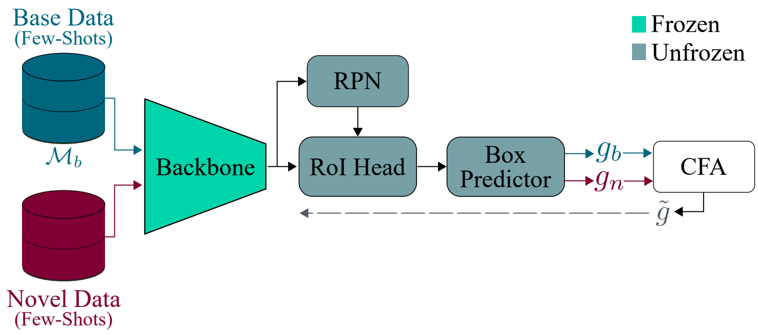


Figure 5.1: A depiction of the finetuning stage using CFA on Faster R-CNN [Ren15]. The base and novel gradients are computed from each mini-batch. CFA then determines the final gradient update rule, which is applied to backpropagate on the unfrozen components of the model. This ensures that the gradients of the novel task do not stray or deviate significantly from the gradients of the base task.

$$\mathcal{L} = \mathcal{L}_{\text{RPN}} + \mathcal{L}^{\text{cls}} + \mathcal{L}_{\text{reg}}. \tag{5.3}$$

$\mathcal{L}_{\text{RPN}}$ is the RPN loss function. $\mathcal{L}^{\text{cls}}$ and $\mathcal{L}_{\text{reg}}$ are the classification and bounding box regression losses, respectively.

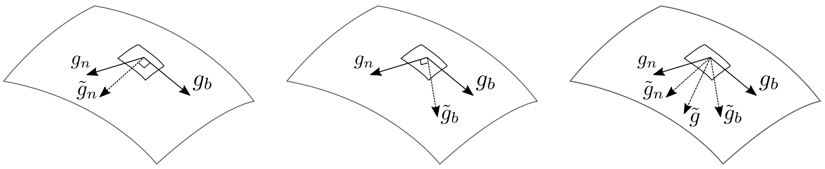
Similar to A-GEM [Cha19a], the optimization problem described in Equation 5.1 can be simplified into a Quadratic Programming (QP) problem:

$$\begin{aligned} \text{minimize}_{\tilde{\boldsymbol{g}}_n} \quad & \frac{1}{2}||\boldsymbol{g}_n - \tilde{\boldsymbol{g}}_n||_2^2 \\ \text{subject to} \quad & \tilde{\boldsymbol{g}}_n^\top \boldsymbol{g}_b \geq 0. \end{aligned} \tag{5.4}$$

$\tilde{\boldsymbol{g}}_n$ is the projected novel gradient. The closed-form gradient update rule can be denoted as:

$$\tilde{\boldsymbol{g}}_n = \boldsymbol{g}_n - \frac{\boldsymbol{g}_n^\top \boldsymbol{g}_b}{\boldsymbol{g}_b^\top \boldsymbol{g}_b} \cdot \boldsymbol{g}_b. \tag{5.5}$$

This indicates that the projection of the novel gradient orthogonal to the base gradient only occurs when the A-GEM constraint is violated.



(a) Vanilla A-GEM **(b)** Intermediate gradient **(c)** CFA gradient

Figure 5.2: The gradient update process for vanilla A-GEM [Cha19a] and the proposed CFA is visualized. Figure 5.2a illustrates the gradient update for the novel task using A-GEM, where the novel task gradient is projected orthogonally to the base task gradient. In Figure 5.2b, the solution for the proposed constraint is depicted, where the base task gradient is projected at a right angle to the novel task gradient. Finally, Figure 5.2c showcases the final gradient update for the CFA algorithm.

5.2.2 Methodology

In the context of G-FSOD, A-GEM as a finetuning approach offers improved regularization to the learning process, effectively preventing early overfitting. However, relying solely on the mentioned constraint may hinder knowledge transfer between tasks for two main reasons. First, the base gradient is only backpropagated when a violation occurs, resulting in limited influence during finetuning when learning the novel classes. Second, orthogonally projecting the novel gradient is overly restrictive for promoting diverse feature learning in the novel task.

Motivated by the observations above, it is proposed to minimize the angle between $\boldsymbol{g}_b$ and $\boldsymbol{g}_n$ instead of always orthogonally projecting $\boldsymbol{g}_n$ in case of a violation. The CFA algorithm is derived from a joint optimization problem that considers both tasks, including an additional constraint to account for the performance on base categories. In CFA, the backpropagated gradients are influenced by both the novel and base gradients rather than relying solely on the novel gradient. The scheme is illustrated in Figure 5.1. If a violation occurs, $\boldsymbol{g}_n$ is projected orthogonally onto $\tilde{\boldsymbol{g}}_n$ with respect to $\boldsymbol{g}_b$, while $\boldsymbol{g}_b$ is, in turn, projected orthogonally onto $\tilde{gb}$ with respect to $\boldsymbol{g}_n$.

Algorithm 5.2.0 CFA

```
1:  procedure TRAIN($f_\theta, \mathcal{D}_b, \mathcal{D}_n$)
2:      $\mathcal{M}_b \sim \mathcal{D}_b$
3:      for $n_{epoch} = 1, \ldots, N_{epoch}$ do:
4:          for $(x_n, y_n)$ in $\mathcal{D}_n$ do
5:              $(x_b, y_b) \sim \mathcal{M}_b$
6:              $g_b \leftarrow \nabla_\theta \mathcal{L}(f_\theta(x_b), y_b)$
7:              $g_n \leftarrow \nabla_\theta \mathcal{L}(f_\theta(x_n), y_n)$
8:              if $g_n^\top g_b \geq 0$ then
9:                  $\tilde{g} \leftarrow \frac{g_n + g_b}{2}$
10:             else
11:                 $\tilde{g} \leftarrow \frac{1}{2}\left(1 - \frac{g_n^\top g_b}{g_b^\top g_b}\right) \cdot g_b + \frac{1}{2}\left(1 - \frac{g_b^\top g_n}{g_n^\top g_n}\right) \cdot g_n$
12:             end if
13:             $\theta \leftarrow \theta - \eta \tilde{g}$
14:         end for
15:     end for
16:     return $f_\theta$
17: end procedure
```

Mathematically, our proposed constrained optimization problem is as follows:

$$\begin{aligned} \text{minimize}_{\tilde{\boldsymbol{g}}_b, \tilde{\boldsymbol{g}}_n} \quad & \frac{1}{2}||\boldsymbol{g}_n - \tilde{\boldsymbol{g}}_n||_2^2 + \frac{1}{2}||\boldsymbol{g}_b - \tilde{\boldsymbol{g}}_b||_2^2 \\ \text{subject to} \quad & \tilde{\boldsymbol{g}}_n^\top \boldsymbol{g}_b \geq 0, \\ & \tilde{\boldsymbol{g}}_b^\top \boldsymbol{g}_n \geq 0. \end{aligned} \tag{5.6}$$

The projected gradient updates for the base and novel tasks are denoted as $\tilde{\boldsymbol{g}}_b$ and $\tilde{\boldsymbol{g}}_n$, respectively. By solving the aforementioned constrained optimization problem using the Lagrange multipliers approach, the gradient update rules can be derived as:

$$\tilde{\boldsymbol{g}}_n = \boldsymbol{g}_n - \left(\frac{\boldsymbol{g}_n^\top \boldsymbol{g}_b}{\boldsymbol{g}_b^\top \boldsymbol{g}_b}\right) \cdot \boldsymbol{g}_b, \tag{5.7}$$

$$\tilde{\boldsymbol{g}}_b = \boldsymbol{g}_b - \left(\frac{\boldsymbol{g}_b^\top \boldsymbol{g}_n}{\boldsymbol{g}_n^\top \boldsymbol{g}_n}\right) \cdot \boldsymbol{g}_n. \tag{5.8}$$

Rather than conducting two separate gradient updates, a single update rule can be achieved by taking the average of $\tilde{\boldsymbol{g}}_b$ and $\tilde{\boldsymbol{g}}_n$:

$$\tilde{\boldsymbol{g}} = \frac{1}{2}\left(1 - \frac{\boldsymbol{g}_n^\top \boldsymbol{g}_b}{\boldsymbol{g}_b^\top \boldsymbol{g}_b}\right) \cdot \boldsymbol{g}_b + \frac{1}{2}\left(1 - \frac{\boldsymbol{g}_b^\top \boldsymbol{g}_n}{\boldsymbol{g}_n^\top \boldsymbol{g}_n}\right) \cdot \boldsymbol{g}_n. \tag{5.9}$$

Specifically, the equation above can be regarded as an adaptive re-weighting of the base and novel gradients during the finetuning stage, balancing their contribution to the final projected gradients. The full derivation of the CFA update rule is presented in Appendix A.1. The advantage of the CFA algorithm, as depicted in Algorithm 5.2.0, can be attributed to two key factors. Firstly, the base gradients consistently contribute to the finetuning process in contrast to previous works [Lop17, Cha19a]. Secondly, the algorithm seeks to determine the optimal direction of backpropagation by weighting each gradient while ensuring that the angle between the last gradient update, $\tilde{\boldsymbol{g}}$, and the base gradient, $\boldsymbol{g}_b$, remains less than $90°$. Figure 5.2 shows a visual representation illustrating the distinctions between the gradient update rules of A-GEM and CFA.

5.2.3 Experimental Evaluations

Implementation Details

Faster R-CNN [Ren15] is adopted as the primary detection framework, utilizing a ResNet-101 backbone [He16] and a FPN [Lin17]. For the base training, the learning rate is set to 0.02, while for the novel training, it is set to 0.001. The model optimization uses SGD with a momentum of 0.9 and a weight decay of 0.0001. The batch size is set to 16, utilizing four Nvidia GeForce 1080Ti GPUs.

Similar to TFA [Wan20a], the evaluation of our method includes a fully-connected base classifier (CFA w/fc) and a cosine similarity-based box classifier (CFA w/cos). Additionally, CFA is applied to the DeFRCN [Qia21] approach (CFA-DeFRCN), following the original hyperparameters described in the paper [Qia21]. In contrast to CFA w/fc and CFA w/cos, CFA-DeFRCN does not incorporate a FPN, similar to the baseline DeFRCN [Qia21].

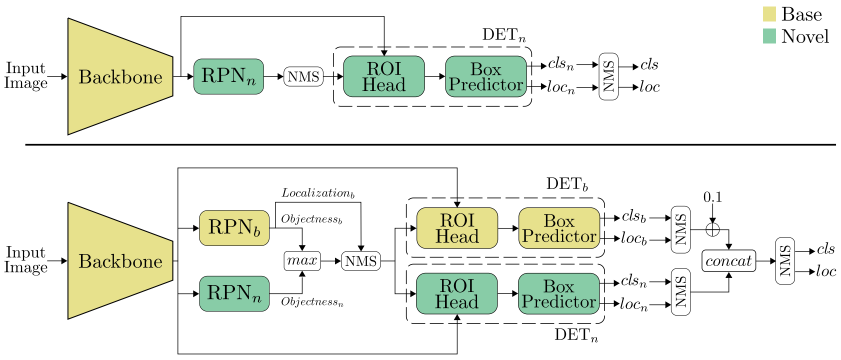


Figure 5.3: Top: Illustration of the single model inference. **Bottom:** A detailed overview of the ensemble model evaluation protocol proposed by Retentive R-CNN [Fan21].

Evaluation Protocols

Retentive R-CNN [Fan21] introduced a model-growth-based evaluation protocol that involves utilizing the base-trained RPN (RPN$_b$) and base-trained detection head (DET$_b$) alongside the finetuned novel RPN (RPN$_n$) and novel detector (DET$_n$) during inference. The protocol includes generating proposals from RPN$_b$ and RPN$_n$ based on the maximum objectness score. These proposals are then fed to DET$_b$ and DET$_n$, where detections from DET$_b$ are favored for base categories $\mathcal{C}_b$ using non-maximum suppression. For a fair comparison, our methods (CFA w/fc, CFA w/cos, and CFA-DeFRCN) are evaluated using this protocol as well.

Figure 5.3 illustrates the evaluation protocols employed in the study. The RPN$_n$ and DET$_n$, which are finetuned with a few shots from the novel data while keeping the backbone frozen, are used for single model inference. The evaluation process is as follows:

1. The input image is fed to the backbone.
2. The RPN$_n$ generates the proposals.
3. Proposals with IoU scores below a predefined threshold are discarded using NMS.

4. The novel detection head DET_n produces the novel classification logits (cls_n) and bounding boxes offsets (loc_n).
5. The final predictions are obtained by applying NMS to filter the outputs.

In contrast, the ensemble inference model incorporates the RPN_b and DET_b from the base-trained model. The inference process is as follows:

1. The input image is fed to the backbone.
2. The image features are fed to both RPN_b and RPN_n to compute the base and novel objectness logits, O_b and O_n, respectively.
3. The maximum between O_b and O_n is used in NMS along with the bounding boxes from RPN_b.
4. The filtered proposals are then fed to both DET_b and DET_n to obtain classification logits and bounding boxes.
5. The predictions from the detectors are separately subjected to NMS, with a bonus of 0.1 added to cls_b.
6. Finally, the outputs from both detectors are concatenated and passed through another NMS to obtain the final predictions.

It is important to note that the ensemble models were not used during finetuning, unlike in Retentive-RCNN [Fan21]. Instead, a single model was finetuned, and both the base and finetuned models were utilized during inference.

Quantitative comparisons are performed against transfer-learning [Wan20a, Wu20a, Fan21, Qia21] and meta-learning based [Yan19, Kan18, Per20, Fan20, Xia20] methods under the G-FSOD setting. Additionally, comparisons are made to Retentive-RCNN [Fan21] using their evaluation protocol. The results for the various proposed model settings (CFA w/fc, CFA w/cos, and CFA-DeFRCN) are reported.

Table 5.1: The G-FSOD results in the $K = 5,10,30$-shot settings on MS-COCO are presented. The metrics reported include AP, bAP, and nAP for all, base, and novel classes, respectively. The best and second-best results are highlighted. w/E indicates whether the ensemble-learning-based evaluation protocol of Retentive R-CNN [Fan21] is employed. An asterisk (*) denotes results reported in Retentive R-CNN [Fan21] and DeFRCN [Qia21], while a dash (-) indicates unreported results in the original work.

Methods / Shots	**w/E**	**5 shot**			**10 shot**			**30 shot**		
		AP	**bAP**	**nAP**	**AP**	**bAP**	**nAP**	**AP**	**bAP**	**nAP**
FRCN-ft-full [Wan20a]	✗	18.0	22.0	6.0	18.1	21.0	9.2	18.6	20.6	12.5
TFA w/ fc [Wan20a]	✗	27.5	33.9	8.4	27.9	33.9	10.0	29.7	35.1	13.4
TFA w/ cos [Wan20a]	✗	28.1	34.7	8.3	28.7	35.0	10.0	30.3	35.8	13.7
MPSR [Wu20a]	✗	-	-	-	15.3	17.1	9.7	17.1	18.1	14.1
DeFRCN [Qia21]	✗	28.7	33.1	**15.3**	30.6	34.6	**18.6**	31.6	34.7	**22.5**
ONCE [Per20]	✗	13.7	17.9	1.0	13.7	17.9	1.2	-	-	-
Meta R-CNN* [Yan19]	✗	3.6	3.5	3.8	5.4	5.2	6.1	7.8	7.1	9.9
FSRW [Kan18]	✗	-	-	-	-	-	5.6	-	-	9.1
FsDetView* [Xia20]	✗	5.9	5.7	6.6	6.7	6.4	7.6	10.0	9.3	12.0
CFA w/ fc	✗	**30.1**	**37.1**	9.0	**30.8**	**37.6**	10.5	**31.9**	**37.7**	14.7
CFA w/ cos	✗	**29.7**	**36.3**	9.8	30.3	**36.6**	11.3	31.7	**37.0**	15.6
CFA-DeFRCN	✗	**30.1**	35.0	**15.6**	**31.4**	35.5	**19.1**	**32.0**	35.0	**23.0**
Retentive R-CNN [Fan21]	✓	31.5	**39.2**	8.3	32.1	39.2	10.5	32.9	**39.3**	13.8
CFA w/ fc	✓	31.8	**39.5**	8.8	32.2	**39.5**	10.4	33.2	**39.5**	14.3
CFA w/ cos	✓	**32.0**	**39.5**	**9.6**	**32.4**	**39.4**	**11.3**	**33.4**	**39.5**	**15.1**
CFA-DeFRCN	✓	**33.0**	38.9	**15.6**	**34.0**	39.0	**18.9**	**34.9**	39.0	**22.6**

Comparison Results

Results on MS-COCO

The CFA is evaluated on MS-COCO in Table 5.1 using the $K = 5,10,30$-shot settings. The standard mean AP metric is employed to measure performance on both the base and novel categories in the G-FSOD setting, denoted by bAP and nAP, respectively. The results demonstrate that, regardless of the architecture, CFA significantly alleviates forgetting on the base categories compared to TFA [Wan20a], while simultaneously improving performance on the novel classes. Table 5.1 reveals that CFA-DeFRCN consistently achieves the best overall and novel performance across the three few-shot configurations compared to both FSOD and G-FSOD methods.

Additionally, the results of the method using the Retentive-RCNN ensemble-based evaluation protocol are reported. When compared to Retentive-RCNN [Fan21], the CFA-finetuned models exhibit superior performance on both the base and novel classes. CFA w/cos demonstrates slightly better performance than CFA w/fc, aligning with previous observations in TFA [Wan20a] and Retentive-RCNN [Fan21] where cosine similarity classifiers demonstrate improved generalization due to their robustness against variations in feature norms between base and novel classes.

Although the ensemble-based evaluation protocol mitigates forgetting the base classes, it is associated with increased inference time and model capacity. The ablation experiments in the following section support this observation. As shown in Table 5.1, individual CFA-finetuned methods achieve comparable performance to Retentive-RCNN [Fan21] on base classes and superior results on novel classes. In the ensemble setting, CFA-DeFRCN incorporates base and novel RPNs, detectors, and backbones. This is facilitated by the gradient decoupling layer, which enables finetuning of the backbone with gradients backpropagated from the RoI-head.

Results on PASCAL-VOC

The overall and novel performance of the G-FSOD models on the PASCAL-VOC dataset are shown in 5.2 and 5.3, respectively. The effectiveness of CFA is highlighted across different splits, where single CFA-finetuned models can generalize better than the ensemble of base and novel models used in Retentive R-CNN [Fan21].

The performance of the G-FSOD models on the PASCAL-VOC dataset is presented in Table 5.2 for the overall performance and in Table 5.3 for the novel performance. Consistent with the results obtained on the MS-COCO dataset, the findings indicate that the proposed CFA enhances both the overall and novel performances across different dataset splits. This highlights the effectiveness of the CFA-finetuned models in terms of generalization compared to the ensemble approach employed in Retentive R-CNN [Fan21].

Table 5.2: The G-FSOD results for the $K = 1,2,3,5,10$-shot settings on the three different sets of PASCAL-VOC (AP50) are presented. w/E indicates whether the ensemble-learning based evaluation protocol of Retentive-RCNN [Fan21] was applied. The best and second-best results are highlighted. An asterisk (*) denotes results reported in [Fan21, Qia21].

Methods / Shots	w/E	All Set 1					All Set 2					All Set 3				
		1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
FRCN-ft-full[Wan20a]	✗	55.4	57.1	56.8	60.1	60.9	50.1	53.7	53.6	55.9	55.5	58.5	59.1	58.7	61.8	60.8
TFA w/ fc[Wan20a]	✗	69.3	66.9	70.3	73.4	73.2	64.7	66.3	67.7	68.3	68.7	67.8	68.9	70.8	72.3	72.2
TFA w/ cos[Wan20a]	✗	69.7	68.2	70.5	73.4	72.8	65.5	65.0	67.7	68.0	68.6	67.9	68.6	71.0	72.5	72.4
MPSR[Wu20a]	✗	56.8	60.4	62.8	66.1	69.0	53.1	57.6	62.8	64.2	66.3	55.2	59.8	62.7	66.9	67.7
DeFRCN[Qia21]	✗	**73.1**	**73.2**	**73.7**	**75.1**	**74.4**	**68.6**	**69.8**	**71.0**	**72.5**	**71.5**	**72.5**	**73.5**	**72.7**	**74.1**	**73.9**
Meta R-CNN*[Yan19]	✗	17.5	30.5	36.2	49.3	55.6	19.4	33.2	34.8	44.4	53.9	20.3	31.0	41.2	48.0	55.1
FSRW[Kan18]	✗	53.5	50.2	55.3	56.0	59.5	55.1	54.2	55.2	57.5	58.9	54.2	53.5	54.7	58.6	57.6
FsDetView*[Xia20]	✗	36.4	40.3	40.1	50.0	55.3	36.3	43.7	41.6	45.8	54.1	37.0	39.5	40.7	50.7	54.8
CFA w/ fc	✗	69.5	68.2	69.8	73.5	**74.3**	66.0	66.9	69.2	70.1	71.1	67.7	69.0	70.9	72.6	73.5
CFA w/ cos	✗	69.1	69.8	71.9	73.6	73.9	64.8	66.5	68.3	69.5	70.5	67.7	69.7	71.9	**73.0**	73.5
CFA-DeFRCN	✗	**73.8**	**74.6**	**74.5**	**76.0**	**74.4**	**69.3**	**71.4**	**72.0**	**73.3**	**72.0**	**72.9**	**73.9**	**73.0**	**74.1**	**74.6**
Retentive R-CNN[Fan21]	✓	71.3	**72.3**	72.1	74.0	74.6	66.8	**68.4**	70.2	70.7	71.5	69.0	70.9	72.3	73.9	74.1
CFA w/ fc	✓	70.3	69.5	71.0	74.4	74.9	**67.0**	68.0	70.2	70.8	71.5	69.1	70.1	71.6	73.3	**74.7**
CFA w/ cos	✓	**71.4**	71.8	**73.3**	**74.9**	**75.0**	66.8	**68.4**	**70.4**	**71.1**	**71.9**	**69.7**	**71.2**	**72.6**	**74.0**	**74.7**
CFA-DeFRCN	✓	**75.0**	**76.0**	**76.8**	**77.3**	**77.3**	**70.4**	**72.7**	**73.7**	**74.7**	**74.2**	**74.7**	**75.5**	**75.0**	**76.2**	**76.6**

Ablation Experiments

Impact of Unfreezing Different Components

The influence of various model components in the G-FSOD setting is examined. The results are presented in Table 5.4. Firstly, it is observed that unfreezing either the RPN or the RoI-head alone yields suboptimal results. Although there is a slight increase in bAP compared to the frozen model, there is a slight decrease in nAP due to overfitting of the unfrozen component to the few novel shots. Secondly, the best results in both TFA [Wan20a] and CFA are obtained when unfreezing both the RPN and RoI-head. However, unfreezing the backbone leads to degraded results. Lastly, CFA demonstrates a superior ability to guide the gradients in the expanded search space when the RPN and RoI-heads are unfrozen.

Table 5.3: The G-FSOD results for the $K = 1{,}2{,}3{,}5{,}10$-shot settings on the three different novel sets of PASCAL-VOC (nAP50) are provided. ẅ/Eïndicates whether the ensemble-learning based evaluation protocol of Retentive-RCNN [Fan21] was applied. The best and second-best results are highlighted. An asterisk (*) denotes results reported in [Fan21, Qia21]. Notably, our approach achieves state-of-the-art performance in terms of nAP50 across the three different dataset splits and various few-shot settings.

Methods / Shots	w/E	Novel Set 1					Novel Set 2					Novel Set 3				
		1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
FRCN-ft-full[Wan20a]	✗	15.2	20.3	29.0	25.5	28.7	13.4	20.6	28.6	32.4	38.8	19.6	20.8	28.7	42.2	42.1
TFA w/ fc[Wan20a]	✗	36.8	29.1	43.6	55.7	57.0	18.2	29.0	33.4	35.5	39.0	27.7	33.6	42.5	48.7	50.2
TFA w/ cos[Wan20a]	✗	39.8	36.1	44.7	55.7	56.0	23.5	26.9	34.1	35.1	39.1	30.8	34.8	42.8	49.5	49.8
MPSR[Wu20a]	✗	42.8	43.6	48.4	55.3	61.2	29.8	28.1	41.6	43.2	47.0	35.9	40.0	43.7	48.9	51.3
DeFRCN[Qia21]	✗	**57.0**	**58.6**	**64.3**	**67.8**	**67.0**	**35.8**	**42.7**	**51.0**	**54.4**	**52.9**	**52.5**	**56.6**	**55.8**	**60.7**	**62.5**
Meta R-CNN*[Yan19]	✗	16.8	20.1	20.3	38.2	43.7	7.7	12.0	14.9	21.9	31.1	9.2	13.9	26.2	29.2	36.2
FSRW[Kan18]	✗	14.8	15.5	26.7	33.9	47.2	15.7	15.3	22.7	30.1	39.2	19.2	21.7	25.7	40.6	41.3
MetaDet[Wan19c]	✗	18.9	20.6	30.2	36.8	49.6	21.8	23.1	27.8	31.7	43.0	20.6	23.9	29.4	43.9	44.1
FsDetView*[Xia20]	✗	25.4	20.4	37.4	36.1	42.3	22.9	21.7	22.6	25.6	29.2	32.4	19.0	29.8	33.2	39.8
CFA w/ fc	✗	40.0	35.5	40.9	54.1	56.9	22.2	27.1	35.2	38.5	40.9	29.7	35.1	39.5	47.2	51.3
CFA w/ cos	✗	41.2	43.6	49.5	56.5	57.3	21.3	27.4	35.3	39.1	42.1	31.7	39.1	44.6	49.9	52.6
CFA-DeFRCN	✗	**58.2**	**63.3**	**65.8**	**68.9**	**67.1**	**37.1**	**45.5**	**51.3**	**55.2**	**53.8**	**54.7**	**57.8**	**56.9**	**60.0**	**63.3**
Retentive R-CNN[Fan21]	✓	**42.4**	**45.8**	45.9	53.7	56.1	21.7	**27.8**	35.2	37.0	40.3	30.2	37.6	43.0	49.7	50.1
CFA w/ fc	✓	39.0	34.9	41.4	54.8	57.0	**21.8**	26.1	**35.3**	37.1	40.1	29.9	34.3	40.1	47.0	52.6
CFA w/ cos	✓	**42.4**	43.9	**50.3**	**56.6**	**57.3**	21.0	27.5	**35.3**	**38.6**	**41.4**	**32.3**	**38.0**	**44.5**	**49.8**	**52.7**
CFA-DeFRCN	✓	**59.0**	**63.5**	**66.4**	**68.4**	**68.3**	**37.0**	**45.8**	**50.0**	**54.2**	**52.5**	**54.8**	**58.5**	**56.5**	**61.3**	**63.5**

Influence of the Number of Base Shots

The impact of using unbalanced datasets by finetuning with different numbers of base shots $K = 1{,}2{,}3{,}5{,}10$ and 10 novel shots is demonstrated in Table 5.5. While both TFA [Wan20a] and CFA achieve similar performance on the novel task, TFA shows higher sensitivity to a lower number of base shots. Specifically, when only one base shot is used, TFA experiences a significant drop in bAP ($\sim 22.4\%$). In contrast, CFA exhibits greater robustness to fewer base shots, with a reduction of approximately ($\sim 7.2\%$) in bAP, thus demonstrating reduced forgetting with fewer base shots. Remarkably, when using only 3 shots, CFA performs similarly to the 10-shot scenario, with only a 0.6 points decrease in bAP. These findings indicate that CFA is more memory efficient as it can effectively leverage fewer base shots to achieve reduced forgetting.

Table 5.4: The impact of unfreezing different components during finetuning with CFA, compared to TFA [Wan20a], is examined. The results are presented for the 10-shot setting on the MS-COCO dataset.

Method	Backone	RPN	RoI Head	AP	bAP	nAP
TFA w/ fc [Wan20a]				27.9	33.9	10.0
		✓		29.9	37.2	7.9
			✓	28.9	35.4	9.6
		✓	✓	28.9	35.1	10.2
	✓	✓	✓	24.1	29.0	9.1
CFA w/ fc				29.6	36.0	10.4
		✓		30.3	37.4	9.3
			✓	**30.8**	**37.8**	9.6
		✓	✓	**30.8**	37.6	**10.5**
	✓	✓	✓	23.9	28.6	10.1
TFA w/ cos [Wan20a]				28.7	35.0	10.0
		✓		28.9	35.8	8.3
			✓	29.0	35.3	10.3
		✓	✓	29.2	35.2	11.2
	✓	✓	✓	24.1	28.5	10.9
CFA w/ cos				29.4	35.9	9.8
		✓		28.7	35.3	8.9
			✓	30.2	**36.8**	10.6
		✓	✓	**30.3**	36.6	**11.3**
	✓	✓	✓	23.6	27.9	10.9

Finetuning with A-GEM

To assess the performance of CFA compared to A-GEM [Cha19a], which serves as its foundation, a comparison is conducted by employing both finetuning methods across the three models utilized in this study. Table 5.6 demonstrates that regardless of the model, CFA exhibits reduced forgetting and enhances performance on the novel task compared to A-GEM. Furthermore, CFA consistently achieves superior overall performance and improves the nAP as well.

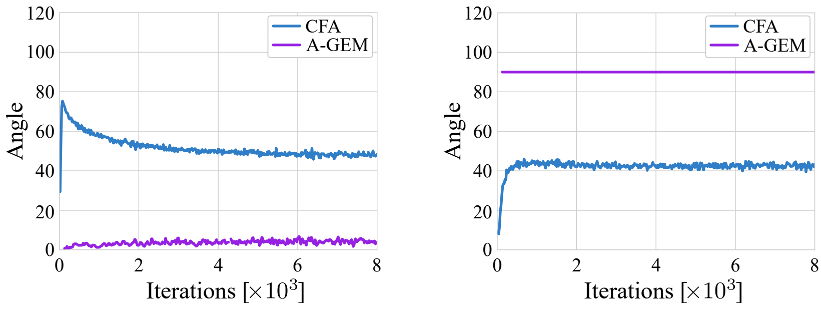
Table 5.5: The effect of the number of base shots on the occurrence of catastrophic forgetting in base classes is investigated, comparing CFA to TFA [Wan20a]. The experiments are performed on the MS-COCO dataset, with 10-shots provided for the novel categories.

Method	Base-Shots	AP	bAP	nAP
TFA w/ fc [Wan20a]	1-Shots	22.2	26.3	9.8
	2-Shots	24.8	29.8	9.9
	3-Shots	26.1	31.5	10.1
	5-Shots	27.0	32.6	10.2
	10-Shots	27.9	33.9	10.0
CFA w/ fc	1-Shots	28.8	34.9	**10.5**
	2-Shots	30.0	36.5	**10.5**
	3-Shots	30.3	37.0	10.3
	5-Shots	30.5	37.2	10.4
	10-Shots	**30.8**	**37.6**	**10.5**
TFA w/ cos [Wan20a]	1-Shots	24.2	28.9	10.0
	2-Shots	26.5	32.0	10.2
	3-Shots	27.2	32.9	10.3
	5-Shots	27.8	33.6	10.3
	10-Shots	28.7	35.0	10.0
CFA w/ cos	1-Shots	28.6	34.3	**11.5**
	2-Shots	29.8	35.9	11.3
	3-Shots	30.0	36.2	11.3
	5-Shots	30.2	36.4	11.3
	10-Shots	**30.3**	**36.6**	11.3

This suggests that the proposed constraint in CFA fosters more effective forward knowledge transfer, as both base and novel gradients actively contribute to the gradient update rule in each step, thereby minimizing the expected risk on both tasks.

Table 5.6: A comparison between finetuning different models using CFA and A-GEM [Cha19a]. The results are presented for the 10-shot setting on the MS-COCO dataset.

Model	AP	bAP	nAP
A-GEM w/ fc	30.1	36.8	10.1
CFA w/ fc	30.8	**37.6**	10.5
A-GEM w/ cos	28.2	34.5	9.3
CFA w/ cos	30.3	36.6	11.3
A-GEM-DeFRCN	30.3	35.6	14.4
CFA-DeFRCN	**31.4**	35.5	**19.1**



(a) Angle between $\tilde{\boldsymbol{g}}$ and $\boldsymbol{g}_n$

(b) Angle between $\tilde{\boldsymbol{g}}$ and $\boldsymbol{g}_b$

Figure 5.4: The angle between the projected gradient $\tilde{\boldsymbol{g}}$ and both the $\boldsymbol{g}_n$ (a) and $\boldsymbol{g}_b$ (b) is visualized for A-GEM [Cha19a] and CFA in the 10-shot setting on the MS-COCO dataset.

Projected Gradients Visualization

Figure 5.4 illustrates the differences between A-GEM [Cha19a] and CFA regarding gradient directions. In Figure 5.4a, it is observed that as the network converges, the projected gradient in A-GEM is aligned closely with the direction of the novel gradient ($\sim 4°$), indicating a bias towards the novel tasks over the base tasks. On the other hand, CFA exhibits a larger angle ($\sim 43°$) in the gradient update, indicating a more balanced consideration of both base and novel tasks. Figure 5.4b further highlights that CFA provides a projected gradient that is much closer to the base task loss gradients ($\sim 45°$) compared to the

Table 5.7: The evaluation protocols have varying implications on inference time and model capacity. Ensemble methods introduce a substantial overhead compared to using a single model. The notation w/E indicates whether the ensemble method is employed.

Method	w/E	Inference Time [ms]	Model Capacity [M]
TFA w/ fc [Wan20a]	✗	85	60.6
TFA w/ cos [Wan20a]	✗	87	60.6
CFA w/ fc	✗	85	60.6
CFA w/ cos	✗	86	60.6
CFA-DeFRCN	✗	147	52.7
CFA w/ fc	✓	211	75.4
CFA w/ cos	✓	211	75.4
CFA-DeFRCN	✓	376	105.3

consistent orthogonal projection employed by A-GEM. This closer alignment between CFA and the base task gradients enables better knowledge transfer while learning novel tasks and helps to mitigate forgetting.

Model Complexity Analysis

In Table 5.7, the influence of the two evaluation protocols on inference time and the number of parameters during inference is examined. While the ensemble model evaluation approach achieves less forgetting, it also results in an average increase of 52% in inference time. Conversely, the number of parameters increases by 50% in CFA w/fc (and w/cos) and by 102% in CFA-DeFRCN, primarily due to the backbone being unfrozen during the finetuning process in DeFRCN [Qia21].

Multiple Runs

The CFA-finetuned models (CFA w/fc, CFA w/cos, and CFA-DeFRCN) are evaluated using 10 different seeds on the MS-COCO and PASCAL-VOC datasets, and compared against the baselines, TFA [Wan20a] and DeFRCN [Qia21]. The results are presented in Table 5.8 for the MS-COCO dataset and Table 5.9 for the

Table 5.8: G-FSOD results for $K = 5,10,30$-shot settings on MS-COCO across 10 different seeds.

Methods / Shots	5 shot			10 shot			30 shot		
	AP	**bAP**	**nAP**	**AP**	**bAP**	**nAP**	**AP**	**bAP**	**nAP**
TFA w/ fc[Wan20a]	25.6±0.5	31.8±0.5	6.9±0.7	26.2±0.5	32.0±0.5	9.1±0.5	28.4±0.3	33.8±0.3	12.0±0.4
TFA w/ cos[Wan20a]	25.9±0.6	32.3±0.6	7.0±0.7	26.6±0.5	32.4±0.6	9.1±0.5	28.7±0.4	34.2±0.4	12.1±0.4
CFA w/ fc	29.1±0.3	36.2±0.3	7.7±0.6	29.9±0.3	36.7±0.2	9.6±0.6	30.8±0.2	36.6±0.2	13.6±0.3
CFA w/ cos	29.3±0.2	36.0±0.2	9.2±0.5	30.2±0.2	36.6±0.1	11.2±0.5	31.1±0.1	36.6±0.1	14.8±0.2
DeFRCN[Qia21]	27.8±0.3	32.6±0.3	13.6±0.7	29.7±0.2	34.0±0.2	16.8±0.6	31.4±0.1	34.8±0.1	21.2±0.4
CFA-DeFRCN	28.4±0.2	32.8±0.2	15.2±0.5	30.2±0.2	34.0±0.2	18.8±0.4	31.7±0.1	34.6±0.1	23.0±0.3

Table 5.9: G-FSOD results for $K = 5,10,30$-shot settings on PASCAL-VOC (AP50) across 30 different seeds.

Set	Methods	Shots				
		1	2	3	5	10
All Set 1	CFA w/ fc	66.3±0.8	68.0±0.5	70.1±0.4	71.7±0.5	73.2±0.5
	CFA w/ cos	66.5±0.9	69.2±0.6	71.1±0.6	72.5±0.4	73.4±0.4
	DeFRCN[Qia21]	67.8±1.4	71.3±0.8	72.6±0.5	73.6±0.5	74.1±0.5
	CFA-DeFRCN	69.0±1.4	72.6±0.7	73.1±0.4	74.0±0.5	74.3±0.4
All Set 2	CFA w/ fc	64.9±0.9	66.4±0.7	68.3±0.5	69.6±0.3	70.8±0.5
	CFA w/ cos	64.1±0.9	66.5±0.5	68.1±0.5	69.3±0.2	70.4±0.4
	DeFRCN[Qia21]	65.2±1.0	68.0±0.8	69.2±0.6	70.6±0.6	71.3±0.5
	CFA-DeFRCN	66.4±1.0	69.0±0.8	70.4±0.7	71.3±0.7	72.1±0.4
All Set 3	CFA w/ fc	65.2±0.8	66.8±0.8	69.1±0.7	70.9±0.6	72.3±0.4
	CFA w/ cos	64.9±1.2	67.5±1.0	69.7±0.8	71.6±0.5	72.7±0.3
	DeFRCN[Qia21]	66.9±2.0	70.6±0.8	71.2±0.6	72.9±0.5	73.5±0.3
	CFA-DeFRCN	68.3±1.6	71.4±0.8	72.3±0.5	73.5±0.5	74.0±0.3

PASCAL-VOC dataset. The same random seeds as TFA [Kan18] and DeFRCN are utilized. CFA consistently improves the overall AP while demonstrating a smaller confidence interval.

Qualitative Results

Qualitative results of CFA w/cos finetuned with the 30-shot setting are depicted in Figure 5.5. The first three columns showcase different successful scenarios,

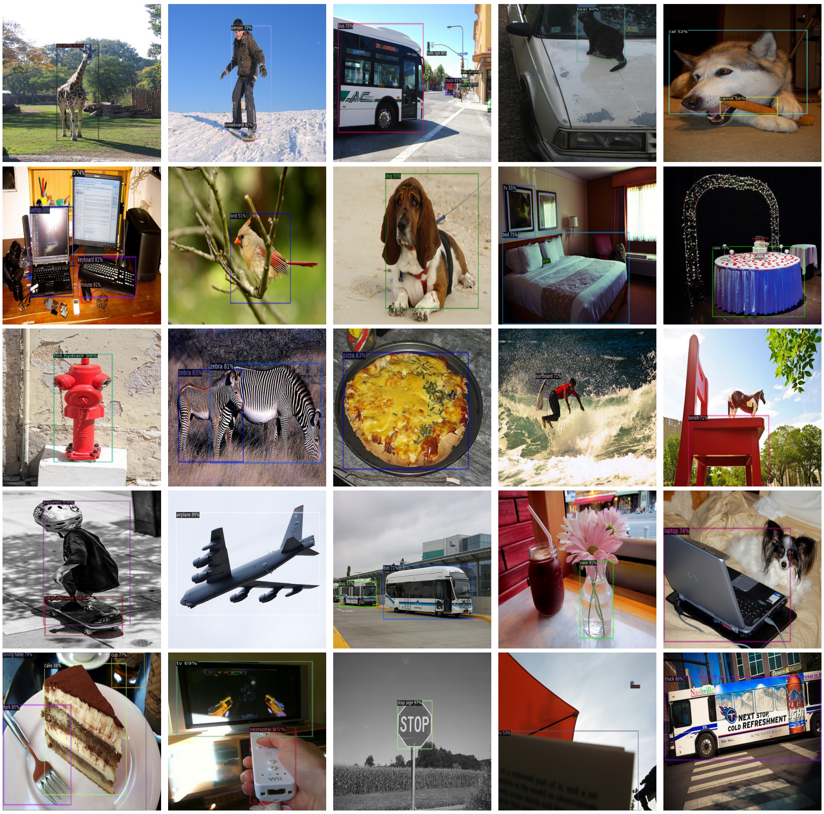


Figure 5.5: A qualitative analysis of the proposed CFA method is conducted on the MS-COCO dataset. The results presented correspond to CFA w/cos finetuned using the 30-shot setting. The first three columns depict instances of success, while the last two columns showcase scenarios of failure.

while the last two columns highlight instances of failure. It is evident that the model exhibits less confidence in predicting novel categories compared to base classes. This reduced confidence can be attributed to the learning of indiscriminative features, leading to occurrences of false positives and false negatives.

5.3 Uncertainty-based Progressive Proposal Refinement

The previous section emphasized the significance of leveraging the information retained by the base model to mitigate forgetting and improve performance on novel tasks. However, the existing G-FSOD approaches [Wan20a, Qia21, Fan21] overlook a valuable source of information, namely uncertainty estimation. By incorporating predictive uncertainties, which offer valuable distributional insights into the base classes, it is possible to mitigate catastrophic forgetting and improve overall performance more effectively [Li16, Ser18, Kur21].

It is also important to note that current G-FSOD frameworks [Wan20a, Qia21, Fan21] are mostly based on a two-stage Faster R-CNN model [Ren15]. One of the main bottlenecks encountered during standard object detection is the poor quality of object proposals [Vu19]. The quality of the object proposals further deteriorates in G-FSOD due to the introduction of new classes. There are three main reasons for this: (1) the training data for these new classes is limited and does not represent the true class distribution, (2) the novel classes might be considered as background by the network due to a low IoU with the ground truth boxes, and (3) the scale distribution of the novel objects differs from that in the base training data. Moreover, the limited novel samples result in higher epistemic uncertainty, because the true data distribution is not fully captured, causing the model to overfit or underfit the data. None of the previous G-FSOD works have explicitly tackled the aforementioned limitations.

In this section, a novel G-FSOD model is proposed, aiming to tackle the limitations mentioned above. Specifically, the goal is to refine the initially low-quality and highly uncertain proposals in a stagewise manner. The model achieves this by utilizing multiple R-CNNs, where each stage is responsible for estimating predictive aleatoric and epistemic uncertainties to generate more confident proposals. Additionally, attention blocks are incorporated during novel training to effectively learn the discriminative spatial features of each class, even when only limited labels are available. The main goal is to design a two-stage G-FSOD framework that enhances the object proposals to improve the overall detection performance without forgetting the base classes.

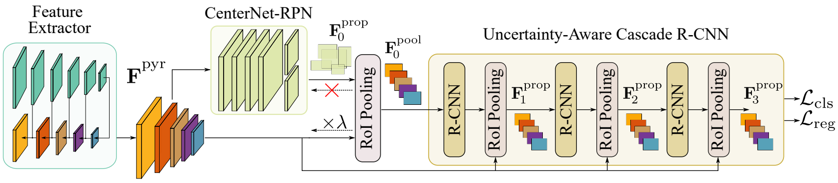

Feature Extractor
CenterNet-RPN
$\mathbf{F}^{\text{pyr}}$
$\mathbf{F}_0^{\text{prop}}$
$\mathbf{F}_0^{\text{pool}}$
$\times\lambda$
RoI Pooling
Uncertainty-Aware Cascade R-CNN
R-CNN
RoI Pooling
$\mathbf{F}_1^{\text{prop}}$
R-CNN
RoI Pooling
$\mathbf{F}_2^{\text{prop}}$
R-CNN
RoI Pooling
$\mathbf{F}_3^{\text{prop}}$
$\mathcal{L}_{\text{cls}}$
$\mathcal{L}_{\text{reg}}$

Figure 5.6: An illustration of the proposed Decoupled Cascade R-CNN (DeCRCN) architecture. The feature pyramid is generated using ResNet-FPN [He16, Lin17], and proposal quality is enhanced through a keypoint-based RPN inspired by CenterNet [Dua19]. Subsequently, Cascade R-CNN is employed to progressively refine proposals. Furthermore, attention modules are integrated within R-CNN stages to prioritize features that correlate with accurate detections.

5.3.1 Architecture Overview

Upon examination of a fundamental two-stage G-FSOD framework [Qia21], encompassing a RPN and a subsequent R-CNN, the direct integration of uncertainty estimation has been observed to yield a deleterious impact on performance. This phenomenon can be ascribed to the potential introduction of heightened complexity or noise into the model as a consequence of employing uncertainty estimation methods [Ken17, Gaw23]. In cases where the model lacks the capacity to effectively manage this augmented complexity or noise, a decline in performance becomes evident. Additionally, the limited availability of data exacerbates this effect, given that uncertainty estimation often relies on the model's comprehension of the true data distribution. In pursuit of mitigating these challenges, several architectural modifications to the DeFRCN model [Qia21] have been proposed. An overview of the proposed model, referred to as Decoupled Cascade R-CNN (DeCRCN), is presented in Figure 5.6.

Multi-Scale CenterNet-RPN

In traditional architectures like Faster R-CNN [Ren15], the conventional RPN component has a tendency to generate suboptimal proposals for the subsequent R-CNN detector. This phenomenon primarily arises from utilizing fixed-sized anchors, which frequently yield a plethora of background and low-quality foreground proposals. Furthermore, the misalignment between these anchors

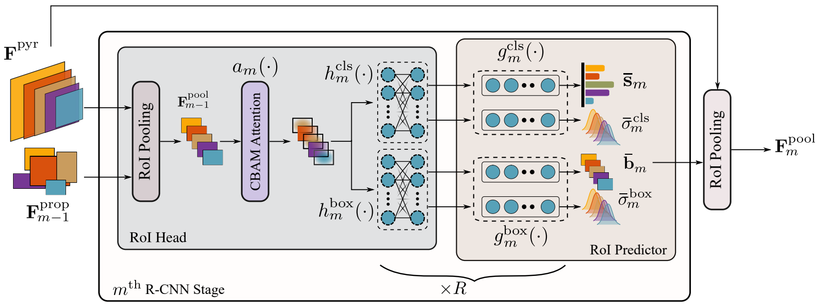

$\mathbf{F}^{\text{pyr}}$
$\mathbf{F}^{\text{prop}}_{m-1}$
RoI Pooling
$\mathbf{F}^{\text{pool}}_{m-1}$
$a_m(\cdot)$
CBAM Attention
$h^{\text{cls}}_m(\cdot)$
$h^{\text{box}}_m(\cdot)$
RoI Head
$g^{\text{cls}}_m(\cdot)$
$\bar{\mathbf{s}}_m$
$\bar{\sigma}^{\text{cls}}_m$
$\bar{\mathbf{b}}_m$
$\bar{\sigma}^{\text{box}}_m$
$g^{\text{box}}_m(\cdot)$
RoI Predictor
$\times R$
m^{th} R-CNN Stage
RoI Pooling
$\mathbf{F}^{\text{pool}}_m$

Figure 5.7: An illustration of the single stage R-CNN at test-time of the cascaded R-CNNs. The dotted neurons represent the dropouts. The epistemic uncertainties are computed by R forward runs and the predictions are averaged.

and convolutional features adds complexity to the bounding box classification task. Conversely, approaches based on keypoints offer a promising remedy by representing object keypoints, which furnish more precise spatial information.

To overcome these limitations, a keypoint-based strategy known as CenterNet-RPN is introduced, which replaces the anchor-based RPN with CenterNet [Dua19]. Additionally, to explicitly address variations in object scale, the feature extractor is augmented through the integration of an FPN [Lin17]. This integration facilitates the refinement of object proposals across different scales, thereby enhancing the overall performance of the model.

Cascade R-CNN

In the context of Cascade R-CNN, the conventional single R-CNN is substituted with a Cascade R-CNN [Cai18], employing increasing IoU thresholds to enhance RPN proposals. Each stage iteratively enhances the quality of object proposals from the preceding stage, thereby augmenting the count of true positives forwarded to the subsequent stage. Within each R-CNN stage, classification and localization features are separated by introducing dual classification and bounding box regressor heads.

Multi-Stage Instance-level Attention

While refining proposals via Cascaded R-CNN stages, it is essential to acknowledge that not all instance-level features possess uniform importance. To prioritize features crucial for accurate classification, attention modules are interspersed with the R-CNN stages. Specifically, a Convolutional Block Attention Module (CBAM) [Woo18] is employed for this purpose, selectively concentrating on the most salient features relevant to the G-FSOD task. The CBAM encompasses channel and spatial attention components, which capture both channel-wise and spatial-wise relationships among instance-level features. This facilitates the model improved understanding of semantically-rich information for both novel and base classes. Notably, the CBAM is favored due to its lightweight design [Woo18], a crucial factor as it is integrated after each R-CNN stage in the network. To ensure an equitable representation of base and novel features, multi-stage attention blocks are exclusively introduced during the novel training phase to prevent the CBAM from exhibiting bias toward the base classes.

5.3.2 Methodology

The proposed DeCRCN leverages uncertainty estimation to harness distributional information from predictive uncertainties, thereby aiding in mitigating forgetting [Kur21]. Furthermore, predictive uncertainties play a pivotal role in enhancing performance on novel classes, given their limited number of examples. In the subsequent sections, an UPPR approach is introduced atop DeCRCN to address the amelioration of forgetting and the augmentation of novel detection performance. More specifically, aleatoric and epistemic uncertainties are utilized to progressively refine object proposals generated by the CenterNet-RPN across the Cascade R-CNN in a stage-wise fashion.

Stage-wise Epistemic Uncertainty-based Refinement

As previously mentioned, the presence of inherent data and model uncertainties mandates their consideration to address the issues of forgetting and to enhance the detection of novel classes [Li16, Ser18, Kur21]. However, the direct integration of uncertainty into the DeFRCN model [Qia21] results in a deterioration of both base and novel detection performance, underscoring the necessity for the architectural design choices introduced in this study. Consequently, the proposed model undertakes the estimation of aleatoric and epistemic uncertainties at each stage within the Cascade R-CNN framework.

During the training phase, epistemic uncertainty is modeled through the introduction of dropout [Nit14] layers within each R-CNN stage. These dropout layers induce an ensemble effect by deactivating different neurons during training, effectively simulating a diverse set of subnetworks. This approach introduces prediction variability, capturing uncertainty regarding which specific neurons are pertinent for a given input [Gaw23]. Furthermore, the ensemble of subnetworks emulates a distribution of models, approximating the concept of Bayesian model averaging [Ken17, Gaw23]. This representation accounts for the model's uncertainty concerning its optimal configuration, resulting in a more robust characterization of epistemic uncertainty.

The process commences by utilizing the pyramid feature maps $\mathbf{F}^{\text{pyr}}$ generated by the backbone network and the object proposals produced by the preceding stage, with CenterNet-RPN serving as the initial stage. Subsequently, proposal features are extracted through RoI-pooling, subjected to the CBAM attention block for focused attention, and then processed by the classification and bounding box regressor heads to derive class scores and bounding box offsets. This sequence constitutes a single forward pass within an R-CNN stage.

During the testing phase, the dropout layers are activated, and multiple forward passes (indicated by R) are executed per stage. Predictions generated from these passes are subsequently aggregated and, in conjunction with $\mathbf{F}^{\text{pyr}}$, passed on to the next stage. The operational procedure of an individual R-CNN stage during testing is illustrated in Figure 5.7.

For M stages, the classification features for the m^{th} stage are formulated as follows:

$$\mathbf{F}_m^{\text{cls}} = h_m^{\text{cls}}(a_m(\mathbf{F}_{m-1}^{\text{pool}})), \tag{5.10}$$

where for stage m, $a_m(\cdot)$ is the CBAM attention module. h_m^{cls} is the classification head in the RoI-head. $\mathbf{F}_{m-1}^{\text{pool}}$ is the pooled instance-level features from the previous stage. Similarly, the bounding-box features are computed as:

$$\mathbf{F}_m^{\text{box}} = h_m^{\text{box}}(a_m(\mathbf{F}_{m-1}^{\text{pool}})), \tag{5.11}$$

where for stage m, h_m^{box} is the bounding-box head in the RoI-head. The $\mathbf{F}_m^{\text{cls}}$ and $\mathbf{F}_m^{\text{box}}$ undergo the RoI-predictor to compute the classification and regression offsets along with their corresponding uncertainties. The RoI-predictor consists of a classifier head $\mathbf{g}_m^{\text{cls}}(\cdot)$ and a box head $\mathbf{g}_m^{\text{box}}(\cdot)$.

During the inference phase, a series of R forward passes is executed with dropouts, and the classification logits $\mathbf{s}_m$ are accumulated, along with the predicted class aleatoric covariance denoted as $\mathbf{\Sigma}_m^{\text{cls}}$. Likewise, the box offsets $\mathbf{b}_m$ are accumulated, along with the predicted box aleatoric covariance referred to as $\mathbf{\Sigma}_m^{\text{box}}$. The final classification logits and the corresponding aleatoric variances are determined by computing the average across the R forward passes, resulting in:

$$\bar{\mathbf{s}}_m^{\text{cls}}, \bar{\mathbf{\Sigma}}_m^{\text{cls}} = \frac{1}{R}\sum_{r=1}^{R} g_m^{\text{cls}}(\mathbf{F}_{m,r}^{\text{cls}}), \tag{5.12}$$

where $\bar{\mathbf{s}}_m^{\text{cls}}$ and $\bar{\mathbf{\Sigma}}_m^{\text{cls}}$ are the average classification logits and class aleatoric covariance across the R iterations, respectively. $\mathbf{F}_{m,r}^{\text{cls}}$ represent different RoI-head classification features each forward run r due to the stochastic dropouts in the classification head. The box offsets and predicted variances are computed as follows:

$$\bar{\mathbf{b}}_m^{\text{box}}, \bar{\mathbf{\Sigma}}_m^{\text{box}} = \frac{1}{R}\sum_{r=1}^{R} g_m^{\text{box}}(\mathbf{F}_{m,r}^{\text{box}}), \tag{5.13}$$

where $\bar{\mathbf{b}}_m^{\text{box}}$ and $\bar{\mathbf{\Sigma}}_m^{\text{box}}$ are the average box offsets and box aleatoric covariance across the R iterations, respectively. $\mathbf{F}_{m,r}^{\text{box}}$ are the RoI-head box features for the forward run r.

Stage-wise Aleatoric Uncertainty-based Refinement

Aleatoric uncertainties are taken into account for both the classification and bounding box regression tasks. Initially, the classification logits are conceptualized as a multivariate Gaussian distribution. This distribution is defined by the mean $\mathbf{s}^{\text{cls}}$ of the predicted classification logits and the diagonal covariance matrix $\mathbf{\Sigma}^{\text{cls}}$, which is computed based on the predicted class variances $\boldsymbol{\sigma}^2_{\text{cls}}$. Subsequently, N^{cls} classification logits $\mathbf{s}^{\text{cls}}_n$ are sampled from this Gaussian distribution. These samples are consolidated into a matrix denoted as $\mathbf{S}^{\text{cls}}$ and can be represented as follows:

$$\mathbf{S}^{\text{cls}} = \{\mathbf{s}^{\text{cls}}_n\}_{n=1}^{N^{\text{cls}}} \in \mathbb{R}^{N^{\text{cls}} \times |\mathcal{C}|},\ \mathbf{s}^{\text{cls}}_n \sim \mathcal{N}(\mathbf{s}^{\text{cls}}, \mathbf{\Sigma}^{\text{cls}}). \tag{5.14}$$

The classification loss is defined as the softmax cross-entropy computed between the stochastic classification logits $\mathbf{S}^{\text{cls}}$ and the corresponding ground-truth labels.

Additionally, the bounding box regression is analogously conceptualized as a Gaussian distribution. In this representation, the mean corresponds to the predicted box offsets $\mathbf{b}^{\text{box}}$, while the diagonal covariance matrix is determined by the predicted box variances, specifically, $(\sigma^2_x, \sigma^2_y, \sigma^2_w, \sigma^2_h)$. Consequently, the loss for bounding box regression is calculated using the negative log-likelihood method as presented in BayesianYOLO [Kra19].

Overall DeCRCN-UPPR Pipeline

The overall DeCRCN-UPPR pipeline can be summarized as follows:

1. Initial proposals generated by CenterNet-RPN, in conjunction with the pyramid feature maps from the backbone, are forwarded to the first R-CNN stage.
2. The RoI-head processes the pooled features, extracting classification and bounding box features. These features are subsequently subjected to the RoI-predictor, yielding classification logits and variances as well as bounding box offsets and variances.

3. To incorporate epistemic uncertainties, stochasticity is introduced through dropout layers during the training phase. During inference, a series of R forward passes are executed, and the network predictions are collected and averaged to produce the final predictions.
4. The predicted box offsets are then applied to the initial proposals, resulting in refined boxes that serve as input for the subsequent R-CNN stage.

This iterative refinement process generates more reliable bounding boxes by capitalizing on the averaged epistemic predictions, which exhibit greater robustness compared to single-run predictions.

5.3.3 Experimental Evaluations

The DeCRCN-UPPR method, as introduced in this study, undergoes evaluation on established benchmarks forG-FSOD, specifically theMS-COCO [Lin14] and PASCAL-VOC [Eve10] datasets. To ensure a equitable comparison with prior research endeavors, the identical data partitions utilized in earlier studies [Wan20a, Qia21, Fan21, Gui22b] are adopted.

Implementation Details

The DeCRCN employs a ResNet-101 [He16] backbone that has undergone pre-training on the ImageNet dataset. It employs three cascaded R-CNN stages characterized by progressively increasing IoU thresholds: 0.5, 0.6, and 0.7, respectively. The network undergoes end-to-end optimization using the Stochastic Gradient Descent (SGD) algorithm, utilizing a mini-batch size of 16. The SGD algorithm is configured with a momentum value of 0.9 and a weight decay factor of $5e^{-5}$.

During base training, the total number of iterations is 110,000, with a learning rate of 0.02. Learning rate decay steps are executed at 85,000 and 100,000 iterations. Gradients stemming from the CenterNet-RPN are attenuated, and

Table 5.10: G-FSOD results on MS-COCO for 5,10,30-shot settings. w/E denotes the ensemble-based evaluation protocol.The best and second-best results are reported.

Methods / Shots	w/E	5 shot			10 shot			30 shot		
		AP	bAP	nAP	AP	bAP	nAP	AP	bAP	nAP
FRCN-ft-full[Wan20a]	✗	18.0	22.0	6.0	18.1	21.0	9.2	18.6	20.6	12.5
TFA w/ fc[Wan20a]	✗	27.5	33.9	8.4	27.9	33.9	10.0	29.7	35.1	13.4
TFA w/ cos[Wan20a]	✗	28.1	34.7	8.3	28.7	35.0	10.0	30.3	35.8	13.7
MPSR[Wu20a]	✗	-	-	-	15.3	17.1	9.7	17.1	18.1	14.1
DeFRCN [Qia21]	✗	28.7	33.1	15.3	30.6	34.6	18.6	31.6	34.7	**22.5**
ONCE [Per20]	✗	13.7	17.9	1.0	13.7	17.9	1.2	-	-	-
Meta R-CNN [Yan19]	✗	3.6	3.5	3.8	5.4	5.2	6.1	7.8	7.1	9.9
FSRW[Kan18]	✗	-	-	-	-	-	5.6	-	-	9.1
FsDetView [Xia20]	✗	5.9	5.7	6.6	6.7	6.4	7.6	10.0	9.3	12.0
CFA w/ fc [Gui22b]	✗	**30.1**	**37.1**	9.0	30.8	**37.6**	10.5	31.9	**37.7**	14.7
CFA w/ cos [Gui22b]	✗	29.7	36.3	9.8	30.3	36.6	11.3	31.7	37.0	15.6
CFA-DeFRCN [Gui22b]	✗	**30.1**	35.0	**15.6**	**31.4**	35.5	**19.1**	**32.0**	35.0	**23.0**
DeCRCN-UPPR	✗	**33.7**	**38.9**	**17.9**	**35.0**	**40.2**	**19.2**	**36.0**	**40.1**	**24.0**
Retentive R-CNN[Fan21]	✓	31.5	39.2	8.3	32.1	39.2	10.5	32.9	39.3	13.8
CFA w/ fc [Gui22b]	✓	31.8	**39.5**	8.8	32.2	**39.5**	10.4	33.2	**39.5**	14.3
CFA w/ cos [Gui22b]	✓	32.0	**39.5**	9.6	32.4	39.4	11.3	33.4	**39.5**	15.1
CFA-DeFRCN [Gui22b]	✓	**33.0**	38.9	**15.6**	**34.0**	39.0	**18.9**	**34.9**	39.0	**22.6**
DeCRCN-UPPR	✓	**35.9**	**41.9**	**17.8**	**36.2**	**41.9**	**19.1**	**37.8**	**42.0**	**23.8**

the R-CNN GDL scale, following the DeFRCN approach [Qia21], is established at $\lambda = 0.75$.

For novel finetuning, the total number of iterations is reduced to 4,000, accompanied by a learning rate of 0.01 and a decay step at 2,000 iterations. The RPN GDL scale is set to $\lambda = 0.04$, and a factor of 0.1 downscales the gradients emerging from the R-CNN layers. In the context of epistemic uncertainty estimation, a total of 40 forward passes are conducted with dropout layers configured with a dropout probability of 0.5. As for aleatoric uncertainty estimation, the number of classification samples is fixed at 10.

Comparison Results

The proposed DeCRCN-UPPR is compared against state-of-the-art G-FSOD [Wan20a, Qia21, Fan21, Gui22b] and FSOD models on MS-COCO and PASCAL-VOC benchmarks.

Results on MS-COCO

Table 5.10 provides an overview of the performance outcomes on the MS-COCO dataset. Notably, DeCRCN-UPPR demonstrates a substantial improvement over prior state-of-the-art results in terms of both AP and bAP across all configurations, while achieving slightly superior nAP. Additionally, the model evaluation under the ensemble evaluation protocol [Fan21] further highlights its superiority over alternative approaches.

PASCAL-VOC Results

Table 5.11 presents the overall performance on PASCAL-VOC (AP50). While Table 5.12 reports the novel performance (nAP50). The adoption of the UPPR methodology yields state-of-the-art outcomes across all shot settings, both with and without the ensemble evaluation protocol.

Ablation Experiments

Impact of Individual Modules

In Table 5.13, an ablation study is conducted to assess the contributions of individual components. Configuration A represents the baseline DeFRCN model. In configuration B, incorporating aleatoric and epistemic uncertainty estimation is introduced solely during novel training, resulting in a reduction in bAP but a marginal enhancement in nAP. Configuration C involves the replacement of the RoI-head with a Cascade R-CNN and the RPN with CenterNet, leading to improvements in the base metrics. This modified model

Table 5.11: The overall G-FSOD results (AP50) on PASCAL-VOC for $K = 1,2,3,5,10$-shot settings for all three splits. The best and second-best results are color coded.

Methods / Shots	w/E	All Set 1					All Set 2					All Set 3				
		1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
FRCN-ft-full[Wan20a]	✗	55.4	57.1	56.8	60.1	60.9	50.1	53.7	53.6	55.9	55.5	58.5	59.1	58.7	61.8	60.8
TFA w/ fc[Wan20a]	✗	69.3	66.9	70.3	73.4	73.2	64.7	66.3	67.7	68.3	68.7	67.8	68.9	70.8	72.3	72.2
TFA w/ cos[Wan20a]	✗	69.7	68.2	70.5	73.4	72.8	65.5	65.0	67.7	68.0	68.6	67.9	68.6	71.0	72.5	72.4
MPSR[Wu20a]	✗	56.8	60.4	62.8	66.1	69.0	53.1	57.6	62.8	64.2	66.3	55.2	59.8	62.7	66.9	67.7
DeFRCN[Qia21]	✗	73.1	73.2	73.7	75.1	**74.4**	68.6	69.8	71.0	72.5	71.5	72.5	73.5	72.7	74.1	73.9
Meta R-CNN[Yan19]	✗	17.5	30.5	36.2	49.3	55.6	19.4	33.2	34.8	44.4	53.9	20.3	31.0	41.2	48.0	55.1
FSRW[Kan18]	✗	53.5	50.2	55.3	56.0	59.5	55.1	54.2	55.2	57.5	58.9	54.2	53.5	54.7	58.6	57.6
FsDetView[Xia20]	✗	36.4	40.3	40.1	50.0	55.3	36.3	43.7	41.6	45.8	54.1	37.0	39.5	40.7	50.7	54.8
CFA w/ fc [Gui22b]	✗	69.5	68.2	69.8	73.5	74.3	66.0	66.9	69.2	70.1	71.1	67.7	69.0	70.9	72.6	73.5
CFA w/ cos [Gui22b]	✗	69.1	69.8	71.9	73.6	73.9	64.8	66.5	68.3	69.5	70.5	67.7	69.7	71.9	73.0	73.5
CFA-DeFRCN [Gui22b]	✗	**73.8**	**74.6**	**74.5**	**76.0**	**74.4**	**69.3**	**71.4**	**72.0**	**73.3**	**72.0**	**72.9**	**73.9**	**73.0**	**74.1**	**74.6**
DeCRCN-UPPR	✗	**74.3**	**75.1**	**75.4**	**76.3**	**75.1**	**71.4**	**72.6**	**73.2**	**74.9**	**73.2**	**73.2**	**74.3**	**74.2**	**75.3**	**75.8**
Retentive R-CNN[**retentive**]	✓	71.3	72.3	72.1	74.0	74.6	66.8	68.4	70.2	70.7	71.5	69.0	70.9	72.3	73.9	74.1
CFA w/ fc [Gui22b]	✓	70.3	69.5	71.0	74.4	74.9	67.0	68.0	70.2	70.8	71.5	69.1	70.1	71.6	73.3	74.7
CFA w/ cos [Gui22b]	✓	71.4	71.8	73.3	74.9	75.0	66.8	68.4	70.4	71.1	71.9	69.7	71.2	72.6	74.0	74.7
CFA-DeFRCN [Gui22b]	✓	**75.0**	**76.0**	**76.8**	**77.3**	**77.3**	**70.4**	**72.7**	**73.7**	**74.7**	**74.2**	**74.7**	**75.5**	**75.0**	**76.2**	**76.6**
DeCRCN-UPPR	✓	**76.1**	**77.0**	**77.9**	**78.2**	**78.4**	**71.3**	**73.5**	**74.4**	**75.1**	**75.2**	**75.1**	**76.9**	**76.2**	**77.3**	**77.5**

is referred to as DeCRCN. Configuration D introduces uncertainty estimation in a stage-wise manner, maintaining the bAP from the previous configuration while increasing the nAP. This underscores the significance of progressively applying uncertainty estimation throughout the stages of the Cascade R-CNN. While the naive application of uncertainty estimation can induce some degree of forgetting, its incremental integration within the Cascade R-CNN contributes to proposal refinement.

Configuration E demonstrates enhancements in all metrics by incorporating uncertainty estimation during base training as well. Finally, configurations F, H, and G explore the influence of attention blocks during novel training. Regardless of uncertainty estimation, including attention blocks during the novel training stage improves both bAP and nAP. However, when attention blocks are integrated into the base training phase, performance declines in the base classes. This underscores the critical nature of the design choice to train attention blocks using a balanced set encompassing both base and novel classes.

Table 5.12: PASCAL-VOC G-FSOD (nAP50) results for $K = 1,2,3,5,10$-shot settings for all three splits are reported. Similar to [Fan21, Gui22b], w/E denotes the ensemble-based inference paradigm [Fan21]. The best and second-best results are color coded.

Methods / Shots	**w/E**	**Novel Set 1**					**Novel Set 2**					**Novel Set 3**				
		1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
FRCN-ft-full[Wan20a]	✗	15.2	20.3	29.0	25.5	28.7	13.4	20.6	28.6	32.4	38.8	19.6	20.8	28.7	42.2	42.1
TFA w/ fc[Wan20a]	✗	36.8	29.1	43.6	55.7	57.0	18.2	29.0	33.4	35.5	39.0	27.7	33.6	42.5	48.7	50.2
TFA w/ cos[Wan20a]	✗	39.8	36.1	44.7	55.7	56.0	23.5	26.9	34.1	35.1	39.1	30.8	34.8	42.8	49.5	49.8
MPSR[Wu20a]	✗	42.8	43.6	48.4	55.3	61.2	29.8	28.1	41.6	43.2	47.0	35.9	40.0	43.7	48.9	51.3
DeFRCN[Qia21]	✗	57.0	58.6	64.3	67.8	67.0	35.8	42.7	51.0	54.4	52.9	52.5	56.6	55.8	**60.7**	62.5
Meta R-CNN[Yan19]	✗	16.8	20.1	20.3	38.2	43.7	7.7	12.0	14.9	21.9	31.1	9.2	13.9	26.2	29.2	36.2
FSRW[Kan18]	✗	14.8	15.5	26.7	33.9	47.2	15.7	15.3	22.7	30.1	39.2	19.2	21.7	25.7	40.6	41.3
MetaDet[Wan19c]	✗	18.9	20.6	30.2	36.8	49.6	21.8	23.1	27.8	31.7	43.0	20.6	23.9	29.4	43.9	44.1
FsDetView*[Xia20]	✗	25.4	20.4	37.4	36.1	42.3	22.9	21.7	22.6	25.6	29.2	32.4	19.0	29.8	33.2	39.8
CFA w/ fc [Gui22b]	✗	40.0	35.5	40.9	54.1	56.9	22.2	27.1	35.2	38.5	40.9	29.7	35.1	39.5	47.2	51.3
CFA w/ cos [Gui22b]	✗	41.2	43.6	49.5	56.5	57.3	21.3	27.4	35.3	39.1	42.1	31.7	39.1	44.6	49.9	52.6
CFA-DeFRCN [Gui22b]	✗	**58.2**	**63.3**	**65.8**	**68.9**	**67.1**	**37.1**	**45.5**	**51.3**	**55.2**	**53.8**	**54.7**	**57.8**	**56.9**	60.0	**63.3**
DeCRCN-UPPR	✗	**60.2**	**64.7**	**66.4**	**70.1**	**68.4**	**38.7**	**46.4**	**52.8**	**56.2**	**54.6**	**55.5**	**58.7**	**57.9**	**61.2**	**64.7**
Retentive R-CNN[**retentive**]	✓	42.4	45.8	45.9	53.7	56.1	21.7	27.8	35.2	37.0	40.3	30.2	37.6	43.0	49.7	50.1
CFA w/ fc [Gui22b]	✓	39.0	34.9	41.4	54.8	57.0	21.8	26.1	35.3	37.1	40.1	29.9	34.3	40.1	47.0	52.6
CFA w/ cos [Gui22b]	✓	42.4	43.9	50.3	56.6	57.3	21.0	27.5	35.3	38.6	41.4	32.3	38.0	44.5	49.8	52.7
CFA-DeFRCN [Gui22b]	✓	**59.0**	**63.5**	**66.4**	**68.4**	**68.3**	**37.0**	**45.8**	**50.0**	**54.2**	**52.5**	**54.8**	**58.5**	**56.5**	**61.3**	**63.5**
DeCRCN-UPPR	✓	**61.0**	**64.5**	**67.8**	**69.7**	**69.0**	**38.5**	**46.9**	**51.4**	**55.9**	**53.6**	**55.3**	**59.4**	**57.5**	**62.8**	**64.1**

Table 5.13: An ablation study performed on MS-COCO for the 10-shot setting to highlight the impact of different design choices. BT and NT denote base training and novel training, respectively. UE denotes uncertainty estimation (aleatoric and epistemic). ATT is the stage-wise instance-level attention.

Model Configuration	Base Training		Base		Novel		Overall	
	AP	AR	bAP	bAR	nAP	nAR	AP	AR
A DeFRCN	38.5	33.2	36.5	32.4	16.8	20.2	31.6	29.4
B DeFRCN (NT: UE)	38.5	33.2	34.9	31.3	17.5	20.8	30.5	28.6
C DeCRCN	41.4	35.7	38.2	34.1	17.5	21.6	33.0	31.0
D DeCRCN (NT: UE)	41.4	35.7	38	34.2	18.2	22.5	33.1	31.3
E DeCRCN (BT + NT: UE)	42.0	36.1	40.2	36.6	19.0	23.6	34.7	32.6
F DeCRCN (BT: UE, NT: ATT)	42.0	36.1	40.2	36.6	19.3	24.2	34.8	32.8
G DeCRCN (BT + NT: UE + ATT)	41.7	36.2	37.3	34.6	18.7	23.6	32.6	31.8
H DeCRCN (BT: UE, NT: UE + ATT) (UPPR)	42.0	36.1	40.5	36.7	19.2	24.0	35.0	32.8

Table 5.14: Multiple runs G-FSOD results for $K = 5{,}10{,}30$-shot settings on MS-COCO for multiple runs using 10 different seeds.

Methods / Shots	5 shot			10 shot			30 shot		
	AP	bAP	nAP	AP	bAP	nAP	AP	bAP	nAP
TFA w/ fc[Wan20a]	25.6±0.5	31.8±0.5	6.9±0.7	26.2±0.5	32.0±0.5	9.1±0.5	28.4±0.3	33.8±0.3	12.0±0.4
TFA w/ cos[Wan20a]	25.9±0.6	32.3±0.6	7.0±0.7	26.6±0.5	32.4±0.6	9.1±0.5	28.7±0.4	34.2±0.4	12.1±0.4
CFA w/ fc [Gui22b]	29.1±0.3	36.2±0.3	7.7±0.6	29.9±0.3	36.7±0.2	9.6±0.6	30.8±0.2	36.6±0.2	13.6±0.3
CFA w/ cos [Gui22b]	29.3±0.2	36.0±0.2	9.2±0.5	30.2±0.2	36.6±0.1	11.2±0.5	31.1±0.1	36.6±0.1	14.8±0.2
DeFRCN[Qia21]	27.8±0.3	32.6±0.3	13.6±0.7	29.7±0.2	34.0±0.2	16.8±0.6	31.4±0.1	34.8±0.1	21.2±0.4
CFA-DeFRCN [Gui22b]	28.4±0.2	32.8±0.2	15.2±0.5	30.2±0.2	34.0±0.2	18.8±0.4	31.7±0.1	34.6±0.1	23.0±0.3
DeCRCN-UPPR	31.8±0.1	36.5±0.1	17.4±0.2	33.7±0.1	38.5±0.1	18.9±0.1	35.6±0.0	39.6±0.1	24.0±0.1

Table 5.15: The G-FSOD multiple runs results for $K = 1{,}2{,}3{,}5{,}10$-shot settings on the three all sets of PASCAL-VOC (AP50).

Set	Methods	Shots				
		1	2	3	5	10
All Set 1	CFA w/ fc [Gui22b]	66.3±0.8	68.0±0.5	70.1±0.4	71.7±0.5	73.2±0.5
	CFA w/ cos [Gui22b]	66.5±0.9	69.2±0.6	71.1±0.6	72.5±0.4	73.4±0.4
	DeFRCN [Qia21]	67.8±1.4	71.3±0.8	72.6±0.5	73.6±0.5	74.1±0.5
	CFA-DeFRCN [Gui22b]	69.0±1.4	72.6±0.7	73.1±0.4	74.0±0.5	74.3±0.4
	DeCRCN-UPPR	70.1±0.9	73.5±0.5	74.9±0.5	75.3±0.4	75.7±0.4
All Set 2	CFA w/ fc [Gui22b]	64.9±0.9	66.4±0.7	68.3±0.5	69.6±0.3	70.8±0.5
	CFA w/ cos [Gui22b]	64.1±0.9	66.5±0.5	68.1±0.5	69.3±0.2	70.4±0.4
	DeFRCN [Qia21]	65.2±1.0	68.0±0.8	69.2±0.6	70.6±0.6	71.3±0.5
	CFA-DeFRCN [Gui22b]	66.4±1.0	69.0±0.8	70.4±0.7	71.3±0.7	72.1±0.4
	DeCRCN-UPPR	67.5±0.9	70.5±0.7	71.7±0.6	72.6±0.6	73.3±0.4
All Set 3	CFA w/ fc [Gui22b]	65.2±0.8	66.8±0.8	69.1±0.7	70.9±0.6	72.3±0.4
	CFA w/ cos [Gui22b]	64.9±1.2	67.5±1.0	69.7±0.8	71.6±0.5	72.7±0.3
	DeFRCN [Qia21]	66.9±2.0	70.6±0.8	71.2±0.6	72.9±0.5	73.5±0.3
	CFA-DeFRCN [Gui22b]	68.3±1.6	71.4±0.8	72.3±0.5	73.5±0.5	74.0±0.3
	DeCRCN-UPPR	69.5±0.8	73.7±0.6	74.7±0.5	75.5±0.4	76.3±0.2

Multiple Runs

To assess the robustness of DeCRCN-UPPR relative to other G-FSOD baselines, multi-seed experiments are conducted on both MS-COCO (using 10 different random seeds) and PASCAL-VOC (using 30 different random seeds). The

Table 5.16: G-FSOD results for $K = 1,2,3,5,10$-shot settings on the three novel sets of PASCAL-VOC (nAP50).

Set	Methods	Shots				
		1	2	3	5	10
Novel Set 1	CFA w/ fc [Gui22b]	28.2±3.1	35.0±1.9	41.9±1.4	47.8±1.6	53.3±1.6
	CFA w/ cos [Gui22b]	30.9±3.9	40.9±2.5	47.8±2.4	53.1±1.4	56.1±1.4
	DeFRCN [Qia21]	43.8±4.3	57.5±2.5	61.4±1.7	65.3±0.9	67.0±1.4
	CFA-DeFRCN [Gui22b]	45.4±4.9	60.3±2.2	62.1±1.4	66.4±0.9	67.6±1.2
	DeCRCN-UPPR	45.8±3.2	59.5±2.0	61.9±1.7	64.5±1.3	66.5±1.3
Novel Set 2	CFA w/ fc [Gui22b]	20.0±3.5	26.4±2.9	32.8±2.2	37.3±1.7	41.8±1.9
	CFA w/ cos [Gui22b]	21.0±3.5	29.0±2.3	34.6±2.3	38.9±1.2	43.0±1.9
	DeFRCN[Qia21]	31.5±3.6	40.9±2.2	45.6±2.0	50.1±1.4	52.9±1.1
	CFA-DeFRCN [Gui22b]	32.9±3.7	42.3±2.2	47.1±1.9	51.2±1.4	55.3±1.3
	DeCRCN-UPPR	29.8±3.4	39.4±1.9	44.7±2.0	49.1±1.8	51.5±1.4
Novel Set 3	CFA w/ fc [Gui22b]	20.3±3.4	26.4±3.1	34.3±2.5	41.2±2.4	46.5±1.6
	CFA w/ cos [Gui22b]	21.5±4.7	30.4±4.1	38.4±2.8	45.5±2.1	49.9±1.0
	DeFRCN [Qia21]	38.2±6.8	50.9±2.8	54.1±2.2	59.2±1.2	61.9±1.3
	CFA-DeFRCN [Gui22b]	41.4±5.8	52.9±3.0	56.1±1.7	60.3±1.1	62.9±0.9
	DeCRCN-UPPR	41.0±2.9	51.9±2.0	56.0±1.7	58.4±1.4	61.8±1.2

results, presented in Tables 5.14 (for MS-COCO) and Tables 5.15 and 5.16 (for PASCAL-VOC), encompass the mean and standard deviation values.

In Table 5.14, the outcomes on MS-COCO demonstrate that DeCRCN-UPPR consistently outperforms the other baseline models across various shot settings. Likewise, in the case of PASCAL-VOC, results for both overall performance (Table 5.15) and novel performance (Table 5.16) illustrate that DeCRCN-UPPR exhibits a higher overall AP50 and maintains competitive novel nAP50 scores for diverse shot settings. These findings align with the observations made in the MS-COCO experiments.

5.4 Discussion

In this chapter, the focus has been on addressing the challenge of forgetting in G-FSOD when base data is available during novel training. Two methods, namely CFA and DeCRCN-UPPR, have been introduced to alleviate forgetting and enhance the performance of novel object detection.

CFA adapts gradient episodic memory techniques by utilizing a static memory buffer to replay a few-shot of base objects. The newly derived gradient update rule is adaptively reweights the base and novel gradients to reduce forgetting. Compared to related GEM-based approaches [Lop17, Cha19a], CFA promotes better knowledge transfer between base and novel classes and can seamlessly integrate with FSOD frameworks without increasing model capacity or inference time.

DeCRCN-UPPR focuses on the design of a detector that refines proposals in a stagewise manner. This is achieved by leveraging predictive uncertainties and careful architectural modifications. Estimating uncertainties provides an extra source of distributional information to reduce forgetting and improve the novel detection performance. During novel training, attention blocks are appended to each R-CNN stage, enabling selective focus on discriminative features for improved classification. Integrating multiple R-CNN stages and attention blocks significantly enhances the detection performance of both base and novel classes.

The introduction of CFA and DeCRCN-UPPR presents valuable contributions to addressing forgetting in G-FSOD. These methods provide innovative strategies for transferring knowledge and improving the performance of novel object detection. The effectiveness of these methods has been demonstrated through experimental evaluations, showcasing their potential to improve the performance of few-shot detectors.

6 Regularized-based G-FSOD

In the preceding chapter, both the proposed and discussed G-FSOD approaches operate under the assumption that the base images are available during the novel training phase. However, this assumption may not hold in scenarios with constraints on data sharing and storage, primarily due to increasing privacy concerns surrounding AI models.

This chapter proposes the first Data-Free Knowledge Distillation (DFKD) approach for G-FSOD. Specifically, the statistics of the RoI features from the base model are utilized to generate instance-level features without direct access to the base images. The contribution of this work can be summarized in three aspects. First, the introduction of a standalone lightweight generator. Second, the incorporation of class-wise heads for generating and replaying diverse instance-level base features to the RoI head during the novel data finetuning process. Third, the distinction from conventional DFKD approaches in image classification, which typically involves inverting the entire network to generate base images. Additionally, careful design choices are made in the novel finetuning pipeline to enforce the regularization of the model. The experimental results demonstrate the effectiveness of the proposed approach in significantly reducing the memory requirements for base images while achieving state-of-the-art performance in G-FSOD on challenging benchmarks such as MS-COCO and PASCAL-VOC.

6.1 Literature Review

6.1.1 Regularization-based Continual Learning Methods

In contrast to the previously discussed replay-based methods in Section 5.1.2, regularization-based methods [Kir16, Alj18, Lee17, Zen17, Liu18b, Cha18, Li16, Jun16, Tri17, Zha20a] take a different approach to address the issue of catastrophic forgetting. Instead of relying on storing or replaying data, regularization-based methods incorporate regularization techniques to retain the knowledge of previous tasks. By adding penalty terms to the loss function, the network parameters are encouraged to remain close to their previous task solutions, ensuring that previous knowledge is preserved while learning new tasks. This approach proves advantageous in situations where there are limitations on privacy or memory resources, as it eliminates the necessity of storing any data related to previous tasks.

Regularization-based methods can be classified into two categories: prior-focused and data-focused approaches. These methods share the common objective of retaining knowledge acquired from previous tasks but differ in prioritizing and utilizing the previous task information. Prior-focused methods [Kir16, Alj18, Lee17, Zen17, Liu18b, Cha18] emphasize the preservation of network parameter values obtained from prior tasks. This is accomplished by incorporating penalty terms into the loss function, which enforces the parameters to remain close to their solutions for previous tasks. In contrast, data-focused methods [Li16, Jun16, Tri17, Zha20a] prioritize conserving the learned data distribution or patterns from past tasks. This is often achieved through generative models, which learn to capture and reproduce the underlying distribution of the data. In this chapter, the focus is directed towards the exploitation of prior-focused methods for G-FSOD. Specifically, the two widely used approaches, namely Elastic Weight Consolidation (EWC) [Kir16] and Memory Aware Synapses (MAS) [Alj18], due to their ease of implementation and established efficacy in retaining prior knowledge during sequential learning tasks.

Elastic Weight Consolidation

The fundamental concept underlying EWC [Kir16] involves constraining the network parameters according to their significance for previous tasks. This is accomplished by introducing a penalty term into the training loss function, which measures the discrepancy between the current and previous task parameters. The penalty term employed in EWC relies on utilizing the Fisher Information Matrix (FIM), which measures the network loss sensitivity to parameter variations. The FIM is computed as the expected value of the outer product of the gradients of the loss with respect to the parameters. By capturing the curvature of the loss landscape around the optimal parameters, the FIM facilitates understanding the importance attributed to each parameter.

Formally, the penalty term can be written as:

$$\frac{1}{2}\lambda_{\text{EWC}} \sum_i \mathbf{FIM}_i (\boldsymbol{\Theta}_i - \boldsymbol{\Theta}_i^*)^2, \tag{6.1}$$

where λ_{EWC} is a scaling factor to weight the penalty term. $\mathbf{FIM}_i$ is the FIM for the i^{th} parameter. $\boldsymbol{\Theta}_i$ is the current value of the network i^{th} parameter. $\boldsymbol{\Theta}_i^*$ is the optimal value of the i^{th} parameter from the previous task.

Memory Aware Synapses

In contrast to EWC, MAS [Alj18] makes use of a memory module to store target values of network parameters acquired from previous tasks. Subsequently, MAS incorporates a penalty term that encourages the current parameters to align with the stored target values.

However, MAS is explicitly tailored for MLPs. As a consequence, when employed with alternative network architectures such as CNNs with shared parameters, MAS might necessitate adjustments. In contrast, EWC exhibits compatibility with a diverse range of neural network architectures and loss functions, rendering it adaptable for various tasks without being constrained to a specific network architecture

6.1.2 Data-Free Knowledge Distillation

DFKD [Mor15, Yin20, Smi21, Cha21] encompasses a range of approaches with a shared objective of transferring knowledge without relying on storing raw data. DFKD methods focus on distilling and transferring knowledge from a teacher network to a student network by generating synthetic images instead of using original data from the previous tasks. A two-step noise optimization paradigm is prevalent among the commonly reviewed approaches in this area. In this paradigm, a noise vector is initially sampled from a Gaussian distribution and then iteratively optimized using SGD to produce a synthesized image. This optimization process aims to minimize the Kullback–Leibler (KL) divergence between the statistical properties of the synthesized images and the gathered statistics, assuming a Gaussian distribution. Standard data-driven Knowledge Distillation (KD) techniques are applied in the second stage, utilizing the synthetic images generated in the first stage. The goal is to transfer knowledge from the teacher network to the student network in a teacher-student fashion [Sha18, Xu20, Xu21], mitigating the issue of forgetting.

DeepDream

DeepDream [Mor15] was a pioneering work in the field of Model Inversion (MI) that focused on generating synthetic images by reversing a pre-trained classifier. It was the first of its kind to demonstrate that neural networks trained for classification tasks contain valuable information that can be utilized for image generation. DeepDream achieves this by synthesizing images that elicit strong responses for specific classes at various layers of the model. Specifically, DeepDream revolves around the fundamental concept of visualizing the patterns and features learned by neural networks during their training on extensive datasets. Departing from the conventional usage of neural networks for image classification, DeepDream takes a reverse approach, employing the network to generate images that optimize the activation of specific patterns and features to the maximum extent possible.

DeepInversion

DeepInversion [Yin20] seeks to generate images that, when fed into a trained neural network, produce desired activation responses. This is accomplished by refining the input image through an optimization process that minimizes the discrepancy between its representation in the network and a target representation. The optimization process entails iteratively adjusting the pixel values of the input image to align it with the target representation. Specifically, the backpropagated gradients are utilized to modify the image to encourage the desired activation patterns in the network. DeepInversion employs a loss function comprising two main elements [Yin20]: a misclassification loss and a regularization loss. The misclassification loss encourages the network to classify the generated image as a specific target class. Meanwhile, the regularization loss promotes smoothness and naturalness in the resulting image, preventing excessive noise or unrealistic features. By iteratively optimizing the input image based on these loss components, DeepInversion aims to discover an image that can activate the desired neurons or layers within the network while maintaining high fidelity.

Always Be Dreaming

Always Be Dreaming (ABD) [Smi21] was the first DFKD in a class-incremental setting. ABD comprises three main components. First, a modified cross-entropy loss that prevents the model from biasing the feature embeddings towards the new classes. Specifically, the cross-entropy loss is computed locally across the new class linear heads without considering the past class linear heads. This ensures that the model does not learn to separate classes based on their domain. Second, ABD minimizes the feature drift over previous task data. Due to the utilization of distillation images originating from a domain distinct from that of the present task images, an issue of feature domain bias arises. To address this, an importance-weighted feature distillation is proposed to reinforce important components of past task data while allowing less important features to adapt to the new task.

DIODE

While the previous approaches tackle the DFKD for the image classification task, DIODE [Cha21] proposed a non-class incremental approach for OD. DIODE consists of two primary components. Firstly, it incorporates a wide range of differentiable augmentations to improve the quality of images and enhance the efficacy of KD. This ensures that the semantic information in inverted images remains consistent despite the augmentations applied, resulting in images that align with natural images [Cha21]. Secondly, DIODE introduces a novel automated scheme for sampling bounding boxes and categories during image synthesis. This enables the generation of a substantial quantity of images that contain objects with diverse spatial distributions and categories. These generated images facilitate the process of DFKD from a teacher object detector to a student detector [Cha21]. The student detector is trained from scratch, starting with no prior information.

Despite DIODE [Cha21] successfully applying MI in the context of OD, it cannot be directly extended to G-FSOD due to a specific limitation. Like other existing works in DFKD, DIODE relies on the statistics of BN layers, which are pre-trained on detection datasets. However, in G-FSOD, the backbone network is pre-trained on ImageNet [Rus15] and kept frozen except for the last residual block during training. It is important to note that unfreezing the earlier backbone layers would change the mature pre-trained parameters and potentially decrease overall performance. Consequently, the running means and variances in the BN layers would no longer accurately represent the underlying base data distribution.

Two main challenges are highlighted at this point. Firstly, performing DFKD in G-FSOD with most of the backbone and all BN layers are frozen. Secondly, in contrast to image classification, the presence of multiple instances per image and the RPN make it difficult to invert the model prior to the RoI-head. Otherwise, generating bounding boxes for the synthesized images would be necessary, resulting in higher complexity and significantly increased computational and memory overhead.

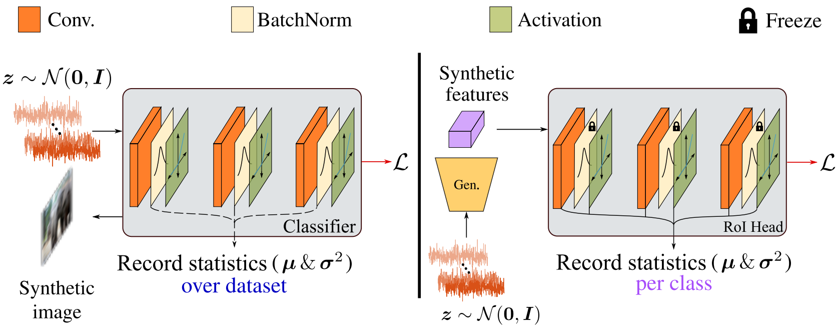

Conv.
BatchNorm
Activation
Freeze
$z \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$
Classifier
$\mathcal{L}$
Synthetic image
Record statistics (μ & σ^2)
over dataset
Synthetic features
Gen.
RoI Head
$\mathcal{L}$
Record statistics (μ & σ^2)
per class
$z \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Figure 6.1: Left: The vanilla DFKD approach using model inversion [Yin20, Smi21]. **Right:** An outline of the proposed DFKD method for G-FSOD, highlighting only a few layers. The key distinctions between the approaches are as follows: (1) Synthesizing features instead of images, (2) A standalone generator is employed rather than inverting the entire model, and (3) The class-wise statistics are recorded, specifically before and after the BN layers in the RoI-head.

6.2 The NIFF Framework

The objective is to develop a G-FSOD pipeline capable of learning novel classes with limited data while considering the privacy and memory limitations. In this section, the proposed approach Neural Instance Feature Forging (NIFF) is presented, comprising two stages: (1) Training a lightweight standalone feature generator, and (2) Conducting novel training using distillation between the base and novel RoI-heads with the trained generator. The abstract concept of the proposed method in comparison to the conventional DFKD approaches is depicted in Figure 6.1.

6.2.1 Feature Generator Training

In the initial stage of NIFF, a standalone feature generator is trained by aligning the class-wise statistics at the RoI-head. Subsequently, in the second stage, features are synthesized from the trained generator during novel training to mitigate forgetting. This section provides an in-depth description of the first stage, focusing on the design and training of the generator.

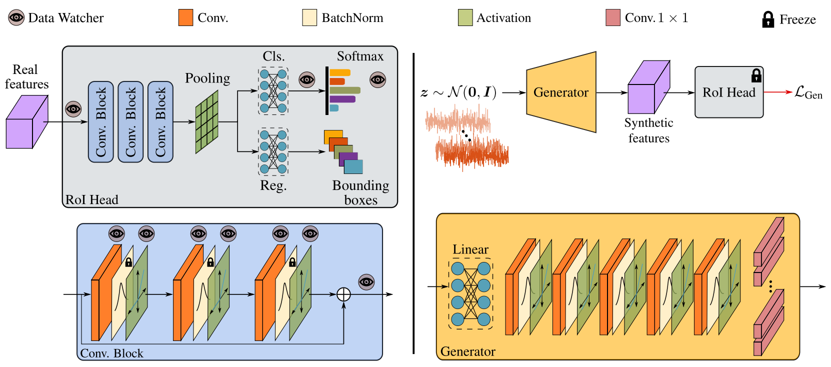


Figure 6.2: Illustration of the first NIFF stage involving the feature generator training. **Left:** An emphasis on the specific locations where feature statistics are collected through the utilization of data watchers. **Right:** A depiction of the generator training pipeline and the relevant architectural details.

Gathering Base Statistics

Considering the frozen state of the BN layers in G-FSOD models, an alternative method is required to obtain meaningful statistics, such as running means and variances, for the base RoI features. Instead of class-agnostic statistics, class-wise statistics are gathered to provide finer control over the number and class of generated features. This design choice compensates the sparser and harder classes in the base dataset through class conditional generation. To achieve this, the introduction of a data watcher block is proposed. Specifically, the data watcher block performs average pooling on the spatial dimensions of the input feature maps and records class-wise mean $\boldsymbol{\mu}_c$ and variance $\boldsymbol{\sigma}_c^2$ vectors, along with the sample size n_c. The statistics are updated using combined mean and corrected variance operations as follows:

$$\hat{\boldsymbol{\mu}}_{c,t} = \frac{\hat{n}_{c,t-1}\hat{\boldsymbol{\mu}}_{c,t-1} + n_{c,t}\boldsymbol{\mu}_{c,t}}{\hat{n}_{c,t-1} + n_{c,t}}, \tag{6.2}$$

$$\hat{\boldsymbol{\sigma}}^2_{c,t} = \frac{(\hat{n}_{c,t-1}-1)\hat{\boldsymbol{\sigma}}^2_{c,t-1} + (n_{c,t}-1)\boldsymbol{\sigma}^2_{c,t}}{\hat{n}_{c,t-1}+n_{c,t}-1} + \frac{\hat{n}_{c,t-1}n_{c,t}(\boldsymbol{\mu}_{c,t}-\hat{\boldsymbol{\mu}}_{c,t-1})}{(\hat{n}_{c,t-1}+n_{c,t})(\hat{n}_{c,t-1}+n_{c,t}-1)}. \tag{6.3}$$

At each iteration step t, the running estimate is denoted by $\hat{(.)}$. In order to enforce a greater diversity in the forged features, the decision is made to position data watchers at various layers within the RoI-head, specifically before the frozen BN layers and after the activations, as illustrated in Figure 6.2 (left). The number of data watchers determines the level of restriction on the forged features. To achieve this, additional data watchers are placed both before and after the softmax layer. These collected statistics should accurately represent a strong prior distribution of the base RoI features.

It is important to emphasize that once the statistics are collected for the base classes, the data is no longer accessible or stored. As a result, the model containing the underlying base statistics can be treated as a black box, preserving data privacy and enabling the sharing of the model with different parties without the need to share or store the data. Moreover, relying solely on the running averages of the RoI statistics leads to information loss, particularly since only RoI pooled features of fixed shape are utilized, making the reconstruction of the training data challenging. NIFF requires even fewer statistics than previous methods like DIODE [Cha21] which employs backbone statistics and, therefore, cannot reconstruct an entire image.

Generator Architecture

The gathered means and variances of the RoI features are leveraged to train the feature generator using SGD. In the right side of Figure 6.2, the proposed lightweight architecture of the generator is presented. It starts with a linear layer that maps the input noise vector $\boldsymbol{z} \in \mathbb{R}^{100}$ to $\mathbb{R}^{392}$, which is then reshaped to $\mathbb{R}^{8\times7\times7}$. Subsequently, the reshaped input is passed through five sequential convolutional blocks, each containing 2D convolutional layers with 8 channels and a kernel size of 3×3. To output class-wise features $\boldsymbol{f}_c \in \mathbb{R}^{1024\times7\times7}$, a number of $|\mathcal{C}_b|$ 1×1 convolutional blocks are appended. Synthetic features for

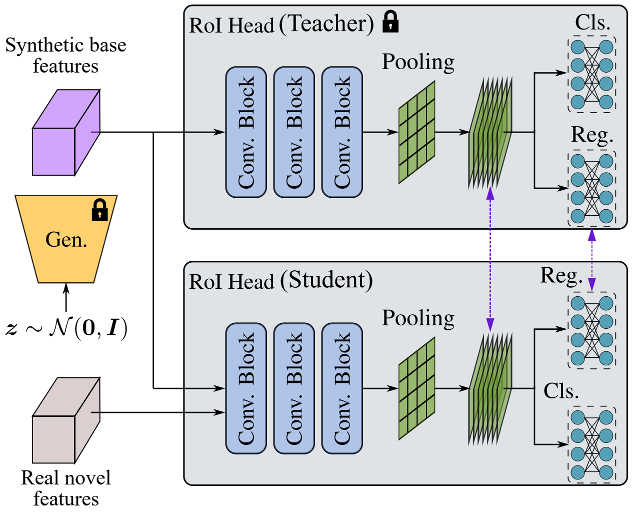


Figure 6.3: An overview of the second NIFF stage, featuring the knowledge distillation between the base and novel RoI-heads. Knowledge distillation is performed via the forged base features during novel training.

class c are generated by sampling noise $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$ and using $\boldsymbol{f}_c = G(c,\boldsymbol{z})$, where G represents our generator model.

Generator Training

The training of the feature generator focuses on forging instance-level base features by aligning the acquired statistics within the RoI-head at the corresponding layers. This alignment ensures the generation of diverse base features. To achieve class-specific feature generation, the class-wise statistics obtained from passing the features $\boldsymbol{f}_c$ through the RoI-head are aligned with the class-wise statistics collected by the data watchers. This approach deviates from aligning the class-agnostic statistics of the entire dataset. Additionally, to encourage each separate head to produce distinct $\boldsymbol{f}_c$ features, a cross-entropy loss is introduced between the target class-label $\boldsymbol{y}_{i,c}$ and the probability $\boldsymbol{p}_{i,c}$ at the final softmax layer.

In summary, the training of the generator is guided by two primary objectives: (1) Alignment of the RoI-head statistics with the gathered base statistics through the use of KL divergence under a Gaussian assumption, and (2) Maximization of the class probability by incorporating a cross-entropy loss. Mathematically, the generator training loss function $\mathcal{L}_{\text{Gen}}$ is denoted by:

$$\begin{aligned}\mathcal{L}_{\text{Gen}} =& \lambda_{\text{KL}} \frac{1}{|\mathcal{C}_b| * d} \sum_{c=1}^{|\mathcal{C}_b|} \sum_{i=1}^{d} \log \frac{\tilde{\boldsymbol{\sigma}}_{c,i}}{\boldsymbol{\sigma}_{c,i}} - \frac{1}{2} \left[1 - \frac{\boldsymbol{\sigma}_{c,i}^2 + (\tilde{\boldsymbol{\mu}}_{c,i} - \boldsymbol{\mu}_{c,i})^2}{\tilde{\boldsymbol{\sigma}}_{c,i}^2} \right] \\ & - \frac{1}{|\mathcal{C}_b| * N_{\text{feat}}} \sum_{i=1}^{|\mathcal{C}_b| * N_{\text{feat}}} \frac{1}{|\mathcal{C}_b|} \sum_{c=1}^{|\mathcal{C}_b|} \boldsymbol{y}_{i,c} \log(\boldsymbol{p}_{i,c}). \end{aligned} \tag{6.4}$$

The weighting factor λ_{KL} is applied to the loss, which is averaged over all data Watchers but excluded above for better readability. The ground-truth vector $\boldsymbol{y}_i$ is represented as a one-hot encoding. The dimension of the pooled features is denoted by d. The feature statistics $\boldsymbol{\mu}_c$ and $\boldsymbol{\sigma}_c^2$ are gathered, while the fake feature statistics $\tilde{\boldsymbol{\mu}}_c$ and $\tilde{\boldsymbol{\sigma}}_c^2$ are generated. The total number of generated features per class is denoted as N_{feat}, and during training $N_{\text{feat}} = 600$. To prevent memory overflow, the features for each class are sequentially fed, and the gradients are accumulated for backpropagation at the end.

6.2.2 Improved Novel Training Pipeline

In the final stage of NIFF, novel finetuning is performed while conducting KD at the RoI-head using a teacher-student approach. As illustrated in Figure 6.3, the forged base instance-level features as well as the novel features, are fed to the base and novel RoI-heads, respectively. To align with the finetuning K-shot setting, a total of $N_{\text{feat}} = K$ features per class are set, ensuring that all base classes are encountered in each iteration. This provides a notable advantage compared to data-dependent approaches such as [Qia21, Fan21, Gui22b], which employ base images containing only a few classes in each iteration. Another advantage of our approach is its generative nature, allowing the sampling of features to produce diverse results. In contrast, [Qia21, Fan21, Gui22b] utilize a fixed number of shots per base class during training, limiting the distribution seen by the model.

The distillation process is carried out in the following manner: Firstly, a weighted feature distillation using L2-norm is employed to penalize the difference between class-wise pooled RoI features of the teacher ($\boldsymbol{F}^T \in \mathbb{R}^{(|\mathcal{C}_b| \cdot K) \times d}$ and student $\boldsymbol{F}^S \in \mathbb{R}^{(|\mathcal{C}_b| \cdot K) \times d}$, where d represents the pooled feature dimension. To account for the generation of class-specific features, the difference between the features is weighted using the weight vector $\boldsymbol{W}^c_{\text{Cls}} \in \mathbb{R}^d$ from the complete classification weight matrix $\boldsymbol{W}_{\text{Cls}} \in \mathbb{R}^{|\mathcal{C}_b| \times d}$ for the corresponding class c. Similarly, the feature differences in regression are also weighted using the weight matrix $\boldsymbol{W}^c_{\text{Reg}} \in \mathbb{R}^{4 \times d}$ obtained from the regression weight tensor $\boldsymbol{W}_{\text{Reg}} \in \mathbb{R}^{(4 \cdot |\mathcal{C}_b|) \times d}$. Secondly, the alignment of class-wise regression logits is achieved by penalizing the drift between the predicted offsets of the teacher $\boldsymbol{reg}^T_c \in \mathbb{R}^4$ and student $\boldsymbol{reg}^S_c \in \mathbb{R}^4$. Formally, the KD loss can be denoted by:

$$\begin{aligned} \mathcal{L}_{\text{KD}} = \lambda_F \frac{1}{|\mathcal{C}_b| \cdot K} \sum_{i=1}^{|\mathcal{C}_b| \cdot K} \|(\boldsymbol{F}^T_i - \boldsymbol{F}^S_i) \cdot \boldsymbol{W}^{c_i}_{\text{Cls}}\|^2_2 \\ + \lambda_F \frac{1}{4 \cdot |\mathcal{C}_b| \cdot K} \sum_{i=1}^{|\mathcal{C}_b| \cdot K} \sum_{j=1}^{4} \|(\boldsymbol{F}^T_i - \boldsymbol{F}^S_i) \cdot \boldsymbol{W}^{c_i,j}_{\text{Reg}}\|^2_2 \\ + \frac{1}{|\mathcal{C}_b| \cdot K} \sum_{i=1}^{|\mathcal{C}_b| \cdot K} \|\boldsymbol{reg}^T_i - \boldsymbol{reg}^S_i\|_1, \end{aligned} \tag{6.5}$$

where λ_F is a hyperparameter to weight the feature distillation. Thirdly, analogous to [Smi21], a cross-entropy loss during finetuning is employed to maximize the confidence of the synthesized features:

$$\mathcal{L}_{\text{conf}} = -\frac{1}{|\mathcal{C}_b| \cdot K} \sum_{i=1}^{|\mathcal{C}_b| \cdot K} \frac{1}{|\mathcal{C}|} \sum_{c=1}^{|\mathcal{C}|} \boldsymbol{y}_{i,c} \log(\boldsymbol{p}_{i,c}). \tag{6.6}$$

Formally, the overall novel training loss $\mathcal{L}_{\text{N}}$ can be written as:

$$\mathcal{L}_{\text{N}} = \mathcal{L}_{\text{Cls}} + \mathcal{L}_{\text{Reg}} + \mathcal{L}_{\text{KD}} + \mathcal{L}_{\text{conf}}. \tag{6.7}$$

$\mathcal{L}_{\text{Cls}}$ and $\mathcal{L}_{\text{Reg}}$ are the cross-entropy and smooth L1 losses, respectively [Ren15].

Additional Regularization

It has been observed that by replaying the generated base features, the proposed model achieves performance that is nearly comparable to the state-of-the-art in terms of overall AP. Throughout the novel training phase, several design choices in the training pipeline have been identified that can enhance the overall detection performance. A chosen approach is to implement the earlier proposed CFA [Gui22b] while utilizing the backpropagated base gradients enabled by the availability of the forged base features. Furthermore, an investigation is conducted into various techniques for pixel-level and parameter-level regularization. Regarding the former, random color jittering, such as brightness, contrast, and saturation, random flipping, and random cropping are applied as augmentations, each with a probability of augmentation $p_{\text{aug}} = 0.5$.

Regarding parameter-level regularization, it has been shown that the utilization of the EWC [Kir16], initially designed for image classification, proves effective in alleviating forgetting. The importance of the parameters is weighted using the diagonal of the FIM, which is computed by squaring the backpropagated gradients obtained during the final epoch of base training. During novel training, EWC penalizes changes in significant parameters based on the FIM, facilitating more efficient knowledge transfer.

However, the FIM consumes a substantial amount of memory due to storing a weight for each model parameter. To mitigate this drawback, an approach is employed to average the weights of each layer in the FIM, resulting in a reduction of memory usage from approximately 200MB to 6.8KB. As a result, a single scalar value represents the importance of each layer. This approximation of EWC is referred to as mean Elastic Weight Consolidation (mEWC).

6.3 Experimental Evaluations

The evaluation of the proposed approach is conducted on widely recognized benchmarks for G-FSOD, namely the MS-COCO [Lin14] and PASCAL-VOC [Eve10] datasets. To ensure a fair comparison, the identical data splits utilized in previous studies [Wan20a, Qia21, Fan21, Gui22b] are employed.

6.3.1 Implementation Details

Generator Training

The generator is trained for 2k iterations using the RoI-head parameters from base training and the collected statistics from the data watchers. SGD is employed to optimize the generator, with a batch size of 600 features. The momentum is set to 0.9, and a weight decay of $5e-5$ is applied. The learning rate is fixed at 0.001. The scaling factor for the KL divergence loss is specified as $\lambda_{\text{KL}} = 5$.

Novel Training

During the process of novel training, the model undergoes optimization using SGD with a batch size of 16. The learning rate is set to 0.005 for MS-COCO and 0.01 for PASCAL-VOC. A warmup period of 200 iterations is implemented. Step decays are performed at specific iterations: 2500, 4000, and 6400 for MS-COCO in the 5-shot, 10-shot, and 30-shot settings, respectively. For PASCAL-VOC, the decay is conducted at 1000 and 1500 iterations for the 5-shot and 10-shot settings, respectively, in the first three shot settings. To enable distillation, the RoI-head is unfrozen, and the learning rate is scaled down by a factor of 0.015. The scaling factors for the mEWC penalty term are set as follows: $\lambda_{\text{F}} = 0.1$ for MS-COCO and $\lambda_{\text{F}} = 0.01$ for PASCAL-VOC. All training activities are conducted using four Nvidia GeForce 1080Ti GPUs.

Table 6.1: The G-FSOD results on MS-COCO for $K = 5{,}10{,}30$-shot settings. w/E indicates the ensemble-based evaluation protocol [Fan21]. w/B denotes whether the base data is available during novel finetuning. The best and second-best results are reported.

Methods / Shots	**w/E**	**w/B**	**5 shot**			**10 shot**			**30 shot**		
			AP	bAP	nAP	AP	bAP	nAP	AP	bAP	nAP
FRCN-ft-full[Wan20a]	✗	✓	18.0	22.0	6.0	18.1	21.0	9.2	18.6	20.6	12.5
TFA w/ fc[Wan20a]	✗	✓	27.5	33.9	8.4	27.9	33.9	10.0	29.7	35.1	13.4
TFA w/ cos[Wan20a]	✗	✓	28.1	34.7	8.3	28.7	35.0	10.0	30.3	35.8	13.7
MPSR[Wu20a]	✗	✓	-	-	-	15.3	17.1	9.7	17.1	18.1	14.1
DeFRCN [Qia21]	✗	✓	28.7	33.1	15.3	30.6	34.6	**18.6**	31.6	34.7	**22.5**
ONCE [Per20]	✗	✓	13.7	17.9	1.0	13.7	17.9	1.2	-	-	-
Meta R-CNN [Yan19]	✗	✓	3.6	3.5	3.8	5.4	5.2	6.1	7.8	7.1	9.9
FSRW[Kan18]	✗	✓	-	-	-	-	-	5.6	-	-	9.1
FsDetView [Xia20]	✗	✓	5.9	5.7	6.6	6.7	6.4	7.6	10.0	9.3	12.0
CFA w/ fc [Gui22b]	✗	✓	**30.1**	**37.1**	9.0	30.8	**37.6**	10.5	31.9	**37.7**	14.7
CFA w/ cos [Gui22b]	✗	✓	29.7	**36.3**	9.8	30.3	**36.6**	11.3	31.7	37.0	15.6
CFA-DeFRCN [Gui22b]	✗	✓	**30.1**	35.0	**15.6**	**31.4**	35.5	**19.1**	**32.0**	35.0	**23.0**
DeFRCN	✗	✗	23.7	26.3	**15.6**	18.2	18.5	17.4	16.3	16.3	21.4
NIFF-DeFRCN	✗	✗	**31.3**	**36.3**	**15.7**	**32.2**	**36.6**	**19.1**	**33.1**	**37.2**	21.0
Retentive R-CNN[Fan21]	✓	✓	31.5	**39.2**	8.3	32.1	39.2	10.5	32.9	**39.3**	13.8
CFA w/ fc [Gui22b]	✓	✓	31.8	**39.5**	8.8	32.2	**39.5**	10.4	33.2	**39.5**	14.3
CFA w/ cos [Gui22b]	✓	✓	32.0	**39.5**	**9.6**	**32.4**	**39.4**	11.3	33.4	**39.5**	15.1
CFA-DeFRCN [Gui22b]	✓	✓	**33.0**	38.9	**15.6**	**34.0**	39.0	**18.9**	**34.9**	39.0	**22.6**
NIFF-DeFRCN	✓	✗	**33.1**	38.9	**15.9**	**34.0**	39.0	**18.8**	**34.5**	39.0	**20.9**

6.3.2 Comparison Results

Results on MS-COCO

The results on the MS-COCO dataset are presented in Table 6.1. The inclusion of base data is indicated by (w/B). To assess the impact of removing base data on G-FSOD, the baseline method DeFRCN [Qia21] is reevaluated without any base data. It is observed that both the base and novel performance exhibit a decline across all shot settings, indicating the importance of base data for knowledge transfer to new tasks. By leveraging NIFF, consistent improvements are achieved in the base performance across all settings, resulting in higher overall AP performance.

Table 6.2: The G-FSOD results for various baselines, with DeFRCN [Qia21] being the base model, on MS-COCO for $K = 5,10,30$-shot settings. w/E indicates the ensemble-based evaluation protocol [Fan21]. w/B denotes whether the base data is available during novel finetuning. $\checkmark_F$ indicates finetuning with offline stored base RoI features. KD is the proposed knowledge distillation approach. The best and second-best results are reported.

Methods / Shots	w/B	5-Shot			10-Shot			30-Shot		
		AP	bAP	nAP	AP	bAP	nAP	AP	bAP	nAP
DeFRCN [Qia21]	✓	28.7	33.1	15.3	30.6	34.6	**18.6**	31.6	34.7	**22.5**
DeFRCN	✗	23.7	26.3	**15.6**	18.2	18.5	17.4	16.3	16.3	21.4
DeFRCN w/ DA	✗	22.6	25.0	15.3	26.4	29.2	17.9	24.2	25.0	**21.8**
DeFRCN w/ MAS	✗	31.0	**36.8**	13.5	31.5	**36.8**	15.3	32.6	36.6	20.4
DeFRCN w/ EWC	✗	**31.1**	**37.1**	13.4	**31.8**	**36.9**	16.6	**33.0**	**37.3**	20.1
DeFRCN	$\checkmark_F$	24.2	27.6	13.6	25.8	28.9	16.6	26.6	29.0	19.7
DeFRCN + CFA	$\checkmark_F$	26.0	29.9	13.7	27.7	31.4	16.6	28.6	31.5	19.9
DeFRCN w/ KD	$\checkmark_F$	25.3	28.8	14.3	27.0	30.2	17.4	27.9	30.3	20.5
DeFRCN w/ KD + CFA	$\checkmark_F$	28.4	33.3	14.1	30.5	34.9	17.1	31.3	35.0	20.3
NIFF-DeFRCN	✗	**31.3**	36.3	**15.7**	**32.2**	36.6	**19.1**	**33.1**	**37.2**	21.0

The model is also evaluated using the ensemble evaluation protocol in Retentive R-CNN [Fan21], where despite the absence of base data (with a 0.4 AP difference in the 30-shot setting), superior performance is achieved compared to other approaches. It is important to note that this evaluation protocol requires the retention of base model parameters [Fan21, Gui22b], leading to increased memory usage and inference time. Furthermore, the proposed approach, NIFF-DeFRCN, (w/o E and w/o B), outperforms Retentive R-CNN [Fan21] (w/E and w/B) in terms of overall AP performance.

Data-Free and G-FSOD Baselines

A comparison is made between our method and various baselines in Table 6.2, including base-data-free and G-FSOD baselines built upon DeFRCN [Qia21]. The data-free baselines are derived from regularization-based CL approaches, such as pixel-level data augmentations, EWC[Kir16], Memory Aware Synapses (MAS) [Alj18] using computed FIM. As CFA [Gui22b] cannot be conducted in a data-free setting, DeFRCN [Qia21] is trained and base RoI features are saved for later application of CFA during novel training. Two additional baselines are introduced: the proposed KD method applied to DeFRCN using saved base RoI

Table 6.3: The G-FSOD results (AP50) on PASCAL-VOC for $K = 1,2,3,5,10$-shot settings for the three different splits. w/E indicates the ensemble-based evaluation protocol [Fan21]. w/B denotes whether the base data is available during novel fine-tuning. The best and second-best results are reported.

Methods / Shots	w/E	w/B	All Set 1					All Set 2					All Set 3				
			1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
FRCN-ft-full[Wan20a]	✗	✓	55.4	57.1	56.8	60.1	60.9	50.1	53.7	53.6	55.9	55.5	58.5	59.1	58.7	61.8	60.8
TFA w/ fc[Wan20a]	✗	✓	69.3	66.9	70.3	73.4	73.2	64.7	66.3	67.7	68.3	68.7	67.8	68.9	70.8	72.3	72.2
TFA w/ cos[Wan20a]	✗	✓	69.7	68.2	70.5	73.4	72.8	65.5	65.0	67.7	68.0	68.6	67.9	68.6	71.0	72.5	72.4
MPSR[Wu20a]	✗	✓	56.8	60.4	62.8	66.1	69.0	53.1	57.6	62.8	64.2	66.3	55.2	59.8	62.7	66.9	67.7
DeFRCN[Qia21]	✗	✓	73.1	73.2	73.7	75.1	**74.4**	68.6	**69.8**	71.0	72.5	71.5	72.5	73.5	72.7	74.1	73.9
Meta R-CNN[Yan19]	✗	✓	17.5	30.5	36.2	49.3	55.6	19.4	33.2	34.8	44.4	53.9	20.3	31.0	41.2	48.0	55.1
FSRW[Kan18]	✗	✓	53.5	50.2	55.3	56.0	59.5	55.1	54.2	55.2	57.5	58.9	54.2	53.5	54.7	58.6	57.6
FsDetView[Xia20]	✗	✓	36.4	40.3	40.1	50.0	55.3	36.3	43.7	41.6	45.8	54.1	37.0	39.5	40.7	50.7	54.8
CFA w/ fc [Gui22b]	✗	✓	69.5	68.2	69.8	73.5	74.3	66.0	66.9	69.2	70.1	71.1	67.7	69.0	70.9	72.6	73.5
CFA w/ cos [Gui22b]	✗	✓	69.1	69.8	71.9	73.6	73.9	64.8	66.5	68.3	69.5	70.5	67.7	69.7	71.9	73.0	73.5
CFA-DeFRCN [Gui22b]	✗	✓	**73.8**	**74.6**	**74.5**	**76.0**	**74.4**	**69.3**	**71.4**	**72.0**	**73.3**	**72.0**	**72.9**	**73.9**	**73.0**	**74.1**	**74.6**
DeFRCN	✗	✗	61.1	48.5	35.9	32.8	20.7	64.7	59.7	58.2	56.9	48.4	56.3	51.2	46.9	38.8	23.9
NIFF-DeFRCN	✗	✗	**75.6**	**76.5**	**76.7**	**77.4**	**76.9**	**70.0**	**71.4**	**73.9**	**74.4**	**74.0**	**74.4**	**75.8**	**76.2**	**76.6**	**76.7**
Retentive R-CNN[Fan21]	✓	✓	71.3	72.3	72.1	74.0	74.6	66.8	68.4	70.2	70.7	71.5	69.0	70.9	72.3	73.9	74.1
CFA w/ fc [Gui22b]	✓	✓	70.3	69.5	71.0	74.4	74.9	67.0	68.0	70.2	70.8	71.5	69.1	70.1	71.6	73.3	74.7
CFA w/ cos [Gui22b]	✓	✓	71.4	71.8	73.3	74.9	75.0	66.8	68.4	70.4	71.1	71.9	**69.7**	71.2	72.6	74.0	74.7
CFA-DeFRCN [Gui22b]	✓	✓	**75.0**	**76.0**	**76.8**	**77.3**	**77.3**	**70.4**	**72.7**	**73.7**	**74.7**	**74.2**	**74.7**	**75.5**	**75.0**	**76.2**	**76.6**
NIFF-DeFRCN	✓	✗	**75.9**	**76.9**	**77.3**	**77.9**	**77.5**	**70.6**	**71.6**	**74.5**	**75.1**	**74.5**	**74.7**	**76.0**	**76.1**	**76.8**	**76.7**

features with and without CFA. Our method consistently outperforms both data-free and data-reliant baselines. The diversity of the forged features is argued to be the factor enabling our method to surpass data-reliant baselines. It is also noteworthy that the stored features require 114.8 MB of memory, which is significantly more than the memory requirements of our generator (3.7 MB) and statistics (12.4 MB). Similarly, EWC and MAS exhibit high memory usage due to the FIM, requiring approximately 200 MB.

Results on PASCAL-VOC

The overall performance on PASCAL-VOC (AP50) and the novel performance (nAP50), are presented in Table 6.3 and Table 6.4, respectively. The adoption of NIFF demonstrates state-of-the-art results, both with and without the ensemble evaluation protocol. Although the primary objective is not focused on the

Table 6.4: The G-FSOD novel results (nAP50) on PASCAL-VOC for $K = 1,2,3,5,10$-shot settings for the three different splits. w/E indicates the ensemble-based evaluation protocol [Fan21]. w/B denotes whether the base data is available during novel fine-tuning. The best and second-best results are reported. '-' represents missing results in previous works. '*' denotes results reported in [Gui22b].

Methods / Shots	w/E	w/B	Novel Set 1					Novel Set 2					Novel Set 3				
			1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
FRCN-ft-full[Wan20a]	✗	✓	15.2	20.3	29.0	25.5	28.7	13.4	20.6	28.6	32.4	38.8	19.6	20.8	28.7	42.2	42.1
TFA w/ fc[Wan20a]	✗	✓	36.8	29.1	43.6	55.7	57.0	18.2	29.0	33.4	35.5	39.0	27.7	33.6	42.5	48.7	50.2
TFA w/ cos[Wan20a]	✗	✓	39.8	36.1	44.7	55.7	56.0	23.5	26.9	34.1	35.1	39.1	30.8	34.8	42.8	49.5	49.8
MPSR[Wu20a]	✗	✓	42.8	43.6	48.4	55.3	61.2	29.8	28.1	41.6	43.2	47.0	35.9	40.0	43.7	48.9	51.3
DeFRCN[Qia21]	✗	✓	57.0	58.6	64.3	67.8	67.0	35.8	42.7	51.0	54.4	52.9	52.5	56.6	55.8	**60.7**	62.5
Meta R-CNN*[Yan19]	✗	✓	16.8	20.1	20.3	38.2	43.7	7.7	12.0	14.9	21.9	31.1	9.2	13.9	26.2	29.2	36.2
FSRW[Kan18]	✗	✓	14.8	15.5	26.7	33.9	47.2	15.7	15.3	22.7	30.1	39.2	19.2	21.7	25.7	40.6	41.3
MetaDet[Wan19c]	✗	✓	18.9	20.6	30.2	36.8	49.6	21.8	23.1	27.8	31.7	43.0	20.6	23.9	29.4	43.9	44.1
FsDetView*[Xia20]	✗	✓	25.4	20.4	37.4	36.1	42.3	22.9	21.7	22.6	25.6	29.2	32.4	19.0	29.8	33.2	39.8
CFA w/ fc [Gui22b]	✗	✓	40.0	35.5	40.9	54.1	56.9	22.2	27.1	35.2	38.5	40.9	29.7	35.1	39.5	47.2	51.3
CFA w/ cos [Gui22b]	✗	✓	41.2	43.6	49.5	56.5	57.3	21.3	27.4	35.3	39.1	42.1	31.7	39.1	44.6	49.9	52.6
CFA-DeFRCN [Gui22b]	✗	✓	**58.2**	**63.3**	**65.8**	**68.9**	**67.1**	**37.1**	**45.5**	**51.3**	**55.2**	**53.8**	**54.7**	**57.8**	**56.9**	60.0	**63.3**
DeFRCN	✗	✗	53.3	47.4	58.7	58.8	59.6	33.0	37.0	49.5	53.8	48.5	47.1	45.8	52.7	52.8	52.6
NIFF-DeFRCN	✗	✗	**62.8**	**67.2**	**68.0**	**70.3**	**68.8**	**38.4**	**42.9**	**54.0**	**56.4**	**54.0**	**56.4**	**62.1**	**61.2**	**64.1**	**63.9**
Retentive R-CNN[Fan21]	✓	✓	42.4	45.8	45.9	53.7	56.1	21.7	27.8	35.2	37.0	40.3	30.2	37.6	43.0	49.7	50.1
CFA w/ fc [Gui22b]	✓	✓	39.0	34.9	41.4	54.8	57.0	21.8	26.1	35.3	37.1	40.1	29.9	34.3	40.1	47.0	52.6
CFA w/ cos [Gui22b]	✓	✓	42.4	43.9	50.3	56.6	57.3	21.0	27.5	35.3	38.6	41.4	32.3	38.0	44.5	49.8	52.7
CFA-DeFRCN [Gui22b]	✓	✓	**59.0**	**63.5**	**66.4**	**68.4**	**68.3**	**37.0**	**45.8**	**50.0**	**54.2**	**52.5**	**54.8**	**58.5**	**56.5**	**61.3**	**63.5**
NIFF-DeFRCN	✓	✗	**63.5**	**67.2**	**68.3**	**71.1**	**69.3**	**37.8**	**41.9**	**53.4**	**56.0**	**53.5**	**55.3**	**60.5**	**61.1**	**63.7**	**63.9**

performance of novel classes, NIFF-DeFRCN achieves competitive results on both datasets in the majority of cases.

6.3.3 Ablation Experiments

Generator Design Choices

Different generator design choices are investigated in Table 6.5 without regularization. This includes the standalone generator, class-wise statistics, separate heads, and the number of channels per layer. Initially, the RoI-head is inverted to generate instance-level features by minimizing the KL loss with respect to the gathered base data statistics. This can be seen as an extension of the

Table 6.5: Investigation of various design choices in the generator on the MS-COCO dataset in the 10-shot setting, without the inclusion of any additional regularization techniques. Memory here refers to the additional storage needed beyond the detection model.

Model Configuration	10-Shot Inference				
	AP	bAP	nAP	AR	Memory [MB]
Inverted RoI head	28.9	32.6	17.7	27.1	**0**
Gen. w/ shared head w/o classwise stats	29.3	33.1	17.8	27.3	1.6
Gen. w/ shared head w/ classwise stats	28.5	32.1	17.7	26.4	1.6
Gen. w/ separate heads w/o classwise stats	29.0	32.8	17.5	27.4	3.7
Gen. w/ separate heads w/ classwise stats **(Ours)**	**30.7**	**35.0**	17.8	28.6	3.7
Ours w/o cross-entropy term	30.0	34.1	17.6	28.6	3.7
Ours w/ (dim = 64)	**30.7**	**35.0**	17.8	**28.8**	28.7
Ours w/ (dim = 32)	30.5	34.8	**17.9**	28.6	14.0
Ours w/ (dim = 16)	**30.7**	34.9	**17.9**	28.7	7.1
Ours w/ (dim = 8)	**30.7**	**35.0**	17.8	28.6	3.7

standard MI approach ABD [Smi21] to G-FSOD, but with synthesizing features instead of images. Subsequently, a standalone generator is trained with a shared head for all classes while minimizing the full base-data statistics. Despite the minimal memory overhead, it outperforms the inverted model, supporting the claim that a separate generator is easier to optimize. However, when trained with class-wise statistics, a slight performance drop exists. Replacing the shared head with separate class-aware heads and minimizing the full base statistics achieves a similar performance as the generator with a shared head (row 2). The best overall performance is attained when combining the separate heads with the class-wise statistics, allowing the model to better account for inter-class variance. Simply extending the model with either class-wise statistics or class-wise heads reduces overall performance.

Table 6.6: The impact of the number of generated features per class and using fixed samples on forgetting and detection performance is examined in the absence of any additional regularization techniques.

Feature(s) per class	10-Shot Inference					
	AP	bAP	nAP	AR	bAR	nAR
1	29.9	34.0	17.6	28.4	31.0	**20.7**
5	30.6	34.9	**17.7**	**28.8**	**31.5**	**20.8**
10	**30.7**	**35.0**	**17.8**	**28.8**	**31.5**	**20.8**
30	**30.8**	**35.1**	**17.8**	**28.8**	**31.5**	**20.8**
10 (fixed)	25.9	28.9	16.9	25.5	27.1	20.6
10 (10 sampled cls)	30.5	34.8	**17.7**	**28.7**	**31.4**	20.6

Additionally, experiments are conducted with the complete generator version but with removing the cross-entropy loss from Equation 6.4, resulting in a slight decrease in overall AP. In the lower part of the table, the trade-off between the overall AP and memory is studied by altering the number of channels per generator layer. Interestingly, it is found that the models with 64 channels and 8 channels perform similarly, leading to the choice of the minimalist design with 8 channels.

Impact of Generated Features Sampling

Table 6.6 investigates the impact of sampling techniques on feature generation during novel finetuning, without any regularization. Initially, only one feature per class is generated, and then the number of features is increased to assess its effect on overall AP. It is observed that the best overall results are achieved when generating $N_{\text{feat}} = 10$ features per class, matching the 10-shot finetuning setting. Further increases in the number of features yield similar results, leading to the consistent choice of setting $N_{\text{feat}} = K$ throughout the experiments on MS-COCO and PASCAL-VOC. In row (5), 10 features per class are generated by sampling only once at the beginning of training and keeping them fixed throughout novel training. It is noted that the performance drops significantly due to the limited diversity of the generated base features, highlighting the importance of sampling features in each iteration. In the final row,

Table 6.7: The placement of data watchers to capture useful statistics for improved feature generation is investigated in relation to the frozen BN layers and the subsequent activations. The results are reported for the 10-shot setting on MS-COCO.

Data Watcher Config.	10-Shot Inference					
	AP	bAP	nAP	AR	bAR	nAR
(1) After Act.	32.0	36.7	18.0	29.9	32.7	21.5
(2) Before FBN	**32.2**	**36.9**	18.1	**30.1**	**33.0**	21.1
(3) Both	**32.2**	36.6	**19.1**	29.6	32.1	**22.3**

a random subset of the base classes $\mathcal{C}_s < |\mathcal{C}_b|$ with a size of 10 base classes, is sampled while still generating $N_{\text{feat}} = 10$ features for each class. Compared to generating features for all the base classes (row 3), a slight decrease in performance is observed, underscoring the significance of a class-balanced sampling scheme. Therefore, the decision is made to generate $N_{\text{feat}} = K$ features per class for all the base classes in each iteration to achieve the best overall AP.

Data Watchers Placement

To investigate the required statistics for capturing the base data distribution, an analysis is performed in Table 6.7. Different placements of data watchers are examined, and generators are trained accordingly. The placement options include: (1) after the activations following the frozen BN (FBN), (2) before the FBN layers, and (3) both locations. The results indicate that locations (2) and (3) demonstrate superior performance compared to location (1). Although locations (2) and (3) achieve the same AP, location (3) is chosen as it exhibits a slightly higher nAP.

Component Analysis

The results presented in Table 6.8 showcase the incremental introduction of our contributions to the DeFRCN model [Qia21]. In configuration 0, which represents the baseline, DeFRCN is trained without utilizing any base data. This leads to a substantial performance drop of 40.5% on the base classes compared to the overall performance. However, as we gradually introduce

Table 6.8: Analysis of the incremental contribution of different components on MS-COCO using the 10-shot setting.

	Model Configuration	10-Shot Inference					
		AP	bAP	nAP	AR	bAR	nAR
0	DeFRCN	30.6	34.6	18.6	29.1	32.0	20.5
A	DeFRCN (Base-Free)	18.2	18.5	17.4	17.5	16.2	21.3
B	+ Generator	30.7	35.0	17.8	28.6	31.3	20.9
C	+ CFA	32.0	36.4	18.5	29.6	32.4	21.3
D	+ DA	**32.2**	**36.8**	18.4	**29.9**	**32.6**	21.7
E	+ mEWC (NIFF)	**32.2**	36.6	**19.1**	29.6	32.1	**22.3**

our contributions, significant improvements are observed. In configuration B, where a standalone lightweight generator is incorporated, the performance is nearly recovered, achieving a performance level close to that of configuration 0 without relying on base data. By applying CFA in configuration C and further including pixel-level data augmentation in configuration D and parameter-level regularization (mEWC) in configuration E, our approach achieves state-of-the-art results in the overall performance. These findings highlight the effectiveness of our incremental contributions in enhancing the performance of the DeFRCN model.

Generator Architecture

To investigate the impact of various architectural design choices on the overall detection performance, experiments were conducted and presented in Table 6.9. The focus was on three factors: the number of layers (L), the kernel size, and the input noise dimension (z) of the generator. The effect of the number of layers on performance was examined. It is shown that increasing the number of layers beyond $L = 5$ resulted in improved overall performance, particularly in the base performance. However, performance started to decline when the number of layers exceeded $L = 7$.

Table 6.9: The influence of different architectural design choices on the generator performance is examined on MS-COCO with a 10-shot setting. The DeFRCN [Qia21] model is fine-tuned without base data using the generator without any regularization techniques.

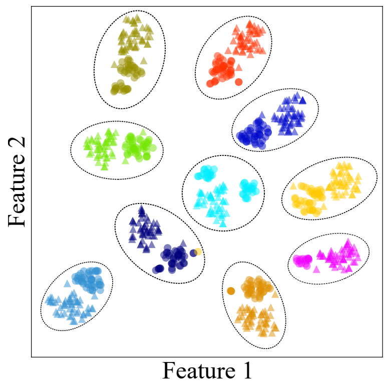
Model Configuration	10-Shot Inference			
	AP	bAP	nAP	AR
Number of Layers (L=3)	27.6	31.3	16.4	27.6
Number of Layers (L=5)	30.7	35.0	17.7	28.8
Number of Layers (L=7)	31.2	35.7	17.7	29.3
Number of Layers (L=10)	31.1	35.6	17.5	29.2
Kernel Size (kernel=1)	30.2	34.7	17.7	28.3
Kernel Size (kernel=3)	30.7	35.0	17.7	28.8
Kernel Size (kernel=5)	30.5	34.8	17.6	28.7
Kernel Size (kernel=7)	30.6	34.8	17.8	28.6
Noise Dimension (z=50)	30.5	34.9	17.6	28.5
Noise Dimension (z=100)	30.7	35.0	17.7	28.8
Noise Dimension (z=1000)	30.7	35.0	17.7	28.8

The impact of kernel size on performance was explored. Surprisingly, it was discovered that increasing the kernel size did not lead to any significant performance gain. Furthermore, the effect of the input noise dimension was investigated. It was observed that increasing the noise dimension beyond $z = 100$ did not result in any noticeable change in performance. This suggests that the generator is capable of generating diverse and high-quality features without requiring high-dimensional noise vectors.

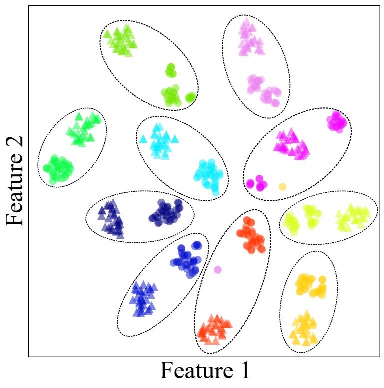
Overall, these experiments provide insights into the importance of architectural design choices in the generator and offer guidance for achieving high detection performance by optimizing the number of layers, kernel size, and input noise dimension.

Generator Training Analysis

A T-distributed Stochastic Neighbor Embedding (TSNE) visualization is presented in Figure 6.4a, depicting the distribution of real and generated instance-level features for 10 randomly selected MS-COCO base classes. The generated



(a) Class-wise statistics



(b) Class-agnostic statistics

Figure 6.4: A TSNE visualization of the real (●) and generated fake features (▲) via class-wise (a) and class-agnostic (b) statistics for 10 random classes.

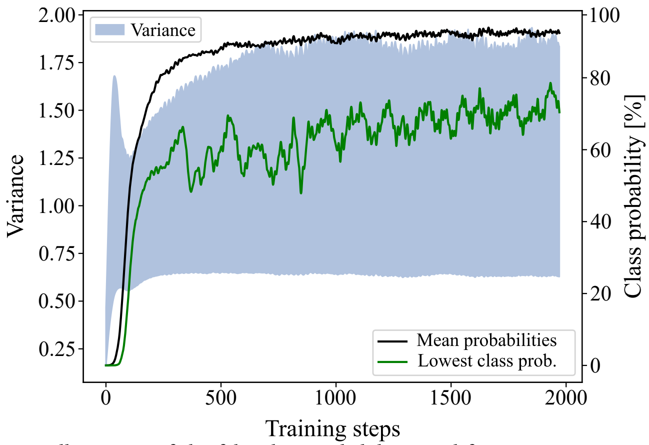


Figure 6.5: An illustration of the fake class probabilities and feature variance is provided by showcasing the lowest class probability and the mean probabilities across all base classes in MS-COCO.

features are produced by generating 30 features per class. The visualization demonstrates that the forged features are consistently located in close proximity to the real base features, with some instances of overlap. This confirms

Table 6.10: A comparison is made between the full FIM variant of EWC and the utilized mean FIM variant (mEWC) in the context of NIFF-DeFRCN on MS-COCO (10-shot). Additionally, an investigation is conducted to examine the effect of scaling the EWC/mEWC regularization term.

Configuration	10-Shot Inference			
	AP	bAP	nAP	AR
EWC ($\lambda_{\text{EWC}} = 1.0$)	33.0	38.1	17.6	30.3
EWC ($\lambda_{\text{EWC}} = 0.1$)	32.9	38.2	17.1	30.6
EWC ($\lambda_{\text{EWC}} = 0.01$)	32.2	37.7	15.9	29.8
EWC ($\lambda_{\text{EWC}} = 0.001$)	31.8	36.7	17.3	29.7
mEWC ($\lambda_{\text{EWC}} = 1.0$)	33.0	38.5	16.5	30.5
mEWC ($\lambda_{\text{EWC}} = 0.1$)	33.0	38.4	16.6	30.5
mEWC ($\lambda_{\text{EWC}} = 0.01$)	32.2	36.6	19.1	29.8
mEWC ($\lambda_{\text{EWC}} = 0.001$)	30.3	33.9	19.3	28.8

the ability of the feature generator to capture and represent the distribution of the base features.

Additionally, Table 6.4b showcases the features generated using class-agnostic statistics. In contrast to the features generated with class-wise statistics, the fake samples are observed to be further away from the real features.

Figure 6.5 presents an analysis of the quality and diversity of the generated features in terms of feature variance. The plot includes the mean class probability (black curve) and the lowest class probability (green curve) across all classes for the generated features. It is evident that the generator is capable of learning diverse features with a high variance, while still maintaining high class probabilities. The mean class probability is $\sim 95\%$, while the lowest class probability is $\sim 75\%$.

mEWC VS. EWC

Table 6.10 compares the performance of vanilla EWC [Kir16] with the full FIM and the proposed mEWC using a mean FIM per parameter. It also explores the impact of different scaling factors (λ_{EWC}) when applying the EWC/mEWC

Table 6.11: The impact of different finetuning loss components on MS-COCO (10-shot) is examined when finetuning DeFRCN [Qia21] without base data using the generator without any regularization techniques.

Model Configuration	10-Shot Inference			
	AP	bAP	nAP	AR
DeFRCN w/G.	30.7	35.0	17.7	28.8
w/o $\mathcal{L}_{\text{conf}}$	28.9	32.6	17.7	26.4
$\mathcal{L}_{\text{conf}}$ using KL	30.7	35.0	17.7	28.2
w/o Weighted feature terms	29.8	33.9	17.6	28.0
w/o L1 Reg. term	30.5	34.8	17.8	28.6

penalty term during novel training. The results reveal that EWC is more effective in maintaining the base performance across different scaling factors, albeit at the expense of the novel performance. On the other hand, reducing the scaling factor for mEWC leads to a decrease in base performance compared to EWC. However, mEWC achieves the same overall AP as EWC at $\lambda_{\text{EWC}} = 0.01$, with a lower bAP and a higher nAP. This setting is used consistently throughout the experiments. It is worth noting that the trade-off between bAP and nAP can be adjusted based on the specific application requirements. Additionally, it is observed that in mEWC, the AP improves with lower λ_{EWC}, but at the expense of lower bAP.

Novel Training Loss Components

The impact of various finetuning loss components is investigated in Table 6.11. In the first row, finetuning of DeFRCN with the proposed generator and the novel training loss $\mathcal{L}_{\text{N}}$ without any regularization is performed. Removing the cross-entropy confidence loss $\mathcal{L}_{\text{conf}}$ (row 2) results in a decrease of 2.4 points in the base performance, leading to a drop in overall AP and AR. This indicates that the confidence loss contributes to generating base features with higher probabilities at the final softmax layer. When the cross-entropy loss is replaced with KL divergence (row 3) between teacher and student logits, similar results are obtained with a slight decrease in AR. Removal of the weighted feature distillation terms (row 4) leads to a drop in both base and overall performance.

Table 6.12: The G-FSOD results for $K = 5,10,30$-shot settings on MS-COCO are reported for multiple runs using 10 different seeds.

Methods / Shots	5 shot			10 shot			30 shot		
	AP	bAP	nAP	AP	bAP	nAP	AP	bAP	nAP
TFA w/ fc[Wan20a]	25.6±0.5	31.8±0.5	6.9±0.7	26.2±0.5	32.0±0.5	9.1±0.5	28.4±0.3	33.8±0.3	12.0±0.4
TFA w/ cos[Wan20a]	25.9±0.6	32.3±0.6	7.0±0.7	26.6±0.5	32.4±0.6	9.1±0.5	28.7±0.4	34.2±0.4	12.1±0.4
CFA w/ fc [Gui22b]	29.1±0.3	36.2±0.3	7.7±0.6	29.9±0.3	36.7±0.2	9.6±0.6	30.8±0.2	36.6±0.2	13.6±0.3
CFA w/ cos [Gui22b]	29.3±0.2	36.0±0.2	9.2±0.5	30.2±0.2	36.6±0.1	11.2±0.5	31.1±0.1	36.6±0.1	14.8±0.2
DeFRCN[Qia21]	27.8±0.3	32.6±0.3	13.6±0.7	29.7±0.2	34.0±0.2	16.8±0.6	31.4±0.1	34.8±0.1	21.2±0.4
CFA-DeFRCN [Gui22b]	28.4±0.2	32.8±0.2	15.2±0.5	30.2±0.2	34.0±0.2	18.8±0.4	31.7±0.1	34.6±0.1	23.0±0.3
NIFF-DeFRCN	31.1±0.1	36.6±0.0	14.6±0.2	32.1±0.1	36.8±0.1	18.0±0.2	33.3±0.0	37.7±0.1	20.0±0.1

Table 6.13: The G-FSOD results (AP50) for $K = 1,2,3,5,10$-shot settings on PASCAL-VOC are reported for multiple runs using 30 different seeds.

Set	Methods	Shots				
		1	2	3	5	10
All Set 1	CFA w/ fc [Gui22b]	66.3±0.8	68.0±0.5	70.1±0.4	71.7±0.5	73.2±0.5
	CFA w/ cos [Gui22b]	66.5±0.9	69.2±0.6	71.1±0.6	72.5±0.4	73.4±0.4
	DeFRCN [Qia21]	67.8±1.4	71.3±0.8	72.6±0.5	73.6±0.5	74.1±0.5
	CFA-DeFRCN [Gui22b]	69.0±1.4	72.6±0.7	73.1±0.4	74.0±0.5	74.3±0.4
	NIFF-DeFRCN	71.2±0.8	74.2±0.4	75.4±0.4	76.3±0.3	76.7±0.3
All Set 2	CFA w/ fc [Gui22b]	64.9±0.9	66.4±0.7	68.3±0.5	69.6±0.3	70.8±0.5
	CFA w/ cos [Gui22b]	64.1±0.9	66.5±0.5	68.1±0.5	69.3±0.2	70.4±0.4
	DeFRCN [Qia21]	65.2±1.0	68.0±0.8	69.2±0.6	70.6±0.6	71.3±0.5
	CFA-DeFRCN [Gui22b]	66.4±1.0	69.0±0.8	70.4±0.7	71.3±0.7	72.1±0.4
	NIFF-DeFRCN	68.0±0.8	70.5±0.5	71.7±0.5	72.8±0.4	73.7±0.3
All Set 3	CFA w/ fc [Gui22b]	65.2±0.8	66.8±0.8	69.1±0.7	70.9±0.6	72.3±0.4
	CFA w/ cos [Gui22b]	64.9±1.2	67.5±1.0	69.7±0.8	71.6±0.5	72.7±0.3
	DeFRCN [Qia21]	66.9±2.0	70.6±0.8	71.2±0.6	72.9±0.5	73.5±0.3
	CFA-DeFRCN [Gui22b]	68.3±1.6	71.4±0.8	72.3±0.5	73.5±0.5	74.0±0.3
	NIFF-DeFRCN	70.7±0.7	73.7±0.5	74.7±0.4	75.5±0.3	76.3±0.2

Additionally, the removal of the proposed L1 regression distillation term (row 5) causes a slight decrease in base and overall performance. Based on these ablations, it is determined that novel training should be performed using the overall loss that includes the confidence and feature distillation loss terms.

Table 6.14: The G-FSOD novel results (nAP50) for $K = 1,2,3,5,10$-shot settings on PASCAL-VOC are reported for multiple runs using 30 different seeds.

Set	Methods	Shots				
		1	2	3	5	10
Novel Set 1	CFA w/ fc [Gui22b]	28.2±3.1	35.0±1.9	41.9±1.4	47.8±1.6	53.3±1.6
	CFA w/ cos [Gui22b]	30.9±3.9	40.9±2.5	47.8±2.4	53.1±1.4	56.1±1.4
	DeFRCN [Qia21]	43.8±4.3	57.5±2.5	61.4±1.7	65.3±0.9	67.0±1.4
	CFA-DeFRCN [Gui22b]	45.4±4.9	60.3±2.2	62.1±1.4	66.4±0.9	67.6±1.2
	NIFF-DeFRCN	46.0±3.0	57.2±1.7	62.0±1.4	65.5±1.1	67.2±1.1
Novel Set 2	CFA w/ fc [Gui22b]	20.0±3.5	26.4±2.9	32.8±2.2	37.3±1.7	41.8±1.9
	CFA w/ cos [Gui22b]	21.0±3.5	29.0±2.3	34.6±2.3	38.9±1.2	43.0±1.9
	DeFRCN[Qia21]	31.5±3.6	40.9±2.2	45.6±2.0	50.1±1.4	52.9±1.1
	CFA-DeFRCN [Gui22b]	32.9±3.7	42.3±2.2	47.1±1.9	51.2±1.4	55.3±1.3
	NIFF-DeFRCN	30.1±3.0	39.6±1.8	45.0±1.9	49.4±1.6	52.8±1.3
Novel Set 3	CFA w/ fc [Gui22b]	20.3±3.4	26.4±3.1	34.3±2.5	41.2±2.4	46.5±1.6
	CFA w/ cos [Gui22b]	21.5±4.7	30.4±4.1	38.4±2.8	45.5±2.1	49.9±1.0
	DeFRCN [Qia21]	38.2±6.8	50.9±2.8	54.1±2.2	59.2±1.2	61.9±1.3
	CFA-DeFRCN [Gui22b]	41.4±5.8	52.9±3.0	56.1±1.7	60.3±1.1	62.9±0.9
	NIFF-DeFRCN	41.1±2.6	52.5±1.8	56.4±1.5	59.7±1.2	62.1±1.0

Multiple Runs

The performance robustness of NIFF-DeFRCN is investigated by running multiple experiments using different seeds. The results for the MS-COCO dataset are presented in Table 6.12. Despite the absence of base data, NIFF-DeFRCN consistently achieves higher AP and bAP scores across all shot settings compared to the baselines [Wan20a, Qia21, Gui22b].

Similarly, for the PASCAL-VOC dataset, the AP50 results in Table 6.13 and nAP50 results in Table 6.14 demonstrate the consistent higher performance of NIFF-DeFRCN compared to the baselines, while also delivering competitive results on nAP50 for various shot settings.

Model Complexity Analysis

The memory requirements for the 10-shot MS-COCO configuration are as follows: The model requires 195.1 MB, the base images require 148.8 MB, and the novel images require 48.6 MB. On the other hand, the generator occupies 3.7 MB, while the base statistics occupy 12.42 MB. Therefore, the proposed model achieves a reduction of 33.8% in memory requirements compared to the initial setup.

In terms of computation, DeFRCN [Qia21] and the generator require 133.46 G and 943.94 K FLOPS, respectively, indicating that the computational overhead is minimal.

The novel training time for DeFRCN 104.5 minutes, while the generator training and data generation require an additional 112 minutes and 27 minutes, respectively.

6.4 Qualitative Results

Figure 6.6 showcases different qualitative results for the MS-COCO dataset in the 10-shot setting. The first column displays images containing only base classes, indicated by green boxes, while the second column shows images with only novel classes, represented by blue boxes. The third column presents images that contain both base and novel classes. These three scenarios are presented to assess the performance of the NIFF approach across different cases. Additionally, the last two columns highlight various failure cases. As previously mentioned, four Nvidia GeForce 1080Ti GPUs were utilized.

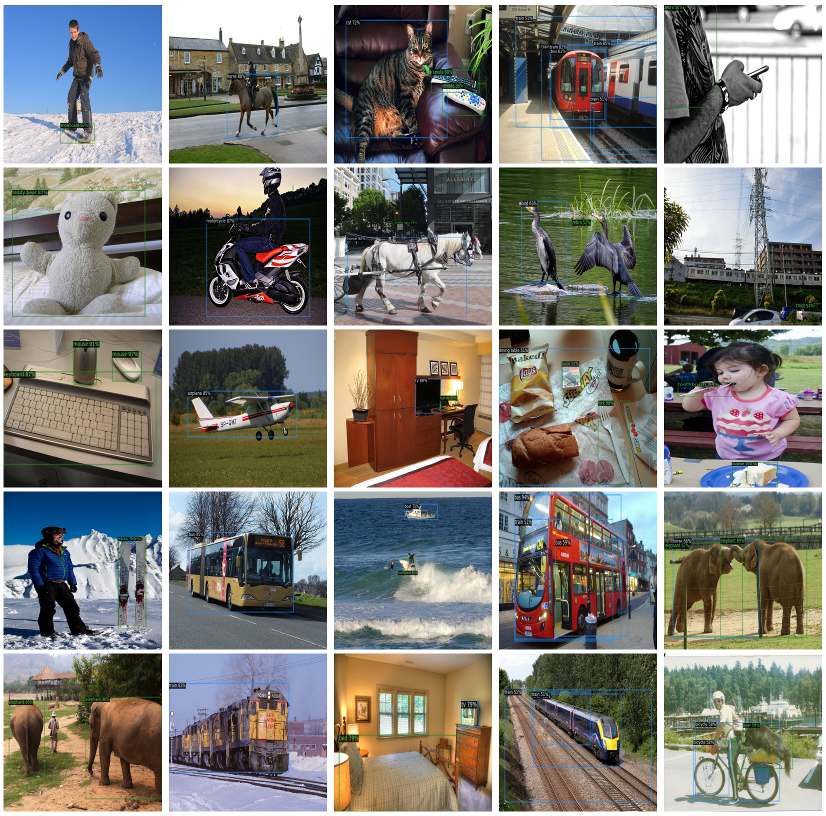


Figure 6.6: The qualitative results of the proposed NIFF method (NIFF-DeFRCN) on the MS-COCO (10-shot) dataset are presented. The first three columns show successful scenarios, where green bounding boxes represent base classes and blue bounding boxes represent novel classes. The last two columns display failure scenarios.

6.5 Discussion

This chapter has presented a new approach to address the limitations of existing G-FSOD methods, which rely on storing and replaying base data. Unlike previous replay-based methods [Wan20a, Qia21, Fan21, Gui22b], the proposed framework, NIFF, is a base-data-free G-FSOD method. NIFF offers several advantages by eliminating the need for storing and replaying base images.

Firstly, NIFF respects privacy constraints by not requiring the storage of sensitive base images. This is particularly beneficial in scenarios where privacy and data protection are paramount, such as in sensitive domains or applications involving personal data.

Secondly, NIFF significantly reduces the memory footprint associated with G-FSOD. Instead of storing and using base images, a standalone generator that forges base instance-level features is introduced. The generator has a negligible memory footprint of approximately 4MB, which is two orders of magnitude lower than the memory required for storing and finetuning base images.

The main contribution lies in the proposed two-stage DFKD paradigm that leverages a tailored standalone feature generator. Specifically, during generator training, the generator aligns class-wise statistics in the RoI-head to forge base instance-level features. By leveraging the proposed base feature generator and knowledge distillation approaches, NIFF surpasses replay-based methods without needing base data.

In summary, NIFF represents a promising step towards more efficient and privacy-conscious G-FSOD methods. By eliminating the reliance on base images, NIFF offers a scalable and memory-efficient solution for continual object detection, making it suitable for real-world applications with privacy considerations and limited computational resources.

7 Towards Efficient Dense Meta Detectors

This chapter seeks to design more embedded-friendly meta-detectors without significantly sacrificing the detection performance. Following that, a thorough investigation is conducted on the commonly used sparse meta-detector, Attention-RPN [Fan20], and two evaluation metrics are proposed to assess the knowledge transfer capability of meta-based detectors for new tasks. The chapter then introduces our dense meta-detector, FSRN [Gui23a], which addresses the observed limitations and provides a more efficient and embedded-friendly meta-detector solution. Lastly, extensive experiments and ablation studies are performed to evaluate the performance of the proposed framework.

7.1 Analyzing Sparse Meta-Detectors

To determine the causes behind the performance gap between one-stage (dense) and two-stage (sparse) meta-detectors, we begin this section by examining a meta sparse detector called Attention-RPN [Fan20]. The adoption of this model is attributed to the fact that the fusion of support and query takes place early on before the RPN, allowing us to assess the RPN component independently as a dense meta-detector.

While the discriminability of the detector is evaluated by computing the average precision on the base classes (referred to as bAP), this alone does not sufficiently pinpoint the bottlenecks that impede successful knowledge transfer from base to novel classes. To effectively assess the transferability of knowledge from base to novel classes, two straightforward metrics are introduced.

Table 7.1: The performance of the RPN in the Attention-RPN [Fan20] as a stand-alone dense meta-detector is analyzed using the proposed evaluation protocol. The 10-shot detection performance on the MS-COCO dataset [Lin14] is evaluated.

Method	Base Performance				Novel Performance				Transferability			
	bAP	bAP50	bAP75	bAR	nAP	nAP50	nAP75	nAR	PT	PT50	PT75	RT
MetaYOLO [Kan18]	13.8	-	-	15.5	5.6	-	-	14.4	0.40	-	-	0.93
Attention-RPN (RPN-only) [Fan20]	5.54	13.35	3.65	21.23	0.98	3.40	0.31	11.84	0.18	0.25	0.08	0.55
Attention-RPN [Fan20]	24.26	38.04	26.44	40.56	11.95	22.37	11.79	30.84	0.49	0.59	0.45	0.76

First, the Precision Transferability (PT) metric, to assess the transferability of precision as follows:

$$PT = \frac{\text{nAP}}{\text{bAP}}. \tag{7.1}$$

Similarly, to examine the transferability of recall, the Recall Transferability (RT) is proposed:

$$RT = \frac{\text{nAR}}{\text{bAR}}. \tag{7.2}$$

A meta-detector would exhibit PT and RT values of 1 in an optimal scenario, indicating perfect transferability. Meaning that the detection model has successfully acquired valuable base knowledge, which is powerful enough to learn novel classes from limited data rapidly. Conversely, lower ratios suggest overfitting to the base classes.

Initiating the analysis, one can perceive the RPN within the two-stage (sparse) meta-detector, Attention-RPN [Fan20], as a standalone one-stage (dense) meta-detector. This characterization arises from the early support and query feature fusion before the RPN module. Therefore, the transferability metrics above are reported on both the Attention-RPN and the RPN module alone as an independent dense meta-detector. The findings are presented in Table 7.1, including MetaYOLO [Kan18] as an additional dense meta-detector. The results indicate that the Attention-RPN component of the sparse Attention-RPN meta-detector performs poorly, similar to the dense Meta-YOLO [Kan18]. This highlights a notable disparity between the RPN and the final detection head of Attention-RPN. The outcomes demonstrate that the RPN of the two-stage Attention-RPN and the one-stage MetaYOLO exhibit inadequate performance.

This discrepancy is particularly evident in the base classes, suggesting low discriminability. Notably, the transferability scores of MetaYOLO and the two-stage Attention-RPN are similar (0.40 vs. 0.49), as shown in Table 7.1. However, the transferability of the RPN in Attention-RPN has decreased by half, which we attribute primarily to the low discriminability of the dense RPN.

On the other hand, the low discriminability of dense detectors can be attributed to multiple factors. Firstly, the absence of an instance-level proposal network such as RPN in one-stage FSODs restricts the receptive field and limits the post-fusion learning capacity. Secondly, the learning signal in dense detectors is weaker than in sparse detectors, as most anchors are classified as background due to the query-set construction strategy, which considers only a single class per image in each task.

7.2 The Few-Shot RetinaNet Framework

In this section, our approach Few-Shot RetinaNet (FSRN) [Gui23a] is introduced to tackle the limitations above. FSRN comprises five main components:

- Multi-Scale Fusion (MSF) module which enables a wide receptive field covering the entire anchor area.
- Multi-Way Support Training (MWST) strategy aimed at increasing the number of foreground samples to enhance the learning signal.
- Multi-Scale Data Augmentation (MSDA) strategy applied to both query and support images during meta-testing, enhancing the diversity of the data distribution.
- Gaussian Prototyping (GP) used during meta-testing to compute representative class prototypes by utilizing the mean and standard deviation of the support features. This approach enables a more accurate determination of class prototypes.
- Adaptive Pooling (AdaPool) to adaptively calibrate the support features maps to compute a more representative support class prototype.

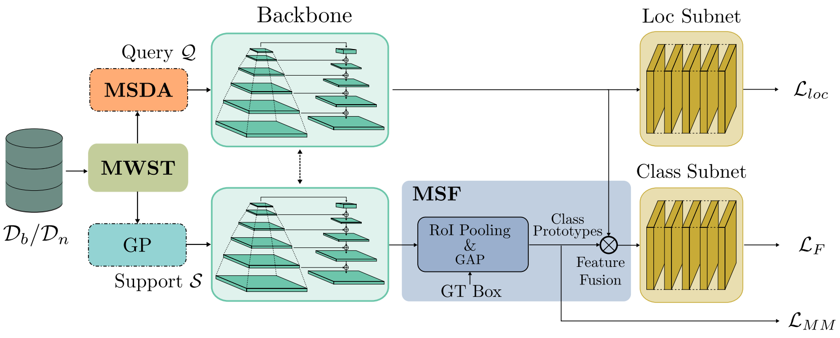


Figure 7.1: An overview of the proposed FSRN architecture. The MWST generates multi-way tasks with multiple positive and negative classes per training episode, increasing foreground anchor sampling and improving discriminability. The MSF module, on top of the FPN, enables a wide receptive field that covers the entire anchor area after fusion. During meta-testing, the introduced MSDA scheme enriches the scale-space, enhancing discriminability for novel classes. Improved class prototypes are achieved through the proposed GP technique.

7.2.1 Architecture Overview

The proposed FSRN model, illustrated in Figure 7.1, extends the well-known one-stage RetinaNet [Lin18] architecture to operate as a dense meta-detector. FSRN consists of two branches: one branch handles the query images, while the other handles the support images. Both branches share a backbone comprising a ResNet-50 with an FPN. In the support branch, the backbone is followed by a RoI-pooling operation to extract relevant feature maps from the support images. Global Average Pooling (GAP) is then applied, averaging across the shots dimension to obtain class prototypes. Subsequently, the MSF module combines the query features and class prototypes before passing them through the classification subnet, while the localization subnet only utilizes the query features.

In the context of FSOD, we observe that relying solely on the focal loss ($\mathcal{L}_F$) as the training objective similar to RetinaNet [Lin18] is insufficient for the backbone to acquire robust disentangled representations for the novel categories. To improve discriminability and ensure stable training, we leverage a max-margin loss inspired by [Li21]. The objective of this loss is to minimize the

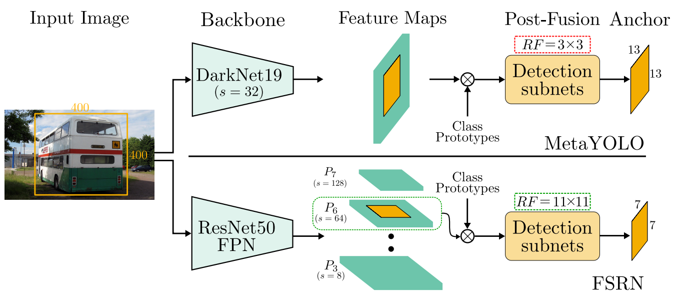


Figure 7.2: The impact of the post-fusion network Receptive Field (RF) is depicted. To demonstrate, an example query image from the MS-COCO dataset with an annotated bounding box of size 400 × 400 is used. The upper part shows that a YOLOv2-based dense meta-detector [Kan18] is affected by a narrow receptive field that is unable to cover the entire anchor area (i.e., $RF = 3 \times 3 < 13 \times 13$). On the other hand, the whole anchor area is processed by the proposed FSRN, which utilizes the introduced MSF and a deeper post-fusion network (i.e., $RF = 11 \times 11 > 7 \times 7$).

intra-class variance while maximizing the inter-class variance. Mathematically,

$$\mathcal{L}_{MM} = \frac{\sum_{\mathrm{c}}^{\mathrm{C}} \frac{1}{K} \sum_{k}^{K} ||\boldsymbol{v}_{\mathrm{c}k} - \boldsymbol{\mu}_c||_2^2}{\sum_{\mathrm{c}}^{\mathrm{C}} \min_{j, j\neq \mathrm{c}} ||\boldsymbol{\mu}_{\mathrm{c}} - \boldsymbol{\mu}_j||_2^2}, \tag{7.3}$$

where $\boldsymbol{v}_{\mathrm{c}j}$ represents the k^{th} prototype vector for class c, and K corresponds to the total number of prototype vectors. $\boldsymbol{\mu}_i$ denotes the mean prototype for class c, while C represents the total number of classes. The overall training objective function can be expressed as follows:

$$\mathcal{L} = \mathcal{L}_F + \mathcal{L}_{loc} + \lambda \mathcal{L}_{MM}, \tag{7.4}$$

where is the original focal loss [Lin18].$\mathcal{L}_{loc}$ is the smooth $L1$-loss for the bounding box regression task [Ren15]. To balance the impact of the max-margin loss with respect to the classification and regression losses, a scaling factor λ is incorporated.

7.2.2 Multi-Scale Feature Fusion

Previous experiments in Table 7.1 have indicated that the limited discriminability can be attributed, in part, to the absence of a deep post-fusion network prior to the final detection layers. During meta-learning, the fusion process filters the global-level *class-agnostic* features from the backbone and generates *class-specific* features. When the support and query features are directly aggregated before reaching the detector, this class-specific information from the support branch is injected. However, the downstream layers struggle to effectively utilize this information due to their small receptive field and limited learning capacity. In the case of the two-stage Attention-RPN [Fan20], the loss of spatial information is mitigated by the presence of a RoI head, which provides a sufficiently large receptive field to learn instance-level features after fusion. This implies that the post-fusion receptive field should cover at least the area occupied by the largest anchor size. In dense detectors, a simple solution is to increase the receptive field by adding multiple layers [Luo16] between the fusion location and the detection head. However, incorporating a significant number of layers becomes inefficient for one-stage detectors to adequately cover the largest anchor size.

To address this concern, we propose the MSF module on top of the FPN. The FPN inherently restricts the size of the largest anchor to 10×10 pixels, ensuring that it can be adequately covered by the downstream subnet, as depicted in Figure 7.2. The fusion process occurs immediately after the FPN, where support-level features are pooled from the corresponding FPN level $\boldsymbol{p}_l$, depending on the size of the ground truth bounding box. Following spatial global averaging of the extracted features from each support shot, the class prototype is computed by averaging across the K support shots. Mathematically, the class prototype $\boldsymbol{\mu}_{\mathrm{c}}$ is calculated as:

$$\boldsymbol{\mu}_{\mathrm{c}} = \frac{1}{K} \sum_{k=1}^{K} GAP(\boldsymbol{v}_{\mathrm{c}k}^{p_l}), \tag{7.5}$$

where $\boldsymbol{v}_{ck}^{p_l}$ is the the support feature of class c from shot k and the corresponding level p_l. Finally, each class prototype attends the multi-level query features

$\boldsymbol{f}_Q^{p_l}$ through a Hadamard product operation to yield output features $\boldsymbol{f}_o^{p_l}$ for each corresponding pyramid level p_l as follows:

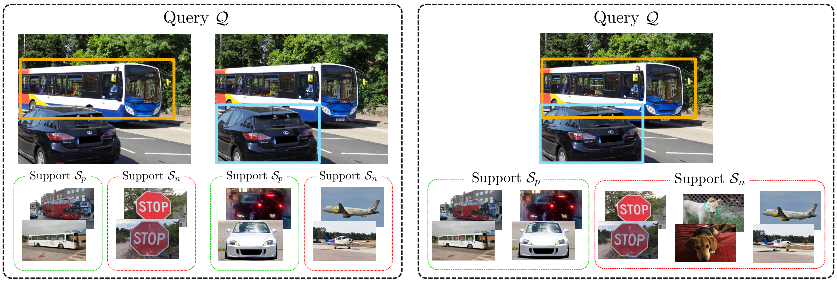
$$\boldsymbol{f}_o^{p_l} = \boldsymbol{\mu}_{\mathrm{c}} \odot \boldsymbol{f}_Q^{p_l}. \tag{7.6}$$

The support and query features are fused only before the classification subnets, while the extracted query features are directly fed to localization subnets without fusion. This separation is due to the differing nature of classification and localization tasks. In more detail, the reason for this separation and distinction lies in the inherent nature of the classification and localization tasks [Wu20b]. Classification necessitates the use of features that are tailored to individual classes or categories. In contrast, the localization task is inherently class-agnostic, which means it is concerned with pinpointing the position or extent of objects without being tied to the specific types of objects.

The choice of the fusion location allows for a deeper post-fusion network, which helps the backbone focus on global-level feature learning while the subnets post-fusion learn the instance-level features. To foster the learning signal of the detection subnets, we increase the number of positive anchors per query image by increasing the number of anchors per feature pixel from 9 in the original RetinaNet [Lin18] to 15.

7.2.3 Multi-Way Support Training Strategy

In meta-based detection, the query-support set construction strategy is to usually sample all annotations in a query image belonging to single class c along with K support shots of the same class [Kan18, Fan20, Xia20], as shown in Figure 7.3. This, in turn, limits each task per episode to a single-class detection. While the said strategy is suitable for image classification, object detection is a more challenging setting, where multiple class instances are present per query image. The binary query-support selection strategy leads to fewer foreground samples and, consequently, fewer positive anchors and fewer positive gradients available during the training. This aggravates an already existing problem in dense detectors, namely the overwhelming number of



(a) Vanilla contrastive training-set construction (b) Multi-way contrastive training-set construction

Figure 7.3: Left: An illustration of the query-support set in a contrastive-based setup [Fan20]. In this approach, for each query image, only one annotation is sampled. Then, K support shots are selected for the same class, and K shots are randomly sampled from negative classes. **Right:** The MWST algorithm is depicted, where a query image with multiple annotations is processed at once with multi-way support examples.

generated static anchors which contain background samples. Although the focal loss addresses the foreground-background imbalance problem, it does not entirely alleviate the issue for meta-detectors.

As a remedy, MWST is introduced. Specifically, it involves loading the query image with all its annotations for each task. Then, random class dropout is performed, which means that when a class is dropped, all corresponding annotations in the query image are removed. Following that, we sample K support shots for each class. To limit the total number of support images required and the associated computational cost, the number of classes is restricted to N per query image. If the number of classes after dropout is smaller than N, we sample negative classes in the support set $\mathcal{S}$. The proposed query-set construction algorithm, outlined in Algorithm 1, enables multi-class contrastive training, resulting in an increase in the number of foreground objects to $\bar{m}/2$ compared to $\bar{m}/\bar{c}$ in binary meta-detection. Here, $\bar{m}$ represents the average number of annotations per query image, and $\bar{c}$ denotes the average number of classes per query image. The difference between the naive contrastive query-set construction and our proposed MWST is illustrated in Figure 7.3. Additionally, the class dropout acts as a data augmentation technique, simulating the random task sampling of a generic meta-learning paradigm and increasing the cardinality of the query set from $\bar{m} \times |\mathcal{D}_b|$ to $2^{\bar{m}} \times |\mathcal{D}_b|$. The task where all classes are dropped is disregarded.

Algorithm 7.2.1 MWST algorithm.

```
Input: Query image $\boldsymbol{Q}_i$ and associated labels $\boldsymbol{y}_i$, Support set $\mathcal{S}$, Set of classes
    $\mathcal{C}_i$ with instances in $\boldsymbol{Q}_i$, C number of classes per task
Output: Multi-way support set $\mathcal{S}_i$ for a query image $\boldsymbol{Q}_i$
1.  initialize $\mathcal{S}_i$ as empty list
2.  randomly drop classes from $\boldsymbol{y}_i$
3.  for every class c in $\boldsymbol{y}_i$
4.      sample different $K$-shots from $\mathcal{S}^c$   ▷ $S^c$ is the support set of class c.
5.      add to $\mathcal{S}_i$
6.  while $\mid \mathcal{S}_i \mid < C$
7.          randomly select class $z$ from $\mathcal{C}_b \setminus \mathcal{C}_i$
8.          sample different $K$-shots from $\mathcal{S}^z$
9.          add to $\mathcal{S}_i$
10. return $\boldsymbol{Q}_i, \boldsymbol{y}_i, \mathcal{S}_i$
```

7.2.4 Multi-Scale Data Augmentation

In [Wu20a], it was observed that the limited amount of novel data during the meta-test phase could lead to a sparse scale space, which may deviate from the learned base distribution. To tackle the issue of scale variation, Wu et al. proposed the Multi-Scale Positive Samples Refinement (MPSR) approach, which leverages the FPN. They introduced an auxiliary branch that generates object pyramids of different scales and refines the predictions accordingly. However, this approach comes with computational, time, and memory costs. Inspired by MPSR, we propose a MSDA module to approximate the multi-positive sample refinement approach during the meta-testing phase. In our approach, we approximate the refinement scheme by applying size jittering to both the query and support images using a logarithmic-based scaling, ensuring equal coverage of all FPN levels. Additionally, we assign higher weights to foreground samples using the parameter α in the focal loss. Empirically, we increase α to $\frac{\alpha+1}{2}$, where $\alpha < 1$. For instance, if $\alpha = 0.5$ during meta-training, we set $\alpha = 0.75$ during the meta-testing phase. This allows for achieving comparable performance without incurring any computational overhead.

7.2.5 Gaussian Prototyping

In addition, we propose a data augmentation scheme GP specifically for the support features during meta-testing. We observed that a simple average of the features from the K support shots does not accurately represent the true distribution of the class prototypes, leading to less diverse prototypes than those obtained during meta-training. Furthermore, there may be significant variance between the K support shots, which may not represent the true class distribution and true class prototypes. To tackle this challenge, we make an assumption that the support feature representation follows a class-conditional Gaussian distribution. To simulate this distribution, we compute the mean feature vector $\bar{\boldsymbol{f}}$ across the K support shots and calculate their corresponding standard deviation, denoted as $\boldsymbol{\sigma}_{\boldsymbol{f}}$. Subsequently, we sample a latent vector $\boldsymbol{z}$ from the Gaussian distribution $\mathcal{N}(\bar{\boldsymbol{f}}, \boldsymbol{\sigma}_{\boldsymbol{f}}^2)$, which becomes the class prototype $\boldsymbol{\mu}_{\mathrm{c}}$. This augmentation strategy aims to prevent overfitting on the novel support data by introducing diversity in the class prototypes and accounting for the inherent variance in the support shots.

7.2.6 Support Class Prototyping

The support feature prototyping technique plays a crucial role in ensuring the robustness of a meta-detector. Its objective is to generate a prototype feature vector that encompasses the discriminative information of a given class [Köh23]. Most meta-detectors rely on simple averaging using GAP across different instances, as shown in Equation 7.5. This means that all support instances are given equal importance, disregarding the possibility of outliers, such as occluded object instances that could significantly deviate the class prototype from its true distribution.

This emphasizes the significance of the class prototyping technique, as it needs to accurately capture representative features and distinctive class characteristics, enhancing discriminability. To this end, different techniques for prototyping have been explored. These techniques involve assigning weights to each support feature instance $\boldsymbol{v}_{ck}$ of class c from the k^{th} shot in order to compute the class prototype $\boldsymbol{\mu}_{\text{c}}$.

Cross-Correlation

Cross-correlation enables measuring the similarity between support and query features, revealing shared patterns and correlations. Additionally, cross-correlation can serve as an indicator of the semantic information contained within the support instances. By calculating these correlations, each support instance can be assigned a weight that reflects its importance. The intuition is that a higher correlation indicates greater importance and semantic richness of the support instance. Subsequently, the class prototype is computed through a weighted average, considering the significance of each support instance. Formally, the cross-correlation map is computed as follows:

$$\boldsymbol{f}_{\text{c}k}^{\text{Corr}}(x,y) = \sum_{i,j,c} \boldsymbol{f}_Q(x+i-1, y+j-1, c) \times \boldsymbol{v}_{\text{c}k}(i,j,c), \tag{7.7}$$

where $\boldsymbol{f}_Q$ denotes the query feature map. (i,j) are indices of the spatial location for the pixel of interest in the feature map. c is the index across the channel dimension. Next, a similarity measure $\text{s}_{\text{c}k}$ is obtained by applying GAP:

$$\text{s}_{\text{c}k} = GAP(\boldsymbol{f}_{\text{c}k}^{\text{Corr}}). \tag{7.8}$$

The similarity measure for each support instance is concatenated in a list $\mathbf{s_c} = [\text{s}_{\text{c}1}, \cdots, \text{s}_{\text{c}K}]$. To compute the support weight $\text{w}_{\text{c}k}$ of the k^{th} support instance for class c, softmax is applied:

$$\text{w}_{\text{c}k} = \frac{\exp(\text{s}_{\text{c}k})}{\sum_{k=1}^{K} \exp(\text{s}_{\text{c}k})}. \tag{7.9}$$

Finally, the class prototype is computed as follows:

$$\boldsymbol{\mu}_{\text{c}} = \sum_{k=1}^{K} \frac{\text{w}_{\text{c}k} \cdot \boldsymbol{v}_{\text{c}k}}{\text{w}_{\text{c}k}}. \tag{7.10}$$

Squeeze-and-Excitation

To focus on discriminative features and suppress less relevant ones, Squeeze-and-Excitation (SE) [Hu18] blocks seek to model channel-wise dependencies and adaptively weight the feature maps with minimal computational overhead, making them highly efficient in practice.

The SE block consists of two primary operations: squeeze and excitation. In the squeeze operation, global information is captured by applying average pooling to the input feature maps, reducing the spatial dimensions while preserving the channel-wise information. Next, the excitation operation takes place to model the relationships among the channels. This is realized through two fully connected layers with non-linear activations, resulting in channel-wise weights that signify the importance of each channel. By multiplying these learned channel-wise weights with the original feature maps, the excitation operation selectively strengthens or reduces specific channels. This adaptive recalibration of feature maps enables the network to prioritize discriminative and informative features while attenuating less relevant or noisy ones.

We utilize the SE blocks to assign weights to the support features, determining their importance for each class. In the squeeze operation, a comprehensive representation of the query feature map is obtained via GAP, capturing its global description. The excitation operation then employs the SE block to recalibrate the support features based on the global information from the query. By reweighting the support features, the SE block ensures that more significant support shots are given higher importance. Finally, to compute the class prototype, the reweighted support feature maps are averaged across the shot dimension, providing a representative prototype for the class of interest.

Cosine Similarity

Cosine similarity is a widely used metric for evaluating the similarity or dissimilarity between two vectors in a multi-dimensional space. It finds applications in diverse fields like computer vision and natural language processing. The fundamental idea behind cosine similarity lies in the geometric understanding of vectors. In vector space, the angle between two vectors can offer valuable information about their similarity. A small angle suggests higher similarity, while a large angle indicates dissimilarity. The cosine similarity precisely captures this concept by measuring the cosine of the angle between the vectors.

The cosine similarity provides two significant benefits. Firstly, it is scale-invariant, where the magnitude of the vectors does not impact the similarity calculation as it solely focuses on the direction of the vectors. Secondly, cosine similarity computation involves simple mathematical operations such as dot product and vector norms, making it computationally efficient and highly scalable. Formally, the cosine similarity between two vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ is computed as follows:

$$\frac{\boldsymbol{v}_1^T \boldsymbol{v}_2}{\| \boldsymbol{v}_1 \| \cdot \| \boldsymbol{v}_2 \|}. \tag{7.11}$$

Based on the motivation above, we utilize cosine similarity to measure the similarity between the query and support features. A higher similarity value indicates a more substantial weight assigned to the corresponding support instance. Similar to cross-correlation prototyping, we further apply a softmax function to the similarity values of the support instances belonging to each class. This softmax operation normalizes the weights, ensuring they sum up to 1. Finally, the class prototype is computed by taking the weighted average of the support instances based on their normalized weights.

Adaptive Pooling

AdaPool [Liu18a] aims to address the presence of noisy samples within a set of K support shots by assigning lower weights to those particular samples. Unlike the previously discussed prototyping methods that determine weights based on the similarity score between query and support shots, the adaptive pooling module calculates weights by comparing the similarity score between the mean prototype and the support shots. Initially, it computes the mean prototype by naively averaging all the available K-shots. Subsequently, the computed class prototype is projected into a different dimensional space using a MLP. The similarity scores of the support shots are then computed in relation to this averaged or mean prototype after the projection. The weights are obtained by subjecting these similarity scores to a softmax layer, ensuring they are appropriately normalized. Mathematically, the similarity scores for a class c is computed as:

$$\mathbf{s}_{\mathrm{c}} = \boldsymbol{v}_{\mathrm{c}} \odot \mathrm{MLP}\left(\frac{1}{K}\sum_{k=1}^{K} GAP(\boldsymbol{v}_{\mathrm{c}k})\right). \tag{7.12}$$

Similar to the cross-correlation prototyping, the weight of the k^{th} support instance for class c is denoted by:

$$\mathrm{w}_{\mathrm{c}k} = \frac{\exp(\mathrm{s}_{\mathrm{c}k})}{\sum_{k=1}^{K}\exp(\mathrm{s}_{\mathrm{c}k})}. \tag{7.13}$$

Finally, the class prototype is calculated in the following manner:

$$\boldsymbol{\mu}_{\mathrm{c}} = \sum_{k=1}^{K} \frac{\mathrm{w}_{\mathrm{c}k} \cdot \boldsymbol{v}_{\mathrm{c}k}}{\mathrm{w}_{\mathrm{c}k}}. \tag{7.14}$$

Note that AdaPool can be viewed as an extension of average pooling, but with the added capability of being learnable. This allows aggregating information from multiple shots while effectively suppressing the noise present in the data [Liu18a].

7.3 Experimental Evaluations

For evaluating the proposed FSRN framework, we adopt the widely-used FSOD benchmarks, as established by previous works [Kan18, Wan20a, Wu20a]. These benchmarks involve conducting experiments on the MS-COCO and PASCAL-VOC datasets. We utilize the same classes and data splits used in the works mentioned above to ensure fair comparisons.

7.3.1 Implementation Details

We utilize a ResNet-50 [He16] as the backbone architecture and incorporate a FPN [Lin17] for our model. During the meta-training phase, we follow the standard training of RetinaNet [Lin18]. The model is trained using SGD for 90k iterations, with a batch size of 16 and a learning rate of 0.01. The learning rate is decayed twice, first at 50k iterations and then at 80k iterations, by a factor of 10. We apply a weight decay of 0.0001 and a momentum of 0.9.

For meta-training, we adopt a 5-way-5-shot setting, where each task consists of 5 classes and 5 support shots per class. This is implemented using the MWST algorithm, with N set to 5. Thus, there are a total of 25 support shots per task. The only data augmentation technique applied during meta-training is horizontal image flipping for the query image.

During meta-testing, we perform 6k iterations with a learning rate of 0.005, which is decayed by a factor of 10 at iteration 4k. To leverage the MWST algorithm, we set N to 15. All experiments are conducted using four Nvidia Tesla V100 GPUs.

7.3.2 Main Results

Results on MS-COCO

The results of the proposed approach on MS-COCO are presented in Table 7.2. The table is divided into two sections, starting with the one-stage FSOD methods such as [Kan18, Wan19c, Per20], followed by the two-stage based approaches including [Che18a, Wan19c, Yan19, Wan20a, Wu20a, Xia20, Fan20, Sun21, Li21, Qia21]. When compared to meta-detectors, not only is FSRN found to outperform dense meta-detectors by a significant margin, but it also achieves better performance than many sparse meta-detectors such as [Yan19, Xia20, Fan20] and is comparable to [Li21].

Table 7.2: The few-shot detection results on the MS-COCO dataset are reported for the 20 novel PASCAL-VOC classes with $K = 5, 10, 30$ shots. The original paper did not provide results for the cases denoted by '-' in the report. The upper section of the table represents the dense meta-detectors only.

Method	5-Shot				10-Shot				30-Shot			
	AP	AP50	AP75	AR	AP	AP50	AP75	AR	AP	AP50	AP75	AR
MetaYOLO [Kan18]	-	-	-	-	5.6	12.3	4.6	14.4	11.3	21.7	8.1	19.2
MetaDet-YOLO [Wan19c]	-	-	-	-	7.1	14.6	6.1	15.5	9.1	19.0	7.6	17.8
ONCE [Per20]	-	-	-	-	5.1	-	-	9.5	-	-	-	-
FSRN	**9.6**	**18.1**	**9.2**	**28.3**	**17.4**	**29.6**	**17.9**	**37.0**	**19.9**	**34.2**	**44.4**	**40.7**
MetaDet [Wan19c]	-	-	-	-	7.1	14.6	6.1	15.5	11.3	21.7	8.1	19.2
Meta-RCNN [Yan19]	-	-	-	-	8.7	19.1	6.6	17.9	12.4	25.3	10.8	21.7
TFA w/fc [Wan20a]	8.4	-	-	-	10.0	17.3	8.5	-	13.4	22.2	11.8	-
TFA w/cos [Wan20a]	8.3	13.3	6.5	-	10.0	17.1	8.8	-	13.7	22.0	12.0	-
MPSR [Wu20a]	-	-	-	-	9.8	17.9	9.7	21.2	14.1	25.4	14.2	24.3
FsDetView [Xia20]	-	-	-	-	12.5	27.3	9.8	25.5	14.7	30.6	12.2	28.4
FSOD-RPN [Fan20]	-	-	-	-	12.0	22.4	11.8	30.8	-	-	-	-
FSCE [Sun21]	-	-	-	-	11.1	-	9.8	-	15.3	-	14.2	-
CME [Li21]	-	-	-	-	15.1	24.6	16.4	-	16.9	28.0	17.8	-
DeFRCN [Qia21]	16.1	-	-	-	18.5	-	-	-	22.6	-	-	-

Table 7.3: The novel detection performance (nAP50) for the five novel categories is reported in the evaluation of few-shot object detection on PASCAL-VOC. The results are presented for all three splits, considering $K = 1,2,3,5,10$ shots.

Method	Novel Set 1					Novel Set 2					Novel Set 3				
	1	2	3	5	10	1	2	3	5	10	1	2	3	5	10
MetaYOLO [Kan18]	14.8	15.5	26.7	33.9	47.2	15.7	15.2	22.7	30.1	40.5	21.3	25.6	28.4	42.8	45.9
MetaYOLO-CME [Li21]	17.8	26.1	31.5	44.8	47.5	12.7	17.4	27.1	33.7	40.0	15.7	27.4	30.7	44.9	48.8
MetaDet-YOLO [Wan19c]	17.1	19.1	28.9	35.0	48.8	18.2	20.6	25.9	30.6	41.5	20.1	22.3	27.9	41.9	42.9
FSRN	**22.9**	**37.1**	**45.5**	**55.1**	**58.3**	**21.7**	**27.9**	**30.5**	**38.4**	**50.7**	**29.9**	**40.2**	**44.4**	**50.7**	**54.9**
Meta R-CNN [Yan19]	16.8	20.1	20.3	38.2	43.7	7.7	12.0	14.9	21.9	31.1	9.2	13.9	26.2	29.2	36.2
MetaDet [Wan19c]	18.9	20.6	30.2	36.8	49.6	21.8	23.1	27.8	31.7	43.0	20.6	23.9	29.4	43.9	44.1
FRCN-ft-full [Wan20a]	15.2	20.3	29.0	25.5	28.7	13.4	20.6	28.6	32.4	38.8	19.6	20.8	28.7	42.2	42.1
TFA w/ fc [Wan20a]	36.8	29.1	43.6	55.7	57.0	18.2	29.0	33.4	35.5	39.0	27.7	33.6	42.5	48.7	50.2
TFA w/ cos [Wan20a]	39.8	36.1	44.7	55.7	56.0	23.5	26.9	34.1	35.1	39.1	30.8	34.8	42.8	49.5	49.8
MPSR [Wu20a]	42.8	43.6	48.4	55.3	61.2	29.8	28.1	41.6	43.2	47.0	35.9	40.0	43.7	48.9	51.3
FsDetView [Xia20]	25.4	20.4	37.4	36.1	42.3	22.9	21.7	22.6	25.6	29.2	32.4	19.0	29.8	33.2	39.8
FSCE [Sun21]	32.9	44.0	46.8	52.9	59.7	23.7	30.6	38.4	43.0	48.5	22.6	33.4	39.5	47.3	54.0
CME [Li21]	41.5	47.5	50.4	58.2	60.9	27.2	30.2	41.4	42.5	46.8	34.3	39.6	45.1	48.3	51.5
DeFRCN[Qia21]	57.0	58.6	64.3	67.8	67.0	35.8	42.7	51.0	54.4	52.9	52.5	56.6	55.8	60.7	62.5

Results on PASCAL-VOC

The performance of FSOD models on the PASCAL-VOC dataset is presented in Table 7.3. In the first section of the table, the results of one-stage FSOD approaches such as [Kan18, Wan19c, Per20] are reported, including the performance of the proposed FSRN. The remaining section of the table shows the results for two-stage FSOD methods [Che18a, Wan19c, Yan19, Wan20a, Wu20a, Xia20, Qia21] and their performance on the PASCAL-VOC dataset. The proposed FSRN achieves considerable improvements as a dense meta-detector across different shot settings. Furthermore, compared to sparse meta-detectors [Yan19, Wan19c, Xia20], FSRN demonstrates competitive performance.

Model Complexity

To evaluate the model complexity, we compare the proposed FSRN model to its two-stage contrastive meta-detector counterpart, Attention-RPN [Fan20], in terms of:

- Number of parameters: The number of model parameters is an important metric as it reflects the capacity and complexity of the model as well as its memory footprint. The parameters are the biases and weights. However, the size of the model and the amount of computation needed are likewise directly correlated to the number of parameters. Models with high parameters count need more computing power for training and inference, which could result in longer training durations and larger memory footprints.
- FLOPS: An essential indicator for gauging computational efficacy is FLOPS. It calculates the floating-point operations per second such as matrix multiplications, convolutions, activation functions, and pooling. Additionally, FLOPS aids in identifying the processing requirements and hence choosing the best hardware for training or inference. The FLOPS capabilities of various hardware accelerators, such as GPUs and Tensor Processing Units (TPUs), vary. Knowing the FLOPS of a model can assess if a particular hardware device can handle the computational workload efficiently. FLOPS can also guide model optimization initiatives. Models with fewer FLOPS can be more embedded-friendly, allowing for quicker training and inference times and using less energy.
- Inference time: The inference time per image is computed on a single GPU for the 10-shot MS-COCO setting.

Table 7.4: A complexity analysis of the proposed FSRN and Attention-RPN [Fan20] on the 10-shot MS-COCO setting using a Nvidia GeForce 1080 GPU.

Model	#Parameters [M]	FLOPS [G]	Inference time [s]
Attention-RPN	55.2	178.8	0.92
FSRN	36.4	100.4	0.49

Table 7.5: Ablation study conducted on a 10-shot MS-COCO dataset explores the incremental contributions of each proposed module.

	Model Configuration	Base Performance				Novel performance				Transferability			
		bAP	bAP50	bAP75	bAR	nAP	nAP50	nAP75	nAR	PT	PT50	PT75	RT
A	Vanilla FSRN	17.7	27.8	19.1	24.1	5.7	10.8	5.2	20.2	0.32	0.39	0.27	**0.84**
B	+ MWST	30.6	45.8	33.4	52.6	12.4	21.2	12.5	30.7	0.40	0.46	0.37	0.58
C	+ Early MSF	**32.5**	**48.6**	**35.0**	**54.0**	15.1	25.3	15.2	32.1	0.46	0.52	0.43	0.59
D	+ MSDA	**32.5**	**48.6**	**35.0**	**54.0**	15.4	25.7	15.9	33.1	0.47	0.53	0.45	0.61
E	+ Gaussian Prototyping	**32.5**	**48.6**	**35.0**	**54.0**	15.8	26.4	15.9	36.0	0.49	0.54	0.45	0.67
F	+ Adaptive Pooling	31.3	47.0	33.4	53.6	**17.4**	**29.6**	**17.9**	**37.0**	**0.56**	**0.63**	**0.54**	0.69

Table 7.4 demonstrates the efficiency and performance of the proposed model. Compared to Attention-RPN, FSRN achieved notable reductions in the number of parameters, with a decrease of 34%. Additionally, the FLOPS count was reduced by 43.8%, indicating improved computational efficiency. Furthermore, FSRN exhibited faster inference times, with a reduction of 430ms per image.

7.3.3 Ablation Experiments

Impact of Individual Modules

We conduct extensive experiments to study the effect of individual modules and their interactions. All experiments are evaluated on the MS-COCO dataset. In Table 7.5, the performance on the base classes is reported *after* the meta-training phase to showcase how the overall discriminability of the model is affected by the different components. We also report the performance of the novel classes and their transferability. In configuration A, we start with a direct extension of the meta-learning paradigm on RetinaNet. This version (vanilla FSRN) utilizes a fusion mechanism directly before the detection head similar to Meta-Yolo [Kan18] and the RPN of Attention-RPN. We find that this configuration has almost the same nAP as Meta-Yolo (5.6 in Table 7.1)

Table 7.6: Ablation study on the various class prototyping approaches using 10-shot MS-COCO setting.

	Prototyping Configuration	Base Performance				Novel performance				Transferability			
		bAP	bAP50	bAP75	bAR	nAP	nAP50	nAP75	nAR	PT	PT50	PT75	RT
A	Naive Average	32.5	48.6	35.0	54.0	15.8	26.4	15.9	36.0	0.49	0.54	0.45	0.67
B	Cross-Correlation	32.5	45.0	35.2	54.2	16.5	27.9	16.4	37.0	0.51	0.62	0.47	0.68
C	Squeeze-and-Excitation	32.7	50.8	36.5	55.4	16.5	27.8	16.7	39.6	0.50	0.55	0.46	0.71
D	Cosine Similarity	32.3	48.9	35.1	54.8	17.0	29.0	17.0	38.2	0.53	0.59	0.48	0.70
E	Adaptive Pooling	31.3	47.0	33.4	53.6	**17.4**	**29.6**	**17.9**	**37.0**	**0.56**	**0.63**	**0.54**	0.69

but a higher bAP, which is attributed to the effect of the focal loss in RetinaNet [Lin18]. Adding the proposed MWST algorithm significantly boosts all metrics by almost doubling the bAP and nAP, and improving transferability. The proposed early fusion further boosts all metrics, especially the nAP. MSDA and Gaussian prototyping are only conducted in meta-testing and thus have no effect on the bAP. Their effect is reflected in the nAP and transferability.

Impact of Class Prototyping

Table 7.6 presents a comparative analysis of different prototyping techniques. The baseline configuration, denoted as configuration A, utilizes naive averaging. Configuration B introduces a weighted average approach based on cross-correlation between the query and support features. This approach demonstrates an increase in novel performance by 0.7 points, along with improved transferability indicated by higher *PT* and *RT* metrics. In configuration C, the SE approach is employed, resulting in improved base performance while maintaining similar novel and *PT* performance, with a slight enhancement in *RT*. However, the SE method cannot effectively recalibrate the feature maps with limited novel data. Configuration D utilizes cosine similarity-based prototyping, leading to a further increase in nAP by 0.5 points and improved *PT*. Unlike SE, this method does not involve learnable parameters, indicating greater robustness to limited novel data. Finally, by incorporating adaptive pooling in configuration E, there is an additional enhancement in nAP and *PT*. Despite being a learnable approach, adaptive pooling demonstrates improved resilience to limited data and a significant overall improvement compared to SE.

Table 7.7: Ablation study on data augmentations. We report the mean Averaged Precision and mean Averaged Recall on the 20 novel classes of MS-COCO in 10-shot setting.

MSF	MWST	$\mathcal{L}_{MM}$	MSDA	GP	AdaptPool	Novel Performance			
						nAP	nAP50	nAP75	nAR
✗	✗	✗	✗	✗	✗	5.7	10.8	5.2	20.2
✗	✗	✗	✓	✗	✗	8.0	14.5	7.7	34.1
✗	✗	✗	✓	✓	✗	9.7	17.2	9.7	34.0
✓	✓	✗	✗	✗	✗	15.1	25.3	15.2	32.1
✓	✓	✗	✓	✓	✗	15.4	25.7	15.9	33.1
✓	✓	✓	✗	✗	✗	15.4	26.4	15.8	34.1
✓	✓	✓	✓	✓	✗	15.8	26.4	15.9	36.0
✓	✓	✓	✓	✓	✓	**17.4**	**29.6**	**17.9**	**37.0**

Effect of Data Augmentations

Table 7.7 presents a study examining the impact of different data augmentations, specifically MSDA and GP. The table showcases the results of various configurations, incrementally adding components such as MWST, MSF, the Max-Margin Loss ($\mathcal{L}_{MM}$), and AdaPool.

The study demonstrates that MSDA and GP affect the performance of the vanilla FSRN model. Applying MSDA alone leads to a 2.3 point increase in nAP, and an additional boost of 1.3 points when GP is applied. Furthermore, introducing MWST, MSF, and/or $\mathcal{L}_{MM}$ without data augmentations results in a noticeable improvement in nAP, indicating that these modules enhance the discriminability of the detector. When data augmentations are applied with these modules, a further increase in nAP is observed, although the improvement is marginal (~ 0.4 points). Additionally, AdaPool offer significant performance gains when added in conjunction with the increments above. The best overall performance is achieved when all the modules above are combined during the meta-testing phase, as shown in the last row of Table 7.7.

Table 7.8: Receptive field effect. We report the mean Averaged Precision and mean Averaged Recall on the 20 novel classes of MS-COCO in 10-shot setting.

Receptive Field	Novel Performance			
/ Biggest Anchor	nAP	nAP50	nAP75	nAR
3/10	13.7	23.8	14.1	31.6
7/10	15.2	26.5	16.0	30.4
11/10	**17.4**	**29.6**	**17.9**	**37.0**
13/10	15.1	25.7	15.6	34.9

Effect of the Post-Fusion Receptive Field

To assess the impact of the receptive field on detection performance, the position of the support-query feature fusion within the network is altered without changing the learning capacity. Specifically, the features are fused at different layers within the classification subnet. When the features are fused just before the classification head, the post-fusion RF is reduced to 3×3. Conversely, fusing the features before the entire subnet (consisting of 5 convolutional layers) results in a RF of 11×11.

Table 7.8 shows the best results when the post-fusion receptive field covers the largest anchor size (10×10). As the receptive field decreases, the nAP experiences a decline. Furthermore, experiments are conducted with the addition of a 6th layer after the fusion to examine if increased model capacity improves precision. However, this modification leads to a degradation in performance, highlighting the significance of the post-fusion receptive field as a more critical design parameter.

Multiple Runs

To ensure a fair comparison with other benchmarks, all experiments were conducted using seed 0. In order to assess the robustness of the proposed model, a multiple runs experiment was performed on a 10-shot MS-COCO benchmark, following the methodology of TFA [Wan20a] and FSDetView [Xia20]. Our

Table 7.9: Effect of FL hyperparameters. We can see that our model is sensitive to the hyperparameters of the focal loss. This sensitivity is a problem faced by all meta-learners.

FL Parameters		Base Performance				Novel Performance			
γ	α	bAP	bAP50	bAP75	bAR	nAP	nAP50	nAP75	nAR
2	0.25	model diverges							
2	0.5								
4	0.25	24.3	38.0	26.4	40.6	12.6	23.0	12.3	30.8
4	0.5	**32.5**	**48.6**	**35.0**	**54.0**	**15.8**	**26.4**	**15.9**	**36.0**

model achieved a nAP of 15.61 ± 0.5, surpassing the performance of both TFA and FSDetView models, despite being a one-stage meta-detector.

Effect of Focal Loss Parameters

The impact of focal loss hyperparameters during the meta-training phase without the AdaPool is examined in Table 7.9. Four different settings of the α and γ hyperparameters are considered. Using the default parameters of RetinaNet [Lin18] results in training divergence. To address this issue, higher values of α and γ are required because the FSOD task has fewer positive anchors than the general object detection task, owing to the smaller number of bounding boxes in the query image. While the MWST helps mitigate this problem to some extent, further improvement is achieved by finetuning the focal loss.

Effect of the Number of Anchors

In Table 7.10, the performance of the meta-detector is shown for different numbers of anchors when trained without the AdaPool. The results highlight the importance of anchor density for meta-detectors. The significance of the number and size of anchors has always been recognized in dense object detectors [Lin18], but their effect becomes more pronounced in FSOD. It is observed in our design that increasing the number of anchors leads to improved performance in both the base and novel classes. The hypothesis is that more anchors provide a stronger learning signal to the post-fusion network, enabling

Table 7.10: A study on the Impact of the number of anchors. Initially, increasing the number of anchors results in performance enhancement. However, it can cause training instability unless the learning capacity is scaled accordingly.

Sizes	Ratios	Base Performance				Novel Performance			
		bAP	bAP50	bAP75	bAR	nAP	nAP50	nAP75	nAR
1	1	17.7	27.8	19.1	40.0	7.6	13.8	7.6	24.0
3	3	27.5	40.5	29.8	46.6	13.0	22.0	13.0	29.6
5	3	**32.5**	**48.6**	**35.0**	**54.0**	**15.8**	**26.4**	**15.9**	**36.0**
5	5	model diverges							
7	7								

Table 7.11: Ablation study on the various backbones for FSRN using 10-shot MS-COCO setting.

	Backbone Configuration	Base Performance				Novel performance				Transferability			
		bAP	bAP50	bAP75	bAR	nAP	nAP50	nAP75	nAR	PT	PT50	PT75	RT
A	ResNet50-FPN	31.3	47.0	33.4	53.6	17.4	29.6	17.9	37.0	0.56	0.63	0.54	0.69
B	ResNet101-FPN	31.9	47.8	34.1	54.5	17.6	30.4	18.1	37.4	0.55	0.64	0.53	0.69
C	Swin-Tiny	22.4	36.0	23.5	47.2	17.6	33.4	16.5	38.1	0.79	0.93	0.70	0.81
D	Swin-Small	26.8	42.0	28.7	50.4	20.6	38.2	19.8	41.1	0.77	0.91	0.69	0.82

Table 7.12: The complexity analysis of different backbones on the various backbones on a $224 \times 224 \times 3$ image on a Nvidia Tesla V100 GPU.

Backbone	#Params [M]	FLOPs [G]	Latency [ms]
RN50-FPN	23.5	4.1	6.0
RN101-FPN	44	7.8	7.8
Swin-T	28	8.9	8.1
Swin-S	50	16.2	10.0

better refinement of instance-level features. It is noted that training becomes unstable when the number of anchors exceeds 15, possibly due to a reduced model capacity for the task.

Backbones Performance and Complexity

In the study shown in Table 7.11, the impact of different backbone choices on the performance and transferability of a novel approach has been studied. A comparison between the CNN-based backbones, such as ResNet50-FPN and ResNet101-FPN, with transformer-based backbones, including Swin-Tiny and

Swin-Small, has been conducted. Two main observations were made. Firstly, using a deeper ResNet backbone, as in configuration B, did not improve performance. This implies either the model capacity is already sufficient for the problem or the architecture cannot capture generalizable knowledge effectively. To get better intuition, the CNN-based backbones are replaced with transformer-based ones. While the Swin-Small transformer backbone significantly boosted the novel and transferability performance, its base performance was noticeably lower. This suggests that transformer-based backbones in this scenario may require more data to improve the base performance and enhance both the novel and transferability performance. This indicates that deeper transformer-based backbones can capture better prior knowledge.

However, although transformer-based backbones can offer substantial gains, there are trade-offs involved. In Table 7.12, a complexity analysis of the utilized backbones is provided. An input image of size $224 \times 224 \times 3$ was used, and the backbones were benchmarked on an Nvidia Tesla V100 GPU. The table reports the number of backbone parameters, FLOPS, and latency for a single image. While the number of parameters in the Swin backbones may not be significantly higher, the number of FLOPS is nearly doubled compared to their ResNet counterparts. Furthermore, the latency significantly increases, with a single image requiring approximately 10.0 ms for a forward pass. Despite Swin-Small delivering significantly better results than ResNets, FSRN utilizes the ResNet50-FPN backbone to maintain a more embedded-friendly meta-detector.

Qualitative Results

Figure 7.4 showcases various scenarios depicting successful and failed detections. In the first row, FSRN demonstrates impressive performance in detecting small-scaled objects like baseball and most persons in the first image. However, it fails to detect the Frisbee in the second image. Moving to the second row, FSRN exhibits some false positives in the first image but successfully identifies all the traffic lights in the second one. The subsequent rows display a mix of correct and false detections.

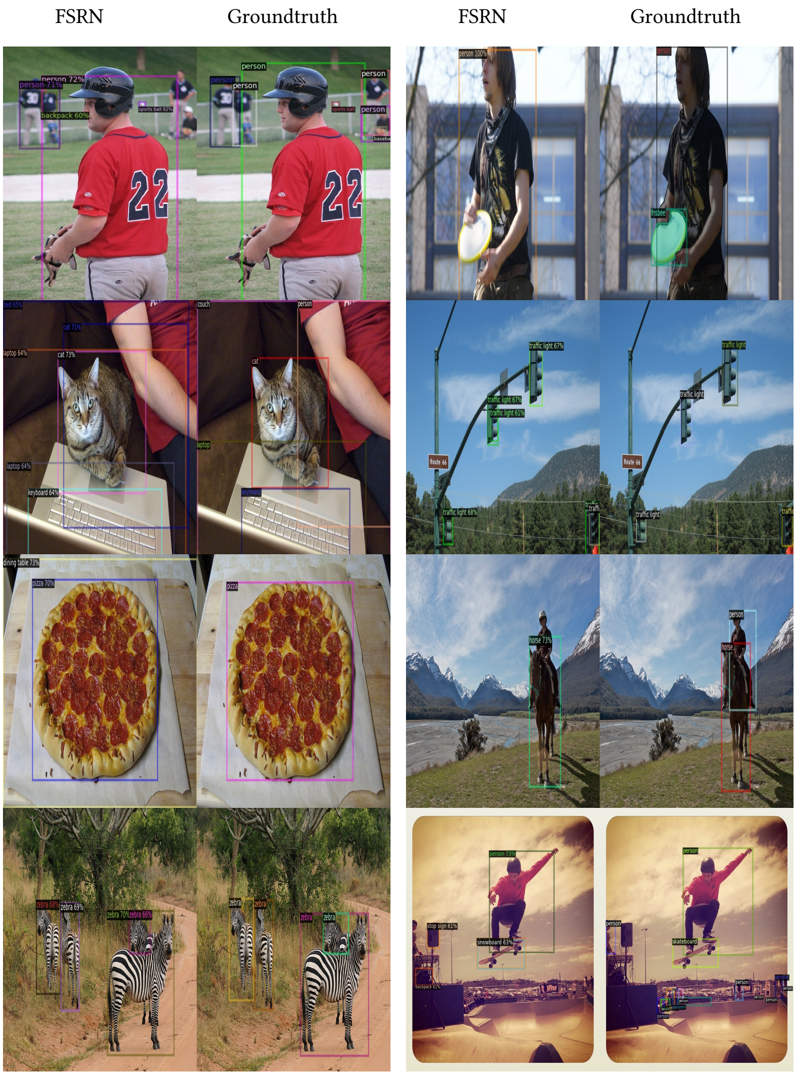


Figure 7.4: Qualitative results for the proposed FSRN model in 10-shot setting on MS-COCO dataset.

7.4 Discussion

The proposed FSRN framework demonstrates strong generalization performance on the demanding MS-COCO and PASCAL-VOC datasets. However, one drawback of our framework is that the MWST introduces additional computational overhead due to the processing of multiple support images. Furthermore, the training process is sensitive to hyperparameters. Future research could focus on improving the training stability of meta-detectors and reducing the memory requirements of data augmentation techniques.

Furthermore, the use of transformer-based backbones can greatly enhance the performance of the novel classes. However, it comes with increased computational requirements, memory footprint, and latency. Nevertheless, considering the rise of modern, efficient transformer architectures, there is a potential to develop a meta-detector that achieves both high base and novel performance and remains suitable for embedded systems deployment.

These contributions would not only benefit FSRN but also have potential applications in other one-stage and two-stage detection models.

8 Conclusions and Outlook

8.1 Conclusions

This dissertation presented significant contributions to the field of FSOD, with a focus on addressing challenges related to limited data, robustness, and efficiency. Multiple novel FSOD frameworks were developed, each specifically tailored to tackle distinct challenges and constraints.

The first contribution was the CFA, designed to alleviate forgetting without increasing model capacity or inference time, ensuring efficient integration with existing few-shot detection frameworks. It derived a new gradient update rule that adaptively reweights the base and novel gradients, effectively reducing forgetting and promoting better knowledge transfer between base and novel classes. CFA serves as a plug-and-play module that can be integrated with various G-FSOD models. At the time of its publication, CFA demonstrated superior performance on both the base and novel detection tasks of MS-COCO and PASCAL-VOC datasets, achieving a new standard.

For the second contribution, a new G-FSOD framework, namely DeCRCN-UPPR, was designed. It leverages predictive uncertainties to refine object proposals in a stagewise manner. Estimating uncertainties provided valuable distributional information to reduce forgetting and enhance the novel detection performance. Moreover, attention blocks were appended to each R-CNN stage during novel training to selectively focus on discriminative features. Integrating multiple R-CNN stages and attention blocks significantly enhanced the detection performance for both base and novel classes.

To maintain privacy and memory constraints, the NIFF framework was introduced as the first DFKD paradigm for G-FSOD to alleviate forgetting without replaying base data during novel training. NIFF leveraged a tailored standalone feature generator, aligning class-wise statistics in the RoI-head to forge base instance-level features during generator training. Through knowledge distillation approaches and the use of the proposed base feature generator, NIFF surpassed replay-based methods such as CFA without relying on base data.

Finally, to tackle the high computational and time overhead in FSOD approaches, the FSRN framework was introduced. FSRN was a one-stage meta-learning FSOD approach that extended the RetinaNet detection model. This framework incorporated various components, including the MSF module, enabling a wide receptive field for comprehensive coverage of the anchor area, and the MWST strategy, aimed at increasing the number of foreground samples to enhance the learning signal. Additionally, the MSDA strategy was applied to both query and support images during meta-testing, improving the diversity of the data distribution. The utilization of GP during meta-testing facilitates more accurate determination of class prototypes, while the adaptive calibration of support feature maps using AdaPool computed more representative support class prototypes, contributing to improved detection performance. FSRN achieved state-of-the-art results on one-stage FSOD on the MS-COCO and PASCAL-VOC datasets.

The outcomes of this research contributes significantly to the advancement of FSOD and G-FSOD, paving the way for more efficient, adaptable, and robust object detection systems in industrial automation, robotics, and beyond. The findings presented in this thesis represents a step toward enhancing productivity, quality assurance, and overall performance in a wide range of real-world industrial settings.

8.2 Outlook

Although the FSOD and G-FSOD methods have demonstrated promising results, further enhancements and extensions are essential to meet the demands of real-world industrial applications. Addressing challenges posed by complex and dynamic environments, occlusions, and cluttered scenes is crucial. Possible research directions might explore novel techniques to improve the detection accuracy, robustness, and generalization capabilities of the models. In the following, various potential research directions are highlighted.

A significant limitation in deploying FSOD and G-FSOD models in practical scenarios is the computational cost. To enable real-time applications, it is imperative to design more efficient FSOD and G-FSOD frameworks that can run on resource-constrained embedded hardware. This necessitates the exploration of more efficient architectures using Neural Architecture Search (NAS) to find solutions that respect hardware limitations.

Uncertainty estimation has shown great potential in improving the reliability of detection systems. Future research should delve deeper into investigating and exploring more recent predictive uncertainty estimation techniques. By incorporating uncertainty measures, the FSOD and G-FSOD models can make more informed decisions and increase their robustness when encountering novel or ambiguous scenarios.

Leveraging recent advances in efficient transformer architectures, researchers can explore new ways to learn robust representations of novel classes without suffering from catastrophic forgetting or overfitting to limited examples. By harnessing the power of transformers, the FSOD and G-FSOD models can be more adaptable to new, previously unseen object categories.

The rise of generative AI approaches and foundational models presents an opportunity to address data scarcity issues in FSOD and G-FSOD. By generating more diverse and realistic samples from available novel data, researchers can significantly improve the robustness and generalization capabilities of the models. Data augmentation through generative techniques can also help alleviate overfitting and improve the models' ability to handle novel scenarios. An alternative avenue worth exploring could involve incorporating LLMs into the FSOD domain. For instance, this approach could entail the utilization of textual prompts as a means to find specific objects or choose the support set within the FSOD framework.

In conclusion, the outlook for FSOD and G-FSOD research is promising, with several potential research directions that can lead to significant improvements in real-world applications. Addressing issues related to efficiency, uncertainty estimation, representation learning, and data augmentation will contribute to the advancement of these methods and their practical utility in industrial contexts.

Bibliography

[Alj17] Aljundi, Rahaf; Chakravarty, Punarjay and Tuytelaars, Tinne: “Expert Gate: Lifelong Learning with a Network of Experts”. In: *IEEE Conference on Computer Vision and Pattern Recognition.* 2017, pp. 7120–7129 (cit. on p. 69).

[Alj18] Aljundi, Rahaf; Babiloni, Francesca; Elhoseiny, Mohamed; Rohrbach, Marcus and Tuytelaars, Tinne: “Memory Aware Synapses: Learning what (not) to forget”. In: *European Conference on Computer Vision.* 2018, pp. 144–161 (cit. on pp. 69, 106, 107, 120).

[Alj19] Aljundi, Rahaf; Belilovsky, Eugene; Tuytelaars, Tinne; Charlin, Laurent; Caccia, Massimo; Lin, Min and Page-Caccia, Lucas: “Online Continual Learning with Maximal Interfered Retrieval”. In: *Advances in Neural Information Processing Systems.* 2019 (cit. on pp. 68, 69, 71).

[Atk18] Atkinson, Craig; McCane, Brendan; Szymanski, Lech and Robins, Anthony V.: “Pseudo-Recursal: Solving the Catastrophic Forgetting Problem in Deep Neural Networks”. In: *CoRR* abs/1802.03875 (2018) (cit. on pp. 68, 69).

[Ba16] Ba, Jimmy; Kiros, Jamie and Hinton, Geoffrey: “Layer Normalization”. In: (July 2016) (cit. on p. 17).

[Bot10] Bottou, Léon: “Large-scale Machine Learning with Stochastic Gradient Descent”. In: *Proceedings of the International Conference on Computational Statistics* 57 (2010), pp. 177–186 (cit. on p. 29).

[Bro20] Brown, Tom et al.: “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems (NeurIPS).* Vol. 33. 2020, pp. 1877–1901 (cit. on p. 1).

[Cai18] Cai, Zhaowei and Vasconcelos, Nuno: “Cascade R-CNN: Delving into High Quality Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 6154–6162 (cit. on pp. 33, 91).

[Cha18] Chaudhry, Arslan; Dokania, Puneet Kumar; Ajanthan, Thalaiyasingam and Torr, Philip H. S.: “Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence”. In: *European Conference on Computer Vision.* 2018, pp. 556–572 (cit. on pp. 69, 106).

[Cha19a] Chaudhry, Arslan; Ranzato, Marc’Aurelio; Rohrbach, Marcus and Elhoseiny, Mohamed: “Efficient Lifelong Learning with A-GEM”. In: *International Conference on Learning Representations.* 2019 (cit. on pp. 68, 69, 71–74, 76, 83, 85, 103).

[Cha19b] Chaudhry, Arslan; Rohrbach, Marcus; Elhoseiny, Mohamed; Ajanthan, Thalaiyasingam; Dokania, Puneet Kumar; Torr, Philip H. S. and Ranzato, Marc’Aurelio: “Continual Learning with Tiny Episodic Memories”. In: *CoRR* abs/1902.10486 (2019) (cit. on pp. 68, 69).

[Cha21] Chawla, Akshay; Yin, Hongxu; Molchanov, Pavlo and Alvarez, Jose: “Data-free Knowledge Distillation for Object Detection”. In: *2021 IEEE Winter Conference on Applications of Computer Vision (WACV).* 2021, pp. 3288–3297 (cit. on pp. 108, 110, 113).

[Che18a] Chen, Hao; Wang, Yali; Wang, Guoyou and Qiao, Yu: “LSTD: A Low-Shot Transfer Detector for Object Detection”. In: *Conference on Artificial Intelligence.* 2018, pp. 2836–2843 (cit. on pp. 49, 151, 153).

[Che18b] Chen, Yuhua; Li, Wen; Sakaridis, Christos; Dai, Dengxin and Gool, Luc Van: “Domain Adaptive Faster R-CNN for Object Detection in the Wild”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3339–3348 (cit. on p. 215).

[Che20a] Chen, Chaoqi; Zheng, Zebiao; Ding, Xinghao; Huang, Yue and Dou, Qi: “Harmonizing transferability and discriminability for adapting object detectors”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 8869–8878 (cit. on p. 215).

[Che20b] Chen, Ting; Kornblith, Simon; Norouzi, Mohammad and Hinton, Geoffrey: “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607 (cit. on p. 222).

[Cor95] Cortes, Corinna and Vapnik, Vladimir: “Support-vector networks”. In: vol. 20. 3. Springer, 1995, pp. 273–297 (cit. on p. 28).

[Dan19] Danielczuk, Michael; Matl, Matthew; Gupta, Saurabh; Li, Andrew; Lee, Andrew; Mahler, Jeffrey and Goldberg, Ken: “Segmenting unknown 3d objects from real depth images using mask r-cnn trained on synthetic data”. In: *International Conference on Robotics and Automation*. 2019, pp. 7283–7290 (cit. on pp. 215, 219).

[Day05] Dayan, Peter and Abbott, Laurence F: Theoretical neuroscience: Computational and mathematical modeling of neural systems. MIT press, 2005 (cit. on p. 10).

[Dos21] Dosovitskiy, Alexey et al.: “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *International Conference on Learning Representations (ICLR)*. 2021 (cit. on p. 23).

[Dua19] Duan, Kaiwen; Bai, Song; Xie, Lingxi; Qi, Honggang; Huang, Qingming and Tian, Qi: “CenterNet: Keypoint Triplets for Object Detection”. In: *IEEE International Conference on Computer Vision*. 2019, pp. 6568–6577 (cit. on pp. 39, 58, 68, 90, 91).

[Eve10] Everingham, Mark; Van Gool, Luc; Williams, Christopher KI; Winn, John and Zisserman, Andrew: “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338 (cit. on pp. 46, 59, 60, 96, 118, 217, 224).

[Fan20] Fan, Qi; Zhuo, Wei and Tai, Yu-Wing: “Few-Shot Object Detection with Attention-RPN and Multi-Relation Detector”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 4012–4021 (cit. on pp. 57, 60, 78, 137, 138, 142–144, 151–154, 218, 224–226, 229).

[Fan21] Fan, Zhibo; Ma, Yuchen; Li, Zeming and Sun, Jian: “Generalized Few-Shot Object Detection Without Forgetting”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2021, pp. 4527–4536 (cit. on pp. 68, 70, 77–82, 89, 96–98, 100, 115, 118–122, 134).

[Fen18] Feng, Di; Rosenbaum, Lars and Dietmayer, Klaus: “Towards Safe Autonomous Driving: Capture Uncertainty in the Deep Neural Network For Lidar 3D Vehicle Detection”. In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3266–3273 (cit. on p. 71).

[Gal16] Gal, Yarin and Ghahramani, Zoubin: “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”. In: *Proceedings of The International Conference on Machine Learning (ICML)*. Vol. 48. Proceedings of Machine Learning Research. 2016, pp. 1050–1059 (cit. on p. 71).

[Gaw23] Gawlikowski, Jakob et al.: “A Survey of Uncertainty in Deep Neural Networks”. In: *Artificial Intelligence Review* (July 2023) (cit. on pp. 70, 90, 93).

[Gir14] Girshick, Ross; Donahue, Jeff; Darrell, Trevor and Malik, Jitendra: “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587 (cit. on p. 28).

[Gir15] Girshick, Ross: “Fast R-CNN”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* 2015, pp. 1440–1448 (cit. on p. 29).

[Goo15] Goodfellow, Ian J.; Shlens, Jonathon and Szegedy, Christian: “Explaining and Harnessing Adversarial Examples”. In: *International Conference on Learning Representations (ICLR).* Ed. by Bengio, Yoshua and LeCun, Yann. 2015 (cit. on p. 220).

[Gui22a] Guirguis, Karim; Eskandar, George; Kayser, Matthias; Yang, Bin and Beyerer, Juergen: “Few-Shot Object Detection in Unseen Domains”. In: *2022 16th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS).* 2022, pp. 98–107 (cit. on pp. 5, 66).

[Gui22b] Guirguis, Karim; Hendawy, Ahmed; Eskandar, George; Abdelsamad, Mohamed; Kayser, Matthias and Beyerer, Jürgen: “CFA: Constraint-based Finetuning Approach for Generalized Few-Shot Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPRW).* 2022, pp. 4039–4049 (cit. on pp. 4, 65, 67, 96–102, 115–122, 131, 132, 134).

[Gui23a] Guirguis, Karim; Abdelsamad, Mohamed; Eskandar, George; Hendawy, Ahmed; Kayser, Matthias; Yang, Bin and Beyerer, Juergen: “Towards Discriminative and Transferable One-Stage Few-Shot Object Detectors”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).* 2023, pp. 3760–3769 (cit. on pp. 4, 66, 137, 139).

[Gui23b] Guirguis, Karim; Meier, Johannes; Eskandar, George; Kayser, Matthias; Yang, Bin and Beyerer, Juergen: “NIFF: Alleviating Forgetting in Generalized Few-Shot Object Detection via Neural Instance Feature Forging”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2023, pp. 24193–24202 (cit. on pp. 4, 65).

[Har20] Harakeh, Ali; Smart, Michael and Waslander, Steven L.: "BayesOD: A Bayesian Approach for Uncertainty Estimation in Deep Object Detectors". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 87–93 (cit. on pp. 70, 71).

[He16] He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing and Sun, Jian: "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on pp. 21, 76, 90, 96, 151).

[He19] He, Zhenwei and Zhang, Lei: "Multi-adversarial Faster-RCNN for Unrestricted Object Detection". In: *IEEE International Conference on Computer Vision*. 2019, pp. 6667–6676 (cit. on p. 215).

[Hod17] Hodan, Tomas; Haluza, Pavel; Obdrzalek, Stepan; Matas, Jiri; Lourakis, Manolis and Zabulis, Xenophon: "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017), pp. 880–888 (cit. on pp. 216, 224).

[Hu18] Hu, Jie; Shen, Li and Sun, Gang: "Squeeze-and-Excitation Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 148).

[Iof15] Ioffe, Sergey and Szegedy, Christian: "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning, ICML*. Vol. 37. 2015, pp. 448–456 (cit. on p. 16).

[Ise18] Isele, David and Cosgun, Akansel: "Selective Experience Replay for Lifelong Learning". In: *Conference on Artificial Intelligence*. 2018, pp. 3302–3309 (cit. on pp. 68, 69).

[Jun16] Jung, Heechul; Ju, Jeongwoo; Jung, Minju and Kim, Junmo: "Less-forgetting Learning in Deep Neural Networks". In: *CoRR* abs/1607.00122 (2016) (cit. on pp. 69, 106).

[Kam17] Kamra, Nitin; Gupta, Umang and Liu, Yan: “Deep Generative Dual Memory Network for Continual Learning”. In: *CoRR* abs/1710.10368 (2017) (cit. on pp. 68, 69).

[Kan18] Kang, Bingyi; Liu, Zhuang; Wang, Xin; Yu, Fisher; Feng, Jiashi and Darrell, Trevor: “Few-Shot Object Detection via Feature Reweighting”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2018, pp. 8419–8428 (cit. on pp. 54, 78, 79, 81, 82, 87, 97, 99, 100, 119, 121, 122, 138, 141, 143, 150–152, 154, 225).

[Ken17] Kendall, Alex and Gal, Yarin: “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?” In: *Advances in Neural Information Processing Systems (NIPS)*. Vol. 30. 2017 (cit. on pp. 70, 90, 93).

[Kho20] Khosla, Prannay; Teterwak, Piotr; Wang, Chen; Sarna, Aaron; Tian, Yonglong; Isola, Phillip; Maschinot, Aaron; Liu, Ce and Krishnan, Dilip: “Supervised Contrastive Learning”. In: *Advances in Neural Information Processing Systems*. Ed. by Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M. F. and Lin, H. Vol. 33. Curran Associates, Inc., 2020, pp. 18661–18673 (cit. on p. 222).

[Kir16] Kirkpatrick, James et al.: “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the National Academy of Sciences* 114 (2016) (cit. on pp. 69, 106, 107, 117, 120, 129).

[Koc15] Koch, Gregory; Zemel, Richard and Salakhutdinov, Ruslan: “Siamese neural networks for one-shot image recognition”. In: *International Conference on Machine Learning Workshops*. 2015 (cit. on p. 222).

[Kod15] Kodirov, E.; Xiang, T.; Fu, Z. and Gong, S.: “Unsupervised Domain Adaptation for Zero-Shot Learning”. In: *IEEE International Conference on Computer Vision*. 2015, pp. 2452–2460 (cit. on p. 216).

[Köh23] Köhler, Mona; Eisenbach, Markus and Gross, Horst-Michael: "Few-Shot Object Detection: A Comprehensive Survey". In: IEEE, 2023 (cit. on p. 146).

[Kra19] Kraus, Florian and Dietmayer, Klaus: "Uncertainty Estimation in One-Stage Object Detection". In: *IEEE Intelligent Transportation Systems Conference (ITSC).* 2019, pp. 53–60 (cit. on pp. 70, 71, 95).

[Kur21] Kurmi, Vinod K; Patro, Badri N; Subramanian, Venkatesh K and Namboodiri, Vinay P: "Do not Forget to Attend to Uncertainty while Mitigating Catastrophic Forgetting". In: *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).* 2021, pp. 736–745 (cit. on pp. 89, 92, 93).

[Lan21a] Lange, Matthias De; Aljundi, Rahaf; Masana, Marc; Parisot, Sarah; Jia, Xu; Leonardis, Ales; Slabaugh, Gregory G. and Tuytelaars, Tinne: "Continual learning: A comparative study on how to defy forgetting in classification tasks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021) (cit. on p. 68).

[Lan21b] Lange, Matthias De and Tuytelaars, Tinne: "Continual Prototype Evolution: Learning Online from Non-Stationary Data Streams". In: *IEEE International Conference on Computer Vision.* 2021 (cit. on pp. 68, 69).

[Lav18] Lavda, Frantzeska; Ramapuram, Jason; Gregorova, Magda and Kalousis, Alexandros: "Continual Classification Learning Using Generative Models". In: *Advances in Neural Information Processing Systems Workshops.* 2018 (cit. on pp. 68, 69).

[LeC98] LeCun, Yann; Bottou, Léon; Bengio, Yoshua and Haffner, Patrick: "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on p. 13).

[Lee17] Lee, Sang-Woo; Kim, Jin-Hwa; Ha, JungWoo and Zhang, Byoung-Tak: "Overcoming Catastrophic Forgetting by Incremental Moment Matching". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4652–4662 (cit. on pp. 69, 106).

[Les93] Leshno, Moshe; Ya. Lin, Vladimir; Pinkus, Allan and Schocken, Shimon: "Multilayer Feedforward Networks with a Non-polynomial Activation Function can Approximate any Function". In: vol. 6. 6. 1993, pp. 861–867 (cit. on p. 11).

[Li16] Li, Zhizhong and Hoiem, Derek: "Learning without Forgetting". In: *European Conference on Computer Vision*. 2016, pp. 614–629 (cit. on pp. 69, 89, 93, 106).

[Li21] Li, Bohao; Yang, Boyu; Liu, Chang; Liu, Feng; Ji, Rongrong and Ye, Qixiang: "Beyond Max-Margin: Class Margin Equilibrium for Few-Shot Object Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 7363–7372 (cit. on pp. 58, 140, 151, 152).

[Li22] Li, Yanghao; Mao, Hanzi; Girshick, Ross and He, Kaiming: "Exploring Plain Vision Transformer Backbones for Object Detection". In: *Proceedings of the IEEE/CVF Proceedings of the IEEE/CVF European Conference on Computer Vision (ECCV) (ECCV)*. 2022, pp. 280–296 (cit. on p. 26).

[Lin14] Lin, Tsung-Yi; Maire, Michael; Belongie, Serge; Hays, James; Perona, Pietro; Ramanan, Deva; Dollár, Piotr and Zitnick, C Lawrence: "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755 (cit. on pp. 31, 42, 46, 59, 96, 118, 138).

[Lin17] Lin, Tsung-Yi; Dollar, Piotr; Girshick, Ross; He, Kaiming; Hariharan, Bharath and Belongie, Serge: "Feature Pyramid Networks for Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 (cit. on pp. 22, 76, 90, 91, 151).

[Lin18] Lin, Tsung-Yi; Goyal, Priyal; Girshick, Ross; He, Kaiming and Dollar, Piotr: “Focal Loss for Dense Object Detection”. In: vol. 42. 2. 2018, pp. 318–327 (cit. on pp. 38, 58, 140, 141, 143, 151, 155, 158).

[Lin20] Linder, Timm; Pfeiffer, Kilian Y.; Vaskevicius, Narunas; Schirmer, Robert and Arras, Kai O.: “Accurate detection and 3D localization of humans using a novel YOLO-based RGB-D fusion approach and synthetic training data”. In: *IEEE International Conference on Robotics and Automation*. 2020, pp. 1000–1006 (cit. on pp. 215, 219).

[Liu18a] Liu, Shu; Qi, Lu; Qin, Haifang; Shi, Jianping and Jia, Jiaya: “Path Aggregation Network for Instance Segmentation”. In: (2018), pp. 8759–8768 (cit. on pp. 149, 150).

[Liu18b] Liu, Xialei; Masana, Marc; Herranz, Luis; Weijer, Joost van de; López, Antonio M. and Bagdanov, Andrew D.: “Rotate your Networks: Better Weight Consolidation and Less Catastrophic Forgetting”. In: *International Conference on Pattern Recognition*. 2018, pp. 2262–2268 (cit. on pp. 69, 106).

[Liu21] Liu, Ze; Lin, Yutong; Cao, Yue; Hu, Han; Wei, Yixuan; Zhang, Zheng; Lin, Stephen and Guo, Baining: “Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10012–10022 (cit. on p. 24).

[Loh19] Loh, Yuen Peng and Chan, Chee Seng: “Getting to Know Low-light Images with The Exclusively Dark Dataset”. In: *Computer Vision and Image Understanding* 178 (2019), pp. 30–42. doi: https://doi.org/10.1016/j.cviu.2018.10.010 (cit. on pp. 217, 224).

[Lop17] Lopez-Paz, David and Ranzato, Marc’Aurelio: “Gradient Episodic Memory for Continuum Learning”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 6470–6479 (cit. on pp. 68, 69, 71, 72, 76, 103).

[Luo16] Luo, Wenjie; Li, Yujia; Urtasun, Raquel and Zemel, Richard: "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems.* Vol. 29. 2016 (cit. on p. 142).

[Mal18a] Mallya, Arun and Lazebnik, Svetlana: "PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning". In: *IEEE Conference on Computer Vision and Pattern Recognition.* 2018, pp. 7765–7773 (cit. on p. 69).

[Mal18b] Mallya, Arun and Lazebnik, Svetlana: "Piggyback: Adding Multiple Tasks to a Single, Fixed Network by Learning to Mask". In: *European Conference on Computer Vision.* 2018, pp. 72–88 (cit. on p. 69).

[Mor15] Mordvintsev, A.; Olah, Christopher and Tyka, Mike: "Inceptionism: Going Deeper into Neural Networks". In: 2015 (cit. on p. 108).

[Mus20] Musgrave, Kevin; Belongie, Serge and Lim, Ser-Nam: "A metric learning reality check". In: *European Conference on Computer Vision.* 2020, pp. 681–699 (cit. on p. 223).

[Nit14] Nitish Srivastava and Geoffrey Hinton and Alex Krizhevsky and Ilya Sutskever and Ruslan Salakhutdinov: "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958 (cit. on pp. 70, 93).

[Pen18] Peng, Kuan Chuan; Wu, Ziyan and Ernst, Jan: "Zero-Shot Deep Domain Adaptation". In: *European Conference on Computer Vision.* 2018, pp. 793–810 (cit. on p. 216).

[Per20] Perez-Rua, Juan-Manuel; Zhu, Xiatian; Hospedales, Timothy M. and Xiang, Tao: "Incremental Few-Shot Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2020, pp. 13843–13852 (cit. on pp. 58, 68, 78, 79, 97, 119, 151, 152).

[Pri23] Prince, Simon J.D.: Understanding Deep Learning. MIT Press, 2023. url: https://udlbook.github.io/udlbook/ (cit. on p. 14).

[Qia21] Qiao, Limeng; Zhao, Yuxuan; Li, Zhiyuan; Qiu, Xi; Wu, Jianan and Zhang, Chi: “Defrcn: Decoupled faster r-cnn for few-shot object detection”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8681–8690 (cit. on pp. 52, 60, 61, 65, 70, 76, 78, 79, 81, 82, 86, 87, 89, 90, 93, 96–102, 115, 118–122, 125, 127, 130–134, 151–153).

[Rah20] Rahman, Shafin; Khan, Salman; Barnes, Nick and Khan, Fahad Shahbaz: “Any-shot object detection”. In: *Proceedings of the Asian Conference on Computer Vision*. 2020 (cit. on p. 225).

[Ram20] Ramapuram, Jason; Gregorova, Magda and Kalousis, Alexandros: “Lifelong generative modeling”. In: *Neurocomputing* 404 (2020), pp. 381–400 (cit. on pp. 68, 69).

[Reb17] Rebuffi, Sylvestre-Alvise; Kolesnikov, Alexander and Lampert, Christoph H.: “iCaRL: Incremental Classifier and Representation Learning”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5533–5542 (cit. on pp. 68, 69).

[Red16] Redmon, Joseph; Divvala, Santosh; Girshick, Ross and Farhadi, Ali: “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788 (cit. on pp. 35, 58).

[Red17] Redmon, Joseph and Farhadi, Ali: “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 7263–7271 (cit. on pp. 37, 54, 58).

[Red18] Redmon, Joseph and Farhadi, Ali: “YOLOv3: An Incremental Improvement”. In: vol. abs/1804.02767. 2018 (cit. on p. 37).

[Ren15] Ren, Shaoqing; He, Kaiming; Girshick, Ross and Sun, Jian: "Faster R-CNN: Towards real-time object detection with region proposal networks". In: *Advances in Neural Information Processing Systems (NIPS).* 2015, pp. 91–99 (cit. on pp. 27, 31, 49, 50, 52, 72, 73, 76, 89, 90, 117, 141, 215, 218).

[Rie19] Riemer, Matthew; Cases, Ignacio; Ajemian, Robert; Liu, Miao; Rish, Irina; Tu, Yuhai and Tesauro, Gerald: "Learning to Learn without Forgetting By Maximizing Transfer and Minimizing Interference". In: *International Conference on Learning Representations.* 2019 (cit. on pp. 68, 69, 71).

[Rol19] Rolnick, David; Ahuja, Arun; Schwarz, Jonathan; Lillicrap, Timothy P. and Wayne, Greg: "Experience Replay for Continual Learning". In: *Advances in Neural Information Processing Systems.* 2019, pp. 348–358 (cit. on pp. 68, 69).

[Ros20] Rosenfeld, Amir and Tsotsos, John K.: "Incremental Learning Through Deep Adaptation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.3 (2020), pp. 651–663 (cit. on p. 69).

[Ros58] Rosenblatt, Frank: "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: vol. 65. 6. 1958, pp. 386–408 (cit. on p. 10).

[Rus15] Russakovsky, Olga; Deng, Jia; Su, Hao; Krause, Jonathan; Satheesh, Sanjeev; Ma, Sean; Huang, Zhiheng; Karpathy, Andrej; Khosla, Aditya; Bernstein, Michael et al.: "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252 (cit. on p. 110).

[Rus16] Rusu, Andrei A.; Rabinowitz, Neil C.; Desjardins, Guillaume; Soyer, Hubert; Kirkpatrick, James; Kavukcuoglu, Koray; Pascanu, Razvan and Hadsell, Raia: "Progressive Neural Networks". In: *CoRR* abs/1606.04671 (2016) (cit. on p. 69).

[Sai19] Saito, Kuniaki; Ushiku, Yoshitaka; Harada, Tatsuya and Saenko, Kate: “Strong-Weak Distribution Alignment for Adaptive Object Detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6956–6965 (cit. on p. 215).

[Sal83] Salton, Gerard and McGill, Michael: Introduction to Modern Information Retrieval. McGraw-Hill, 1983 (cit. on p. 41).

[Sch15] Schroff, Florian; Kalenichenko, Dmitry and Philbin, James: “FaceNet: A unified embedding for face recognition and clustering”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 815–823 (cit. on p. 222).

[Ser18] Serrà, Joan; Surís, Dídac; Miron, Marius and Karatzoglou, Alexandros: “Overcoming catastrophic forgetting with hard attention to the task”. In: *International Conference on Machine Learning*. 2018, pp. 4555–4564 (cit. on pp. 69, 89, 93).

[Sha18] Sharma, Abhishek; Somanath, Gowri and Namboodiri, Anoop M: “Object Detection Knowledge Transfer via Distillation of Object Proposals”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 694–709 (cit. on p. 108).

[Sim14] Simonyan, Karen and Zisserman, Andrew: “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (2014) (cit. on pp. 20, 29).

[Smi21] Smith, James; Hsu, Yen-Chang; Balloch, Jonathan; Shen, Yilin; Jin, Hongxia and Kira, Zsolt: “Always Be Dreaming: A New Approach for Data-Free Class-Incremental Learning”. In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 9354–9364 (cit. on pp. 108, 109, 111, 116, 123).

[Sun18] Sundermeyer, Martin; Marton, Zoltan-Csaba; Durner, Maximilian; Brucker, Manuel and Triebel, Rudolph: “Implicit 3d orientation learning for 6d object detection from rgb images”. In: *European Conference on Computer Vision*. 2018, pp. 699–715 (cit. on pp. 215, 219, 220).

[Sun21] Sun, Bo; Li, Banghuai; Cai, Shengcai; Yuan, Ye and Zhang, Chi: “FSCE: Few-Shot Object Detection via Contrastive Proposal Encoding”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2021, pp. 7352–7362 (cit. on pp. 50, 151, 152).

[Tob17] Tobin, Josh; Fong, Rachel; Ray, Alex; Schneider, Jonas; Zaremba, Wojciech and Abbeel, Pieter: “Domain randomization for transferring deep neural networks from simulation to the real world”. In: *IEEE International Conference on Intelligent Robots and Systems.* 2017, pp. 23–30 (cit. on pp. 215, 219).

[Tri17] Triki, Amal Rannen; Aljundi, Rahaf; Blaschko, Matthew B. and Tuytelaars, Tinne: “Encoder Based Lifelong Learning”. In: *IEEE International Conference on Computer Vision.* 2017, pp. 1329–1337 (cit. on pp. 69, 106).

[Uly16] Ulyanov, Dmitry; Vedaldi, Andrea and Lempitsky, Victor: “Instance Normalization: The Missing Ingredient for Fast Stylization”. In: *CoRR* (2016) (cit. on p. 17).

[Vas17] Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz and Polosukhin, Illia: “Attention is All you Need”. In: *Advances in Neural Information Processing Systems (NIPS).* Vol. 30. 2017 (cit. on p. 23).

[Vu19] Vu, Thang; Jang, Hyunjun; Pham, Trung X and Yoo, Chang D: “Cascade RPN: Delving into High-Quality Region Proposal Network with Adaptive Convolution”. In: *Conference on Neural Information Processing Systems (NeurIPS).* 2019 (cit. on p. 89).

[Wan18] Wang, Mei and Deng, Weihong: “Deep visual domain adaptation: A survey”. In: *Neurocomputing* 312 (2018), pp. 135–153 (cit. on p. 213).

[Wan19a] Wang, Jinghua and Jiang, Jianmin: “Conditional Coupled Generative Adversarial Networks for Zero-Shot Domain Adaptation”. In: *IEEE International Conference on Computer Vision.* 2019, pp. 3374–3383 (cit. on p. 216).

[Wan19b] Wang, Tao; Zhang, Xiaopeng; Yuan, Li and Feng, Jiashi: “Few-shot Adaptive Faster R-CNN”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7173–7182 (cit. on p. 215).

[Wan19c] Wang, Yu-Xiong; Ramanan, Deva and Hebert, Martial: “Meta-Learning to Detect Rare Objects”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9924–9933 (cit. on pp. 55, 82, 100, 122, 151–153).

[Wan20a] Wang, Xin; Huang, Thomas E.; Darrell, Trevor; Gonzalez, Joseph E. and Yu, Fisher: “Frustratingly Simple Few-Shot Object Detection”. In: *International Conference on Machine Learning*. 2020, pp. 9919–9928 (cit. on pp. 49, 60, 61, 68, 70, 76, 78–84, 86, 87, 89, 96–101, 118, 119, 121, 122, 131, 132, 134, 150–153, 157).

[Wan20b] Wang, Yaqing; Yao, Quanming; Kwok, James T and Ni, Lionel M: “Generalizing From a Few Examples: A Survey on Few-Shot Learning”. In: vol. 53. 3. ACM New York, NY, USA, 2020, pp. 1–34 (cit. on pp. 46, 56).

[Wen20] Wen, Bowen; Mitash, Chaitanya; Ren, Baozhang and Bekris, Kostas E: “se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains”. In: *IEEE International Conference on Intelligent Robots and Systems*. 2020, pp. 10367–10373 (cit. on p. 215).

[Wir19] Wirges, Sascha; Reith-Braun, Marcel; Lauer, Martin and Stiller, Christoph: “Capturing Object Detection Uncertainty in Multi-Layer Grid Maps”. In: *IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 1520–1526 (cit. on p. 71).

[Woo18] Woo, Sanghyun; Park, Jongchan; Lee, Joon-Young and Kweon, In So: “CBAM: Convolutional Block Attention Module”. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 3–19 (cit. on p. 92).

[Wu18] Wu, Yuxin and He, Kaiming: “Group Normalization”. In: *Proceedings of the IEEE/CVF Proceedings of the IEEE/CVF European Conference on Computer Vision (ECCV) (ECCV).* 2018, pp. 3–19 (cit. on p. 17).

[Wu20a] Wu, Jiaxi; Liu, Songtao; Huang, Di and Wang, Yunhong: “Multi-Scale Positive Sample Refinement for Few-Shot Object Detection”. In: *Proceedings of the IEEE/CVF European Conference on Computer Vision (ECCV).* 2020, pp. 456–472 (cit. on pp. 50, 78, 79, 81, 82, 97, 99, 100, 119, 121, 122, 145, 150–153).

[Wu20b] Wu, Yue; Chen, Yinpeng; Yuan, Lu; Liu, Zicheng; Wang, Lijuan; Li, Hongzhi and Fu, Yun: “Rethinking Classification and Localization for Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2020 (cit. on p. 143).

[Xia20] Xiao, Yang and Marlet, Renaud: “Few-Shot Object Detection and Viewpoint Estimation for Objects in the Wild”. In: *Proceedings of the IEEE/CVF European Conference on Computer Vision (ECCV).* 2020, pp. 192–210 (cit. on pp. 58, 60, 78, 79, 81, 82, 97, 99, 100, 119, 121, 122, 143, 151–153, 157).

[Xu18] Xu, Ju and Zhu, Zhanxing: “Reinforced Continual Learning”. In: *Advances in Neural Information Processing Systems.* 2018, pp. 907–916 (cit. on p. 69).

[Xu20] Xu, Xudong; Zhao, Ruizhi; Chen, Rui and Wang, Zulin: “Distilling Object Detectors with Fine-grained Feature Imitation”. In: *Proceedings of the European Conference on Computer Vision (ECCV).* 2020, pp. 102–118 (cit. on p. 108).

[Xu21] Xu, Mengde; Zhang, Zheng; Hu, Han; Wang, Jianfeng; Wang, Lijuan; Wei, Fangyun; Bai, Xiang and Liu, Zicheng: “End-to-End Semi-Supervised Object Detection with Soft Teacher”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021) (cit. on p. 108).

[Yan19] Yan, Xiaopeng; Chen, Ziliang; Xu, Anni; Wang, Xiaoxi; Liang, Xiaodan and Lin, Liang: "Meta R-CNN: Towards general solver for instance-level low-shot learning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* 2019, pp. 9577–9586 (cit. on pp. 56, 78, 79, 81, 82, 97, 99, 100, 119, 121, 122, 151–153).

[Yin20] Yin, Hongxu; Molchanov, Pavlo; Alvarez, Jose M.; Li, Zhizhong; Mallya, Arun; Hoiem, Derek; Jha, Niraj K. and Kautz, Jan: "Dreaming to Distill: Data-Free Knowledge Transfer via DeepInversion". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2020, pp. 8712–8721 (cit. on pp. 108, 109, 111).

[Zai22] Zaidi, Syed Sahil Abbas; Ansari, Mohammad Samar; Aslam, Asra; Kanwal, Nadia; Asghar, Mamoona and Lee, Brian: "A Survey of Modern Deep Learning based Object Detection Models". In: *Digital Signal Processing* 126 (2022), p. 103514 (cit. on p. 27).

[Zen17] Zenke, Friedemann; Poole, Ben and Ganguli, Surya: "Improved multitask learning through synaptic intelligence". In: *International Conference on Machine Learning.* 2017, pp. 3987–3995 (cit. on pp. 69, 106).

[Zha19] Zhang, Hui; Tian, Yonglin; Wang, Kunfeng; He, Haibo and Wang, Fei-Yue: "Synthetic-to-Real Domain Adaptation for Object Instance Segmentation". In: *International Joint Conference on Neural Networks.* 2019, pp. 1–7 (cit. on p. 215).

[Zha20a] Zhang, Junting; Zhang, Jie; Ghosh, Shalini; Li, Dawei; Tasci, Serafettin; Heck, Larry P.; Zhang, Heming and Kuo, C.-C. Jay: "Class-incremental Learning via Deep Model Consolidation". In: *IEEE Winter Conference on Applications of Computer Vision.* 2020, pp. 1120–1129 (cit. on pp. 69, 106).

[Zha20b] Zhao, Wenshuai; Queralta, Jorge Peña and Westerlund, Tomi: "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: *IEEE Symposium Series on Computational Intelligence*. 2020, pp. 737–744 (cit. on p. 215).

[Zha21] Zhang, Youshan: "A Survey of Unsupervised Domain Adaptation for Visual Recognition". In: *CoRR* abs/2112.06745 (2021) (cit. on p. 215).

[Zho22] Zhou, Kaiyang; Liu, Ziwei; Qiao, Yu; Xiang, Tao and Loy, Chen Change: "Domain Generalization: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2022), pp. 1–20 (cit. on p. 213).

[Zhu19] Zhu, Xinge; Pang, Jiangmiao; Yang, Ceyuan; Shi, Jianping and Lin, Dahua: "Adapting Object Detectors via Selective Cross-Domain Alignment". In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 687–696 (cit. on p. 215).

[Zhu20] Zhuang, Chenfan; Han, Xintong; Huang, Weilin and Scott, Matthew: "iFAN: Image-instance full alignment networks for adaptive object detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 2020, pp. 13122–13129 (cit. on p. 215).

Own publications

[1] Guirguis, Karim; Hendawy, Ahmed; Eskandar, George; Abdelsamad, Mohamed; Kayser, Matthias and Beyerer, Jürgen: “CFA: Constraint-based Finetuning Approach for Generalized Few-Shot Object Detection”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPRW).* 2022, pp. 4039–4049.

[2] Guirguis, Karim; Abdelsamad, Mohamed; Eskandar, George; Hendawy, Ahmed; Kayser, Matthias; Yang, Bin and Beyerer, Juergen: “Towards Discriminative and Transferable One-Stage Few-Shot Object Detectors”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).* 2023, pp. 3760–3769.

[3] Guirguis, Karim; Meier, Johannes; Eskandar, George; Kayser, Matthias; Yang, Bin and Beyerer, Juergen: “NIFF: Alleviating Forgetting in Generalized Few-Shot Object Detection via Neural Instance Feature Forging”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2023, pp. 24193–24202.

[4] Guirguis, Karim; Eskandar, George; Kayser, Matthias; Yang, Bin and Beyerer, Juergen: “Few-Shot Object Detection in Unseen Domains”. In: *2022 16th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS).* 2022, pp. 98–107.

[5] Eskandar, George; Palaniswamy, Janaranjani; Guirguis, Karim; Somashekar, Barath and Yang, Bin: “HALS: A Height-Aware Lidar Super-Resolution Framework for Autonomous Driving”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPRW).* 2022.

[6] Eskandar, George; Guo, Diandian; Guirguis, Karim and Yang, Bin: “Towards Pragmatic Semantic Image Synthesis for Urban Scenes”. In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–8.

[7] Eskandar, George; Farag, Youssef; Yenamandra, Tarun; Cremers, Daniels; Guirguis, Karim and Yang, Bin: “Urban-StyleGAN: Learning to Generate and Manipulate Images of Urban Scenes”. In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–8.

[8] Eskandar, George; Marsden, Robert A.; Pandiyan, Pavithran; Döbler, Mario; Guirguis, Karim and Yang, Bin: “An Unsupervised Domain Adaptive Approach for Multimodal 2D Object Detection in Adverse Weather Conditions”. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS*. 2022, pp. 10865–10872.

List of Figures

List of Tables

Acronyms

ABD	Always Be Dreaming
AdaPool	Adaptive Pooling
AI	Artificial Intelligence
ANN	Artificial Neural Network
AP	Average Precision
AR	Average Recall
bAP	base Average Precision
bAR	base Average Recall
BN	Batch Normalization
CBAM	Convolutional Block Attention Module
CFA	Constraint-based Finetuning Approach
CFCE	Contrastive Foreground-Class Embedding
CL	Continual Learning
CNN	Convolutional Neural Network

CPU	Central Processing Unit
CV	Computer Vision
DA	Domain Adaptation
DeFRCN	Decoupled Faster R-CNN
DFKD	Data-Free Knowledge Distillation
DG	Domain Generalization
DL	Deep Learning
DR	Domain Randomization
EWC	Elastic Weight Consolidation
FCNN	Fully Connected Neural Network
FIM	Fisher Information Matrix
FLOPS	Floating Point Operations Per Second
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
FSL	Few-Shot Learning
FSOD	Few-Shot Object Detection
FSRN	Few-Shot RetinaNet

GAP	Global Average Pooling
GDL	Gradient Decoupling Layer
G-FSOD	Generalized Few-Shot Object Detection
GN	Group Normalization
GP	Gaussian Prototyping
GPU	Graphics Processintg Unit
IN	Instance Normalization
IoU	Intersection over Union
KD	Knowledge Distillation
KL	Kullback–Leibler
LLM	Large Language Model
LN	Layer Normalization
MAE	Mean Absolute Error
mAP	mean Average Precision
mAR	mean Average Recall
MAS	Memory Aware Synapses
MDTS	Mixed Domain Training Strategy
mEWC	mean Elastic Weight Consolidation

MHSA	Multi-Head Self-Attention
MI	Model Inversion
ML	Machine Learning
MLP	Multi Layer Perceptron
MPSR	Multi-Scale Positive Samples Refinement
MS-COCO	Microsoft Common Objects in Context
MSDA	Multi-Scale Data Augmentation
MSE	Mean Squared Error
MSF	Multi-Scale Fusion
MWST	Multi-Way Support Training
nAP	novel Average Precision
nAR	novel Average Recall
NIFF	Neural Instance Feature Forging
NLP	Natural Language Processing
NMS	Non-Maximum Suppression
NN	Neural Network
OD	Object Detection
OHEM	Online Hard Example Mining

PASCAL-VOC	PASCAL Visual Object Classes
PReLU	Parametric Rectified Linear Unit
PT	Precision Transferability
QP	Quadratic Programming
R-CNN	Region Convolutional Neural Network
ReLU	Rectified Linear Unit
ResNet	Residual Neural Network
RF	Receptive Field
RGB	Red-Green-Blue
RoI	Region of Interest
RPN	Region Proposal Network
RT	Recall Transferability
SE	Squeeze-and-Excitation
SGD	Stochastic Gradient Descent
SS	Selective Search
SVM	Support Vector Machine
TN	True Negative

TP	True Positive
TPU	Tensor Processing Unit
TSNE	T-distributed Stochastic Neighbor Embedding
UDA	Unsupervised Domain Adaptation
UPPR	Uncertainty-based Progressive Proposal Refinement
ViT	Vision Transformer
ViTD	Vision Transformer Detector
YOLO	You Only Look Once
ZDA	Zero-Shot Domain Adaptation
ZDA-FSOD	Zero-Shot Domain Adaptive FSOD

A Derivations

A.1 Analytical Proof of CFA

In this section, we mathematically derive the update rules for the proposed CFA method. We formulate our objective function as follows:

$$\begin{aligned}
\text{minimize}_{\tilde{\boldsymbol{g}}_n,\tilde{\boldsymbol{g}}_b} \quad & \frac{1}{2}||\boldsymbol{g}_n - \tilde{\boldsymbol{g}}_n||_2^2 + \frac{1}{2}||\boldsymbol{g}_b - \tilde{\boldsymbol{g}}_b||_2^2 \\
\text{subject to} \quad & \tilde{\boldsymbol{g}}_n^\top \boldsymbol{g}_b \geq 0, \\
& \tilde{\boldsymbol{g}}_b^\top \boldsymbol{g}_n \geq 0,
\end{aligned} \tag{A.1}$$

where $\boldsymbol{g}_n$ and $\boldsymbol{g}_b$ represent the proposed gradient update for the novel and base task, respectively. $\tilde{\boldsymbol{g}}_n$ and $\tilde{\boldsymbol{g}}_b$ denote the projected gradient update for the novel and base task, respectively. If both constraints are satisfied, the update rule will be the average of $\boldsymbol{g}_n$ and $\boldsymbol{g}_b$. Otherwise, we solve the constrained optimization problem using the method of Lagrange multipliers.

First, we reformulate the problem in the standard form as follows:

$$\begin{aligned}
\text{minimize}_{\boldsymbol{z}_n,\boldsymbol{z}_b} \quad & \frac{1}{2}\boldsymbol{z}_n^\top \boldsymbol{z}_n - \boldsymbol{g}_n^\top \boldsymbol{z}_n + \frac{1}{2}\boldsymbol{z}_b^\top \boldsymbol{z}_b - \boldsymbol{g}_b^\top \boldsymbol{z}_b \\
\text{subject to} \quad & -\boldsymbol{z}_n^\top \boldsymbol{g}_b \leq 0, \\
& -\boldsymbol{z}_b^\top \boldsymbol{g}_n \leq 0,
\end{aligned} \tag{A.2}$$

where $\tilde{\boldsymbol{g}}_b$ and $\tilde{\boldsymbol{g}}_n$ are denoted as $\boldsymbol{z}_b$ and $\boldsymbol{z}_n$, respectively. We ignore the constant terms $\boldsymbol{g}_n^\top \boldsymbol{g}_n$ and $\boldsymbol{g}_b^\top \boldsymbol{g}_b$. In addition, the sign of the inequality constraints is changed. Then, the Lagrangian can be formulated as:

$$\begin{aligned}\mathcal{L}(\boldsymbol{z}_n, \boldsymbol{z}_b, \alpha_1, \alpha_2) =& \frac{1}{2}\boldsymbol{z}_n^\top \boldsymbol{z}_n - \boldsymbol{g}_n^\top \boldsymbol{z}_n - \alpha_1 \boldsymbol{z}_n^\top \boldsymbol{g}_b \\ &+ \frac{1}{2}\boldsymbol{z}_b^\top \boldsymbol{z}_b - \boldsymbol{g}_b^\top \boldsymbol{z}_b - \alpha_2 \boldsymbol{z}_b^\top \boldsymbol{g}_n,\end{aligned} \tag{A.3}$$

where α_1 and α_2 are the dual variables. To find the solution of the primal variables $\boldsymbol{z}_n^*$ and $\boldsymbol{z}_b^*$, we need to find the lower bound solution of the primal problem by computing the solution of the dual problem:

$$\theta_{\mathcal{D}}(\alpha_1, \alpha_2) = \min_{\boldsymbol{z}_n, \boldsymbol{z}_b} \mathcal{L}(\boldsymbol{z}_n, \boldsymbol{z}_b, \alpha_1, \alpha_2). \tag{A.4}$$

We find $\boldsymbol{z}_n^*$ and $\boldsymbol{z}_b^*$ as a function of dual variables α_1 and α_1, respectively, by minimizing the Lagrangian $\mathcal{L}(\boldsymbol{z}_n, \boldsymbol{z}_b, \alpha_1, \alpha_2)$. This is achieved by setting its derivatives w.r.t $\boldsymbol{z}_n$ and $\boldsymbol{z}_b$ to zero,

$$\begin{aligned}\nabla_{\boldsymbol{z}_n}\mathcal{L}(\boldsymbol{z}_n, \boldsymbol{z}_b, \alpha_1, \alpha_2) &= 0, \\ \boldsymbol{z}_n^* &= \boldsymbol{g}_n + \alpha_1 \boldsymbol{g}_b,\end{aligned} \tag{A.5}$$

$$\begin{aligned}\nabla_{\boldsymbol{z}_b}\mathcal{L}(\boldsymbol{z}_n, \boldsymbol{z}_b, \alpha_1, \alpha_2) &= 0, \\ \boldsymbol{z}_b^* &= \boldsymbol{g}_b + \alpha_2 \boldsymbol{g}_n.\end{aligned} \tag{A.6}$$

Next, we can find the solution of the primal variables by solving the dual problem. We substitute eq. (A.5) and eq. (A.6) in eq. (A.4). Now, the dual problem can be rewritten as:

$$\begin{aligned}\theta_{\mathcal{D}}(\alpha_1, \alpha_2) &= \frac{1}{2}(\boldsymbol{g}_n^\top \boldsymbol{g}_n + 2\alpha_1 \boldsymbol{g}_n^\top \boldsymbol{g}_b + \alpha_1^2 \boldsymbol{g}_b^\top \boldsymbol{g}_b) \\ &- \boldsymbol{g}_n^\top \boldsymbol{g}_n - 2\alpha_1 \boldsymbol{g}_n^\top \boldsymbol{g}_b - \alpha_1^2 \boldsymbol{g}_b^\top \boldsymbol{g}_b \\ &+ \frac{1}{2}(\boldsymbol{g}_b^\top \boldsymbol{g}_b + 2\alpha_2 \boldsymbol{g}_b^\top \boldsymbol{g}_n + \alpha_2^2 \boldsymbol{g}_n^\top \boldsymbol{g}_n) \\ &- \boldsymbol{g}_b^\top \boldsymbol{g}_b - 2\alpha_2 \boldsymbol{g}_b^\top \boldsymbol{g}_n - \alpha_2^2 \boldsymbol{g}_n^\top \boldsymbol{g}_n \\ &= -\frac{1}{2}\boldsymbol{g}_n^\top \boldsymbol{g}_n - \alpha_1 \boldsymbol{g}_n^\top \boldsymbol{g}_b - \frac{1}{2}\alpha_1^2 \boldsymbol{g}_b^\top \boldsymbol{g}_b \\ &- \frac{1}{2}\boldsymbol{g}_b^\top \boldsymbol{g}_b - \alpha_2 \boldsymbol{g}_b^\top \boldsymbol{g}_n - \frac{1}{2}\alpha_2^2 \boldsymbol{g}_n^\top \boldsymbol{g}_n.\end{aligned}$$

Next, we find the solution α_1^* and α_2^* of dual problem as follows:

$$\begin{aligned}\nabla_{\alpha_1}\theta_{\mathcal{D}}(\alpha_1, \alpha_2) &= 0, \\ \alpha_1^* &= -\frac{\boldsymbol{g}_n^\top \boldsymbol{g}_b}{\boldsymbol{g}_b^\top \boldsymbol{g}_b},\end{aligned} \tag{A.7}$$

$$\begin{aligned}\nabla_{\alpha_2}\theta_{\mathcal{D}}(\alpha_1, \alpha_2) &= 0, \\ \alpha_2^* &= -\frac{\boldsymbol{g}_b^\top \boldsymbol{g}_n}{\boldsymbol{g}_n^\top \boldsymbol{g}_n}.\end{aligned} \tag{A.8}$$

Given the solutions of the dual problem, we can find closed-form solutions of $\tilde{\boldsymbol{g}}_n$ and $\tilde{\boldsymbol{g}}_b$ by substituting the dual solutions α_1^* eq. (A.7) and α_2^* eq. (A.8) in eq. (A.5) and eq. (A.6), respectively:

$$\boldsymbol{z}_n^* = \boldsymbol{g}_n - \frac{\boldsymbol{g}_n^\top \boldsymbol{g}_b}{\boldsymbol{g}_b^\top \boldsymbol{g}_b}\boldsymbol{g}_b = \tilde{\boldsymbol{g}}_n, \tag{A.9}$$

$$\boldsymbol{z}_b^* = \boldsymbol{g}_b - \frac{\boldsymbol{g}_b^\top \boldsymbol{g}_n}{\boldsymbol{g}_n^\top \boldsymbol{g}_n}\boldsymbol{g}_n = \tilde{\boldsymbol{g}}_b. \tag{A.10}$$

After finding the closed-form solution, a single update rule can be realized as:

$$\tilde{\boldsymbol{g}} = \frac{\tilde{\boldsymbol{g}}_n + \tilde{\boldsymbol{g}}_b}{2}. \tag{A.11}$$

B Domain Generalization for FSOD

In the previous chapters, the FSOD and G-FSOD approaches discussed operate under the assumption that the data originates from the same source domain, meaning the data belongs to the same distribution. However, in real-life scenarios, it is common for models to be deployed in varied operating conditions where the test-time domain differs from the trained domain reflected in a distribution shift. For example, suppose a detection model is trained on images captured by a specific sensor on a production line and subsequently deployed on another production line with dissimilar sensors, lighting conditions, backgrounds, or poses. In that case, the detection model will likely fail due to the resulting domain shift. Nevertheless, collecting and annotating datasets for each new task and domain is costly and time-consuming, and acquiring sufficient training data may not always be feasible.

Domain Generalization (DG) [Zho22], a sub-discipline of transfer learning, enables models to acquire generalized representations capable of handling variations across multiple domains. In contrast to DA [Wan18], which focuses on adapting models from a specific source domain to a target domain, DG aims to develop models that perform well across numerous unseen domains without domain-specific training.

The significance of DG arises from several factors [Zho22]:

- Rather than relying on labeled data from each potential target domain, DG empowers models to leverage knowledge gained from diverse source domains. Consequently, this reduces the cost and time involved in data acquisition and enables the utilization of various models in domains where obtaining labeled data proves challenging, such as industrial robots.

- Improves the resilience and dependability of the models. Exposing models to various distributional shifts during training can capture and generalize underlying patterns and concepts that remain consistent across different environments. As a result, these models become better equipped to handle variations in data distribution, noise, and other factors encountered during deployment.
- Facilitates knowledge transfer across diverse domains. Models trained using DG can serve as a foundation for various downstream tasks, significantly saving time and resources. For instance, a model trained on various indoor scenes could be employed in tasks such as object detection, even in previously unseen environments.

This chapter presents and tackles the challenging task of ZDA-FSOD. Specifically, the assumption is that neither images nor labels of the novel classes in the target domain are available during training. A two-fold approach is proposed for solving the domain gap. Firstly, a meta-training paradigm is leveraged, where the domain shift is learned on the base classes, and the domain knowledge is subsequently transferred to the novel classes. Secondly, various data augmentation techniques are proposed to account for all possible domain-specific information within the few shots of novel classes. To ensure that the network is constrained to encode domain-agnostic class-specific representations only, a contrastive loss is proposed. This contrastive loss aims to maximize the mutual information between foreground proposals and class embeddings, while reducing the network bias towards the background information from the target domain. Experimental evaluations conducted on datasets with considerable domain shifts demonstrate that the proposed approach successfully alleviates the domain gap considerably without utilizing labels or images of novel categories from the target domain.

B.1 Literature Review

The DA methods with limited or no target domain data can be categorized as follows: Unsupervised Domain Adaptation (UDA), DR, and Zero-Shot Domain Adaptation (ZDA).

B.1.1 Unsupervised Domain Adaptation

UDA [Zha21] thrives when labeled data exists solely in the source domain, while the target domain lacks corresponding labeled samples. The main objective of UDA is to obtain domain-agnostic knowledge by leveraging the labeled data in the source domain. This knowledge aims to capture the common underlying representations across diverse domains and generalize to the target domain without ground-truth labels.

UDA methods employ either feature-level adaptation in latent space or pixel-level adaptation through image-to-image translation techniques. Domain Adaptive Faster R-CNN (DA-FRCNN) [Che18b] combines Faster R-CNN [Ren15] with adversarial training. DA-FRCNN aims to align both the image and instance distributions across domains while utilizing consistency regularization to learn a domain-invariant RPN [Che18b]. Subsequently, several adversarial-based methods have been proposed by other researchers [Zhu19, Wan19b, Sai19, He19, Zha19, Zhu20, Che20a], building upon this work.

B.1.2 Domain Randomization

Alternatively, DR [Zha20b] aims to acquire domain-invariant features by generating images with randomized attributes such as illumination, pose, and background, simulating real-world distributions. In recent years, DR methods have been introduced in robotics applications, including 6D object detection [Sun18], 6D object tracking [Wen20], object localization [Tob17], person detection [Lin20], and segmentation [Dan19]. However, these DR methods typically rely on using a blender or a game engine to generate semi-realistic images, which can be ineffective in terms of time and cost.

B.1.3 Zero-Shot Domain Adaptation

ZDA [Kod15, Pen18, Wan19a] tackles the more challenging scenario where no labeled data from the target domain is accessible during training. The objective of ZDA is to enable the adaptation of a model, originally trained on a source domain, to effectively perform on a target domain without relying on any labeled samples from the target domain. To address this challenge, ZDA approaches utilize auxiliary sources of information, such as domain-specific attributes or additional data, to bridge the gap between the source and target domains.

B.2 Datasets

Since ZDA-FSOD is a new task, it necessitates using datasets encompassing different domains to assess the performance of the proposed approach. In light of this, datasets were carefully chosen to ensure the availability of distinct domains, enabling a comprehensive evaluation of the effectiveness of the proposed method in addressing the challenges posed by ZDA-FSOD.

B.2.1 T-Less

T-Less [Hod17] dataset was designed for the OD and pose estimation tasks in industrial settings. Specifically, the T-Less [Hod17] dataset consists of 30 industrial objects without any notable texture, distinct color, or reflectance features. Two different versions of T-Less are available: synthetic rendered images, comprising a total of 50 scenes with approximately 1000 images each, and real images captured from 20 different scenes. The synthetic and real datasets are used as source and target domain data, respectively. It is important to note that the RGB data alone is employed in all experiments.

To adapt the T-Less dataset for FSOD, train and test splits are proposed for both the base and novel classes. The dataset is divided into 19 base classes and 11 novel classes. Additionally, the 20 scenes containing real images are split into

8 training scenes $(2, 3, 5, 6, 7, 9, 11, 12)$, which mainly feature the base classes, while the remaining 12 scenes contain both training and testing scenes for the novel classes. The inference is performed on unseen classes, specifically the 11 novel objects. Inference results are reported using $K = 5$,10-shot settings.

B.2.2 Ex-Dark

The ExDark [Loh19] dataset is an adaptation of the widely-utilized PASCAL-VOC dataset [Eve10], tailored to tackle the difficulty of object detection in extremely poor illumination conditions. This modification involves reducing the brightness levels of the images to emulate circumstances with minimal illumination, where objects may be barely detectable.

ExDark [Loh19] consists of 12 classes, with 10 classes overlapping with the PASCAL-VOC dataset. To adapt for FSOD, the dataset is partitioned into 7 base classes and 5 novel classes. The novel classes align with the classes present in the PASCAL-VOC dataset, while the remaining classes are categorized as base classes. For testing, only the novel classes within the ExDark test set are used. The reported results are based on the $K = 5$,10-shot settings.

B.3 Zero-Shot Domain Adaptive FSOD

B.3.1 Problem Formulation

In ZDA, annotated abundant base target domain data is assumed available. Formally, $\boldsymbol{X}_B = \{\boldsymbol{X}_B^S, \boldsymbol{X}_B^T\}$ and $\boldsymbol{Y}_B = \{\boldsymbol{Y}_B^S, \boldsymbol{Y}_B^T\}$, where $\boldsymbol{X}_B$ are the base training examples with the corresponding annotations $\boldsymbol{Y}_B$. $\boldsymbol{X}^S$ and $\boldsymbol{X}^T$ denote the source and target domain data samples, respectively. Similarly, the $\boldsymbol{Y}^S$ and $\boldsymbol{Y}^T$ are source and target domain annotations, respectively.

In ZDA-FSOD, only a few shots of source domain data are available, represented by the sets $\boldsymbol{X}_N = \{\boldsymbol{X}_N^S\}$ and $\boldsymbol{Y}_N = \{\boldsymbol{Y}_N^S\}$. A harsher constraint is imposed in this work, assuming that the target data is either unavailable in the base task

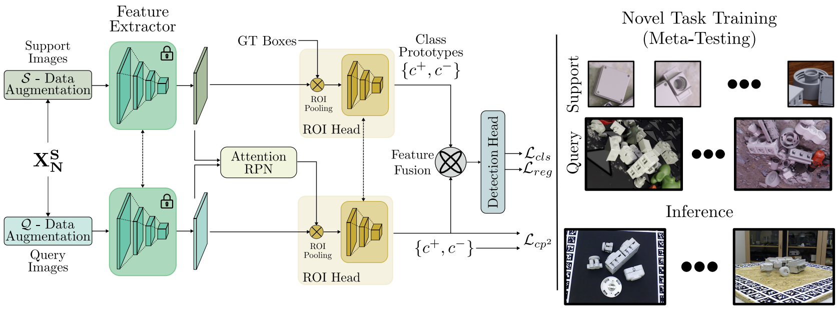


Figure B.1: Left: The framework utilized for the proposed ZDA-FSOD task during the novel task training (meta-testing) phase is represented. The network receives the novel source data $\boldsymbol{X}_N^S$ after applying query and support level augmentations. **Right:** Examples of the support and query images during the meta-testing phase (upper) and the inference stage (lower). The support images used during inference are the same ones utilized during the meta-testing and belong to the source domain.

or accessible only with a few shots. The former setting is more challenging as it lacks any knowledge about the target domain.

B.3.2 Baseline Description

To set up a baseline for the ZDA-FSOD task, the meta-learning based Attention-RPN model [Fan20], which is based on Faster-RCNN [Ren15]. In the Attention-RPN model, a two-stage meta-detector is employed to detect objects in a query image that belong to the same class as the support images. As previously discussed in Section 3.2.2, the RPN is modified to filter out irrelevant object proposals by combining the query and support features. Additionally, the detection head is altered to integrate the query and support feature embeddings through concatenation, followed by an attention mechanism operating at a global, pixel, and patch level. The choice of Attention-RPN [Fan20] as the baseline is motivated by its contrastive training paradigm that encourages better domain-invariant features. Figure B.1 depicts the baseline architecture and our contributions in the meta-testing phase.

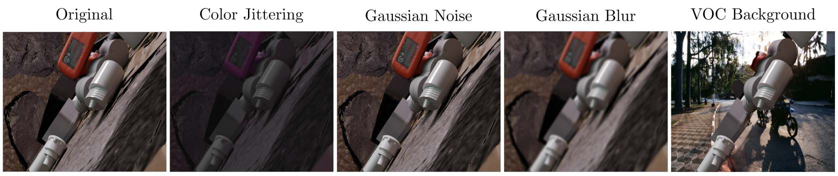


Figure B.2: A depiction of different DAs implemented on a query image originating from a T-Less source domain image to perform DR.

B.3.3 Data Augmentation Schemes

Given the supervised learning of novel classes, the model is highly susceptible to overfitting the domain information due to data scarcity. To address this issue, various data augmentation techniques are proposed during the meta-testing phase. In contrast to previous domain randomization approaches [Tob17, Sun18, Dan19, Lin20], using a graphics engine to generate multiple poses and lighting conditions is avoided. Instead, pixel-level and feature-level augmentations are leveraged, enabling the proposed method to handle any domain gap without being limited to simulation-to-real applications. The following augmentations are proposed:

- Color Jittering: To address diverse point-wise lighting variations, color jittering is randomly applied by modifying the brightness, contrast, hue, and saturation of both the query and support images. This measure aids in reducing the network susceptibility to overfitting the source domain colors while promoting acquiring more domain-invariant features.
- Gaussian Blur: To mitigate the impact of noisy low-light captured images or out-of-focus frames, the resulting images can confuse the detector. To address this issue, it is proposed to randomly apply Gaussian blur with random kernel sizes and standard deviations to both the support and query images. This measure aims to account for such distortions and enhance the robustness of the detector.
- Gaussian Noise: The limited novel examples may lead to a highly noisy learning signal, harming the quality of learned representations. To address this issue, Gaussian noise with different small standard

deviation values is randomly added to both the support and query images. This augmentation draws inspiration from the work on adversarial attacks on neural networks [Goo15], where it was observed that even small imperceptible perturbations could deceive the network decision-making and shift a sample away from the classifier decision boundary. The objective is to encourage the network to be less sensitive to the absolute pixel values of the few available support and query images, thereby reducing the number of false positives and negatives. Consequently, the enhanced network robustness against pixel perturbations leads to higher domain transferability.

- Background Augmentation: Overfitting of the backgrounds in training images, which are not transferable across domains, can occur when training on abundant source domain data. However, foreground object classes should have consistent feature representations across domains. To avoid domain confusion arising from different backgrounds during test-time, the proposal is to extract the objects from the given source query images using their masks or bounding boxes and embed them onto real background images from the PASCAL-VOC datasets, similar to the approach in [Sun18]. Unlike applying this augmentation to a single object, it is applied to the query image containing multiple foreground objects. Specifically, an image is randomly sampled from the PASCAL-VOC dataset and resized to match the resolution of the source query image. The extracted objects are then placed at the same location in the PASCAL-VOC image to keep the bounding box labels unchanged.

The abovementioned data augmentations are visually depicted in Figure B.2.

B.3.4 Mixed Domain Training Strategy

To mitigate the risk of overfitting on domain-specific features, the model should encounter multiple domains during meta-training. Due to the absence of target domain data in the base task, data augmentations outlined in Section B.3.3 are applied to the source data. Consequently, the training set is represented as $\{(\boldsymbol{X}_B^S)^t, \boldsymbol{Y}_B^S\}$, where $t \in \boldsymbol{T}$ denotes a transform. By incorporating these augmentations, the model is compelled to observe more than one domain at this stage, as the feature extractor remains frozen in the subsequent stage and subsequently transfers the acquired knowledge from the base task to the novel tasks.

In scenarios where few-shots of target domain data are available in the base task, the meta-learning paradigm is leveraged by redefining the episodic base tasks during the meta-training phase. Instead of exclusively learning the base tasks using source domain data, the base source data is augmented with a few-shots from the target domain, resulting in a combined dataset $\{(\boldsymbol{X}_B^S, \boldsymbol{Y}_B^S), (\boldsymbol{X}_B^T, \boldsymbol{Y}_B^T)\}$ to extract cross-domain knowledge. This meta-training paradigm is denoted by MDTS.

To enable the MDTS at both the query and support levels, each episode in the meta-training involves query images drawn from either the source or target domain, while the support images may be sampled from a single domain or a mixture of both. The proposed MDTS facilitates the observation of visual cues in the query image that are less domain-dependent. Subsequently, the acquired domain knowledge is transferred to the novel task by freezing the feature extractor and finetuning the RoI and detection heads.

B.3.5 Contrastive Foreground-Class Embedding Loss

Learning discriminative features across different domains poses a significant challenge for novel classes with limited data. One common drawback observed in such scenarios is increased confusion, leading to higher false positives and negatives, especially for classes with similar appearances or between foreground and background in the new domain. A crucial requirement for a robust feature representation is its ability to encode class-specific domain-agnostic information, indicating sensitivity only to the shape of the foreground object.

To enforce semantic consistency between foreground and background embeddings, a CFCE loss is proposed. This loss functions by comparing the foreground proposals from the query image with both the positive and negative support class embeddings. Contrastive losses have been successfully

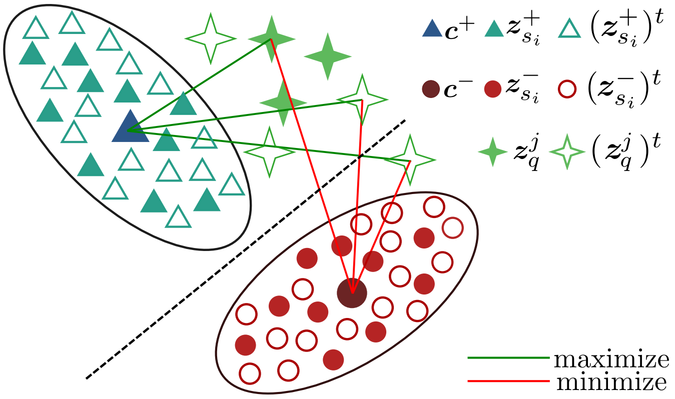


Figure B.3: A depiction of the proposed CFCE contrastive loss is presented. The primary objective is to maximize the similarity between the features of the foreground query proposals, denoted as $\boldsymbol{z}_q^j$, and the positive support class prototype $\boldsymbol{c}^+$. Simultaneously, the goal is to minimize the similarity between $\boldsymbol{z}_q^j$ and the negative support prototype $\boldsymbol{c}^-$. Furthermore, the utilization of augmented features, represented as $(.)^t$, complements the CFCE loss, promoting the learning of robust cross-domain features.

employed to map two different views of the same scene to a similar point in the representation space [Sch15, Koc15, Che20b, Kho20]. More specifically, the proposed CFCE loss seeks to maximize the mutual information between instance-level features in the query and class embeddings from the support images, regardless of their augmentations. The objective is to bias the decision boundary of the detector towards the topology and semantics of the objects rather than being influenced by domain information.

Formally, foreground features of the q^{th} query proposals are denoted as $\{\boldsymbol{z}_q^j\}_{j=1}^{P_f}$, where P_f is the number of foreground proposals. Let $\{\boldsymbol{z}_{s_i}^+\}_{i=1}^{K}$ represent positive support features that are present in the query image while $\{\boldsymbol{z}_{s_i}^-\}_{i=1}^{K}$ are the negative support features that are randomly sampled class which are absent in the query image. s_i denotes the i^{th} support image and K is the total number of support shots per class. $\boldsymbol{c}^+$ and $\boldsymbol{c}^-$ are the positive and negative class embeddings and are computed by averaging $\{\boldsymbol{z}_{s_i}^+\}_{i=1}^{K}$ and $\{\boldsymbol{z}_{s_i}^-\}_{i=1}^{K}$, respectively.

The objective of the CFCE loss is to ensure that the foreground query proposals are near $\boldsymbol{c}^+$ while simultaneously pushing them further away from $\boldsymbol{c}^-$. For this purpose, a triplet margin contrastive loss is employed, which aims to reduce the distance between $\boldsymbol{z}_q^j$ and $\boldsymbol{c}^+$ while increasing the distance between $\boldsymbol{z}_q^j$ and $\boldsymbol{c}^-$ beyond a hyperparameter margin m. This triplet margin loss allows for a more flexible feature space that accommodates the inter-class distances [Mus20]. The cosine similarity distance is preferred over the Euclidean distance as it considers the angle between two similar data objects whose features have a large magnitude in the feature space. The CFCE loss is mathematically expressed as follows:

$$\mathcal{L}_{CFCE} = \frac{1}{P_f}\sum_{j}^{P_f} \max\left(d(\boldsymbol{z}_q^j, \boldsymbol{c}^+) - d(\boldsymbol{z}_q^j, \boldsymbol{c}^-) + m, 0\right), \tag{B.1}$$

$$d(\boldsymbol{v}_1, \boldsymbol{v}_2) = \frac{\boldsymbol{v}_1^T \boldsymbol{v}_2}{\parallel \boldsymbol{v}_1 \parallel \parallel \boldsymbol{v}_2 \parallel}. \tag{B.2}$$

The proposed CFCE loss is presented in Figure B.3.

B.3.6 Feature-Level Class Embedding Augmentations

The direct averaging of K-support shot features for computing class embeddings may not accurately represent the true class embedding distribution, especially when support shots are scarce or have a high standard deviation. To address these concerns, a data augmentation scheme is proposed at the support feature level. Specifically, assuming that the support feature representation follows a Gaussian distribution, the mean feature $\bar{\boldsymbol{f}}$ is computed over the K-shots, along with their standard deviation, $\boldsymbol{\sigma}_f$. During each iteration in the meta-testing phase, a latent vector is sampled from the Gaussian distribution $\mathcal{N}(\bar{\boldsymbol{f}}, \boldsymbol{\sigma}_f^2)$ and serves as the class embedding. This feature-level augmentation strategy is employed during inference to enhance the class embeddings.

The contrastive losses and data augmentations complement each other by introducing perturbations in the feature space, allowing the network to explore more examples. The data augmentation may shift an object beyond the decision boundary, mimicking the effect of a new domain during inference. However, in experiments, it is observed that strong augmentations can sometimes destabilize training on the novel task. Conversely, the contrastive losses help attract these distant features together by enforcing semantic consistency, resulting in smoother training and higher average precision.

B.4 Experimental Evaluations

The evaluation of the proposed ZDA-FSOD necessitates a dataset encompassing multiple domains. Given the specific nature of this problem, the most appropriate publicly available datasets for evaluation are T-LESS [Hod17], PASCAL-VOC [Eve10], and ExDark [Loh19]. The investigation focuses on two domain gaps: (T-Less Synthetic $\rightarrow$ Real) and (PASCAL-VOC $\rightarrow$ ExDark).

B.4.1 Implementation Details

For a fair quantitative comparison, all models are meta-trained on the base task for 5 epochs using an SGD optimizer with the default parameters as specified in Attention-RPN [Fan20], and a batch size of 8. The base learning rate is set to 0.004 for the first 3 epochs and 0.0004 for the subsequent 2 epochs. During meta-testing, 6k iterations are performed with a learning rate of 0.001. The shorter side of the query image is resized to 600 pixels, and the longer side is resized to 1000. Additionally, each support image is cropped around the target object with a 16-pixel image context, zero-padded, and then resized to 320×320.

Table B.1: Results on T-Less Synthetic → Real domain gap. MDTS denotes the mixed domain training strategy (source + few-shot target data) introduced in Section B.3.4. MDTS-Aug denotes source data augmented with DR.

Methods	Meta-Training Data Domain	Meta-Testing Data Domain	5-Shot				10-Shot			
			AP	AP50	AP75	AR	AP	AP50	AP75	AR
Baselines	(a) Source	Source	1.7	3.0	1.8	4.1	6.1	10.1	6.5	13.3
	(b) Target		6.8	12.6	6.9	19.9	10.2	16.0	11.6	21.9
	(c) MDTS		12.9	22.1	13.0	35.2	23.5	34.0	25.7	49.0
Ours	Source	Source	6.5	11.0	6.7	32.8	17.3	27.2	18.9	44.2
	MDTS-Aug		10.1	17.7	10.3	21.5	26.3	39.8	29.7	49.2
	MDTS		**17.4**	**26.5**	**18.9**	**39.9**	**31.2**	**45.8**	**34.0**	**61.0**
Oracle	Target	Target	37.5	50.8	42.5	53.9	51.7	68.4	58.7	62.7

B.4.2 Baseline Models

The performance of the proposed model is evaluated against four baselines: (a) Attention-RPN [Fan20] meta-trained solely on source data, (b) Attention-RPN [Fan20] meta-trained solely on target data, and (c) Attention-RPN [Fan20] meta-trained on both source data and few-shot target data. All models undergo fine-tuning on novel classes from the source domain and are tested on novel classes from the target domain. It is important to note that baseline (b) assumes abundant target data in the base task, which provides it with an advantage over the proposed model.

Comparison with other FSOD models is not conducted in this task, as they possess different model capacities and report diverse performances on the same datasets. Such a comparison in a DA setting would lead to an unfair evaluation. Although a few works [Rah20, Kan18] have presented cross-dataset results, specifically from MS-COCO to PASCAL-VOC, these methods reported the results as an experiment to assess generalization on more novel classes, rather than as a direct solution to ZDA. Moreover, there is no apparent domain gap between MS-COCO and PASCAL-VOC.

Table B.2: Results on PASCAL-VOC → ExDark domain gap. MDTS-Aug denotes source data augmented with DR. Note that no PASCAL-VOC background augmentation is used since PASCAL-VOC is the source domain dataset.

Methods	Meta-Training	Meta-Testing	5-Shot				10-Shot			
	Data Domain	Data Domain	AP	AP50	AP75	AR	AP	AP50	AP75	AR
Baselines	(a) Source	Source	10.1	26.4	5.4	16.9	12.1	32.4	6.0	20.0
	(b) Target		5.3	16.1	1.8	11.1	6.9	21.9	2.9	14.1
	(c) MDTS		11.8	29.9	6.7	21.3	13.5	34.3	7.7	21.8
Ours	Source	Source	11.0	28.6	6.5	20.7	13.6	34.7	9.4	23.6
	MDTS-Aug		**13.0**	**32.6**	**7.4**	**23.5**	**14.2**	**36.7**	**9.6**	**25.0**
	MDTS		11.9	30.6	7.1	21.5	13.9	35.5	9.0	23.4
Oracle	Target	Target	10.0	24.7	7.2	21.2	14.2	35.4	9.4	25.5

B.4.3 Comparison Results

Three variants of the proposed model are tested. The first variant is meta-trained solely on source data, the second one is trained on source data augmented with proposed DR techniques, and the final variant uses MDTS with source and few-shot target data. The three variants are meta-tested with the proposed DR techniques, CFCE loss, and feature-level augmentations. As an oracle (upper bound), an Attention-RPN [Fan20] model meta-trained and meta-tested on actual target data is considered.

Results on T-Less Synthetic → Real

Table B.1 presents the results on the T-Less domain gap for $K = 5$ and $K = 10$-shot settings. It is observed that the source-only (baseline (a)) experiences a significant performance drop in AP and AR, declining from 51.7 to 6.1, and 62.7 to 13.3, respectively, when no DR is applied. On the other hand, MDTS (variant (c)) outperforms meta-training on either domain alone (baseline (b) and (c)). Contrary to expectations, MDTS even surpasses target-only meta-training. This can be attributed to the fact that the model reaches a better optimum when exposed to two different data distributions of the same classes during meta-training.

Additionally, the proposed DR approach can considerably improve the AP and AR in different meta-training settings. When performed on a source-only meta-trained model, DR boosts the AP by 11.2 points under the 10-shot setting (row 5). The model can reach an AP of 26.3, almost 51% of the oracle performance, without seeing any target domain sample in the meta-training and meta-testing phases (row 5 in Table B.1). This variant already surpasses the MDTS training in baseline (c) while seeing less data. For the best performance on $K = 5, 10$-shot settings, 17.4 and 31.2, were reached using mixed domain samples in meta-training and DR in meta-testing (row 6 in Table B.1). Nevertheless, the relative performance increases under the 10-shot settings (60% of the oracle) compared to 5-shot settings (46.4% of the oracle). This is attributed to the sparsity of the latter setting.

Results on PASCAL-VOC $\rightarrow$ ExDark

Table B.2 presents the results on the PASCAL VOC $\rightarrow$ ExDark datasets to examine the generalization of our model on a different domain gap (day $\rightarrow$ night). In contrast to T-Less, it is observed that meta-training on the target domain (b) does not result in any improvement over source-only training (a). The drop in baseline (b) can be attributed to two factors. Firstly, ExDark contains fewer base classes in meta-training compared to PASCAL-VOC. Secondly, the domain gap is considerably more challenging to learn in a few-shot setting than the real-to-synthetic domain gap in T-Less. This is due to some objects being hardly visible in ExDark, leading to confusion with the background by the network. Nonetheless, our models consistently outperform all baselines on this new domain gap, achieving the best performance without utilizing any target domain data during meta-training (row 6). It is worth noting that several models in the table outperform the oracle, which is attributed to the target data (ExDark) containing fewer base classes than the source data (PASCAL-VOC).

Table B.3: The ipact of meta-testing on T-Less data with meta-trained weights on mixed domain samples in the base task with the MDTS.

	Data Configuration		10-Shot Inference			
	Support	Query	AP	AP50	AP75	AR
A	Source Only	Source Only	23.5	34.0	25.7	49.0
B	+ Color Jittering	+ Color Jittering	26.3	38.3	27.9	59.0
C	+ Gaussian Blur	+ Gaussian Blur	26.8	39.9	30.0	58.1
D	+ Gaussian Noise	+ Gaussian Noise	29.3	43.8	31.9	59.2
E		+ VOC Background	30.2	44.3	32.6	57.6
F	+ Feature Noise		30.4	44.8	33.4	**63.1**
G	+ CFCE Loss		**31.2**	**45.8**	**34.0**	61.0

B.4.4 Ablation Experiments

Meta-Testing via DR

Table B.3 presents various experiments conducted to examine the impact of different proposed augmentations during meta-testing. The network used in all experiments is meta-trained on the source and few-shot target domain data. The experiments start from source-only meta-testing (configuration **A**), and the augmentations are incrementally applied with the results reported. First, color jittering in configuration **B** leads to a significant AR improvement of 10 points, indicating a notable decrease in false negatives and an AP improvement of 2.2 points. Introducing Gaussian blur (**C**) further improves the AP by 0.6 points but slightly deteriorates the AR by 0.9 points. Adding Gaussian noise (**D**) enhances the AP by 2.5 points and improves the AR by a surplus of 1.1 points, demonstrating the importance of pixel-level perturbations in enhancing model robustness to simple distortions. Next, in configuration **E**, adding PASCAL-VOC background augmentation on the query images results in an improved AP. Feature-level augmentations in configuration **F** slightly increase the AP, with their main impact reflected in the high AR. Finally, the best result in our experiments is achieved by adding the CFCE loss, boosting the AP by 0.8 points and slightly reducing the AR to 61, which is close to the oracle AR of 62.7. We have also observed that the introduced contrastive loss and

Table B.4: An investigation into cross-domain training settings in both the meta-training and meta-testing phases on the T-Less dataset is conducted. The Attention-RPN [Fan20] serves as the baseline for all experiments. $\checkmark^{few}$ indicates the presence of few-shot samples from the base task in the target domain. The rows highlighted in gray indicate the use of our proposed DR approach.

Meta-Training				Meta-Testing	Inference							
Support		Query			5-shots				10-shots			
Source	Target	Source	Target		AP	AP50	AP75	AR	AP	AP50	AP75	AR
✓		✓		Source Only	1.7	3.0	1.8	4.1	6.1	10.1	6.5	13.3
	✓		✓		6.8	12.6	6.9	19.9	10.2	16.0	11.6	21.9
✓	$\checkmark^{few}$	✓	$\checkmark^{few}$		12.9	22.1	13.0	35.2	23.5	34.0	25.7	49.0
✓	$\checkmark^{few}$	✓	$\checkmark^{few}$		**17.4**	**26.5**	**18.9**	**39.9**	**31.2**	**45.8**	**34.0**	**61.0**
✓		✓		Target Only	38.3	54.7	43.0	56.5	48.7	65.2	55.0	64.6
	✓		✓		37.5	50.8	42.5	53.9	51.7	68.4	58.7	62.7
✓	$\checkmark^{few}$	✓	$\checkmark^{few}$		39.9	54.0	44.6	58.9	48.3	62.4	53.7	66.1
✓	$\checkmark^{few}$	✓	$\checkmark^{few}$		**39.9**	**55.2**	**44.6**	**62.9**	**49.2**	**65.0**	**54.6**	**69.7**

feature-level augmentation stabilize the meta-testing process while reducing the number of false negatives and positives.

Cross-Domain Training

Table B.4 explores different cross-domain settings to analyze existing domain gaps in meta-training and meta-testing. The proposed DR approach is selectively applied in the highlighted rows of Table B.4. The following deductions are made.

Firstly, the domain gap can be significantly reduced by the existence of target-domain samples for novel classes, even in cases where no target-domain samples were seen during meta-training. Secondly, the MDTS in meta-training remains beneficial regardless of the samples seen during meta-testing. Rows $3, 4, 7, 8$ in Table B.4 demonstrate that the model achieves higher AR than the oracle under the 5,10-shot settings, higher AP under the 5-shot setting, and an AP close to the oracle under the 10-shot setting. Thirdly, applying the DR approach during target-only meta-testing improves performance compared to the oracle in 5-shot settings (AR and AP) and 10-shot settings (AR).

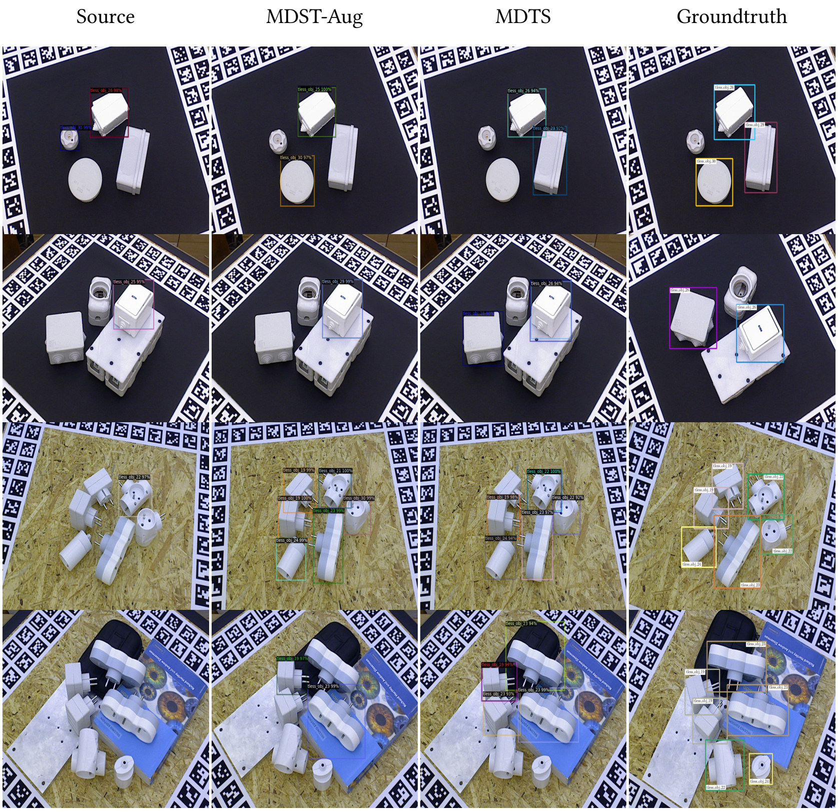


Figure B.4: Qualitative results comprising the three model variants meta-trained on source, MDTS-Aug, and MDTS using the 10-shot setting on the T-Less dataset.

However, it is essential to note that the assumption of having target domain samples during meta-testing may not be valid in many real-world applications, where differences in background, camera sensor, or simulation-to-real domain gaps are common. Additionally, frequent finetuning of the pre-trained model on new domains, such as when installing a new camera or encountering environmental changes, may not be practical.

B.5 Qualitative Results

Qualitative results of the three proposed model variants are depicted in Figure B.4. The results showcase various success and failure scenarios.

B.6 Discussion

This thesis chapter introduces and addresses the new and challenging problem of Zero-Shot Domain Adaptation for Few-Shot Object Detection (ZDA-FSOD). This task involves scenarios where only a limited number of source domain data shots are available, while target data is either completely absent in the base task or accessible only with a few shots.

To tackle this novel problem, the adoption of a meta-learning paradigm led to the proposal of several techniques. One of the key contributions was using Domain Randomization through pixel-level and feature-level augmentations. This approach allowed the proposed method to effectively handle various domain gaps without relying on external tools such as blenders or game engines.

The MDTS was introduced, significantly improving the performance of the meta-trained models. By exposing the network to both source and target domain data during meta-training, better generalization to unseen domains in the meta-testing phase was observed.

A novel contrastive loss called CFCE loss was introduced to ensure semantic consistency between foreground and background embeddings. This loss encourages foreground query proposals to be close to positive class features while being pushed away from negative ones, ensuring a more robust feature representation.

Another important contribution was the feature-level class embedding augmentation. During each iteration in the meta-testing phase, a latent vector was sampled from a Gaussian distribution drawn from the available support features, serving as the class embedding. This feature-level augmentation strategy improved the robustness of the class embeddings during inference.

Extensive experiments on two challenging domain gaps demonstrated the effectiveness of the overall approach. The meta-detector exhibited impressive performance, successfully generalizing to unseen domains even when no target domain data for the novel classes were available during both meta-training and meta-testing.

The proposed method is believed to benefit numerous practical, real-world robotic autonomous systems, and further research is encouraged on tackling domain shifts when learning novel objects in unseen domains.

Lehrstuhl für Interaktive Echtzeitsysteme
Karlsruher Institut für Technologie

Fraunhofer-Institut für Optronik, Systemtechnik und Bildauswertung IOSB Karlsruhe

Artificial Intelligence is transforming the future of manufacturing, driving unparalleled efficiency, cutting costs by up to 20%, and boosting revenue by 10%. At the heart of this transformation lies object detection, an AI powerhouse identifying patterns and anomalies across dynamic production lines.
This book dives into breakthrough methods in Few-Shot Object Detection (FSOD), where models learn new product classes from minimal data, bypassing the need for extensive data labeling. Explore three innovative FSOD frameworks that tackle real-world challenges, like catastrophic forgetting and high computational demands, without compromising accuracy.
Offering industry-changing solutions, this work enables faster knowledge acquisition, reduced labeling efforts, and superior production quality-giving manufacturers a robust edge in a rapidly evolving AI-driven landscape.

ISSN 1866-5934
ISBN 978-3-7315-1389-6

Gedruckt auf FSC-zertifiziertem Papier

ISBN 978-3-7315-1389-6
9 783731 513896