

Recognizing Traffic Signs using Fine-tuning Based Few-Shot Object Detection

Authors

Md. Atiqur Rahman

180041123

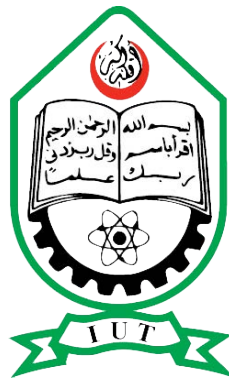
Nahian Ibn Asad

180041136

Md. Mushfiqul Haque Omi

180041140

**A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of
Bachelor of Science in CSE**



**Department of Computer Science and Engineering
Islamic University of Technology
Organization of the Islamic Cooperation (OIC)**

Dhaka, Bangladesh

May, 2023

Recognizing Traffic Signs using Fine-tuning Based Few-Shot Object Detection

Authors

Md. Atiqur Rahman

180041123

Nahian Ibn Asad

180041136

Md. Mushfiqul Haque Omi

180041140

Supervisor

Dr. Md. Hasanul Kabir

Professor

Department of Computer Science and Engineering

Islamic University of Technology

Co-Supervisor

Sabbir Ahmed

Assistant Professor

Department of Computer Science and Engineering

Islamic University of Technology

**A thesis submitted to the Department of CSE
in partial fulfillment of the requirements for the degree of
Bachelor of Science in CSE**



**Department of Computer Science and Engineering
Islamic University of Technology
Organization of the Islamic Cooperation (OIC)**

Dhaka, Bangladesh

May, 2023

Declarations of the Candidates

This is to certify that the work presented in this thesis, titled, “Recognizing Traffic Signs using Fine-tuning Based Few-Shot Object Detection”, is the outcome of the investigation and research carried out by Md. Atiqur Rahman, Nahian Ibn Asad, Md. Mushfiqul Haque Omi, under the supervision of Professor Dr. Md. Hasanul Kabir and Assistant Professor Sabbir Ahmed. It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma, or other qualifications.

Authors:



Md. Atiqur Rahman
Student ID: 180041123

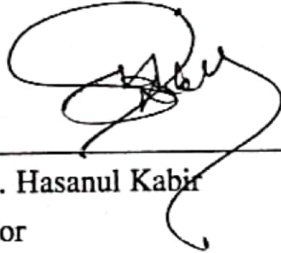


Nahian Ibn Asad
Student ID: 180041136



Md. Mushfiqul Haque Omi
Student ID: 180041140

Supervisor:



Dr. Md. Hasanul Kabir
Professor
Department of Computer Science and Engineering
Islamic University of Technology

Co-supervisor:



Sabbir Ahmed
Assistant Professor
Department of Computer Science and Engineering
Islamic University of Technology

Dedication

Devoted to the unwavering support of our parents, who have consistently stood by our side throughout the years, offering their love, guidance, and encouragement.

Acknowledgement

We would like to express our sincere gratitude to everyone who helped us complete our thesis. We are grateful for the inspiration, encouragement, and guidance that we received from our Supervisor, **Professor Dr. Md. Hasanul Kabir**, Dept. of Computer Science and Engineering, Islamic University of Technology (IUT) and Co-supervisor **Assistant Professor Sabbir Ahmed**, Dept. of Computer Science and Engineering, Islamic University of Technology (IUT). They provided us with constant guidance, support and invaluable insights. We could not have completed our thesis without them.

Finally, we would like to thank our parents for their unwavering love and support. They have always been there for us, and we could not have done this without them.

We are grateful for the opportunity to have worked on this thesis, and we are excited to share our findings with the world.

Contents

<i>Declarations of the Candidates</i>	i
<i>Dedication</i>	ii
<i>Acknowledgement</i>	iii
List of Figures	vi
List of Tables	vii
<i>Abstract</i>	viii
1 Introduction	1
1.1 Motivation and Scope	3
1.2 Problem Statement	4
1.3 Research Challenges	4
1.4 Research Contributions	7
1.5 Organization	7
2 Background Study	8
2.1 Meta-Learning Based Approach	9
2.2 Metric Learning Based Approach	11
2.3 Fine-tuning Based Approach	14
3 Methodology	23
3.1 Domain Adaptation	25
3.2 Warm Model	26
3.3 Pseudo Support Set	28
3.4 Instance-Level Feature Normalization	30
4 Results and Discussions	32
4.1 Dataset	32
4.2 Experimental Setup	33

4.3	Evaluation Metric	33
4.4	Quantitative Analysis	35
4.5	Qualitative Analysis	40
4.6	Ablation Studies	43
5	Conclusion	46
	References	47

List of Figures

1.1	Overview of General Object Detection Techniques shown in [1]	2
1.2	Traffic sign recognition shown in [2]. Detection is done in Phase #1, followed by a classification step.	3
1.3	Traffic sign in rainy weather shown in [3]	5
1.4	Occluded Traffic sign shown in [3]	6
2.1	Spatial misalignment due to convolution-based aggregation addressed in [4]	10
2.2	Proposed Meta-Classifer with Attentive Feature Alignment [4].	11
2.3	Multi-Relation Detector in [5]. Different relation heads model different relationships between the query and support image.	13
2.4	Illustration of Two Stage Fine-Tuning Approach (TFA) [6]	15
2.5	Two-phase single-branch architecture of transfer learning used in [7]	17
2.6	Comparison of Faster R-CNN and DeFRCN from [8]	18
2.7	The architecture of DeFRCN [8].	19
3.1	Basic Architecture of Faster RCNN	24
3.2	Proposed Architecture	24
3.3	Domain adaptation on Proposed Architecture	26
3.4	Freezing layers in TFA [6]	26
3.5	Freezing layers in FSCE [9]	27
3.6	Unfreezed Layers(Warm Model) in Proposed Architecture	28
3.7	Pseudo-Support Set incorporated in Our Architecture	30
3.8	Instance Level Feature Normalization incorporated in Our Architecture	31
4.1	Qualitative Analysis on TFA	40
4.2	Qualitative Analysis on FRCN-ft	41
4.3	Qualitative Analysis on CD-FSOD	42
4.4	Qualitative Analysis of Ours Model	43

List of Tables

2.1	Experimental results of our model on FSOD test set with different numbers of training categories and images in the 5-way 5-shot evaluation.	14
2.2	The freezing module comparison for existing FSOD approaches	20
4.1	DeFRCN on Bangladeshi Traffic Sign	36
4.2	Methods tested on Bangladeshi Traffic Sign Dataset, mAP @ 0.5	36
4.3	Comparison between our implementations and SOTA Architectures(mAP@0.5, 5 way n-shot; n=1,2,3,5,10)	37
4.4	Ablation Studies on FSBD(2x) on 5 way 3 shot	44

Abstract

The most critical task for the Advanced Driver Assistance System (ADAS) which is generally used in autonomous vehicles is to develop a reliable and fast Traffic Sign Recognition (TSR) system. TSR identifies the traffic sign from an image and then determines its category. The majority of widely used TSR techniques that rely on deep convolutional neural networks (DCNNs) emphasize on discriminating feature learning against visual differences of different traffic signs. But these techniques perform poorly if the number of samples available for each of the category is limited to model training. To overcome this problem, few-shot learning can be used where the approach focuses on learning common but distinctive qualities of class-specific objects with few training samples, as opposed to depending heavily on supervision to learn discriminating features. In this work, we have used fine-tuning approach for few-shot learning in order to recognize traffic signs with only a limited number of samples per category. We have introduced Domain Adaptation, Warm Model, Pseudo-Support Set and Instance-Level Feature Normalization in our base architecture. Our model outperformed all state-of-the-art (SOTA) architectures for few-shot learning across different shot settings, including 2, 3, 5, and 10 shots. Particularly, our model achieved remarkable results in 3-shot and 5-shot scenarios, with an additional mAP improvement of 3.53 and 3.73, respectively.

Chapter 1

Introduction

In this chapter, we provide a brief overview of our thesis. At first, we provide the application and research area of generalized object detection. In this work, we use the term 'detection' in order to represent detection followed by classification. In this chapter, we talk about a specific type of object detection called few-shot object detection. It's a technique that helps us train our models with very limited examples to tackle the challenge of having insufficient annotated data. We also discuss a related area called few-shot traffic sign detection, which is all about identifying and classifying traffic signs using this technique. After that, we mention our problem statement along with the challenges faced by our research domain. We conclude this chapter with the organization of the rest of the thesis

Object Detection : When it comes to identifying objects in images or videos, computers have a tough time keeping up with humans. That's where object detection [10] techniques come in, using algorithms based on deep learning [11] or machine learning [12] to improve their accuracy. Object detection is a crucial component of Advanced Driver Assistance Systems (ADAS), as well as other applications such as image retrieval and video surveillance.

Figure 1.1 shows the overview of the object detection. Object detection involves two phases: training and testing. During training, features are extracted from images for training the classifier. During testing, new images are analyzed and the model predicts the object's class. A post-processing step is then taken to display the results.

Deep learning-based object detectors require a ton of annotated data to train effectively. However, we want our computer models to be as good as humans, who can recognize

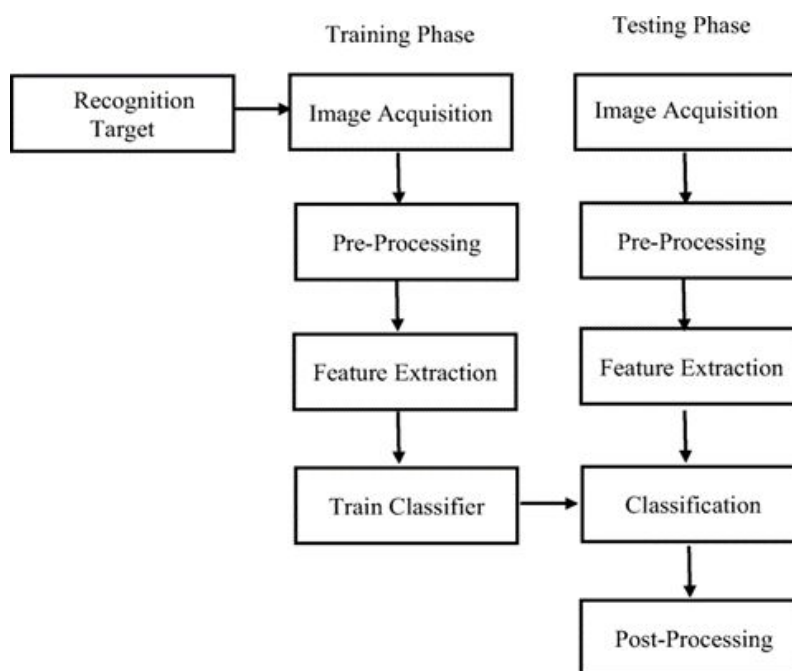


Figure 1.1: Overview of General Object Detection Techniques shown in [1]

new objects with just a few examples. To achieve this, a new technique called few-shot object detection was developed. It aims to learn from a small number of object examples of new categories, so there's no need for vast amounts of annotated data. Few-shot object detection can save time and money while also improving accuracy and efficiency, and it has the potential to create even smarter and more accurate object detection models that surpass human capabilities in certain scenarios.

Few-Shot Object Detection: In the field of object detection, deep learning has become the standard method. However, one of the main issues with deep learning-based object detectors is that they require an extensive amount of annotated data to achieve accurate results. This presents a challenge because we want object detection models to perform as well as humans, who can recognize new objects with just a few examples. To address this problem, a new technique known as few-shot object detection [13] was developed. This approach aims to improve the performance of object detection models even when there is limited data available for each class. By learning from a small number of object examples for new categories, few-shot object detection eliminates the need for acquiring and annotating vast amounts of data, making it a valuable tool in the field of object detection.

Few-Shot Traffic Sign Detection: In order to detect traffic signs, modern object detection techniques such as YOLO [14], and Faster RCNN [15] are used. However, when these techniques are applied to TSR in real world settings, they face challenges to detect traffic signs. For example, physical damages, partial occlusions of traffic signs, lighting

conditions, and motion blur makes it difficult to recognize traffic signs for TSR tools. Hence, a few-shot object detection techniques can be used for better performance by training the model used in TSR with a limited number of samples.

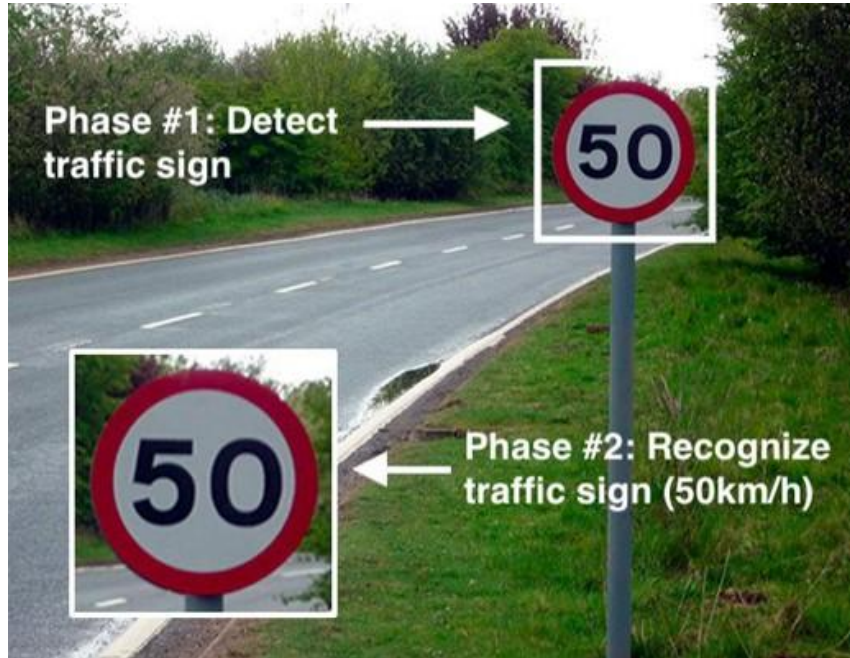


Figure 1.2: Traffic sign recognition shown in [2]. Detection is done in Phase #1, followed by a classification step.

1.1 Motivation and Scope

The recognition of traffic signs plays a crucial role in Autonomous Driving systems, requiring both speed and accuracy. However, existing approaches that rely on object detection techniques for traffic sign recognition often struggle with a large number of samples per class and perform poorly on unseen classes. In this thesis, we aim to address this challenge by utilizing few-shot object detection techniques. By doing so, we can ensure that our proposed system performs well even with a limited number of instances for each traffic sign class. The main contribution of this work is to develop a network that can rapidly and accurately recognize traffic signs in real-world scenarios, leveraging a small training dataset. This advancement would significantly benefit the Traffic Sign Recognition system, enabling the construction of a system that can quickly and accurately identify traffic signals in real-world situations, even when the availability of training samples for each class is limited.

1.2 Problem Statement

Since the Traffic Sign Recognition system is used for Autonomous Driving, recognition of traffic signs should be fast and accurate. Recent approaches for traffic signs recognition object detection techniques need a large number of samples under each class and they under-perform in unseen classes. In this work, we want to use few-shot object detection techniques so that the proposed system does not underperform with limited instances for each class. An important contribution to the TSR system can be proposing a network that can quickly and accurately recognize traffic signs in real world settings given a limited number of samples for training. The TSR system would benefit greatly from this since it would make it possible to build a system that can recognize traffic signals in real-world situations rapidly and accurately, even when there aren't many training samples available for each class. The problem statement can be summarized as "Designing a few-shot object detection model that uses fine-tuning approach for recognizing traffic signs with limited data and is robust in real world settings".

1.3 Research Challenges

In the TSR system, a camera mounted in a vehicle captures an image of a real traffic scene, which is then processed to recognize traffic signs. During this process, many challenges may be faced by TSR system such as:

- (a) **Blurring:** Images may get out of focus and complicated if the camera is not positioned properly. It can be difficult for the Traffic Sign Recognition (TSR) technology to distinguish and classify such unclear images. If the camera is not fixed properly, the pictures can become blurry. Blurred pictures are difficult to recognize by the TSR system. Even if the TSR model has been trained to recognize traffic signs accurately, the blurred images can create confusion and lead to incorrect classification. It is essential to make sure the camera is positioned and oriented correctly in order to capture images of the traffic signs that are clear and sharp. This will improve the TSR system's precision and efficiency, especially in real-world scenarios where the environment can be complicated.
- (b) **Lighting Conditions:** Image acquired under different lighting conditions such as by daylight or night will be different and the same traffic sign may end up in different classes. When traffic signs are captured in different lighting conditions, the resulting images can appear quite different. Image acquired under different lighting

conditions such as by daylight or night will be different and the same traffic sign may end up in different classes. For example, a sign that is captured during the day may appear differently than the same sign captured at night. Due to this variation in lighting conditions, the same traffic sign can end up being classified into different classes by a Traffic Sign Recognition system. This is why it is important to account for variations in lighting when training and testing the system, to ensure that it can accurately recognize traffic signs regardless of the lighting conditions.

- (c) **Weather conditions:** In different weather conditions such as rainy, foggy, and snowy, the pictures taken by the camera might not be up to the mark for traffic sign detection. Because weather conditions have the potential to have a considerable impact on both image quality and the TSR system's functionality. For instance, if it's raining, the camera lens can get wet and produce hazy pictures. Similar to when it's foggy outside, poor visibility might lead to hazy pictures. Additionally, in snowy situations, the camera may record overexposed photos, which may impair the sight of traffic signs in the picture. In order to provide accurate traffic sign detection in all weather circumstances, it is essential to take weather factors into account when developing the TSR system.



Figure 1.3: Traffic sign in rainy weather shown in [3]

- (d) **Occlusion:** One of the biggest challenges in traffic sign recognition is occlusion. It refers to the situation where objects like poles, trees, buildings, or even other vehicles cover the traffic sign either partially or completely. These occlusions can make it difficult for the TSR system for detecting and recognizing the traffic sign

accurately. In some cases, the TSR system may ignore the occluded traffic sign altogether, while in other cases, it may misclassify the occluded image as a different traffic sign.



Figure 1.4: Occluded Traffic sign shown in [3]

- (e) **Faded color:** Faded traffic signs can pose a significant challenge to the TSR system. Because of exposure to sunlight, climatic conditions, and other environmental variables, traffic signs' colors may deteriorate over time. As a result, the colors can become less distinct, making it challenging for the system to correctly detect the sign. This is especially problematic when the meaning of the sign is largely dependent on the color. A fading red stop sign, for example, could be misinterpreted for a yield sign, posing serious safety issues. To mitigate this issue, it is essential to ensure that the TSR system is trained with a diverse set of images that captures different levels of color degradation.
- (f) **Similarity:** In some cases, the traffic sign recognition (TSR) system might confuse objects that resemble traffic signs with actual traffic signs. This could potentially lead to incorrect information being displayed to the driver, causing confusion or even danger on the road. For instance, a TSR system may misinterpret a circular logo on a truck as a "Stop" sign, leading to unnecessary and dangerous braking by the driver. In such cases, it is important for the TSR system to be able to differentiate between actual traffic signs and other objects that may resemble them, such as logos or street art.
- (g) **Scene complexity:** The TSR system may struggle to precisely detect every traffic sign when there are several of them in an image. This is because it can be difficult

for the system to tell some indicators apart from others when they overlap. The TSR system can also have a hard time recognizing all of the traffic signs because there are other things in the image. In such circumstances, the system might give priority to detecting one sign over the others, potentially misclassifying or failing to detect other signs.

1.4 Research Contributions

- **Domain Adaptation:** The model is trained on a source traffic sign dataset and fine-tuned on a target traffic sign dataset to improve performance in the real-world environment with various lighting, angles, and settings.
- **Warm Model:** All layers, including pre-trained layers, are optimized and learned, allowing the model to extract more detailed features and enhance the ability to differentiate between different traffic signs.
- **Pseudo support set:** Generated during training, it simulates a new class and aids in increasing the training examples through strong and weak augmentation strategies such as scaling, flipping, and rotation, improving the model's generalization capabilities.
- **Instance-level feature normalization:** Normalizing the model's retrieved features at each instance level enhances generalization capability and reduces bias.

1.5 Organization

We have organized the rest of our thesis as follows:

In chapter 2, we discuss some existing work related to few-shot object detection. Various methods for few-shot object detection in existing literature are discussed in depth in this section.

Chapter 3 contains our methodology with the necessary diagrams for our thesis.

In chapter 4, we analyze the result we obtain from our methodology. A comparison between existing methods and proposed methods is also shown followed by a brief discussion and analysis of the obtained result.

In the fifth chapter of our thesis, we will wrap up by providing an analysis of the factors contributing to the accuracy of our model.

Chapter 2

Background Study

After going through the introduction we can easily understand our generalized research domain is *Few-shot object detection* and our specific research domain is *Few-shot traffic sign detection and recognition*. Now due to a severe lack of research on few-shot traffic sign detection, we could only read about generalized *few-shot object detection*. And from that, we accumulated the knowledge of generalized few-shot object detection and applied that knowledge on our specific domain few-shot traffic sign detection. As far as we know there is only one paper available on few-shot traffic sign detection which is “Meta-YOLO: Meta-Learning for Few-Shot Traffic Sign Detection via Decoupling Dependencies” [16]. For some obvious reasons, we are not covering this paper in our report. First of all, this paper only detects and does not recognize. Second, in the result section, they compared their model with some models which are not few-shot models. And these comparisons are done on a dataset called MIST [A multi-illuminant synthetic image test set] which is not a traffic sign dataset. They compare their model with other basic models like different versions of YOLO [14] using the traffic sign dataset but they did not use any state-of-the-art architecture. They did not even give the code repository so that we can try and implement it. Moreover, this paper used Meta-learning and in the next subsections of this section, we are going to discuss why we will not use meta-learning.

Before discussing why we will not use meta-learning we should discuss how many types of learning we have in a few shot object detection. There are three types of learning which are,

1. Meta-Learning [17–19]
2. Metric Learning [20]
3. Transfer learning [6,9]

Before going in depth about each learning type we want to state that we will use Transfer Learning for our proposed method. Now, why did we not use the fine-tuning [21] term here? Because fine-tuning is a type of transfer learning. Transfer learning means we trained on the source domain but we are getting tested with the target domain. And fine-tune means before getting tested on the target domain, we will train our network with a few samples of the target domain. Since samples are scant we call this approach a few-shot approach. Hence we are using a fine-tuning approach to transfer learning in our proposed method.

Now we would like to cover six papers in our background study. And with these six papers, we will also be able to go through all the types of learning.

We will be going through Meta-RCNN [18] to cover meta-learning. Then we have ARPN [5] to give a brief overview of Metric Learning in few-shot object detection. At last we will cover 4 papers TFA [6], FSCE [9], De-FRCNN [8] and CD-FSOD [22] for transfer learning. Since we will be using transfer learning in our proposed method we are giving more emphasis on this topic and that's why we are discussing four papers on transfer learning. Another thing to mention before starting is that we will also use cosine similarity [23] in our proposed method. If we use layman's terms, using cosine similarity we can find how much two vectors are similar. Now the use of it here is we can find how much our proposed regions are similar to our target. If it is more than a threshold value then, we get our targeted object in our proposed region. But the use of this cosine similarity is a bit different in our proposed method. In our proposed method we use cosine similarity score to boost our softmax score so that we get better classification results. More details will be given in the later sections. We will find this cosine similarity very common in transfer learning to get a better classification score.

2.1 Meta-Learning Based Approach

Meta-learning refers to learning something that is not learned in the standard problem. It is a way of hyperparameter optimization. Meta-learning speeds up and optimizes the hyperparameters of the networks that are not trained yet.

The model is trained in multiple stages for meta-learning. Initially, a model is trained using the base dataset. Then, an episodic training scheme is applied on the resulting model where each episode from the base dataset has the N-way-K-shot. This sequence of training is known as meta-training. When the generated model is applied to novel categories, this phenomenon is referred to as meta-testing. After the meta-training, the

finetuning is done on a uniform dataset of base classes and novel classes so that the model can detect both base and novel categories. Otherwise, meta-finetuning is applied with only the novel class in order to detect only the novel categories.

Better region proposals and fewer missed detection result from an aggregation process when Faster R-CNN [24] is used as the detector. Usually, the support vectors are aggregated with the features of query RoI. The RPN (Region Proposal Network) must, however, generate one RoI at least for each relevant object in order to do this. Unless this is the case, not even the finest aggregation method will help in locating the expected object. Nonetheless, the RPN is only trained on base categories. The RPN may not generate suitable RoIs for identifying objects if the novel categories diverge significantly from the base categories. As a result, in the base case, the channel-wise multiplication also known as the Hadamard product of the query features and the support vectors is generated. Additionally, channel-wise multiplication is combined with subtraction and concatenation to perform a more difficult computation. Such aggregation is built upon by Meta Faster R-CNN [4]. Moreover, AttentionRPN architecture designed by Fan et al. [5] aggregates query and support features prior to the region proposal network. The spatial correlations between the support image and the query image aren't always aligned. Meta Faster R-CNN [4] also addresses this spatial misalignment.

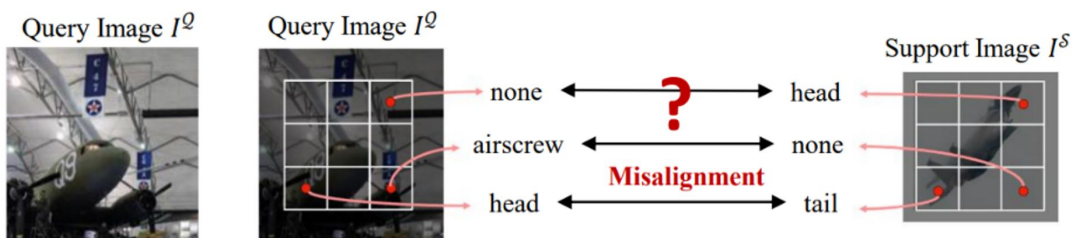


Figure 2.1: Spatial misalignment due to convolution-based aggregation addressed in [4]

The support features and RoI features are spatially aligned using two attention modules and then the foreground areas are emphasized. In addition, redundant information can be reduced while improving feature sensitivity and stability for both novel and base categories via a new aggregation approach of features using a learnable soft-threshold operator.

However, the episodic training method for meta-learning is complicated. In addition, a dual-branch design makes it more difficult to achieve. In more recent research, certain categories of meta-learning have received less attention, and their performance is lagging behind. The population size needed also grows rapidly as the number of parameters to learn increases. Meta-learning may require careful hyperparameter optimization since it might be sensitive to the mutation technique. Furthermore, especially for big models like

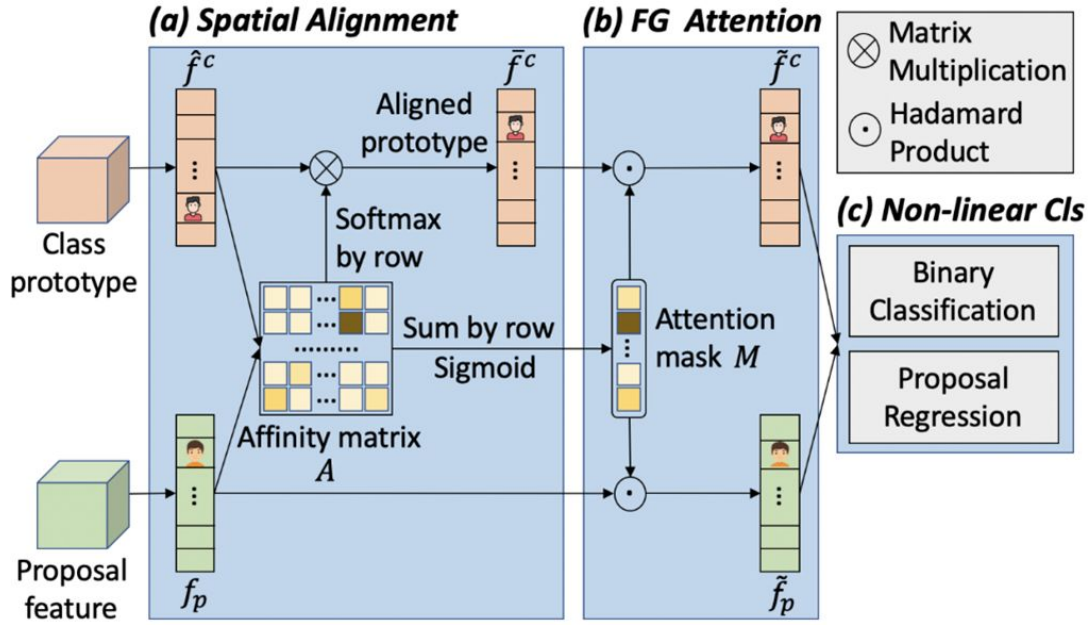


Figure 2.2: Proposed Meta-Classifier with Attentive Feature Alignment [4].

CNNs, their fitting performance is typically worse than gradient-based approaches.

2.2 Metric Learning Based Approach

In this section, we will learn about the usage of metric learning [20] in few-shot object detection. Let's first discuss metric learning in image classification, we assign different islands for different categories or classes of samples we are trying to recognize or classify. Then in the inference stage, we take a query image and make a query island and calculate the shortest distance between the query island and all other islands. The class of that island which has the shortest distance from our query island is the proposed class for the query image. This way we do image classification using metric learning and the same goes for object detection. We just use different objects to make different islands. And using objects is possible because we have bounding boxes to get the objects.

Now authors of this paper stated that their motivation was the work of Siamese Neural Network for One Shot Image Classification [25]. The Siamese network has an architecture that has a severe influence on metric learning. Hence we can say metric learning also has an influence on A-RPN [5]. But most of the time metric learning is used as a booster for better classification scores. Because according to FSCE [9] and DeFRCNN [8] the main error is classifying novel objects as learned base classes. That's why both of these papers used other modules to improve classification scores. FSCE used a contrastive

head aligned with the regressor head and classifier head. Whereas DeFRCNN used a Prototypical Calibration Block to improve the softmax classification scores. More details will be provided in later sections. We can see that though FSCE and DeFRCNN both have a fine-tuning approach they heavily used metric learning to improve their classification score. That's why we also used cosine similarity to improve our classification scores in our proposed method. Because we know cosine similarity is the building block of metric learning.

Now let's dive deep into A-RPN [5]. The authors of this paper claim they have two contributions.

A general few-shot object detection network: This network has similar architecture to Faster RCNN [24]. The authors added two things extra to the architecture and one new strategy for training. The first one is attention RPN(Region Proposal Network) and the other one is a multi-relational detector. And the strategy is a contrastive training strategy. A-RPN is used at an early stage to get less and compact region proposals and a multi-relation detector is used at a later stage which suppresses & filters out false detection in the confusing background.

Attention RPN: Our RPN in our vanilla FRCNN proposes regions that can have our target objects. But in a few shot evaluation settings, we give some support samples and give a query image and tell our network to find the target objects in the query image using the information of support samples. Here we use A-RPN so that we can use the information of support samples. So we learn from supporting samples that what we need to find in our query image. Here using supporting samples we are giving extra attention to query images so that we can get fewer and better region proposals for the next stage.

Now X is the feature map of the support set and Y is the feature map of the query set. We use X as kernel and slide it through on the Y in a depth-wise cross-correlation manner [26]. In the below equation, we can see G , and here G is the resultant attention feature map. Now the value of $G_{h,w,c}$ is the score of cross correlation of X and Y from h,w coordinate on channel c . 'From h,w coordinate' means we put the kernel X 's $(0,0)$ element on the (h,w) element of Y . Then put X 's $(0,1)$ on Y 's $(h, w+1)$ and this goes on until we cover all the elements of X .

$$\mathbf{G}_{h,w,c} = \sum_{i,j} X_{i,j,c} \cdot Y_{h+i-1,w+j-1,c}, \quad i, j \in \{1, \dots, S\} \quad (2.1)$$

Multi-Relation Detector: From Figure 2.3, we can get three relation heads. Global relation means similarity between two images. Local relation means similarity between

one pixel of one image and one pixel at the same position of another image. Patch relation is the complex one where we try to find similarity between one pixel of one image to all the pixels of another image and vice versa. The local relation head is basically depth-wise cross-correlation but for box predictors. Global relation is deep embedding for global matching and patch relation is a deep nonlinear metric for patch matching. After doing experiments authors came to the decision that it is better to use all the relation heads. Patch relation being the most complex one should work better alone but due to complexity it is data-hungry and does not perform well like global and Local. Nonetheless, we need a patch relation head to get nonlinear learning which boosts AP.

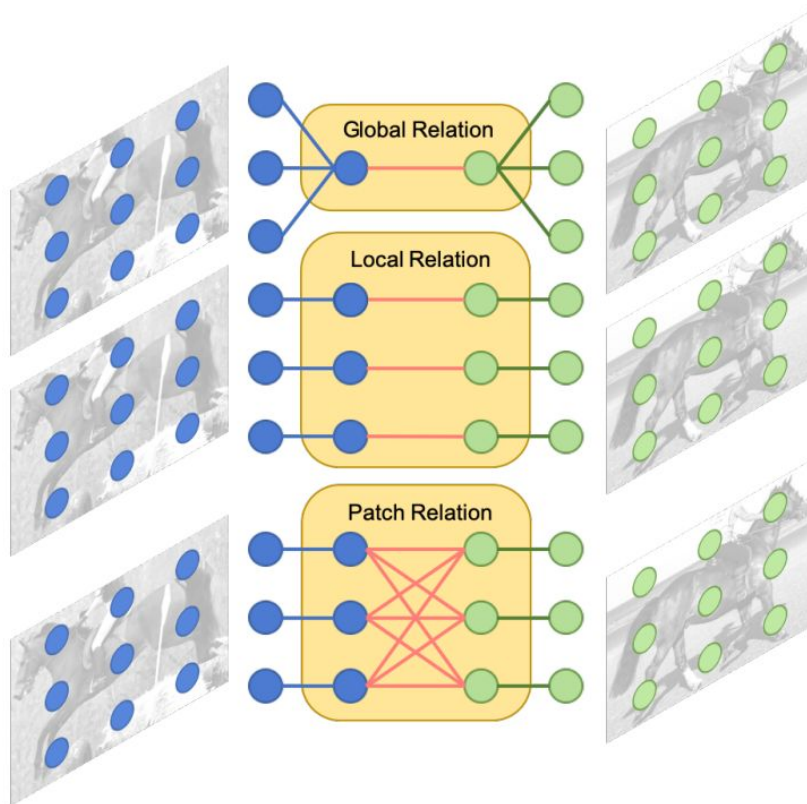


Figure 2.3: Multi-Relation Detector in [5]. Different relation heads model different relationships between the query and support image.

Preparing a new dataset FSOD : A new dataset is proposed by authors claiming present datasets do not follow the few shot evaluation settings. According to them, category diversity is more important than the amount of per category samples. And Table 2.1 proves that their claim is right.

We can see that with each increment of the number of categories we have better AP and each one has more AP than MS COCO.

Table 2.1: Experimental results of our model on FSOD test set with different numbers of training categories and images in the 5-way 5-shot evaluation.

Dataset	No. of Class	No. of Images	AP_{50}	AP_{75}
COCO	80	115k	49.1	28.9
FSOD	300	26k	60.3	39.1
FSOD	500	26k	62.7	41.9
FSOD	800	27k	64.7	42.6

Now let’s discuss some important characteristics of the FSOD dataset. FSOD dataset has 1000 categories whereas MS COCO has only 80 categories. Though it has a tremendous amount of categories, the amount of samples per category is much less than MS COCO. FSOD has a total of 66000 images whereas MS COCO has a total of 123287 samples.

FSOD also follows the same convention of MS COCO for naming their categories. They have 83 parent semantics which have further split into 1000 categories. Here one example can be parent semantic is CAR for both AUDI and MERCEDES leaf categories.

The images of FSOD are from Open Image Dataset [27] v4 and Imagenet [28]. They processed the samples before adding it to the main dataset. Like there are some samples that have categories like ice bear and polar bear. But they both mean the same thing. So they had to use one single-parent semantic for those categories. 531 categories are from imagenet and 469 categories are from Open Image Dataset. The split is 800/200 while training/testing.

2.3 Fine-tuning Based Approach

Wang et al. used fine-tuning based approach in Frustratingly simple few-shot object detection [6] also known as the Two-Stage Fine Tuning Approach(TFA). In this paper, we got introduced to the fine-tuning based approach and the later papers got better and better with the fine-tuning approach. Here we also explain the problems with meta-learning and why A-RPN is not a fine-tuning based approach.

Problems with meta learning: In meta-learning we learn for specific tasks and then accumulate the knowledge to do better in novel sets. But this mainly works in episodes. So with increasing classes of support sets, the meta-learning approach becomes memory inefficient. And the algorithm of meta-learning is so complex that they tend to overfit on few samples. That’s why normal fine-tuned FRCNN works better than some meta-learning approaches. Another thing is that most of the meta-learning approach is used for image classification. Object detection is very different from image classification be-

cause in object detection we also need to do both localization and recognition, not just recognition like image classification.

The training strategy for A-RPN [5] is quite different from the usual fine-tuning based approaches. Here we give the support set to networks and give query sets to another network. These networks share weights. So here we don't fine-tune our networks with support sets, rather we give it to networks like online processing in the inference stage and get the output of final proposals with corresponding scores from networks that work on query sets. The network architecture is like below image,

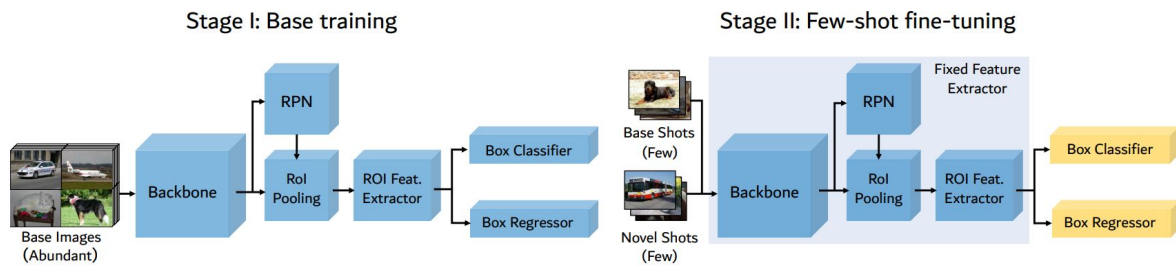


Figure 2.4: Illustration of Two Stage Fine-Tuning Approach (TFA) [6]

So authors of this paper implemented TFA with two intuitions,

- Feature representation learned from base classes is enough and we don't need our backbone to learn the feature representation of novel sets. We can just transfer the knowledge learned from base classes to novel classes. Hence we don't need to do backpropagation on backbone, RPN and ROI feature extractor while we are in the fine-tuning stage. We can just do backpropagation on the box predictor or the last two layers of FRCNN network.
- This works because backbone and RPN do not learn about classes. They just learn how to differentiate foreground from background. Then in the second stage, we do classification and regression. That's why we don't need to fine-tune the feature extractor (backbone, RPN, ROI).

With the above two intuitions in mind, authors froze the feature extractor and region proposal networks and just fine-tune the last two layer classifier head and regressor head with support sets. All other steps are quite similar to normal FRCNN training. Up to this is called a few shots with fine-tuning. Then they also tried another variant by adding cosine similarity which is called TFA w/cos. Now in the below equation, we can find there is $F(x)$ which is an input feature and we find the cosine similarity of it with the weight of different classes. Our final output class will be, whose weight has the highest score with our input feature $F(x)$.

$$s_{i,j} = \frac{\alpha \mathcal{F}(x)_i^T w_j}{\|\mathcal{F}(x)_i\| \|w_j\|} \quad (2.2)$$

We have previously mentioned that deep detectors encounter significant issues with overfitting in scenarios involving few-shot learning. The disparity between few-shot detection and general object detection is more pronounced compared to the gap observed in few-shot image classification. The FSCE [9] method can be considered as the next best alternative to TFA. MPSR which focuses on multi-scale positive sample refinement for few-shot object detection, addresses the scale bias inherent in few-shot datasets and represents an improvement over TFA. However, it should be noted that MPSR’s positive refinement branch requires manual selection, which can be seen as somewhat less elegant. In the baseline TFA approach, the performance for novel classes diminishes when the RPN and ROI feature extractor are unfrozen. Nevertheless, we have discovered that this behavior can be reversed and actually improve the results for novel detection if the training is executed properly. In FSCE, only the backbone is frozen, while an additional head with two existing box predictor heads is added. This allows the authors to fine-tune the FPN pathway and RPN while keeping the backbone fixed. This approach has proven effective in coordinating the activation of backbone feature maps for novel objects, while also mitigating the risk of overfitting. The authors of FSCE employed the FRCNN, a two-stage detection framework, where the RPN utilizes backbone feature maps to generate region proposals. The ROI head then classifies each region proposal and performs bounding box regression if an object is predicted to be present. The authors realized that the primary source of error lies in misclassification after successfully detecting the target instances. As a result, they introduced a contrastive head, consisting of a classifier head and a regressor head. This contrastive head aims to group instances of the same class together while creating a separation between instances of different classes. From our perspective, the key contributions of this paper are:

Unfrozen RPN and ROI: The authors kept RPN and ROI unfrozen with two modifications. The backbone feature extractor is frozen during fine-tuning while the ROI feature extractor is supervised by a contrastive objective. First, they took twice as many proposals as TFA after Non-Max Suppression (NMS). This secures the potential positive anchors which have low scores. Second, take half of the sampled proposals suggested by ROI because these regions mostly contain background.

Contrastive learning: Authors applied batch contrastive learning used in Supervised Contrastive Learning [29] to introduce more intra-class similarity and more inter-class distinction.

CPE (Contrastive Proposal Encoding) loss: Sun et al. criticized methods with complex algorithms that can easily overfit and perform poorly in few-shot evaluation settings. That’s why they have chosen contrastive learning to learn discriminative object proposal representations without complexing the model.

A supervised contrastive objective with specific considerations for detection will be optimized to reduce the variance of object proposal embeddings from the same category while pushing different category instances away from each other. A common approach to do this same thing is to use a large margin classifier used in Large margin deep network for classification [30], but with our trials, category-level positive-margin-based classifiers do not work in this data-hungry setting.

With contrastive learning, the algorithm learns to build representations that do not concentrate on pixel-level details but encode high-level features effective enough to distinguish different images.

Transfer learning methodologies follow a fundamental two-phase approach using a single-branch architecture [7]. Initially, the detector is trained on base categories, with the weights of all components frozen except for the RoI head responsible for bounding box regression and classification [7].

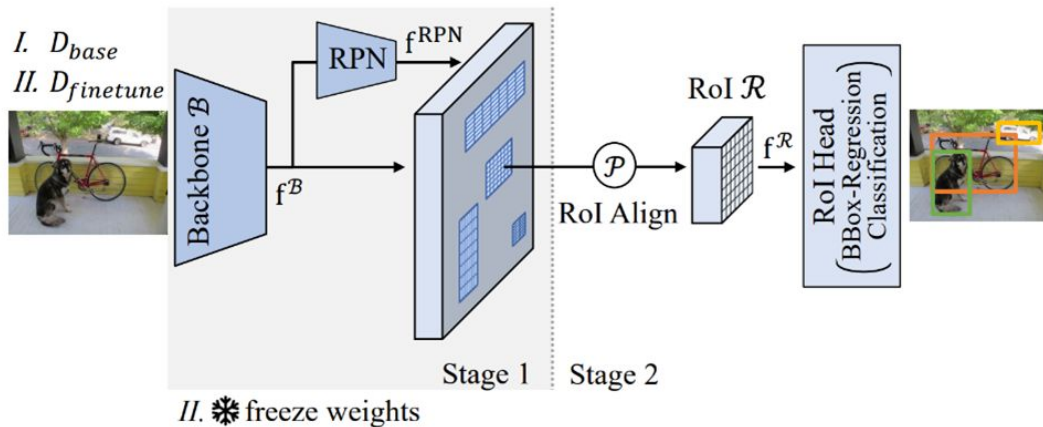


Figure 2.5: Two-phase single-branch architecture of transfer learning used in [7]

In the second phase, fine-tuning takes place on the last layers for both base and novel categories, using a training set with balanced data selections and K shots per category [7]. To account for feature variations between base and novel categories, a modification is made to Faster R-CNN, utilizing cosine similarity for classification [7]. This straightforward approach outperforms complex meta-learning strategies and has been enhanced through various techniques [7].

To further optimize training, the training objective can be updated with revised loss func-

tions, focusing on foreground areas or specific features [7]. Limiting gradient flow within the detector or making minor adjustments to the training process can also improve the training of detector components [7].

In their work on Decoupled Faster R-CNN (DeFRCN), Qiao et al. propose modifying the backbone in both training phases. They discovered that contradictions arise during training because the ROI head aims to differentiate categories while the RPN aims to learn class-agnostic region proposals. They found it crucial to scale the gradient from the ROI head to the backbone, rather than allowing it to flow from the RPN to the backbone. During the first phase on the base dataset, the gradient from the ROI head is scaled by 0.75, ensuring that the backbone learns slightly less than the rest of the detector. Throughout the training on the base and novel datasets in the second phase, scaling the gradient by 0.01, which effectively freezes the backbone, is essential. This scaling greatly enhances performance, especially in the second phase when the gradient from the RPN is stopped and the gradient from the ROI head is scaled. The authors also noted that when trained with ample data, Faster R-CNN benefits from gradient scaling when used as a general object detector.

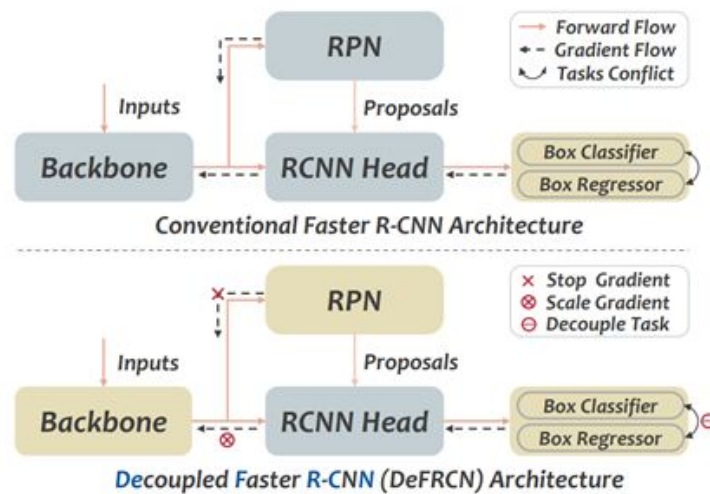


Figure 2.6: Comparison of Faster R-CNN and DeFRCN from [8]

To prevent catastrophic forgetting for base categories at the finetuning phase, In CFA, Guirguis et al. [7] developed a new gradient update strategy that takes into consideration the angle between gradients for samples of the base dataset and samples of the novel dataset. Although the main goal of this gradient optimization method is to conserve the performance of base categories, it also has a positive effect on the performance of novel categories.

Regarding optimal gradient flow and inter-class separability, the loss has to be reduced.

Similar to two-branch meta-learning, a contrastive loss in an auxiliary branch can contribute to enhance the distinct ability of the features.

Benchmark outcomes testify to the effectiveness of Faster R-two-stage CNN's detector for transfer learning methods. Li et al. [31] identified that false positive classifications, or category confusion, are the principal cause of the performance decrease for novel categories in Faster R-CNN. As a result, they improve the classification in a separate branch that increases classification ability and is trained on samples that were incorrectly categorized as false positives. The cropped picture of the object is processed directly. The classification outcome is then integrated with one of the initial branches of the Faster R-CNN.

Many novel categories have low classification scores, as stated by Qiao et al. in DeFRCN [8]. They get to the same conclusion as Li et al. [31], that it is troublesome to have criteria for classification that has translation invariant features and localization that has translation covariant features. They provide a prototype calibration block (PCB) to address this problem by performing score refinement to get rid of false positive classifications with high scores. Hence, a score refinement can help in reducing the number of false positive classifications in detectors based on Faster R-CNN.

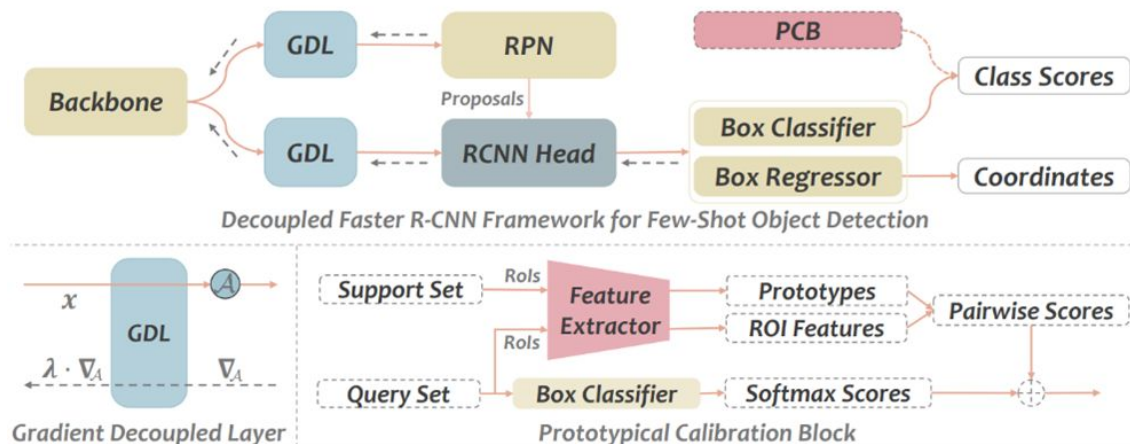


Figure 2.7: The architecture of DeFRCN [8].

As opposed to meta-learning, which requires complex episodic training, transfer learning processes offer a considerably simplified training pipeline. Transfer learning processes can achieve state-of-the-art performance by adding certain methods to be able to fine-tune as many detector components as possible, such as changing the training objective or transmitting knowledge between the base and novel categories.

Cross-domain means we train on the source domain and get tested on the target domain. But what all the above papers did was a train on one domain and get tested on unseen

samples of that same domain. Only DeFRCNN tried a cross-domain approach where authors trained their network with COCO and tested it with PASCAL VOC. Even that is not a complete cross-domain approach because COCO and PASCAL VOC have some categories which are the same or at least they are from the same domain. One example can be that both COCO and PASCAL VOC have animal images though the class might be different, the domain is the same which is ANIMALS.

In CD-FSOD: A Benchmark For Cross-domain few-shot object detection [22] find this huge flaw in the present few-shot evaluation settings. That’s why they proposed a new benchmark or evaluation setting called CD-FSOD (cross-domain few-shot object detection). Now after forming this new evaluation setting, they tested all the models suggested by the above papers. And their performance was not satisfactory. Even one FRCN just fine-tuned did better than all the above models (Meta-RCNN, TFA, FSCE, DeFRCN). Obviously, they introduced some changes to make vanilla FRCN work better. We will first discuss the intuitions behind the changes and then the changes.

1. First of all, why did all the above models underperform? Because during the fine-tuning stage, there are always some modules of those models which got frozen so that they will not overfit the few training samples. Which modules of which models got frozen is clearly illustrated in Table 2.2.

Method	Backbone	RPN	ROI Head
Attention-RPN	✓		
Meta-RCNN	✓		
H-GCN	✓		
TFA w/cos	✓	✓	✓
FSCE	✓		
DeFRCN			✓

Table 2.2: The freezing module comparison for existing FSOD approaches

Now, these models really don’t get overfitted but that was in a few-shot evaluation setting where the source and target domains are the same. But in cross-domain few shot evaluation settings, these models get overfitted with the source domain when the modules of feature extractors are frozen and since it got overfitted with the source domain, it underperforms when tested with the target domain. That’s why authors experimented with unfrozen modules in cross-domain few-shot evaluation settings and the performance got better.

2. Just unfreezing modules did very well but our authors did not stop there, they proposed a new method that does not require changing the internal architecture of the

model. They just used a very interesting and elegant fine-tuning strategy. They divided their approach into the training and testing stage. The testing phase has different stages.

(a) **Training Stage:**

In this stage, we train our detector (Basic FRCN) with the source domain (MS COCO).

(b) **Testing Stage:**

We copy the weights of the detector and copy it to a student (another basic FRCN). Then we train the student with support sets just like FRCN-ft. Here we keep all the modules of the student unfrozen. Then we get another basic FRCN as a teacher and copy the weights of the students to the teacher. This is the initiation of the distillation step. So we have our support images and we augment those images in two different ways. Strong augmentation has color jittering, grayscale conversion, cutout patches, and gaussian blur. Weak augmentation has only random horizontal flips. Then we pass the weak augmented support set image through the teacher and got the pseudo labels. Teachers will put bounding boxes where he is confident enough that there is a target object. Now the bounding boxes given by the teacher are called pseudo labels. Then we pass strongly augmented images through the student and get bounding boxes where he is confident enough that there is a target object. We compare students' bounding boxes with ground truth which gives us supervised loss (L_s) and by comparing students' bounding boxes with pseudo labels we get distillation loss (L_D). We compute the total loss L using the equation below,

Now we need to update weights. So, using the total loss L we update the weights of the student using the equation below,

$$W_s \leftarrow \alpha W_s + \gamma \frac{\delta L}{\delta W_s} \quad (2.3)$$

Then we use student's weights to update teacher weights using the below equation,

$$W_t \leftarrow \alpha W_t + (1 - \alpha) W_s \quad (2.4)$$

The way a teacher's weights get updated is called EMA (Exponential mean average) update. We are storing the average of students in the teacher. Since we are averaging one's learning it means that we are getting more generalized

knowledge that is not overfitted with any support sets. Again, taking distillation loss works like regularization which also prevents overfitting to the source domain. Then at the inference stage, we use our teacher model and we don't do any sort of augmentation on the query sets.

Now that all the intuitions and changes are discussed let's discuss the cross-domain datasets. So base training is done on MS COCO. It also experimented with PASCAL VOC but the result was not up to the mark. The cause is clear with a more diversified category and an ample number of samples, the network produces better results.

After base training on the source domain MS COCO, authors use ArTaxOr, UODD, DIOR, and ChestX as target domains. The reasons for using them are,

1. **ArTaxOr:** This has the image of arthropods. These are natural images but fine-grained specific to biology. Problem is that these require lots of annotations.
2. **UODD:** This has the image of arthropods. These are natural images but fine-grained specific to biology. Problem is that these require lots of annotations.
3. **DIOR:** Satellite images. These were chosen because they have lost perspective but have higher resolution and discriminative images.
4. **ChestX:** This might be the most difficult one in these four target domain datasets. This has radiological images. Lost two color channels and not a natural scene. This might have more differences with source domain COCO than all the other three target domain datasets.

Now the authors chose the FRCN because it works well on the ChestX. Since it works well on the most difficult dataset, we can say that it has the capability of being the most generalized network among Fully convolutional one-stage detector(FCOS) [32], RetinaNet and Deformable-DETR.

Chapter 3

Methodology

Our proposed methodology is a multi-step process designed to improve the accuracy and robustness of the traffic sign detection system using few-shot learning. The methodology is built on the framework of few-shot learning, which entails training a model to learn from a small set of examples. The goal in this scenario is to detect traffic signs from a small number of samples. We have included a number of innovative features that improve the model's accuracy and capacity to detect traffic signs in real-world environments.

We begin with domain adaptation, in which the model is trained on a source domain and then fine-tuned on a target domain to increase performance in the target domain. The target domain in the context of traffic sign detection is the real-world environment in which traffic signs may appear under various lighting situations, angles, and settings.

After that we unfreeze component while training allows the model to optimize and learn the parameters of all layers, including the pre-trained layers. This allows the model to learn more detailed features from the data and improve its ability to distinguish between different traffic signs.

We also introduce the usage of a pseudo support set, which is a collection of samples generated during training to simulate a new class. We start with a step of preprocessing that comprises both strong and weak augmentation strategies. The goal is to create new images from the existing dataset by applying transformations such as scaling, flipping, and rotation. The goal of this step is to increase the amount of training examples and improve the model's generalization capabilities.

Finally, our suggested methodology incorporates instance-level feature normalization, which is a strategy for normalizing the model's retrieved features at each instance level. This strategy aids in improving the model's generalization capability and reducing the

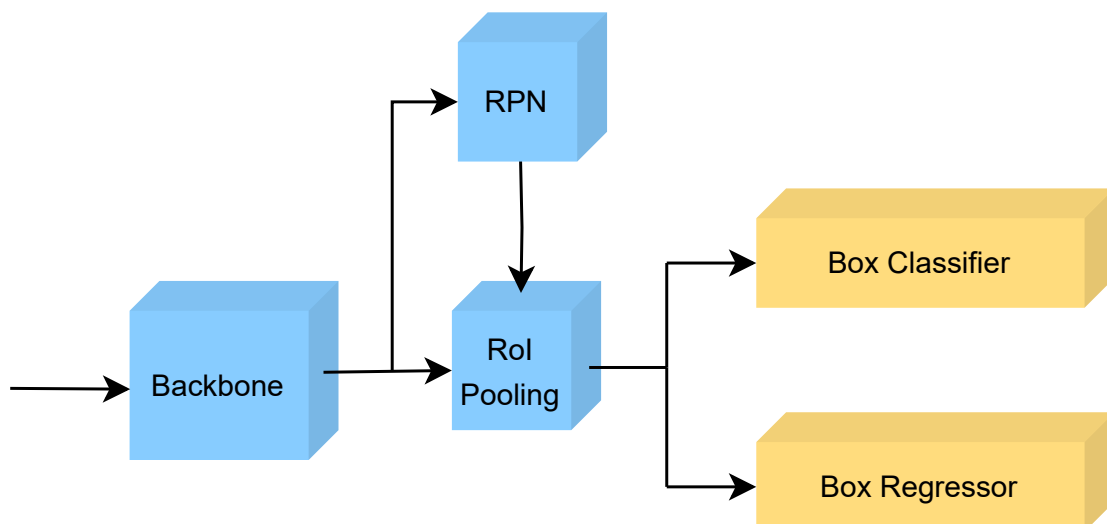


Figure 3.1: Basic Architecture of Faster RCNN

impact of intra-class variability. When these characteristics are coupled, the suggested methodology becomes a robust and successful strategy for few-shot traffic sign detection. Now we'll dive into a detailed explanation of the proposed methodology's aspects.

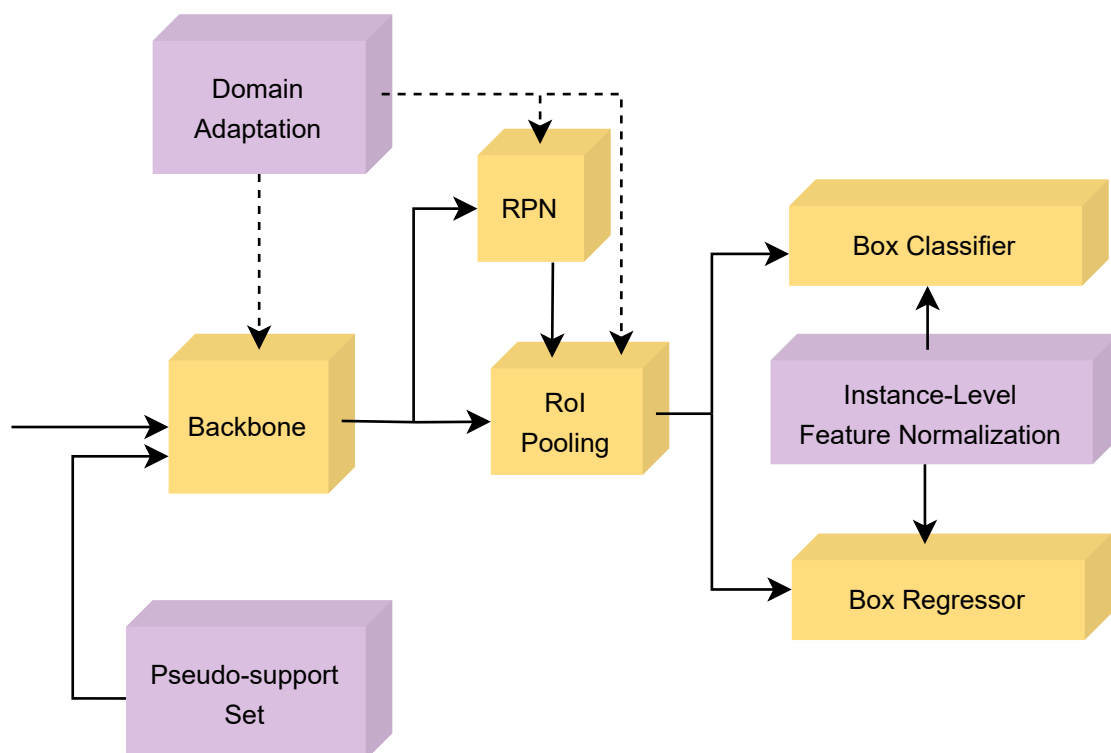


Figure 3.2: Proposed Architecture

3.1 Domain Adaptation

Domain adaptation is a process of adapting a model trained on one domain (source domain) to work well on another domain (target domain).

Our proposed methodology for traffic sign detection builds on the concept of domain adaptation, which has been explored in several existing works such as DEFRCN [8], CDFSOD [22], and ARPN [5]. In DEFRCN, the MSCOCO dataset was divided into base and novel classes for training and testing respectively. Similarly, CDFSOD used MSCOCO as the base dataset for training and ArTaxOr, UODD, and DIOR datasets for testing.

However, we were particularly interested in the approach taken in the ARPN [5] paper, where they created a new dataset called FSOD specifically designed for few-shot learning. This dataset had 800 base classes for training and 200 novel classes for testing.

As we aim to build a traffic sign detection system, our source and target domains do not necessarily have to be distinct. Therefore, we decided to merge different traffic sign datasets. The datasets we have used are:

1. German traffic sign dataset
2. LISA dataset
3. Chinese dataset

German traffic sign dataset, LISA dataset and Chinese dataset to create a base dataset as the source domain for our model. We then used the Bangladeshi Traffic Sign dataset as our target domain.

To implement domain adaptation in our proposed architecture, we first trained our model on the source domain and then fine-tuned it on the target domain.

The incorporation of domain adaptation in our methodology improves the model's ability to generalize and perform well on unseen data. This is because it allows the model to learn and recognize common features between the source and target domains and adjust its weights accordingly. This reduces the risk of overfitting to the training data and improves the model's ability to accurately detect traffic signs in real-world scenarios.

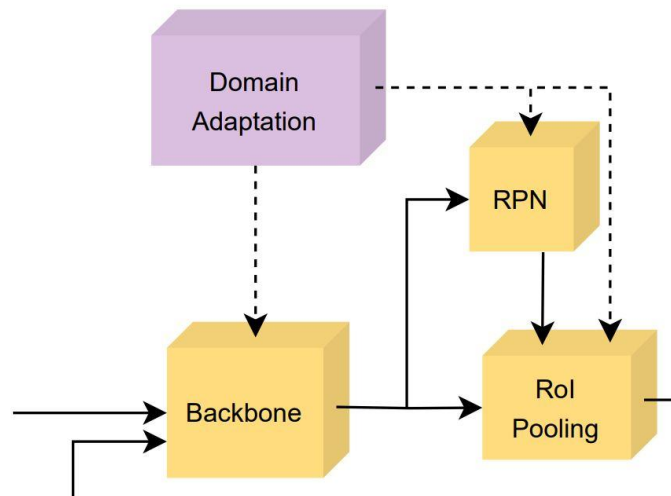


Figure 3.3: Domain adaptation on Proposed Architecture

3.2 Warm Model

While performing background studies, we have found that TFA, CDFSOD and Defrcn are all papers that propose methods for few-shot learning using deep learning models. In order to improve the performance of their models, they all employ the strategy of freezing certain layers during training. However, it has been observed that this approach can result in a decrease in accuracy.

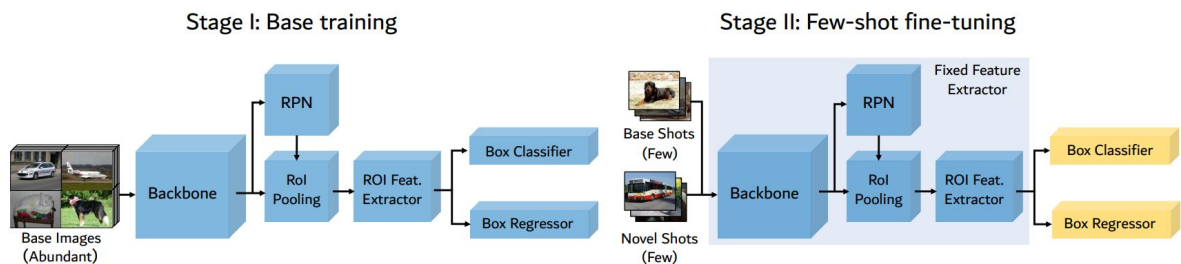


Figure 3.4: Freezing layers in TFA [6]

The reasons behind decreasing accuracy are:

1. Frozen layers are not updated during training, which means that they may not be able to learn to represent the features of the input data that are important for traffic sign detection.
2. Frozen layers can cause the model to become overconfident, as they may not be able to learn to represent the uncertainty in the data.

3. Frozen layers can make the model less resistant to changes in the input data because they can't adjust to new data distributions.

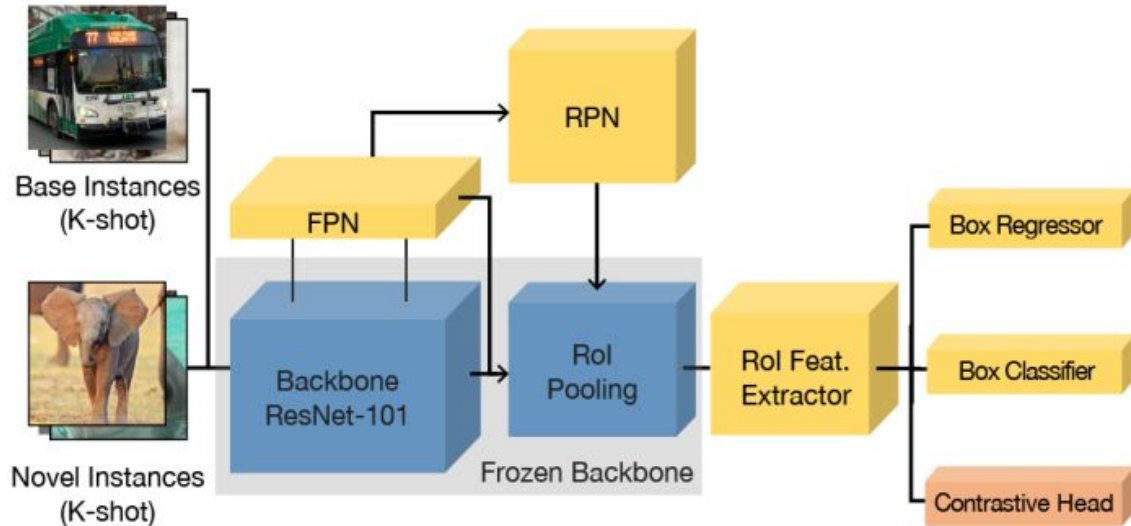


Figure 3.5: Freezing layers in FSCE [9]

Furthermore, when a layer is frozen during training, its weights remain constant and are not changed during the training process. When the layer is a pre-trained feature extractor that has acquired useful features for the job at hand, freezing it allows the model to concentrate on learning other parts of the task.

However, if too many layers are frozen, the model may not be able to adapt to the specific characteristics of the dataset it is being trained on, and its performance may suffer as a result. This is because the fixed weights of the frozen layers may not be able to capture the nuances and variations in the data that are important for the task.

In contrast, training a model without freezing any layers allows all of the weights to be updated during training, which can help the model adapt to the specific characteristics of the dataset and improve its performance. This is especially important in few-shot learning, where the model needs to learn from a small number of examples and may not have access to a large pre-existing set of features to leverage. As we are not freezing any layers we call our model warm model, because the model will not be frozen it will remain warm!

Therefore, in our methodology for traffic sign detection, we have chosen to train the FRCN layers of our model without freezing anything. This allows the model to learn from the support set and adapt to the target domain, improving its accuracy in real-world scenarios.

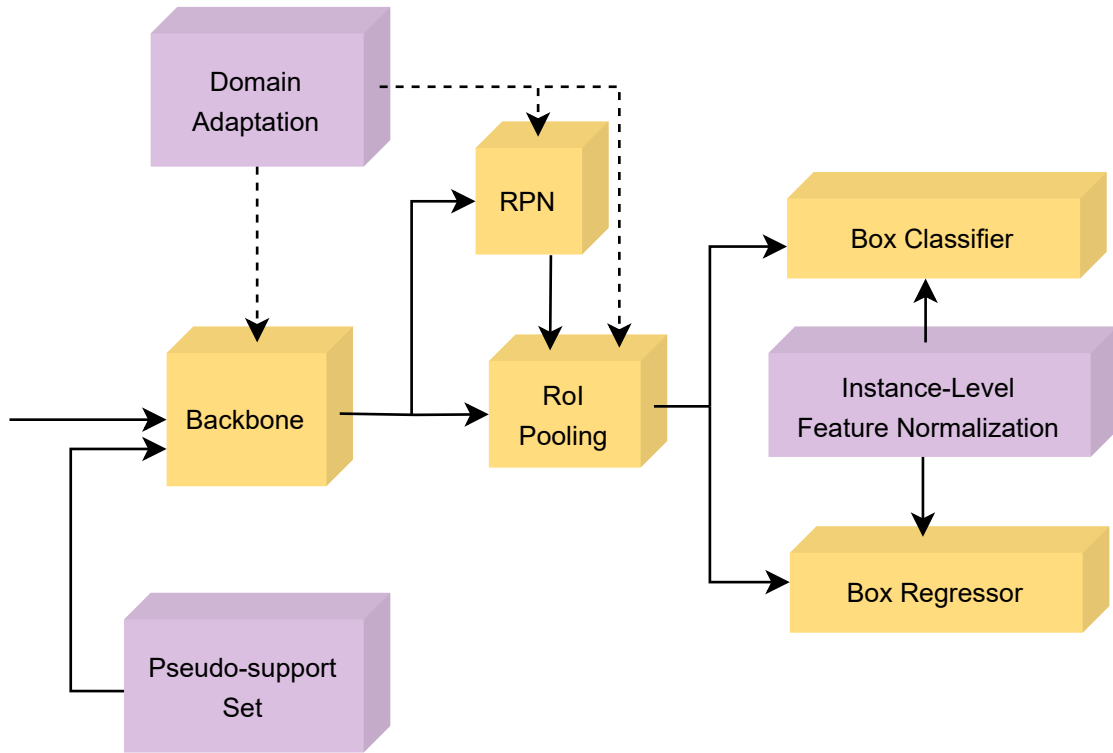


Figure 3.6: Unfrozen Layers(Warm Model) in Proposed Architecture

3.3 Pseudo Support Set

Data augmentation is a technique that involves creating new synthetic data samples from existing training data to improve the performance of machine learning models. The CDF-SOD paper implemented a combination of strong and weak augmentation techniques, which played a critical role in improving the accuracy of their model. Strong augmentation techniques such as Gaussian blur, color jitter, and cutout helped to introduce variability in the training data, making the model more robust to changes in lighting conditions, image quality, and other factors that can affect the performance of the model in real-world scenarios. On the other hand, weak augmentation techniques such as horizontal flip helped to introduce diversity in the dataset, enabling the model to learn more generalizable features that can be applied to different scenarios.

Inspired by this approach, we decided to experiment with data augmentation in our own model, using the Bangladeshi traffic sign dataset. We fine-tuned the CDF-SOD model with various hyperparameter values to determine the best combination for traffic sign detection. We found that setting alpha and gamma to 0 resulted in the best performance.

In CDFSOD the total Loss were,

$$L = L_S + \lambda L_D \quad (3.1)$$

Now if we put $\lambda=0$ then the equation becomes,

$$L = L_S \quad (3.2)$$

Besides, we needed to perform EMA update in CDFSOD following this equation,

$$W_t \leftarrow \alpha W_t + (1 - \alpha) W_s \quad (3.3)$$

Using the hyperparameter value for $\alpha = 0$ we get,

$$W_t \leftarrow W_s \quad (3.4)$$

The absence of the distillation loss and the collapsing of the teacher model into the student model were the results of our hyperparameter experiments. This means that we can now eliminate the teacher model from our architecture, and instead focus solely on the student model. Previously, we had employed the EMA update method to adjust the teacher model's parameters using those of the student model, but now that we have trained our student model by combining three traffic sign datasets, it has surpassed the performance of the teacher model. This allows us to skip the EMA update step altogether.

To enhance the performance of the student model, we applied strong data augmentation techniques such as Gaussian blur, cutout, and color jitter to the support set. We referred to this augmented support set as the "pseudo support set." Instead of using the teacher model, we trained our FRCN model again using this pseudo support set. As a result, our 5-way 5-shot model is now essentially a 5-way 10-shot model, since we are using the same support set twice during the training process. Overall, these adjustments have enabled our model to achieve higher accuracy and better performance on traffic sign detection tasks.

This approach allowed our model to achieve better accuracy, especially in scenarios where there is limited training data or when the data has limited variability. Overall, the incorporation of data augmentation in our model enhanced the robustness and generalization capabilities of the model, making it more effective for real-world applications.

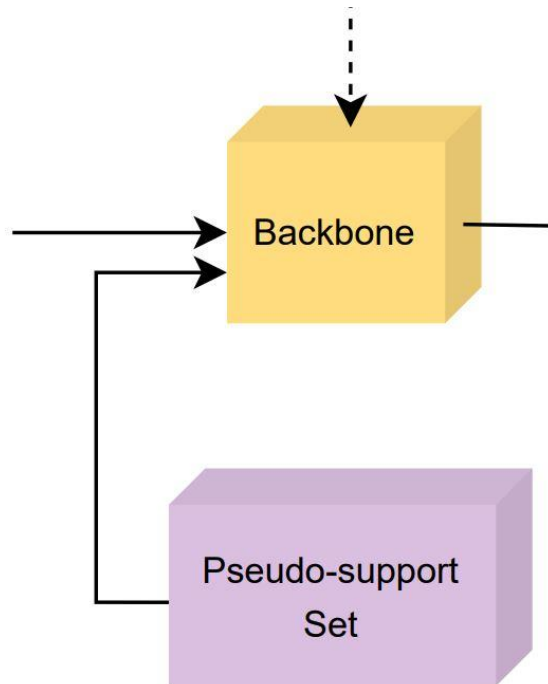


Figure 3.7: Pseudo-Support Set incorporated in Our Architecture

3.4 Instance-Level Feature Normalization

In the context of FRCNN, it is important to have translation invariant features for the box classifier, as it should be robust to changes in object position. On the other hand, the box regressor requires translation covariant features to accurately predict the bounding box coordinates.

Instance-level feature normalization is a technique used to normalize the features of each instance in a dataset. This is done by subtracting the mean of the features and dividing by the standard deviation of the features. This ensures that all of the features have a mean of 0 and a standard deviation of 1.

By making the features more comparable, instance-level feature normalization can be utilized to improve the performance of machine learning models. This is significant because distinct features may have varying scales and ranges. We can verify that all of the characteristics are on a same scale by normalizing the features, making it easier for the machine learning model to learn from the data. The prediction in a normal neural network is computed using the dot product of the weight vector and the input features:

$$\hat{y} = w^T * x \quad (3.5)$$

Instance-level feature normalization, on the other hand, uses a separate algorithm to nor-

malize the features and assure instance-specific modifications. The formula to be used is:

$$s_{i,j} = \frac{\alpha \mathcal{F}(x)_i^T w_j}{\|\mathcal{F}(x)_i\| \|w_j\|} \quad (3.6)$$

Instance-level feature normalization aids in the normalization of our model's scores or values. In comparison to the previous implementation, this normalization reduces or brings the scores closer together. As a result, by modifying the weights, our model attempts to improve these ratings. This change results in higher confidence scores, showing better assurance in our model's predictions.

We provide a more balanced and consistent depiction of the data by normalizing the scores. This balance enables our algorithm to better comprehend and compare the various ratings, resulting in a more accurate estimate of the level of confidence associated with each prediction. In other words, the normalization procedure scales up the scores, making them more comparable and reliable.

The goal of this continuous weight updating method is to maximize the scores and, thus, the model's confidence. As the weights are adjusted, the model becomes more fine-tuned and flexible to the precise patterns and properties of the data it is trained on. This adaptability enables the model to better capture the complexities of the input data and produce more accurate predictions.

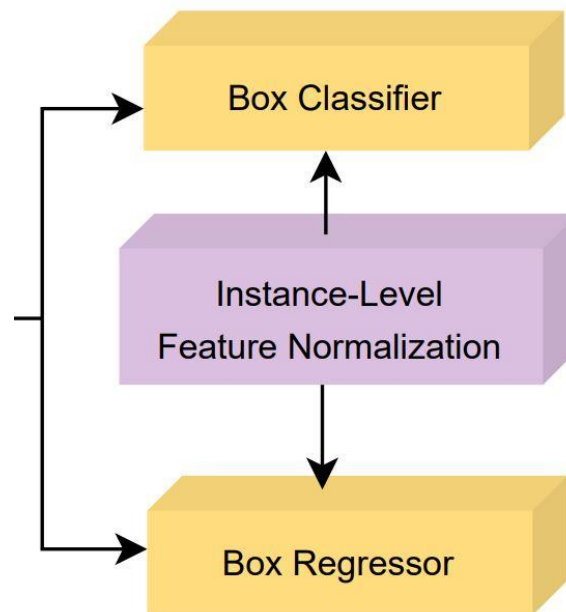


Figure 3.8: Instance Level Feature Normalization incorporated in Our Architecture

Chapter 4

Results and Discussions

4.1 Dataset

To ensure the effectiveness and robustness of our model, we diligently generated a thorough training and testing dataset in the dataset portion of our thesis. We combined three separate datasets to train our model:

1. German traffic sign dataset
2. Lisa dataset
3. Chinese dataset

This merging approach enabled us to use a wide variety of traffic sign samples, capturing a wider diversity of forms, colors, and patterns often observed in real-world circumstances.

We used the Bangladeshi Traffic Sign dataset for assessment and testing, since it provided a solid baseline for analyzing our model's performance in identifying and categorizing traffic signs particular to the Bangladeshi environment. By utilizing this dataset, we aimed to validate the generalizability and adaptability of our model across different geographical regions, ensuring its suitability for practical applications.

By incorporating these datasets into our study, we strived to create a comprehensive and representative collection of traffic sign images, enabling our model to learn and recognize the distinctive characteristics of traffic signs in a wide range of scenarios.

4.2 Experimental Setup

Throughout our research, we ensured a fair and consistent experimental setup to accurately compare the performance of different models. Here are the specific details of our setup:

Optimization Parameters: We utilized Stochastic Gradient Descent (SGD) as the optimizer, with a learning rate of 0.01 to control parameter adjustments based on gradients. Momentum with a value of 0.9 was applied to aid convergence by keeping the model moving in the right direction. Weight decay, with a coefficient of $5e^{-4}$, was implemented to prevent overfitting and promote generalization..

Backbone Networks: We used Faster RCNN as our base detector and Resnet50 as the backbone architecture for feature extraction.

4.3 Evaluation Metric

The performance metric we are going to use for our model is Mean Average Precision(mAP). Generally, mAP is used to evaluate object detection models. Ours is a traffic sign detection model which falls under the category of object detection. Hence, we are using mAP as our evaluation metric. To compute mAP, we need to first compute the Average Precision (AP) for each class of traffic signs in the dataset. AP is the area under the precision-recall curve for a given class. In information retrieval and classification tasks, precision and recall are frequently employed metrics. To calculate precision and recall in a detection or classification task, we use two different formulas. We will briefly discuss these formulas in more details in next few paragraphs.

To compute AP, we first sort the detection results by their confidence scores, from high to low. The precision and recall at that point in the list are then computed for each detection result. These values are then used to plot the precision-recall curve. The area under this curve is denoted by AP. This procedure is repeated for each kind of traffic signs in the dataset. Finally, we compute the mean of all AP values to obtain the mAP for our model.

Using mAP as our evaluation metric allows us to measure the overall performance of our traffic sign detection model. It takes into account both the precision and recall of our model for each class of traffic signs, which gives us a more comprehensive understanding of its effectiveness in real-world scenarios.

To properly understand mAP we need to understand the following metric first:

- (a) **Intersection over Union(IoU):** In object detection tasks, one of the key evaluation metrics used is Intersection over Union (IoU). IoU, which stands for Intersection over Union, is a common evaluation metric used in object detection tasks. The IoU is a way to check how well the predicted bounding box matches the actual ground truth bounding box.

IoU measures the overlap between the two boxes, and the resulting value ranges from 0 to 1. When the IoU value is 1, it means that the predicted box perfectly aligns with the ground truth box. Bounding by a threshold IoU value we can decide whether the prediction is True Positive(correctly classified as positive), True Negative(correctly classified as negative), False Positive(incorrectly classified as positive), False Negative(incorrectly classified as negative).The IoU threshold plays a crucial role in determining the accuracy of the model, and it is often optimized during the training process to improve performance.

- (b) **Precision:** Precision is the measurement of the proportion of predicted positives that are actually correct. Precision is a metric that tells us how many of the predicted positive results are actually correct. It helps us measure the accuracy of our model's predictions by comparing the number of true positive results (correctly detected traffic signs) to the total number of predicted positive results. In simpler terms, precision tells us how precise or exact our model's positive predictions are. Mathematically we can write,

$$Precision, P = \frac{TruePositive}{TruePositive + FalsePositive} \quad (4.1)$$

- (c) **Recall:** Recall is the measurement of the proportion of actual positives that were predicted correctly. In other words, recall indicates how many of the actual positive examples were correctly identified by the model. When the recall value is low, it means that the model is not able to detect all the objects of a particular class. This can result in important objects being overlooked or incorrectly classified, making it essential to have a high recall value. Mathematically we can write,

$$Recall, R = \frac{TruePositive}{TruePositive + FalseNegative} \quad (4.2)$$

- (d) **Average Precision:** Average precision is the area under the Precision-Recall graph. Average Precision is class specific. We need to calculate Average Precision(AP) for each class. The average precision is a way of measuring the performance of the model by considering both precision and recall. It's calculated by calculating the area under

the precision-recall curve. Calculating the value and then averaging it across all classes in the dataset yields the average precision for each class. This enables us to assess the model’s effectiveness for each particular class and pinpoint any areas that require improvement. We can check that the model is doing well overall and make improvements as needed by looking at the performance for each class.

- (e) **Mean Average Precision:** Mean Average Precision(mAP) is the average of AP(Average Precision) over all detected class. It is a metric that calculates the average precision score for all the different classes present in the dataset. Essentially, we calculate the average precision for each class, and then take the mean of all these values to get the mAP score. This allows us to evaluate the overall performance of the model across all the different classes and ensure that it is able to detect and classify all relevant objects accurately. Mathematically we can write,

$$mAP = \frac{\sum AP}{n} \quad (4.3)$$

Here, n indicates the number of classes.

When we see mAP@0.75, it means that the metric was calculated by setting the Intersection over Union (IoU) threshold to 0.75. The IoU measures how well the predicted bounding box overlaps with the ground truth bounding box.

In the past, mAP was evaluated only at a single IoU threshold of 0.5. However, the COCO challenge made the evaluation more challenging by redefining mAP@0.5 to mAP@[0.5:0.05:0.95]. This means that the mAP is now calculated and averaged over a set of ten different IoU thresholds ranging from 0.5 to 0.95, with a step frequency of 0.05. This allows for a more comprehensive evaluation of the model’s performance across different levels of overlap between the predicted and ground truth bounding boxes.

4.4 Quantitative Analysis

Since we have discussed the core idea of the evaluation metric mAP, we can evaluate our model performance. We have implemented the DeFRCN model and measured the performance of the model in the Bangladeshi Traffic Sign Dataset. Table 4.1 shows the results of DeFRCN on the Bangladeshi traffic sign dataset on different AP such as 50,75 and different support set sizes such as 1 shot, 2 shot, 10 shot, etc. Here nAP means novel class Average Precision.

Table 4.1: DeFRCN on Bangladeshi Traffic Sign

	Shots	AP	AP50	AP75	nAP	nAP50	nAP75	Time Taken
Repeat 0	1	25.949	40.482	26.593	25.949	40.482	26.593	eta: 1:22:23
	2	28.978	42.966	33.469	28.978	42.966	33.469	eta: 2:02:21
	3	30.447	46.044	35.418	30.447	46.044	35.418	eta: 2:46:37
	5	33.279	49.507	37.624	33.279	49.507	37.624	eta: 3:34:56
	10	35.817	51.08	42.418	35.817	51.08	42.418	eta: 7:11:36
Repeat 1	1	27.282	42.299	28.351	27.282	42.299	28.351	
	2	29.332	42.687	34.705	29.332	42.687	34.705	
	3	30.506	46.161	35.686	30.506	46.161	35.686	
	5	33.05	48.72	37.943	33.05	48.72	37.943	
	10	35.868	51.725	42.323	35.868	51.725	42.323	
Repeat 2	1	27.177	42.318	27.492	27.177	42.318	27.492	
	2	29.41	43.014	34.222	29.41	43.014	34.222	
	3	29.96	45.529	34.392	29.96	45.529	34.392	
	5	32.488	47.771	37.703	32.488	47.771	37.703	

We have also measured the performance of TFA w/FC [6], FRCN-ft [24], CD-FSOD [22] on Bangladeshi Traffic Sign Dataset. Table 4.4 shows the comparison between these methods. For comparison, we have used 5 way 15 shot. Comparing the mAPs of these methods we see, CD-FSOD performs better than FRCN-ft [24] and TFA w/FC [6]. The reason behind this result is that the strategy CD-FSOD [8] uses that is EMA update performs better in cross-domain as well as traditional settings. Although in FRCN-ft [24] some of the layers are fine-tuned, it can not exceed the performance of CD-FSOD. On the other hand, TFA w/FC [6] gives a very poor mAP as it freezes all the layers.

After conducting an extensive series of experiments, we have obtained valuable insights into the performance and accuracy of our model under various settings. The results of these experiments are meticulously documented and presented in the form of a comprehensive Table 4.4, which provides a detailed overview of the achieved accuracy across different configurations and parameters.

Table 4.2: Methods tested on Bangladeshi Traffic Sign Dataset, mAP @ 0.5

Method/Shot	5 way 15 shot
TFA w/FC	10.346
FRCN-ft	20.048
CD-FSOD	24.7996

Throughout our experimentation and evaluation process, we have employed the widely recognized mean average precision (mAP) metric with an intersection over union (IoU) threshold of 0.5 as our chosen accuracy measure. This metric provides

a comprehensive assessment of the model’s ability to accurately detect and localize traffic signs in various scenarios.

By setting the IoU threshold at 0.5, we ensure that the predicted bounding boxes align sufficiently with the ground truth annotations, indicating a successful detection. The mAP score takes into account both precision and recall, providing a holistic measure of the model’s performance across different classes and detection thresholds.

Table 4.3: Comparison between our implementations and SOTA Architectures(mAP@0.5, 5 way n-shot; n=1,2,3,5,10)

Method \ Shot	1 Shot	2 shot	3 Shot	5 Shot	10 Shot
TFA	5.17	9.48	4.67	11.52	12.43
TFA w/cos	12.44	13.09	12.96	15.03	18.68
FRCN-ft	16.16	24.99	25.88	39.2	58.10
DeFRCN	24.63	23.55	26.74	27.87	35.84
CD-FSOD ($\lambda = 2, \alpha = 0.4$)	14.91	25.46	25.29	39.36	56.15
Ours 2x	21.39	26.44	30.27	42.02	57.79
Ours 4x	19.13	29.47	29.88	43.09	63.57

In our research, we have implemented and evaluated our model using a 5-way n-shot framework, where n takes values from the set 1, 2, 3, 5, 10. This approach allows us to assess the performance of our model across different levels of shot, representing the number of support examples available for each class during training.

During our evaluation, we compared the performance of several existing models, including TFA, TFA with cosine similarity, FRCN-ft, DeFRCN, CDFSOD, as well as our own models labeled as Ours(2x) and Ours(4x). These models were selected based on their relevance and potential to address the challenges of few-shot learning in the context of traffic sign detection.

In Ours(2x), we introduced a strong augmentation strategy where we performed intensive data augmentation techniques on the images, resulting in the generation of additional augmented images. This effectively doubled the size of the support set, hence the name 2x. By augmenting the data in this manner, we aimed to enhance the model’s ability to generalize and learn robust representations for improved few-shot learning performance.

Similarly, in Ours(4x), we further extended the augmentation approach by generating three additional augmented images from each original image, resulting in a

total of four augmented images per sample. This intensive augmentation strategy, labeled as 4x, aimed to provide the model with an even more diverse and comprehensive set of training examples, facilitating better learning and adaptation to novel classes.

The table clearly illustrates the performance of different models in terms of accuracy. TFA emerges as the worst performer, followed by TFA with cosine similarity, and then DeFRCN. FRCN-ft shows a slightly better performance, ranking as the third-best model, with accuracy comparable to that of CDFSOD.

However, our model outperforms all these state-of-the-art (SOTA) models in every shot except for the 1-shot scenario. Whether it's the 2x or 4x variant, our model surpasses the others in terms of accuracy. This highlights the effectiveness and superiority of our proposed approach in few-shot learning tasks.

The results presented in the table provide a clear indication of the performance differences among the models evaluated. TFA consistently demonstrated the lowest accuracy across all shot levels.

The performance of TFA was observed to be the lowest among the evaluated models due to its approach of freezing all layers except for the head. By freezing the majority of the layers, the model's ability to adapt to the specific characteristics of the dataset is limited. As a result, its performance suffers, especially in the context of few-shot learning tasks where the model needs to generalize well with limited training examples. In contrast, other models that employ different strategies such as fine-tuning or partial freezing of layers demonstrate better performance as they allow for more flexibility in adapting to the dataset.

However, when TFA was combined with cosine similarity, we observed significant improvements in accuracy. For example, in the 1-shot scenario, the accuracy increased from 5.17% to 12.44%.

TFA with cosine similarity and instance level normalization at the head outperformed TFA without these modifications. The inclusion of instance level normalization at the head of the TFA model helps to reduce the intra-class variance and improve the accuracy of novel classes, while minimizing the decrease in detection accuracy for base classes. This normalization technique ensures that the features of each instance in the dataset are appropriately normalized, leading to improved model performance. Therefore, the utilization of instance level normalization in TFA with cosine similarity contributes to its superior performance compared to the original TFA model.

On the other hand, DeFRCN outperformed both prominent models and our own

model in the 1-shot setting. This can be attributed to the design of DeFRCN, which is specifically tailored to excel when the number of images in the support set is limited. As a result, as the number of shots increases in DeFRCN, the improvement in mean Average Precision (mAP) becomes less substantial. Additionally, other prominent models surpass DeFRCN when the number of shots exceeds 1.

DeFRCN is specifically designed for robust generalization, which is crucial for few-shot learning tasks. It incorporates a Gradient Decouple Layer that decouples the Region Proposal Network (RPN) and ROI (Region of Interest) head from the backbone network. This decoupling allows DeFRCN to achieve the highest level of generalization among the models evaluated.

In the context of 1-shot learning, where there is limited training data available, high generalization power becomes particularly important. DeFRCN excels in this regard by effectively decoupling the RPN and ROI head, enabling it to adapt and generalize well to novel classes with minimal training examples.

By prioritizing generalization and leveraging the benefits of decoupling, DeFRCN surpasses other models in terms of performance, making it the top performer in the 1-shot scenario. Its ability to generalize effectively even with limited training data sets it apart as a powerful model for few-shot learning tasks.

In the FRCN-ft (Faster R-CNN fine-tuned) model, no layers are frozen during the fine-tuning process. This means that all layers of the model, including both the backbone and the detection heads, are updated and adjusted based on the target dataset.

Contrary to this approach, CDFSOD claims that by fine-tuning their model with certain frozen layers, their method can surpass state-of-the-art (SOTA) architectures. They argue that their specific combination of frozen and unfrozen layers allows their model to adapt better to the target dataset, resulting in improved performance.

CDFSOD perform same as FRCN-ft. Because since our evaluation setting is in neither like cross domain nor like FSOD, the student model in FSOD is already a well-performing model. Further generalizing this model may actually lead to a decrease in performance. Therefore, with the specific hyperparameter values of $\alpha = 0.4$ and $\lambda = 2$, CDFSOD does not significantly modify its pre-trained model based on FRCN-ft. Instead, it retains the knowledge and characteristics learned from the FRCN-ft model. So with CDFSOD does not change its burned model on FRCN-ft that much.

We have enhanced the basic architecture of Faster RCNN by incorporating Domain Adaptation, Warm Model, Pseudo-Support Set, and Instance Level Feature Normal-

ization. By leveraging the best features from state-of-the-art (SOTA) architectures, our model outperforms others in most categories. The only exception is DeFRCN, which has a specific advantage as explained earlier. Notably, our model surpasses all other models in the 2-shot, 3-shot, and 5-shot categories when using 2x augmentation. In the 1-shot category, DeFRCN performs better, and in the 10-shot category, FRCN-ft outperforms our variant, Ours(2x). However, our other variant, Ours(4x), outperforms all models except DeFRCN in the 1-shot category.

4.5 Qualitative Analysis



Figure 4.1: Qualitative Analysis on TFA

In our qualitative analysis, we conducted a thorough examination of the model's predictions on various test samples. We assessed the model's performance in terms of object detection accuracy, localization precision, and its ability to handle challenging scenarios and diverse object classes.

Qualitative analysis plays a crucial role in evaluating the performance and capabilities of our proposed model. By examining the qualitative aspects, we can gain

insights into the model's behavior, its ability to generalize, and its overall effectiveness in solving the problem at hand.

In our qualitative analysis, we conducted a thorough examination of the model's predictions on various test samples. We assessed the model's performance in terms of object detection accuracy, localization precision, and its ability to handle challenging scenarios and diverse object classes.

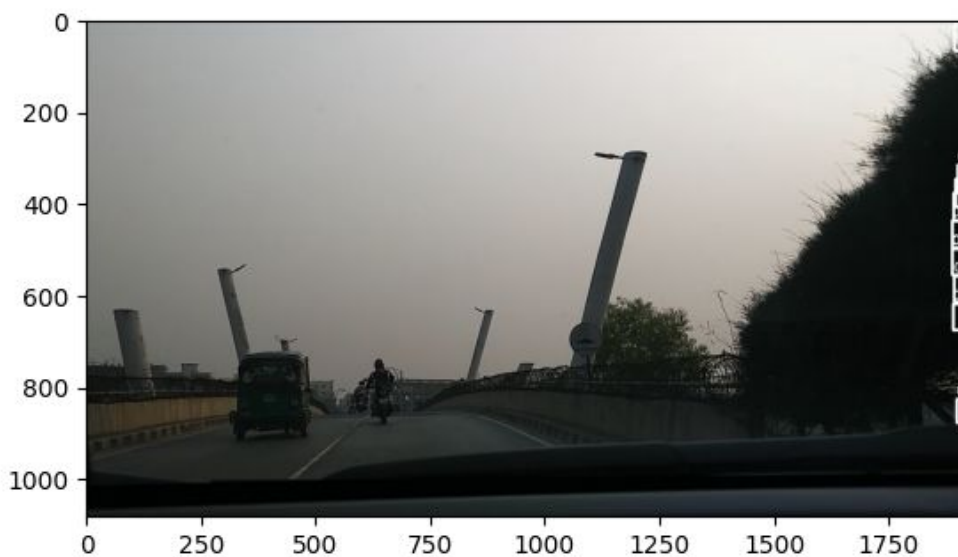


Figure 4.2: Qualitative Analysis on FRCN-ft

From Figure 4.1 it is clear that, TFA's performance in traffic sign detection is sub-optimal as it tends to erroneously identify unrelated objects as traffic signs. This deficiency hinders its ability to accurately distinguish true traffic signs from irrelevant objects within the scene, potentially leading to incorrect interpretations and decisions in real-world applications.

Furthermore, Figure 4.2 shows that the performance of FRCN-ft in detecting traffic signs in the given image is inadequate. Despite fine-tuning the model, it fails to accurately identify and localize the traffic signs present in the image. This limitation suggests the need for further improvements or modifications to enhance its performance specifically for traffic sign detection tasks.



Figure 4.3: Qualitative Analysis on CD-FSOD

Additionally, as depicted in Figure 4.3, the performance of CDFSOD in the detection of traffic signs in the provided image is unsatisfactory. Despite undergoing fine-tuning, the model struggles to precisely recognize and locate the traffic signs depicted in the image. Although this model has detected the traffic sign, it has also detected random object as traffic signs. This drawback emphasizes the necessity for additional enhancements or alterations to improve its effectiveness, particularly in the context of traffic sign detection tasks.

On the other hand, Figure 4.4 shows that even though other models could not detect the traffic sign properly, Ours has exhibited exceptional performance in this regard. Not only did our model successfully identify and locate the traffic sign, but it also exhibited a notable absence of false detections or misclassifications of unrelated objects as traffic signs. This substantiates the superior performance and effectiveness of our model compared to the other models under evaluation.

We observed that our proposed model consistently demonstrated robust object detection capabilities across different environmental conditions, lighting variations, and occlusion scenarios. The model successfully identified and localized traffic signs with high accuracy, providing reliable results in real-world scenarios.

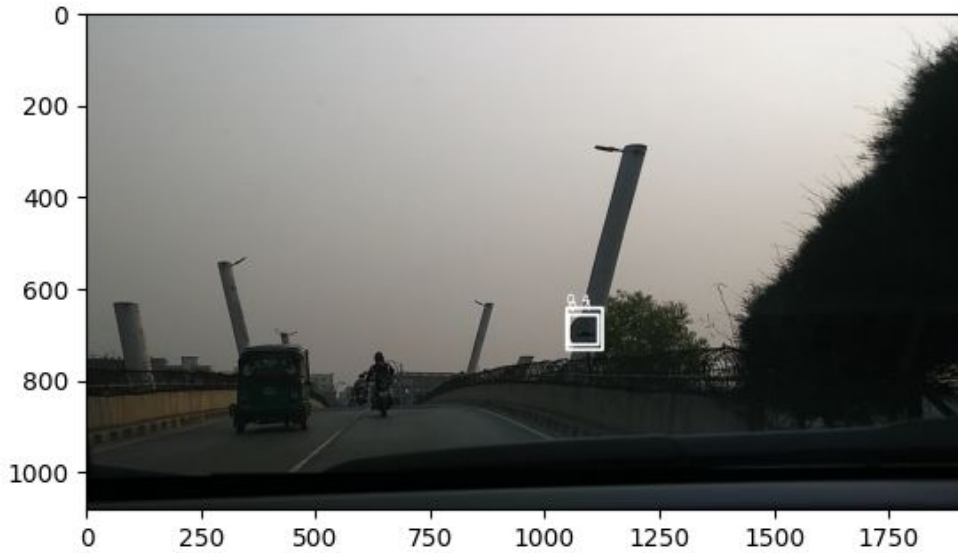


Figure 4.4: Qualitative Analysis of Ours Model

Furthermore, our model showcased a remarkable ability to generalize well to novel traffic sign classes and unseen test samples. It effectively learned and recognized various traffic sign categories, including stop signs, speed limit signs, yield signs, and more. This capability is crucial for real-world applications where the model must adapt and respond accurately to new and evolving traffic sign variations.

4.6 Ablation Studies

In order to understand the impact of each component in our model architecture, we conducted an ablation study where we systematically selected specific components while removing others. This allowed us to evaluate the contribution of each component towards the overall performance of our model. The following ablation experiments were performed:

Table 4.4: Ablation Studies on FSBD(2x) on 5 way 3 shot

DA	WM	PS	ILN	3-shot mAP@0.5
				2.657
✓				8.91
✓		✓		11.01
✓			✓	11.08
✓	✓			24.19
✓	✓	✓		28.70
✓	✓	✓	✓	29.49

- (a) **Removing Domain adaptation, Warm Model, Pseudo Support Set and Instance Level Normalization:** Without incorporating any additional components or modifications as proposed in our model, the mean Average Precision at IoU threshold of 0.50 (mAP@0.5) achieved a low value of 2.657%. This outcome validates our hypothesis that without implementing the proposed enhancements, the model’s performance would not meet the desired expectations. Thus, it emphasizes the necessity of incorporating the proposed components to improve the overall performance and effectiveness of our model.
- (b) **Adding only Domain adaptation:** By incorporating Domain Adaptation with our base model, we have observed a significant improvement in the mean Average Precision at IoU threshold of 0.50 (mAP@0.5), achieving a value of 8.91%. This substantial increase in performance highlights the importance of Domain Adaptation in enhancing the model’s ability to adapt and generalize to different datasets. As a result, we have consistently utilized Domain Adaptation in our subsequent experiments and model iterations, as it has consistently proven to be effective in boosting the detection accuracy of our model.
- (c) **Adding Domain Adaptation and Pseudo-Support Set:** The integration of Domain Adaptation and the utilization of a Pseudo-Support Set have resulted in a mean Average Precision at IoU threshold of 0.50 (mAP@0.5) of 11.01%. By incorporating Domain Adaptation, our model has become more adept at adapting to the specific characteristics of the target dataset, enhancing its performance in terms of detection accuracy. The inclusion of the Pseudo-Support Set has further bolstered our model’s ability to handle the few-shot learning scenario by providing additional training samples that simulate novel class instances.
- (d) **Adding Domain Adaptation and Instance Level Feature Normalization:** The inclusion of Domain Adaptation and Instance Level Feature Normalization techniques has resulted in a mean Average Precision at IoU threshold

of 0.50 (mAP@0.5) of 11.08% . By incorporating Domain Adaptation, our model is able to adapt to the specific characteristics of the target dataset, improving its performance in terms of detection accuracy. Additionally, the application of Instance Level Feature Normalization has further enhanced the model's ability to reduce intra-class variance and improve the detection accuracy of novel classes.

- (e) **Adding Domain Adaptation, Warm Model:** The incorporation of Domain Adaptation and Warm Model techniques has significantly improved the performance of our model, resulting in a mean Average Precision at IoU threshold of 0.50 (mAP@0.5) of 24.19%. The addition of the Warm Model component alone has led to a substantial increase in performance, boosting the mAP@0.5 from 11.08% to 24.19%. This highlights the importance of using a warm model training approach, as it allows the model to adapt and fine-tune its parameters specifically to the target dataset, resulting in enhanced detection accuracy.
- (f) **Adding Domain Adaptation, Warm Model and Pseudo-Support Set:** The inclusion of Domain Adaptation and Warm Model techniques significantly improved the performance of our model, resulting in a mean Average Precision at IoU threshold of 0.50 (mAP@0.5) of 28.70%. This demonstrates the effectiveness of these strategies in enhancing the accuracy of our traffic sign detection system. By adapting the model to different domains and utilizing a warm model training approach, we were able to better capture the specific characteristics of the dataset and improve the overall detection performance. The higher mAP@0.5 achieved indicates the model's ability to accurately identify and localize traffic signs in diverse scenarios.
- (g) **Adding All the Components:** Inclusion of all the proposed components in our model architecture resulted in the highest mean Average Precision at IoU threshold of 0.50 (mAP@50) across various settings. This indicates that our model outperformed other state-of-the-art (SOTA) architectures in terms of detection accuracy. By integrating domain adaptation, warm model training, pseudo-support set augmentation, and instance level feature normalization, we were able to enhance the performance of our model and achieve superior results. The comprehensive combination of these components contributed to the improved accuracy and effectiveness of our traffic sign detection system.

Chapter 5

Conclusion

In this work, we propose a method for recognizing traffic signs using fine-tuning based few shot object detection. Since our presented method follows few-shot learning, it performs well even with a limited number of instances per class. Furthermore, our method uses a fine-tuning based model which exceeds the alternative approach which is meta-learning. We believe our proposed method will outperform all the existing works related to the recognition of traffic signs using a few shot object detection techniques. We can perform computational and analytical analysis after we are done experimenting with our proposed methodology. Apart from that, we intend to include the contrastive head used in FSCE [9] in our modified classifier head in the future to increase intra-class similarity and inter-class variance. We hope the addition of this contrastive head in our proposed method will increase our performance in the future.

References

- [1] M. Kumar, S. Gupta, and A. Garg, “Improved object recognition results using sift and orb feature detector,” *Multimedia Tools and Applications*, vol. 78, 12 2019.
- [2] A. Rosebrock, “Traffic sign classification with keras and deep learning,” *Traffic Sign Classification with Keras and Deep Learning*, vol. 11, no. 4, p. 12019, 2019.
- [3] C. M. Nestel, “Designing an experience: Maps and signs at the archaeological site of ancient troy,” *Cartographic Perspectives*, no. 94, pp. 25–47, 2019.
- [4] G. Han, S. Huang, J. Ma, Y. He, and S.-F. Chang, “Meta faster r-cnn: Towards accurate few-shot object detection with attentive feature alignment,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 1, 2022, pp. 780–789.
- [5] Q. Fan, W. Zhuo, C.-K. Tang, and Y.-W. Tai, “Few-shot object detection with attention-rpn and multi-relation detector,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4013–4022.
- [6] X. Wang, T. E. Huang, T. Darrell, J. E. Gonzalez, and F. Yu, “Frustratingly simple few-shot object detection,” *arXiv preprint arXiv:2003.06957*, 2020.
- [7] K. Guirguis, A. Hendawy, G. Eskandar, M. Abdelsamad, M. Kayser, and J. Beyerer, “Cfa: Constraint-based finetuning approach for generalized few-shot object detection,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 4039–4049.
- [8] L. Qiao, Y. Zhao, Z. Li, X. Qiu, J. Wu, and C. Zhang, “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.

- [9] B. Sun, B. Li, S. Cai, Y. Yuan, and C. Zhang, “Fsce: Few-shot object detection via contrastive proposal encoding,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7352–7362.
- [10] Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey,” *arXiv preprint arXiv:1905.05055*, 2019.
- [11] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M. P. Reyes, M.-L. Shyu, S.-C. Chen, and S. S. Iyengar, “A survey on deep learning: Algorithms, techniques, and applications,” *ACM Computing Surveys (CSUR)*, vol. 51, no. 5, pp. 1–36, 2018.
- [12] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–38, 2019.
- [13] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, “Generalizing from a few examples: A survey on few-shot learning,” *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.
- [14] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, “A review of yolo algorithm developments,” *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.
- [15] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [16] X. Ren, W. Zhang, M. Wu, C. Li, and X. Wang, “Meta-yolo: Meta-learning for few-shot traffic sign detection via decoupling dependencies,” *Applied Sciences*, vol. 12, no. 11, p. 5543, 2022.
- [17] Y.-X. Wang, D. Ramanan, and M. Hebert, “Meta-learning to detect rare objects,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9925–9934.
- [18] X. Yan, Z. Chen, A. Xu, X. Wang, X. Liang, and L. Lin, “Meta r-cnn: Towards general solver for instance-level low-shot learning,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9577–9586.
- [19] A. Nichol, J. Achiam, and J. Schulman, “On first-order meta-learning algorithms,” *arXiv preprint arXiv:1803.02999*, 2018.
- [20] A. Bellet, A. Habrard, and M. Sebban, “Metric learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 9, no. 1, pp. 1–151, 2015.
- [21] Y. Guo, H. Shi, A. Kumar, K. Grauman, T. Rosing, and R. Feris, “Spot-tune: transfer learning through adaptive fine-tuning,” in *Proceedings of the*

- IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4805–4814.
- [22] W. Xiong and L. Liu, “Cd-fsod: A benchmark for cross-domain few-shot object detection,” *arXiv preprint arXiv:2210.05311*, 2022.
- [23] F. Rahutomo, T. Kitasuka, and M. Aritsugi, “Semantic cosine similarity,” in *The 7th international student conference on advanced science and technology ICAST*, vol. 4, no. 1, 2012, p. 1.
- [24] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, vol. 28, 2015.
- [25] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, “Siamese neural networks for one-shot image recognition,” in *ICML deep learning workshop*, vol. 2. Lille, 2015, p. 0.
- [26] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, “Siamrpn++: Evolution of siamese visual tracking with very deep networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4282–4291.
- [27] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Mallocci, A. Kolesnikov *et al.*, “The open images dataset v4,” *International Journal of Computer Vision*, vol. 128, no. 7, pp. 1956–1981, 2020.
- [28] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.
- [29] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, “Supervised contrastive learning,” *Advances in Neural Information Processing Systems*, vol. 33, pp. 18 661–18 673, 2020.
- [30] G. Elsayed, D. Krishnan, H. Mobahi, K. Regan, and S. Bengio, “Large margin deep networks for classification,” *Advances in neural information processing systems*, vol. 31, 2018.
- [31] Y. Li, H. Zhu, Y. Cheng, W. Wang, C. S. Teo, C. Xiang, P. Vadakkepat, and T. H. Lee, “Few-shot object detection via classification refinement and distractor retreatment,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 395–15 403.

-
- [32] Z. Tian, C. Shen, H. Chen, and T. He, “Fcos: Fully convolutional one-stage object detection,” in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 9627–9636.