FSDCNN: A FEW SHOT DETECTION MECHANISM THAT PRESERVES ITS SUPERVISED NATURE

A Thesis

by

MAYANK AGARWALA

B.Tech, Vellore Institute of Technology, 2019

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

May 2021

May 2021

FSDCNN: A FEW SHOT DETECTION MECHANISM THAT PRESERVES ITS SUPERVISED NATURE

A Thesis

by

MAYANK AGARWALA

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Scott A. King, PhD
Chair

Longzhuang Li, PhD
Committee Member

Minhua Huang, PhD
Committee Member

May 2021

ABSTRACT

Object detection has become better with the advent of deep convolution neural networks. However, the challenge of training a fully supervised system when there is a small amount of training samples available still remains. Another issue with fully supervised systems is seen upon encountering novel classes. It is difficult to retrain the model as it is a time-consuming and tedious process. Inspired by a human's ability to learn at a rapid rate, few-shot learning models have seen rapid development. In contrast to fully supervised systems, these learn from just a few samples. We propose a few-shot detection model, FSDCNN, based on a two-stage detector, that optimizes both the region proposal network and the object detector with the help of few-shot learning. FSDCNN performs similar to other models when only 1 or 3 new samples are seen but outperforms them when 5 or 10 samples of the new classes are seen and it preserves the fully-supervised nature of the base two stage detector.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my supervisor and committee chair, Dr. Scott A. King, for his invaluable guidance, continued support, and patience during my thesis. His experience and wisdom as a researcher and a professor along with his ability to make time for my study have encouraged me to complete this thesis. I would also like to thank the committee members, Dr. Longzhuang Li and Dr. Minhua Huang, for their support as professors over the past two years. Their classes have helped me sharpen my technical skills.

I am grateful to my friends and members of iCORE group for their advice, support, and encouragement. The meetings have been very informative and motivational, especially during the trying times of the pandemic. They have also been helpful in providing constructive feedback throughout the thesis.

Finally, I would like to thank my parents and the rest of my family who have been a pillar of support, without whom I wouldn't have been able to see this through. The same goes for my friends back in India who despite being thousands of miles away, kept me sane during this pandemic.

Thank you.

DEDICATION

I would like to dedicate this thesis to my parents, my brother and the rest of my family on both sides of my parents.

I would also like to honor all the teachers who have helped me in one way or the other over the years. Though I cannot mention every name here, some need special mentions for their humongous support. Priya Miss, the matron who looked after me during my time in boarding school, Isha ma'am, who played the role of an elder sister during the last two years of my high school and Dr. Khan, my mentor in my undergraduate years.

I dedicate this to my friends as well for their continued support and encouragement.

TABLE OF CONTENTS

CONTENTS PAGE

ABSTRACT iv

ACKNOWLEDGEMENTS v

DEDICATION vi

TABLE OF CONTENTS vii

LIST OF FIGURES ix

LIST OF TABLES xiv

CHAPTER I: INTRODUCTION 1

1.1 Problem Statement 2

1.2 Contributions 3

1.3 Outline 3

CHAPTER II: REVIEW OF THE LITERATURE 4

2.1 Object Detection 5

2.2 Cross-domain Object Detection 5

2.3 Few-Shot Learning 6

Matching Networks 6

Prototypical Networks 7

Meta Learning 7

Metric Learning 8

CHAPTER III: SYSTEM DESIGN 10

3.1 Problem Setting 10

3.2 Faster r-CNN 11

3.3 Methodology 12

Meta-Training 13

Meta-Testing 16

CHAPTER IV: EXPERIMENTS 17
CHAPTER V: RESULTS 19
5.1 Discussion 20
CHAPTER VI: SUMMARY AND CONCLUSIONS 56
NOTES 59
REFERENCES 60

LIST OF FIGURES

FIGURES PAGE

3.1 System Overview: The Dense CNN for support and query set is the same. The Meta-learner module is for the entire feature map generation and region proposal network. The idea is to learn which regions should be considered for objects. 12

3.2 The proposed architecture in [1]. It uses a class-attentive vector and perform a channel-wse product with the RoI features. Meta-learner is used after the RoI stage. 15

5.3 The plot shows mAP values obtained by the different models for novel classes for the different K-shot settings 20

5.4 The mAP increases with the increase in number of samples available for a novel class. This shows that the supervised nature of the base model isn't compromised. 21

5.5 The mAP achieved for novel classes bird, bus, cow, mbike and sofa over thousand runs in 1-shot setting. 21

5.6 The mAP achieved for novel classes aero, bottle, horse, boat and cat over thousand runs in 1-shot setting. 22

5.7 The mAP achieved for novel classes sheep, plant, tv, dog and bike over thousand runs in 1-shot setting. 22

5.8 The mAP achieved for novel classes train, person, table, car and chair over thousand runs in 1-shot setting. 23

5.9 The mAP achieved for novel classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 1-shot setting. 23

5.10 The mAP achieved for novel classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 1-shot setting. 24

5.11 The mAP achieved for novel classes bird, bus, cow, mbike and sofa over thousand runs in 3-shot setting. 24

5.12 The mAP achieved for novel classes aero, bottle, horse, boat and cat over thousand runs in 3-shot setting. 25
5.13 The mAP achieved for novel classes sheep, plant, tv, dog and bike over thousand runs in 3-shot setting. 25
5.14 The mAP achieved for novel classes train, person, table, car and chair over thousand runs in 3-shot setting. 26
5.15 The mAP achieved for novel classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 3-shot setting. 26
5.16 The mAP achieved for novel classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 3-shot setting. 27
5.17 The mAP achieved for novel classes bird, bus, cow, mbike and sofa over thousand runs in 5-shot setting. 27
5.18 The mAP achieved for novel classes aero, bottle, horse, boat and cat over thousand runs in 5-shot setting. 28
5.19 The mAP achieved for novel classes sheep, plant, tv, dog and bike over thousand runs in 5-shot setting. 28
5.20 The mAP achieved for novel classes train, person, table, car and chair over thousand runs in 5-shot setting. 29
5.21 The mAP achieved for novel classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 5-shot setting. 29
5.22 The mAP achieved for novel classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 5-shot setting. 30
5.23 The mAP achieved for novel classes bird, bus, cow, mbike and sofa over thousand runs in 10-shot setting. 30
5.24 The mAP achieved for novel classes aero, bottle, horse, boat and cat over thousand runs in 10-shot setting. 31

5.25 The mAP achieved for novel classes sheep, plant, tv, dog and bike over thousand runs in 10-shot setting. 31
5.26 The mAP achieved for novel classes train, person, table, car and chair over thousand runs in 10-shot setting. 32
5.27 The mAP achieved for novel classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 10-shot setting. 32
5.28 The mAP achieved for novel classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 10-shot setting. 34
5.29 Comparison of mAP achieved by the models over different k-shot settings. 36
5.30 Comparison of mAP achieved by Meta R-CNN and FSDCNN over different k-shot settings. 36
5.31 Comparison of mAP achieved by YOLO-Few-Shot and FSDCNN w/o over different k-shot settings. 37
5.32 Comparison of mAP achieved by FSDCNN w/o and FSDCNN over different k-shot settings. This shows that the meta-learner in FSDCNN helps in achieving better performance. 38
5.33 The mAP achieved for base classes bird, bus, cow, mbike and sofa over thousand runs in 1-shot setting. 39
5.34 The mAP achieved for base classes aero, bottle, horse, boat and cat over thousand runs in 1-shot setting. 39
5.35 The mAP achieved for base classes sheep, plant, tv, dog and bike over thousand runs in 1-shot setting. 40
5.36 The mAP achieved for base classes train, person, table, car and chair over thousand runs in 1-shot setting. 40
5.37 The mAP achieved for base classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 1-shot setting. 41

5.38 The mAP achieved for base classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 1-shot setting. 41
5.39 The mAP achieved for base classes bird, bus, cow, mbike and sofa over thousand runs in 3-shot setting. 42
5.40 The mAP achieved for base classes aero, bottle, horse, boat and cat over thousand runs in 3-shot setting. 42
5.41 The mAP achieved for base classes sheep, plant, tv, dog and bike over thousand runs in 3-shot setting. 43
5.42 The mAP achieved for base classes train, person, table, car and chair over thousand runs in 3-shot setting. 43
5.43 The mAP achieved for base classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 3-shot setting. 44
5.44 The mAP achieved for base classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 3-shot setting. 44
5.45 The mAP achieved for base classes bird, bus, cow, mbike and sofa over thousand runs in 5-shot setting. 45
5.46 The mAP achieved for base classes aero, bottle, horse, boat and cat over thousand runs in 5-shot setting. 45
5.47 The mAP achieved for base classes sheep, plant, tv, dog and bike over thousand runs in 5-shot setting. 46
5.48 The mAP achieved for base classes train, person, table, car and chair over thousand runs in 5-shot setting. 46
5.49 The mAP achieved for base classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 5-shot setting. 47
5.50 The mAP achieved for base classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 5-shot setting. 47

5.51 The mAP achieved for base classes bird, bus, cow, mbike and sofa over thousand runs in 10-shot setting. 48
5.52 The mAP achieved for base classes aero, bottle, horse, boat and cat over thousand runs in 10-shot setting. 48
5.53 The mAP achieved for base classes sheep, plant, tv, dog and bike over thousand runs in 10-shot setting. 49
5.54 The mAP achieved for base classes train, person, table, car and chair over thousand runs in 10-shot setting. 49
5.55 The mAP achieved for base classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 10-shot setting. 50
5.56 The mAP achieved for base classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 10-shot setting. 50
5.57 Violin plot showing the distribution of mAP values for 1-shot setting over 1000 runs. . 52
5.58 Violin plot showing the distribution of mAP values for 3-shot setting over 1000 runs. . 53
5.59 Violin plot showing the distribution of mAP values for 5-shot setting over 1000 runs. . 54
5.60 Violin plot showing the distribution of mAP values for 10-shot setting over 1000 runs. . 55

LIST OF TABLES

TABLES PAGE

5.1 Performance comparison of the different models in the experimented 4 Novel/Base split settings for 1-shot, 3-shot, 5-shot and 10-shot. Our model performs really well in 10-shot. Furthermore, the deviation of mAP for different settings is less. 33

5.2 Performance comparison of the different models in classifying each of the 20 unseen classes over a thousand runs for 1-shot, 3-shot, 5-shot and 10-shot. (Color guide: red is for the best performance, green is for second best and blue is for third best.) 35

5.3 Performance comparison of the different models in classifying each of the 20 base classes over a thousand runs for 1-shot, 3-shot, 5-shot and 10-shot. (Color guide: red is for the best performance, green is for second best and blue is for third best.) 51

CHAPTER I: INTRODUCTION

With advances in the field of deep learning, especially convolution networks [2, 3, 4], the performance of object detectors and image classifiers have seen quite a boost in recent years. However, the time-consuming nature of these state-of-the-art models mean that when there is a large-scale dataset, the amount of time spent for training the models is very high. These systems are also data-hungry, meaning inadequate number of training images will limit performance of the systems. In addition to this, the objects in the data need to be annotated with precise bounding boxes, which is a laborious task. If a user wishes to expand the categories of objects that can be recognized by these systems those new objects need to be annotated in the existing dataset and quite likely new images are needed which also need annotation. Finally, the resulting dataset will be used to train a final model. It is crucial to have enough training samples for the new image class in order to avoid overfitting on a small population of data. The generalization of these systems is usually very poor.

It is worth noting that the applications of systems that learn from a few samples for image classification and object detection in general are many. Few-shot learning is defined as the art of training a model from a few samples in contrast to the large number of samples used by fully supervised learning. As applications for real-time object detection increases, especially in real-world applications, the ability of few-shot detection mechanisms to recognize novel categories with just a few examples or a few updates has great utility.

Our goal is to create a model that can perform reasonably well on novel classes with just a few examples without the need to retrain. The aim is also to preserve the fully-supervised nature of the base model, that is, the model should perform better with the increase in the number of training samples. Besides this, Few-Shot object detection has also been explored in the field of Hyperspectral Image classification [5, 6]. Few-Shot learning is also applicable in the field of sound recognition [7], however, it lies beyond the scope of research of this article.

1.1 Problem Statement

The inspiration for few-shot object classification has been drawn from human's ability to learn novel object categories with only a few examples. The existing few-shot detection models, despite performing well on novel classes with few examples, are nowhere as good in comparison to fully supervised state-of-the-art models when sufficient examples are available to train the model. However, in real-world scenarios, it is a given that new objects or circumstances will need to be tackled. In such scenarios, we believe it would not be ideal to retrain a fully supervised model that can recognize the new object class. There are two reasons for this. One reason is that there might not be sufficient samples available to train a deep neural network. The second is that even if there were enough samples available, it would be very time consuming to retrain the model.

The main problem statement of this project is to come up with a model that can perform reasonably well on novel classes with just a few examples without the need to retrain. The goal is to preserve the fully-supervised nature of the base model, which is, the performance of the model is enhanced with the increase in the number of training samples. Ideally, the performance of a fully supervised model would have a better performance when there are 5 samples for a class of objects in contrast to when there are 3 samples. The idea is to address the problem of few-shot learning from the perspective of meta learning in order to preserve the performance on base classes. We use meta learning approach for Few-Shot detection which deploys a two-stage detection mechanism as object detector.

It is also worth noting that the applications of Few-Shot learning for image classification and object detection in general are many. In the age, where autonomous vehicles are a reality, real-time object detection is a necessity. This is where the capability of Few-Shot detection mechanisms to recognize novel categories with few examples or a few updates, will come in handy.

It is quite evident that there has been quite a lot of work in the field of Few-Shot detection that have laid the baseline for future work. This area of research has a promising potential.

1.2 Contributions

The main contribution is a system (FSDCNN) that makes use of the Few-Shot paradigm on both stages of a two-stage detector. Few shot learning is carried out by using two different mechanisms, matching and meta-learning. The two stages in a two-stage detector are the region proposal stage, which identifies the region with features to use for classification and the classification stage. This in addition to meta learning for the optimization of the model, enhances the performance of the model on novel classes and preserves the performance of the model on base classes. The system also preserves the supervised learning nature of the base model.

1.3 Outline

Chapter II provides a review of the existing methods to solve the problem of classifying images with few samples and the methods used in this research. It is followed by Chapter III that outlines the proposed method and the system overview. The conducted experiments are discussed in Chapter IV, and the evaluations and results are discussed in Chapter V. We close with conclusions and future work in Chapter VI.

CHAPTER II: REVIEW OF THE LITERATURE

One approach to classify objects is to use the guidance of a weakly supervised learning mechanism. Though this reduces the dependence of performance on the annotated data, the number of samples required to train the data are similar. Saenko et al. [8] suggest a method to transfer knowledge from one domain to another. The issue with such an approach is that often in these cases, the dataset bias hampers the generalization of these models. Saito et al. [9] propose a mechanism based on weak global domain alignment which enhances the performance of the baseline Faster R-CNN [10] by using a domain classifier.

On the other hand, the human visual system is capable of easily learning novel object categories, or concepts in general, with one or few examples, and later recognize them by relying on these encounters. The assumption behind this is that the human system is able to exploit its past experiences about the world. This inspired the concept of few-shot learning [11, 12] and zero-shot learning. Few-shot learning is comprised of techniques where models are able to recognize new categories based on updates of a few steps or even at once with few examples [11, 12]. While some [13, 14, 15] address the few-shot learning mechanism from the perspective of meta-learning, the others address it from the perspective of metric learning . Few-shot learning has also been explored in other domains such as Hyperspectral Image classification [5, 6] and sound recognition [7].

One could argue that there was an issue with Few-Shot detection mechanism as the researchers, in the beginning, had neglected the need of remembering the base categories. In other words, when the model kept on updating itself to be able to recognize the novel categories, it had a tendency to forget the base categories. However, Gidaris and Komodakis [16] developed an approach that was able to counter this. In fact, their model beat the state-of-the-art mechanisms by a significant margin.

Fu et al. [14] created a novel approach that addressed few-shot learning from the perspective

of meta-learning. Their system had two components to it which helped them in attaining a good performance and forming a baseline for further research into the field of meta-learning based few-shot detection. The results were promising but it still remains a field with minimal exploration, which is why many researchers are keen to explore this further. However, they did not make use of a two-stage detector as we do here. We extend the work of Fu et al. to use a two-stage detector and use meta-learning on both the stages of it.

Mishar et al. [17] developed a mechanism for zero-shot learning and generalized it to recognize actions on the basis of few-shot learning. Their results show performance equivalent to that of state-of-the-art with action recognition performing on par with fully supervised learning mechanisms.

2.1 Object Detection

With improvements in deep neural networks [2], the performance of object detection mechanisms has been quite remarkable. They can be classified into two categories: one-stage and two-stage or multi-stage detectors. However, all of them require very large datasets to be trained, which is burdensome for annotation and time-consuming. This proves that they cannot be directly applicable to unseen domains.

2.2 Cross-domain Object Detection

Most of the recent works on domain adaptation with the help of CNNs mainly address the simple task of classification. Only few of them consider object detection. For example, [18] proposed a framework weaken the domain shift problem of deformable part-based model. Domain adaptation for R-CNN based on subspace alignment was proposed in [19]. A two-stage iterative domain transfer and pseudo-labeling approach was proposed in [20] which helps to tackle cross-domain weakly supervised object detection. The authors of [21] came up with three modules for unsupervised domain adaptation of the object detector.

2.3 Few-Shot Learning

Few-shot learning was introduced to learn a new class of objects with very few examples [11]. Initially, most of the few-shot learning classifiers were able to outperform the state-of-the-art fully supervised models for novel classes. However, they fell short when it came to classifying the base classes. The later works explore the feasibility of devising a mechanism that would not only perform well on new classes, but also maintain their performance levels for base classes. Lu et al. [22] introduce a prototypical network for few-shot learning tasks with the introduction of a domain alignment module. This module takes into account the domain shifts between existing categories. Compared to the existing simple Prototypical Networks, the proposed system is able to abate the distribution differences among the data of training and test classes, further optimizing the embedding space of prototype features for each category and then boosting few-shot recognition. A method that exploits information across label semantics and image domains is proposed by Chu and Wang [23]. This ensures that regions of interest are carefully attended which will help in better classification. The proposed module is able to focus on the most relevant regions in an image, while the attended image samples allow data augmentation and alleviate possible overfitting during Few-Shot Learning training. Rahman et al. [24] propose a generalized few-shot learning approach based on Class Adaptive Principal Directions. This approach allows multiple embedding of image features into semantic space.

Matching Networks

Matching Networks [25] were the first that were trained and tested on K-shot M-way tasks. The idea of this is simple — training and testing on similar tasks improves the performance for the target task in a start to finish style. Prior methodologies, for example, siamese networks [26], utilize a pairwise verification loss to perform metric learning and afterward in a different stage utilize the metric space to perform nearest neighbors classification. This isn't ideal as the initial function

is actually trained to optimize performance on a different task altogether. However, Matching Networks form an end-to-end nearest neighbor classifier that is differentiable, by the combination of both embedding and classification.

Prototypical Networks

Snell et al. [27] introduce prototypical networks. A compelling inductive bias is applied in the form of class prototypes, aimed at achieving a very impressive Few-shot performance, outperforming Matching Networks [25]. By assuming that there exists an embedding in which samples from each and every class cluster around a single prototypical representation that is simply the mean of the individual samples Prototypical Networks streamline n-shot classification in the case of $n > 1$ to just taking the label of the closest class prototype. This removes the need for full context embeddings used in Matching Networks [25].

Meta Learning

Meta-learning is the basis for the majority of few-shot learning mechanisms. The literature has seen meta-learning being used for roughly four categories of tasks. The first category uses meta-learning to accelerate the fine-tuning process by learning how to initialize the parameters [28, 29]. Another category is to use meta-learning to store training samples or to encode adaptation algorithms [30, 31]. A third category uses meta-learning to generate new samples for novel classes [32] and the fourth category comprise of works that focus on learning the parameters for image classification using meta-learning [33, 16].

Gidaris and Komodakis [16] suggest a dynamic few-shot mechanism that performs well on the base classes as well as novel classes. This is achieved by extending the object detector, in this case, a cosine-similarity based convolution neural network, with an attention based few-shot classification weight generator. They also propose to redesign the classifier of a convolution neural network model as the cosine similarity function between feature representations and classification

weight vectors. They use parametric classifier in both stages of their system. Their model learns in episodes in stage 2, just like the work suggested by Lifchitz et al. [34]. Instead of remembering the training data for base classes, it stores the weight parameters of the base classes.

Qi et al. [35] suggest a method where the weights of the final layers of a convolution network are set directly by the novel class examples. They name this process as imprinting weights. Though it uses a parametric classifier in both its stages just like Gidaris and Komodakis, the difference is that it remembers the entire training data for the base class examples.

Ye et al. [15] propose an approach called Meta-relations which is based on relation networks and Model Agnostic Meta Learning training methods. This model is trained end-to-end. This approach can quickly learn a novel class from limited samples, after the initial training with the help of Model Agnostic Meta Learning. It can also classify new classes of images by calculating the scores between query images and few examples of each new class.

The concepts of transfer learning and meta-learning are used in combination to come up with an approach called Meta-Transfer Learning (MTL) in [13]. The authors achieve transfer learning by learning how to scale and shift functions of the weights of deep neural networks for each task. The authors also propose a new strategy called Hard Task Meta batch scheme as their learning strategy. The proposed system yields good performance and fast convergence.

Metric Learning

Metric Learning is a task that learns the distance function over objects. It is a common approach used in few-shot learning. Some work [36, 37, 38, 39] has proposed improvements of standard entropy and cross-entropy loss functions to address few-shot problem from the perspective of metric learning. Siamese networks, which is a traditional method, is also considered along with different models that acquire knowledge by comparing two objects. Quite a number of proposed methods in the literature deploy a parametric classifier [16, 34, 35] rather than metric learning in general. Lifchitz et al. [34] propose a variant of a parametric classifier for few-shot detection in their stage

1 where, instead of the usual pooling or flattening, their approach deploys a dense classifier. This dense classifier, instead of considering the most discriminative features in an image, takes every tiny detail into account. In stage 2, they learn in episodes and have come up with a mechanism called *implants*. This freezes the base parameters of the model when the model is encountering novel classes. This avoids the performance drop of models when it comes to base models. Their proposed method outperformed most of the state-of-the-art mechanisms by a huge margin. Karlinsky et. al. [40] develop a new approach called RepMet, which uses a Representative-based Metric Learning for object detection. Their proposed method is able to learn network parameters, the embedding space and the multi-modal distribution of training categories in the space, in one training process.

An approach of imprinting weights is proposed in [35]. The authors propose a method of setting weights of the final layer of convolution neural networks for the classification of novel classes. The inspiration for this approach was drawn from the human capability of identifying new categories from limited examples. After training a base classifier, the embedding vectors of new low-shot examples are used to imprint weights for new classes in the extended classifier.

CHAPTER III: SYSTEM DESIGN

This section is divided into three sections. The first section highlights the problem setting for few-shot detection. An overview of Faster r-CNN is discussed in the following section. We outline and explain our proposed model in the third section which includes a deeper understanding of meta-training and meta-testing.

3.1 Problem Setting

Assume there are two sets of images to be classified. The first set of images consists of classes of images that have a large number of samples, denoted by L. The second set of images consists of classes of images that have a small number of samples, denoted by S. L has a set of categories denoted by L_c and S has a set of categories denoted by S_c. None of the categories are present in both. Our goal is to train a model using the annotated images in L corresponding to L_c and S_c which will be able to detect the images in S that are not labelled.

The aim of the proposed system is to come up with a generalized detection framework with adaptability to detect different categories with limited annotated training samples. The standard method of meta learning has been followed. Learning has been split into two stages: meta-training followed by meta-testing, and optimization of model over various few-shot tasks which are simulated from the data for meta-training. To be specific, we sample few-shot tasks from the large sample L during meta-training. A support set of images and a query set are present in each and every few-shot task. For a given task T_i, K images each of M ways (or classes) are chosen in random from L_c in order to make a support set $\mathrm{T}_i^{\mathrm{L,s}}$. With Q images per class chosen at random from L, we make the query set $\mathrm{T}_i^{\mathrm{L,q}}$.

The support set is used for optimization whereas predictions are made on the query set. Calculated loss on the query set is used to update the model. Few-shot tasks sampled from the smaller

set S are used for the meta-testing phase. The obtained results averaged across multiple few-shot tasks is then used to evaluate the performance of the few-shot model.

3.2 Faster r-CNN

The state-of-the-art detection model, Faster r-CNN, proposed by Ren et al. [10] has been used as our base model. Faster r-CNN is made up of two components: the region proposal network (RPN) and Fast r-CNN. The RPN is used for proposal generation whereas the Fast r-CNN is used for region classification. RPN generates the proposals which are then classified into categories by the region classifiers in Fast r-CNN. To be specific, a feature vector is extracted by the RPN from each region. This is done by scanning the whole image with the use of sliding windows. A binary classifier to classify the objects vs. background and a bounding box regressor to filter the easy negatives follows the RPN. RoI Pooling Layers are used to extract a feature vector of fixed length for each proposal. This feature vector is fed into a sequence of densely connected layers that have two output branches. One of those branches represents the softmax probability over $M+1$ classes, that is, M classes and one background. The other output branch is responsible for encoding four real-values that refine the bounding box position. The network can be optimized by minimizing loss which is calculated using the loss function:

$$L(p,x,b^x,y) = L_{\text{cls}}(p,x) + \lambda\,[x \geq 1] L_{\text{loc}}(b^x,y) \tag{3.1}$$

where x denotes the class and y denotes bounding box label. p denotes the predicted probability distribution over C classes. b^x denotes bounding box prediction of a class x. The trade-off parameter is denoted by λ. L_{cls} is used to denote softmax loss whereas L_{loc} denotes SmoothL1 loss.

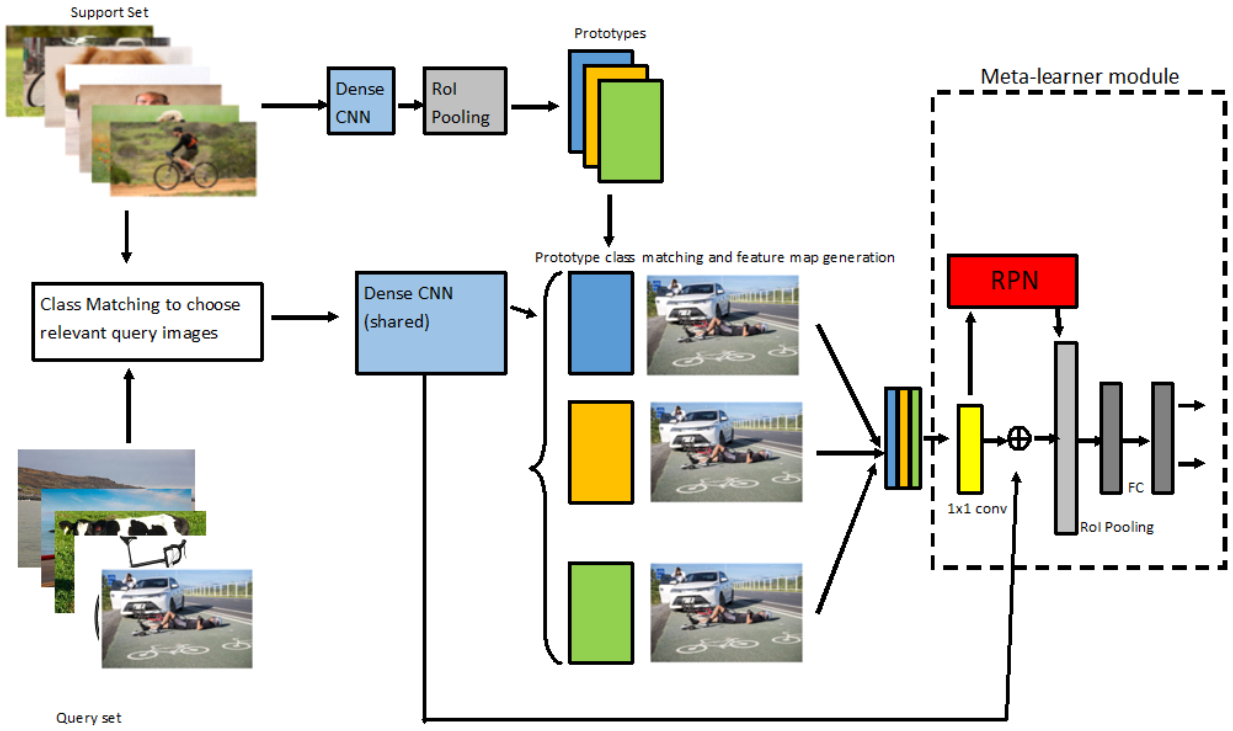

Figure 3.1
System Overview: The Dense CNN for support and query set is the same. The Meta-learner module is for the entire feature map generation and region proposal network. The idea is to learn which regions should be considered for objects.

3.3 Methodology

We combine Faster R-CNN, a two-stage detector with meta-learning to perform few-shot image classification. The main idea behind using a two-stage detector is that the performance of these models is better than the one-stage detectors. Faster R-CNN is about ten times faster than Fast R-CNN. Faster R-CNN uses region proposal network (RPN) which reduces the computational requirements.

We use meta-learning to not only learn the detection branch of the Faster R-CNN but also the RPN branch in order to learn where to look in the images to extract features for classification.

The learning will take place in two stages and the tasks are performed in episodes. The dataset is divided into two parts, the meta-train set and the meta-test set. In every episode, there is training and testing, which is on the meta-train set. The training during every episode updates the Faster R-CNN. The images used in this phase are called Support Images, or images that belong to a base class. The testing phase of every episode is used to update the meta-learning loss. The images in this set are called Query Images, or images of a novel class. This episodic training and testing can be considered as a meta-training phase. The model is then tested upon the meta-test set. See figure 3.1 for an overview of the system.

The hyperparameters of the Faster R-CNN are updated after every episode or batch of images with the help of meta-information that the meta-learner has learned due to back propagation of meta-training loss. The meta-learner learns about RPN as well to choose regions for feature extraction and image classification. The meta-information updates the region feature map generator, which generates the prototypes or the categories of interest. It is further described in Section 3.3.

Meta-Training

For meta-training, several K shot-M class tasks are carried out from the dataset which is already annotated. To fit in the memory size of the training environment, the meta-training stage is used to train the model by implementing 1 shot-5 class tasks. In such tasks, only 5 query images are used, that is 1 query image per classes. Therefore, one task has 10 images as 1 image per class each in support set and query set. The implementation of meta-training is therefore not too difficult. Figure 3.1 outlines the system overview.

For every task T_i^L, the region features of the support images $T_i^{L,s}$ are generated by feeding them into the Faster R-CNN. In the given image, for every category of the object of interest, based on the respective region features, we generate a prototype P_c.

The generated prototypes are fed into a class matching network based on siamese networks. These help in choosing strong images from the query set $T_i^{L,q}$ on the basis of these generated

prototypes and fed into the same Faster R-CNN model. This results in obtaining an image feature map.

$$F_c = f \odot \phi(P_c) \tag{3.2}$$

where $\odot$ represents element-wise multiplication. ϕ is a fully connected layer that encodes P_c and f is a feature map.

A category-specific feature map is then generated for every category on the basis of the input query image and the respective prototype.

For each and every category c, there is a new generated feature map F_c that highlights the objects of c. All the generated feature maps are then concatenated to form a single feature map F. Feature map F is tasked with learning of general representation of M-classes and every sub-channel has the information of the distinct classes of interest. A 1x1 convolution layer based on the feature map F reduces the cost of computation. The Region Proposal Network is then used to produce region proposals. Since there was some loss in information, we need to combine the generated feature map F, which is an M-region classifier, with the original feature map using element-wise summation, and then crop the region features on the basis of a new generated map. At the end, an M+1 region classifier and the bounding box regressors are optimized using the information obtained from the query set $T_i^{L,q}$:

$$L(T_i^{L,q}, T_i^{L,s}, \theta) = L_{\text{RPN}} + L_{\text{cls}} + L_{\text{loc}} \tag{3.3}$$

where θ is the parameters of FSDCNN.

The generated loss at the end of this step needs to be minimized. The information from the meta-learner is used to alter the hyperparameters of the model to maintain the supervised nature of the model and to preserve the performance on base classes. The meta learner also updates the

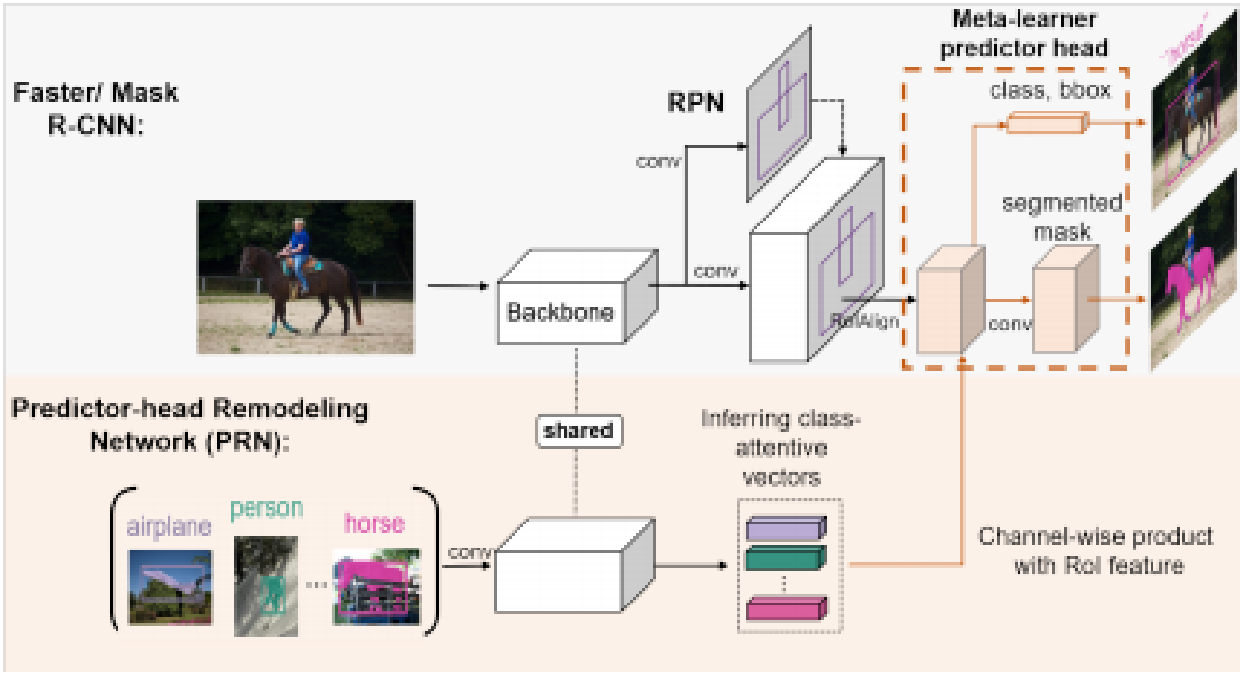

Figure 3.2
The proposed architecture in [1]. It uses a class-attentive vector and perform a channel-wse product with the RoI features. Meta-learner is used after the RoI stage.

network used for class matching for selection of query images. Our approach is inspired by [1] with changes to how the images are chosen, how and where meta-learning is applied and a few changes to the architecture of the system such as the use of a feature map generator in contrast to a class-attentive vector. Yan et. al. [1] have taken the class-attentive vectors and performed a channel-wise product with the RoI features. We use the generated prototypes along with the query image to generate a feature map before a 1x1 convolution after which the feature map passes through the RPN. To counter the loss of information due to 1x1 convolution, the feature map of the original query image is combined with the generated feature map using element-wise summation, after which RoI pooling is applied. Furthermore, our meta-learner learns the entire process from 1x1 convolution of the generated feature map to classification of the image in contrast to the use of meta-learner after the RoI stage in [1].

Meta-Testing

For meta-testing, several few-shot detection tasks are chosen from the set S_c. The images in the support set $T_i^{S,s}$ are annotated. Predictions are made on the query image set $T_i^{S,q}$ for the evaluation of the performance of the proposed system. As mentioned earlier, for each and every task T_i^S, we generate a prototype from the images available in the support set $T_i^{S,s}$. This is later used in the generation of new category-specific feature maps of those images that are present in the query set $T_i^{S,q}$. The model is fine-tuned in this stage with the use of labeled images in the support set. This fine-tuning addresses the limitation one encounters when learning of non-parametric methods when there are more labeled images that are provided. At the end, the output from the query set is evaluated as a traditional detection problem.

$$p_c, b_c = \text{FSDCNN}(T_i^{S,q}, T_i^{S,s}, \theta) \tag{3.4}$$

where p_c represents the probability vector of classes and b_c is a set of location of bounding boxes.

CHAPTER IV: EXPERIMENTS

We use PASCAL VOC dataset [41] to train and test our model. PASCAL VOC is a collection of datasets for object detection. The images in the dataset have been obtained from flickr. It provides standardized image datasets for object class recognition. It has been widely used to benchmark most of the existing works in the literature. The PASCAL VOC dataset has 20 classes. We setup the experiment in such a manner that there are four different splits for the novel/base classes. Each split has 15 base classes and 5 novel classes. As there are 20 classes in the dataset, we make four different novel/base splits. Each of the classes appear in the novel split only once to avoid overlaps.

During meta-training, a total of one thousand distinct tasks are generated. Each class has 10 images in the query set in order to update the model weights for 10 epochs. We set the initial learning rate to 0.001 and reduce it to 0.0001 after 600 tasks. The batch size is set to 5 during query updates.

The parameters of FSDCNN are set up similar to the vanilla Faster R-CNN [10]. If there is a proposal overlap of objects that is greater than 0.5, it is considered as positive. The ones below 0.3 are considered negative. The ones between 0.3 and 0.5 are not seen. We select the top 128 confident proposals for training and the top 300 proposals that have the largest confidence are selected for evaluation. The proposed FSDCNN model is based on ResNet101 which is pretrained on ImageNet.

Two versions of the model are trained and evaluated. One with the use of meta-learning that implements the meta-learning at the last stage that updates the feature map generation between tasks. The other model does not include the meta-learning optimization of the feature map generation. The main idea behind having two different models was to evaluate the performance of the models, one with meta-training of the feature map generator and the use of few-shot paradigm on the query set and the other model with just the few-shot paradigm. In the meta-training phase, the

base classes are used whereas in the meta-testing phase novel classes are used. We test the model on 1-shot, 3-shot, 5-shot and 10-shot. We have performed ten runs on every split for each of the K-shot settings.

A benchmark testbed was built on PASCAL VOC in order to evaluate the performance of the few-shot object detection in meta-learning settings. For the chosen benchmark, the proposed system is evaluated on numerous tasks that are set up with different K shot settings and splits. We use mean Average Precision (mAP) to evaluate the performance of our proposed model. The mean Average Precision (mAP) is defined as the mean of all average precision of the different novel classes encountered by the model. mAP is given by:

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q} \tag{4.5}$$

where Q is the number of queries in the set and *AveP(q)* is the average precision for a given query q.

After the initial set of experiments, whose results are discussed in Section **??**, we performed some more analysis whose results are displayed in Section 5.1. To be surer of our results, we perform the experiments a thousand times with different permutation and combinations of Novel/Base splits in contrast to the four distinct Novel/Base splits taken before. During each of these thousand runs, we recorded the average precision values of each class when it was novel and base separately. The mean of average precision of each class in the thousand runs in calculated to see the performance of our model in each of the classes in comparison to other models. The performance on base classes was recorded to check if there was any drop in performance level over them. This was done to see if the model is able to remember the base classes.

CHAPTER V: RESULTS

We compare the performance of our model on novel classes against the performances of the model Meta R-CNN [1], Faster R-CNN [10], Faster R-CNN-dual (it has been trained on both base and novel classes), Meta-SSD [14] and YOLO-Few-Shot [42]. We are comparing against Meta R-CNN [1] because it is a bit similar to ours as it is also based on Faster R-CNN. The performance against Faster R-CNN [10] is compared as it is our base model. The performances of Meta-SSD [14] and YOLO-Few-Shot [42] are compared as they are one-stage detectors that have addressed the few-shot problem from a similar perspective as ours. The results are displayed in Figure 5.3 which shows the mAP over novel classes for our comparison group. FSDCNN w/o denotes our model without meta-learning whereas FSDCNN is our model with meta-learning. Meta-SSD performs the best for 1-shot, but it improves very slowly as more examples of the novel classes are seen. FSDCN and Meta R-CNN show the best performance of the comparison group across all test, with Meta R-CNN having a slight edge in 1-shot and 3-shot, however, our model outperforms it in 5-shot and 10-shot.

Upon further analysis into the performance of each model over the four selected Novel/Base split settings, it was seen that the performance of our model was more consistent. Table 5.1 shows the tabulated results for each Novel/Base split setting for all the four K-shot settings. These are the averages of 10 runs of the experiment.

Figure 5.4 shows that the mAP increases with the increase in number of samples of novel classes. As we know, a fully supervised learning system performs better with the increase in number of training samples up to a certain point. This indicates that the proposed system doesn't compromise with the supervised nature of the base Faster R-CNN in order to perform better on novel classes.

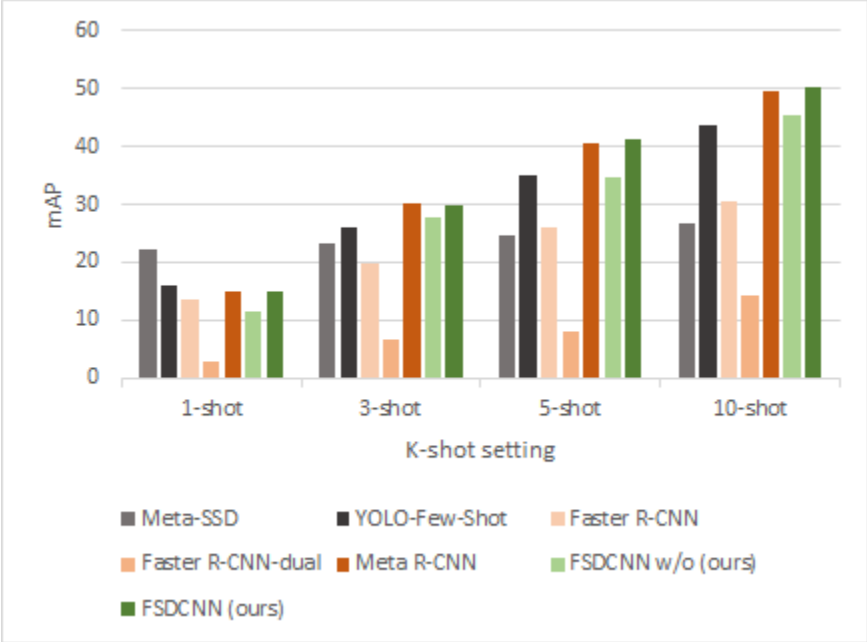

Figure 5.3
The plot shows mAP values obtained by the different models for novel classes for the different K-shot settings

5.1 Discussion

After receiving the initial results, we performed one thousand runs over the PASCAL VOC dataset with different permutations and combinations of Novel/Base splits. The results for the different K-shot settings were recorded and plotted for each of the models. Instead of taking the mAP over all the classes, we took mAP for each class when it was a novel class over the thousand runs.

Figures 5.5- 5.10 display the performances of the models on the 20 classes for 1-shot setting. Meta-SSD performs consistently better in classifying bird, sofa, sheep, plant, tv, bike, train, table and chair. However, our model is close to Meta-SSD's performance for classes mbike, bike, train and chair. It outperforms all the other models in classifying cars and persons. Our model's performance is close to that of Meta R-CNN for classes bird, cow, sofa, plant, tv, bike, aero, and horse. It outperforms Meta R-CNN in sofa, horse, boat, cat, dog and train besides cars and persons. Table

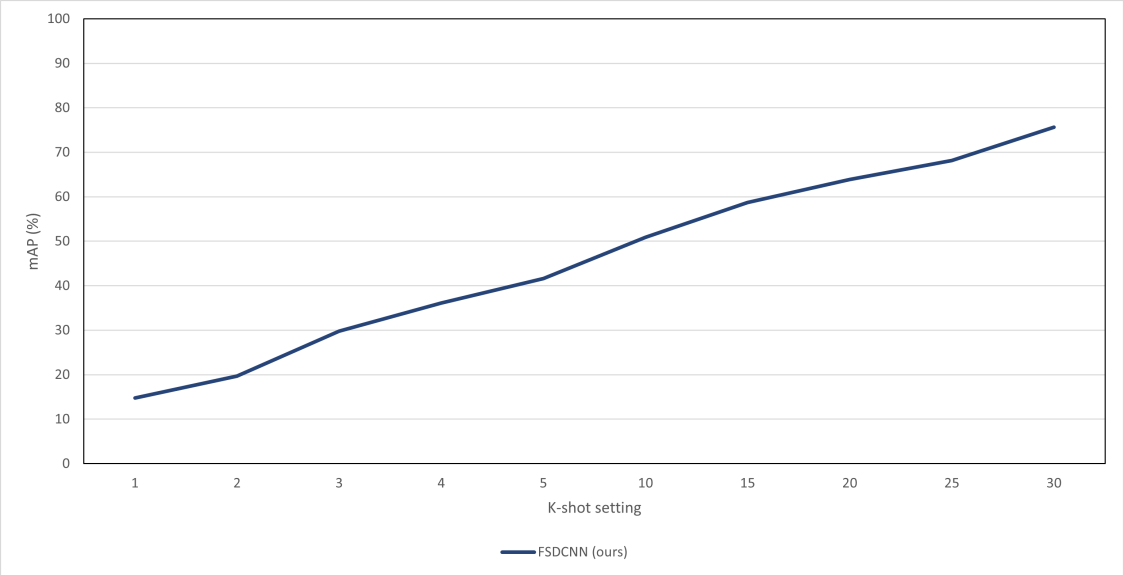

Figure 5.4
The mAP increases with the increase in number of samples available for a novel class. This shows that the supervised nature of the base model isn't compromised.

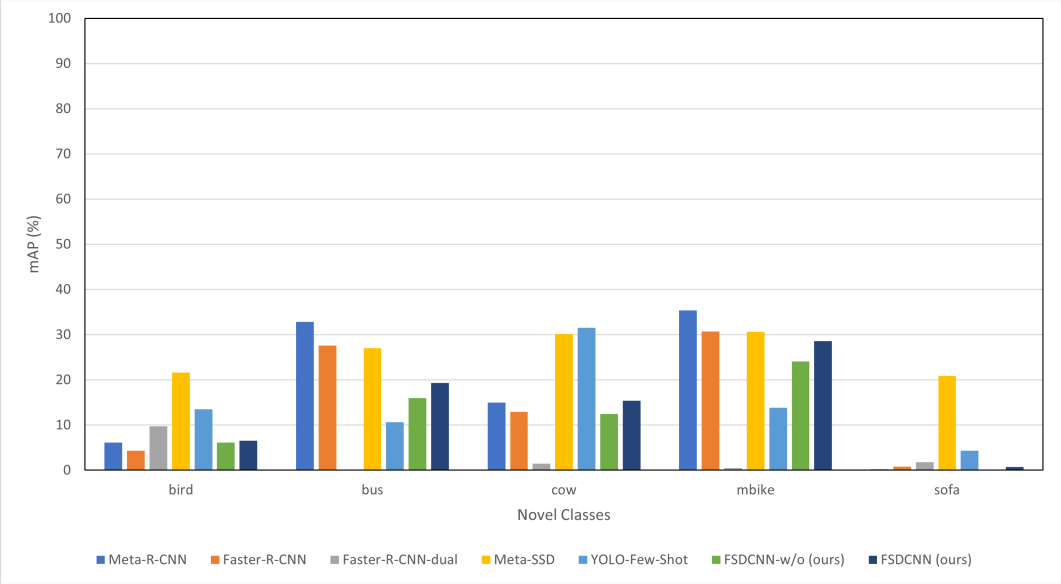

Figure 5.5
The mAP achieved for novel classes bird, bus, cow, mbike and sofa over thousand runs in 1-shot setting.

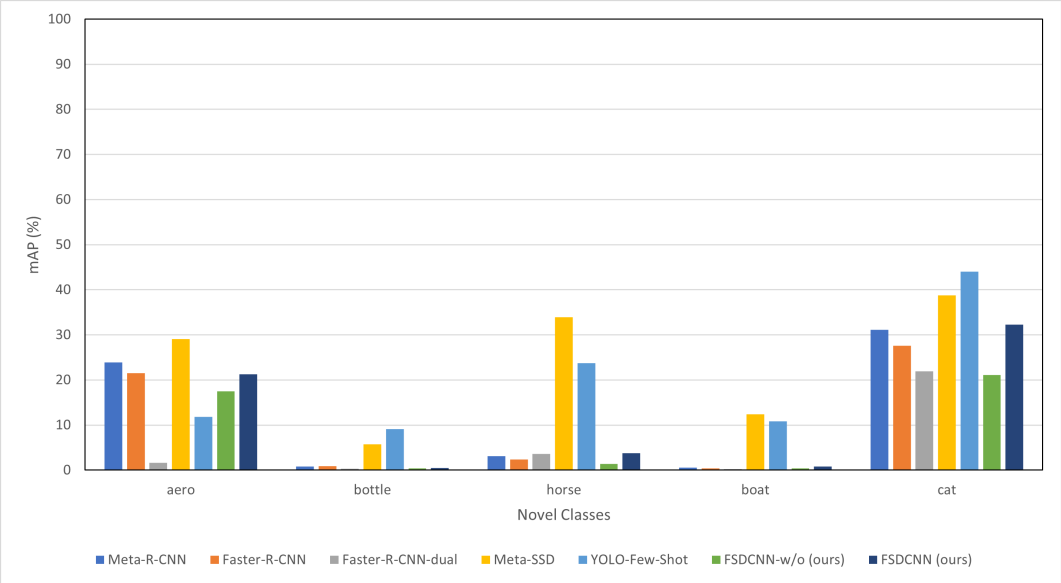

Figure 5.6
The mAP achieved for novel classes aero, bottle, horse, boat and cat over thousand runs in 1-shot setting.

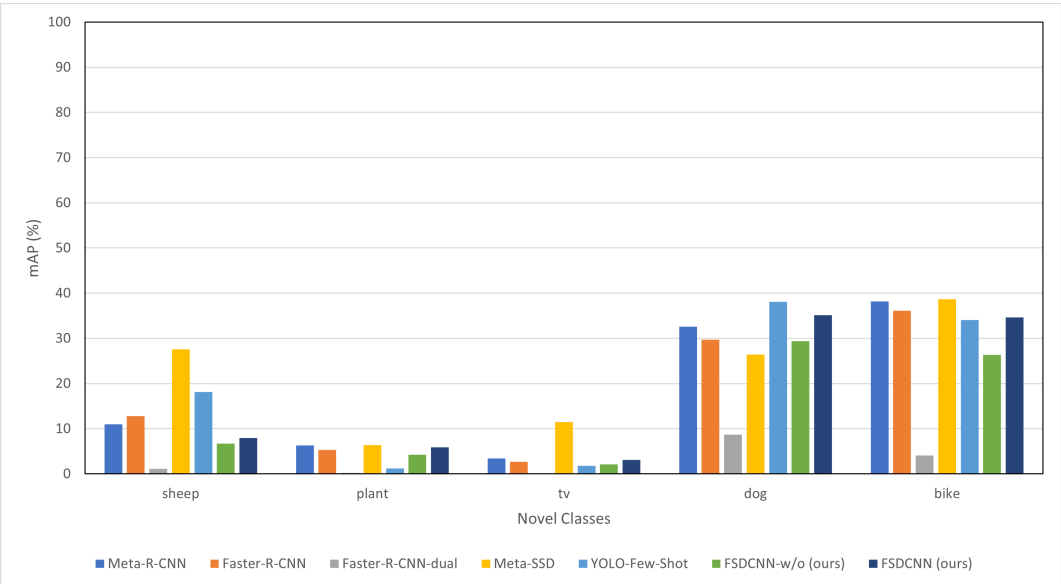

Figure 5.7
The mAP achieved for novel classes sheep, plant, tv, dog and bike over thousand runs in 1-shot setting.

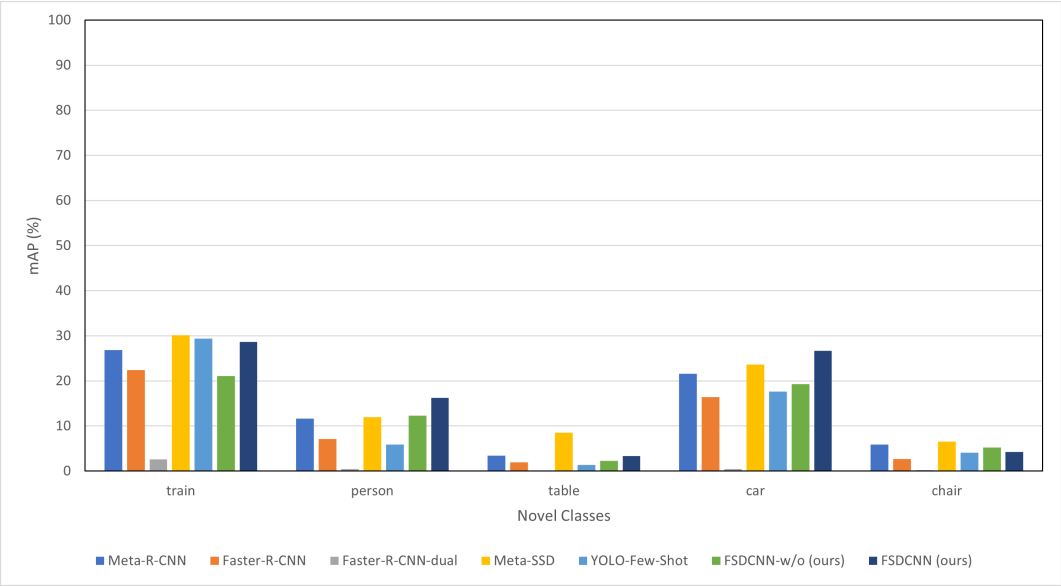

Figure 5.8
The mAP achieved for novel classes train, person, table, car and chair over thousand runs in 1-shot setting.

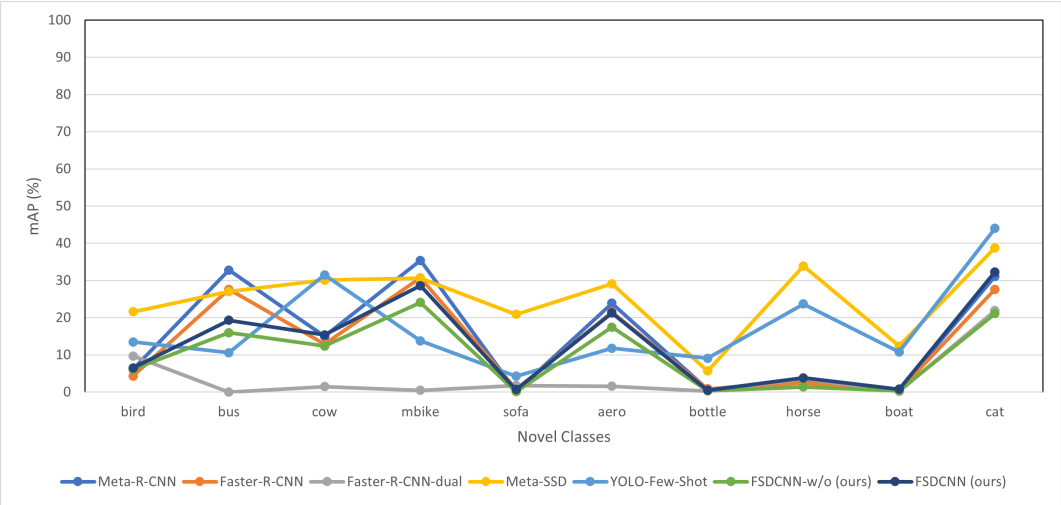

Figure 5.9
The mAP achieved for novel classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 1-shot setting.

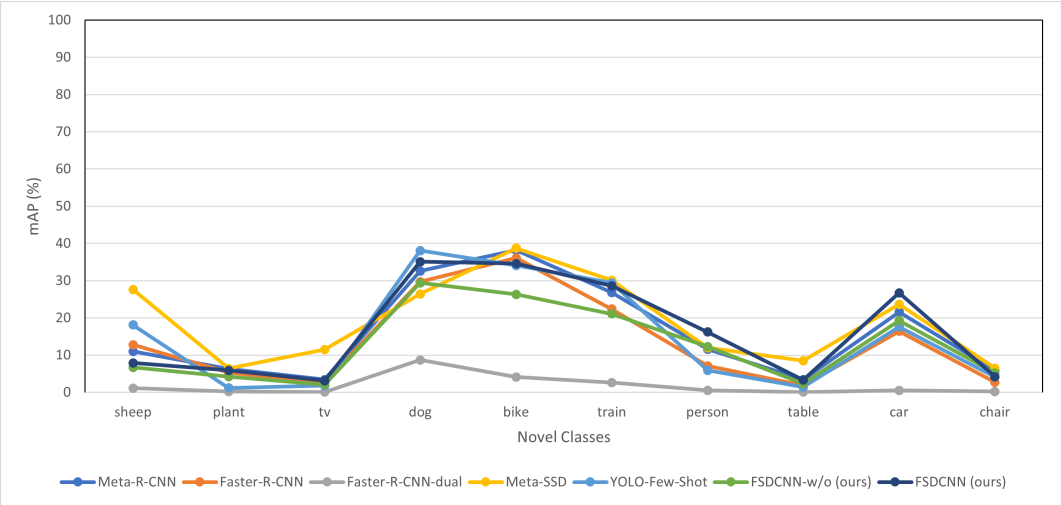

Figure 5.10
The mAP achieved for novel classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 1-shot setting.

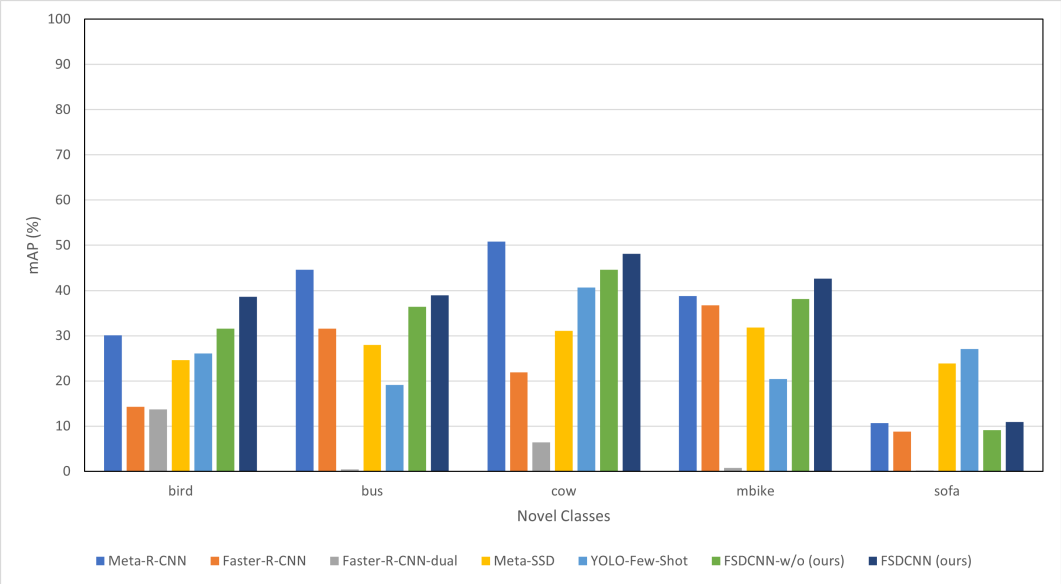

Figure 5.11
The mAP achieved for novel classes bird, bus, cow, mbike and sofa over thousand runs in 3-shot setting.

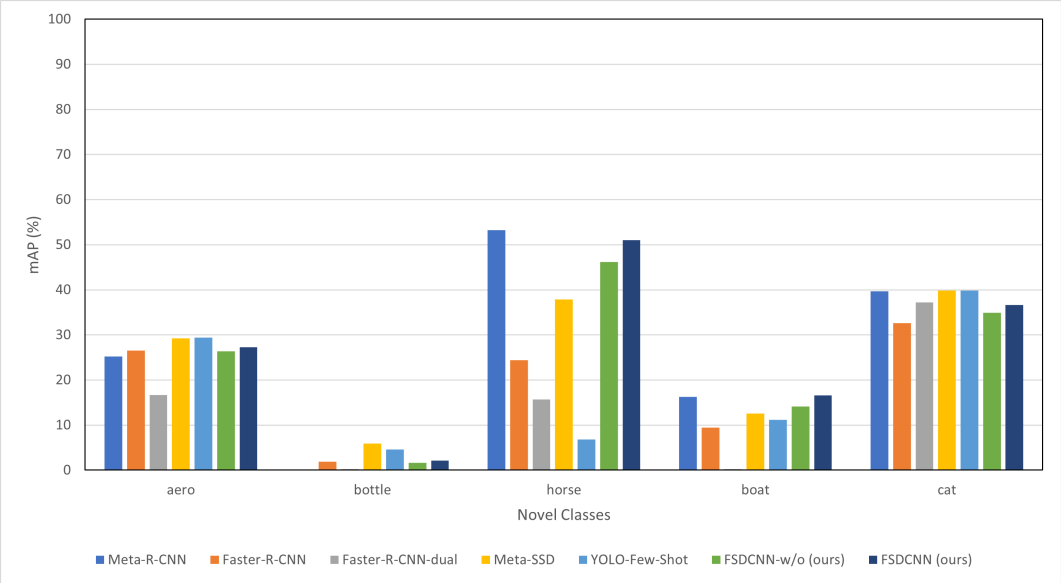

Figure 5.12
The mAP achieved for novel classes aero, bottle, horse, boat and cat over thousand runs in 3-shot setting.

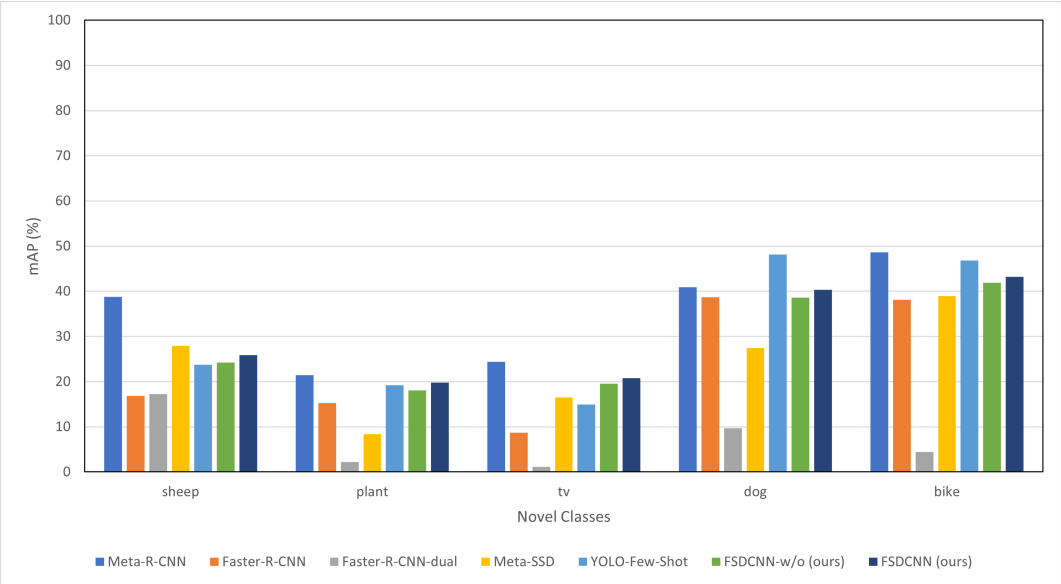

Figure 5.13
The mAP achieved for novel classes sheep, plant, tv, dog and bike over thousand runs in 3-shot setting.

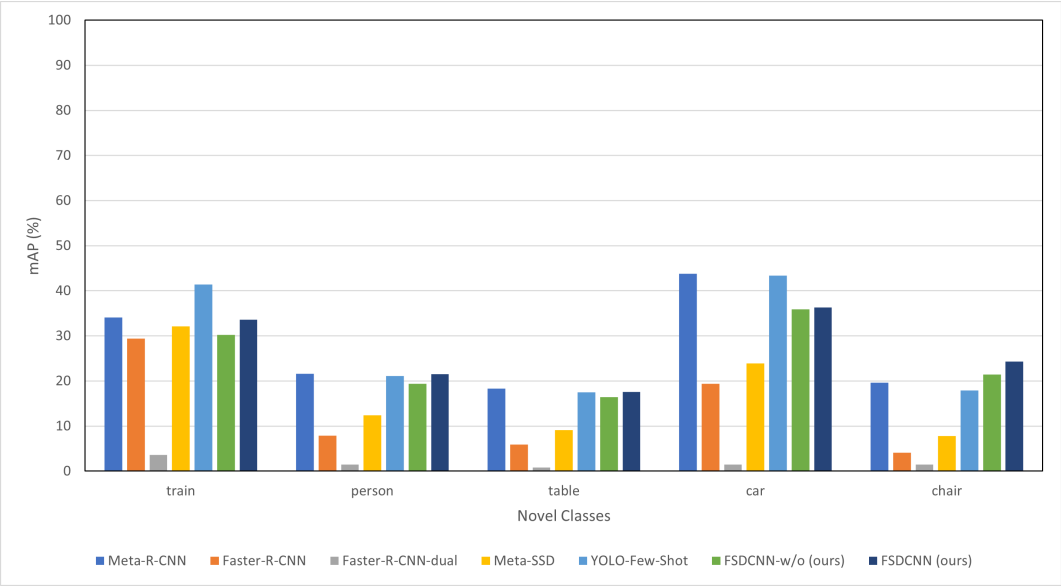

Figure 5.14
The mAP achieved for novel classes train, person, table, car and chair over thousand runs in 3-shot setting.

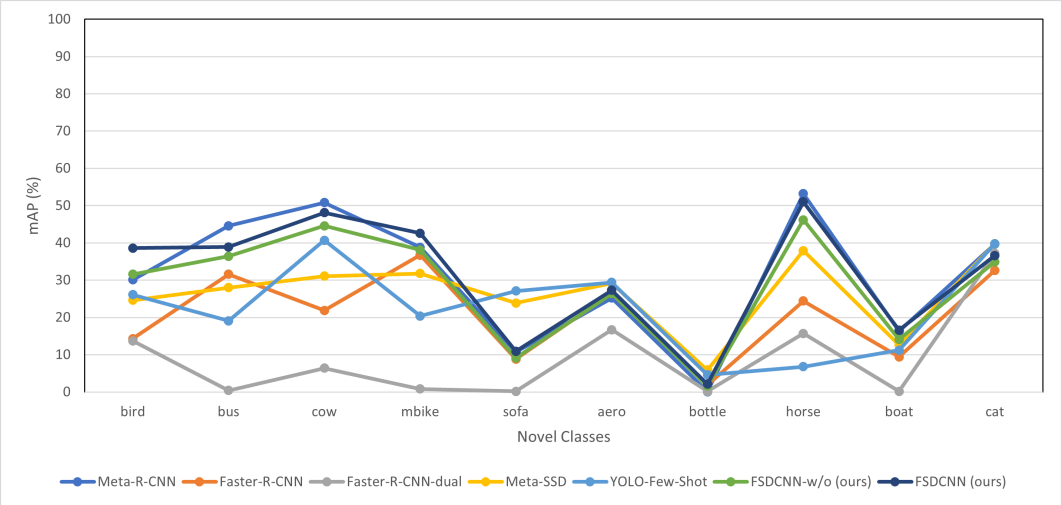

Figure 5.15
The mAP achieved for novel classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 3-shot setting.

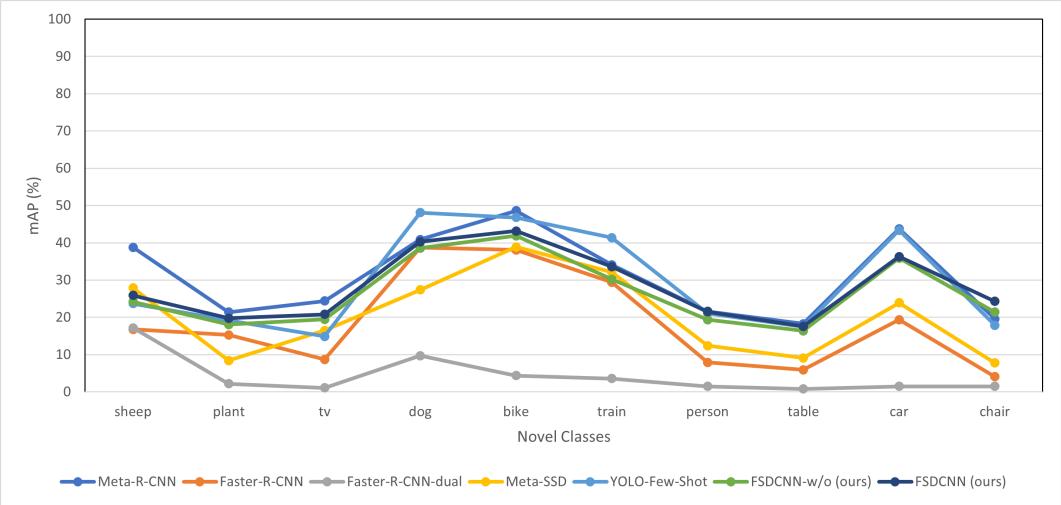

Figure 5.16
The mAP achieved for novel classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 3-shot setting.

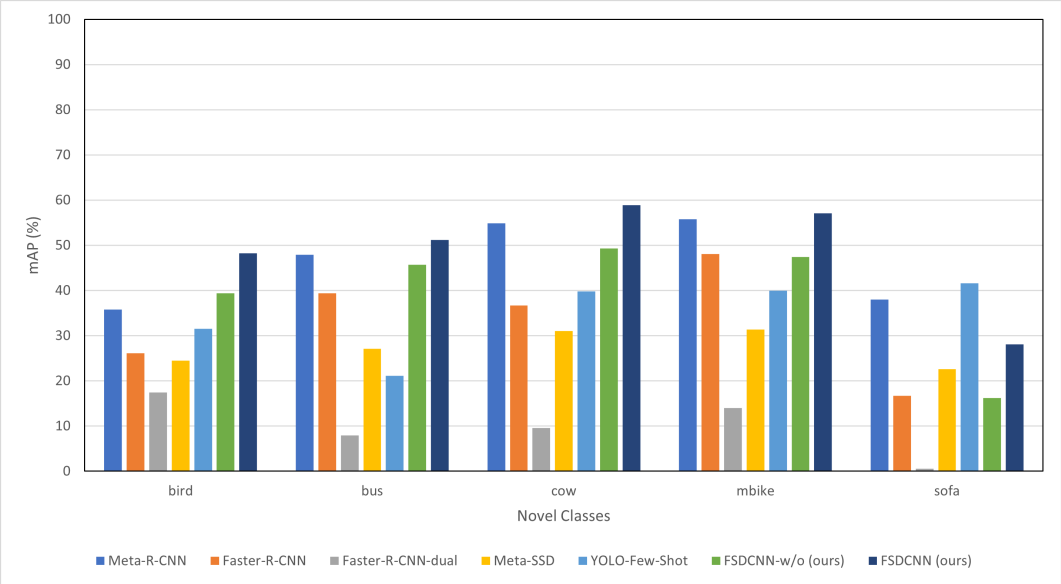

Figure 5.17
The mAP achieved for novel classes bird, bus, cow, mbike and sofa over thousand runs in 5-shot setting.

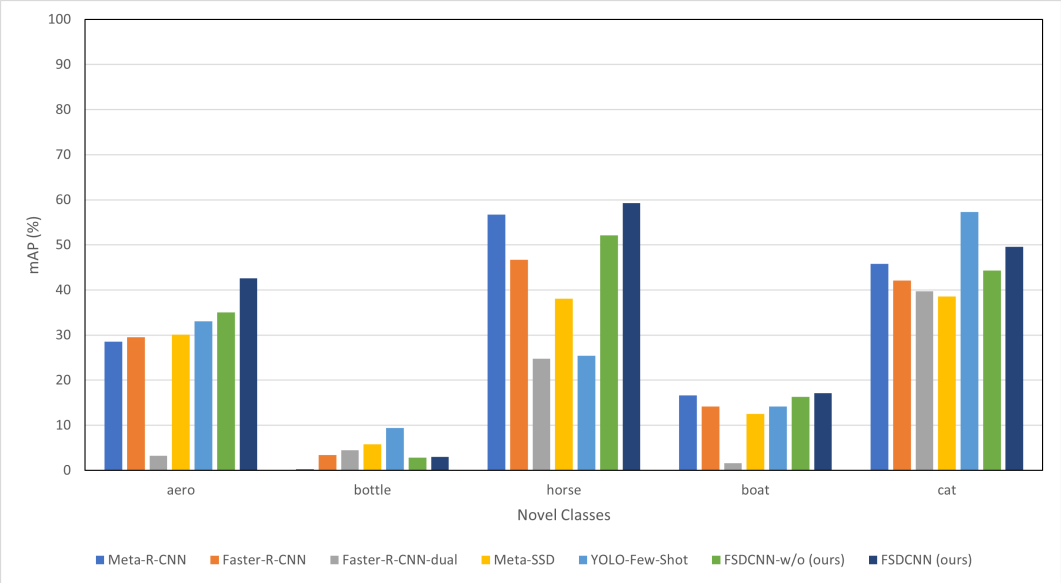

Figure 5.18
The mAP achieved for novel classes aero, bottle, horse, boat and cat over thousand runs in 5-shot setting.

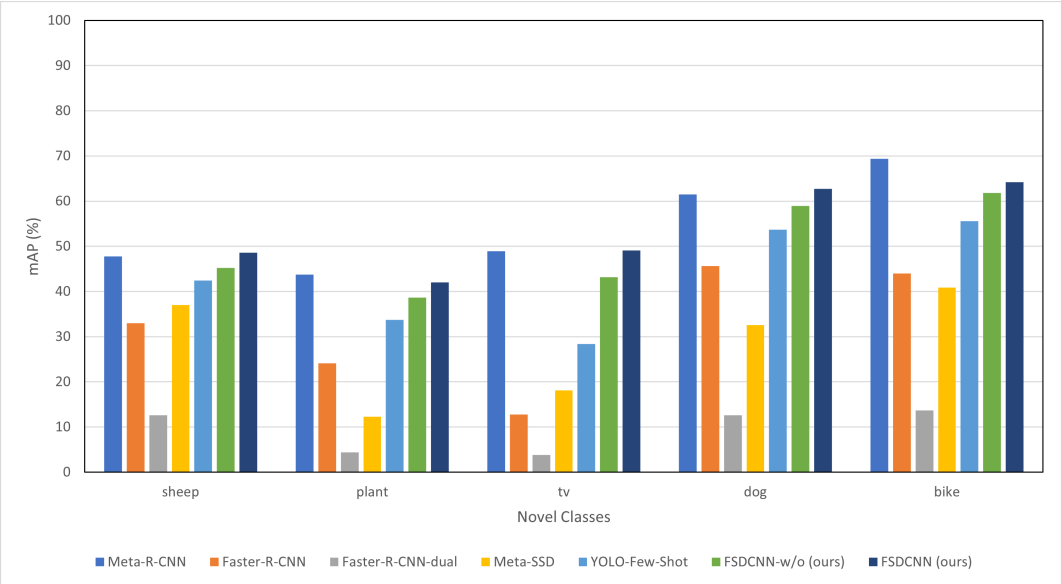

Figure 5.19
The mAP achieved for novel classes sheep, plant, tv, dog and bike over thousand runs in 5-shot setting.

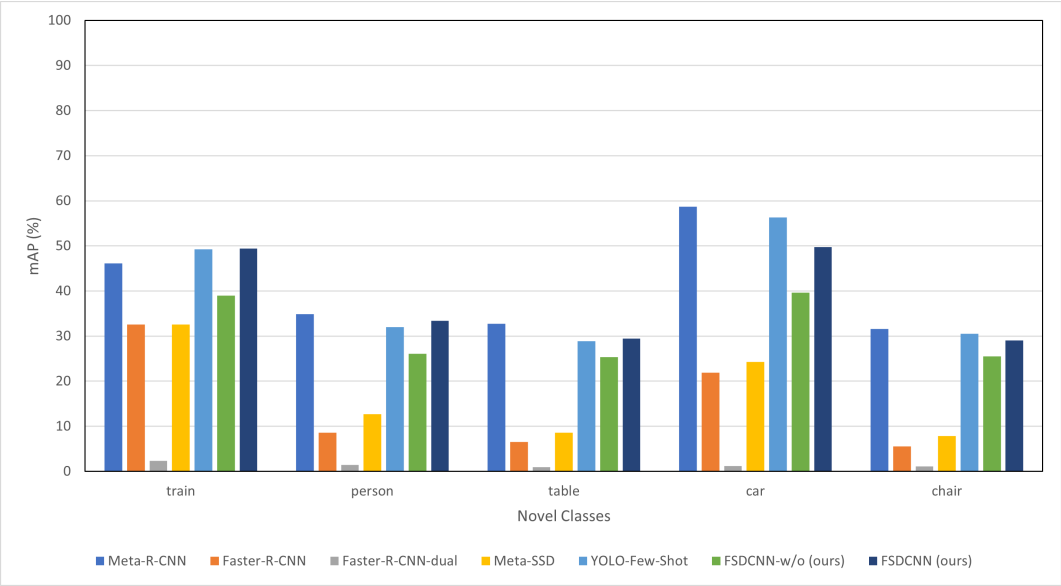

Figure 5.20
The mAP achieved for novel classes train, person, table, car and chair over thousand runs in 5-shot setting.

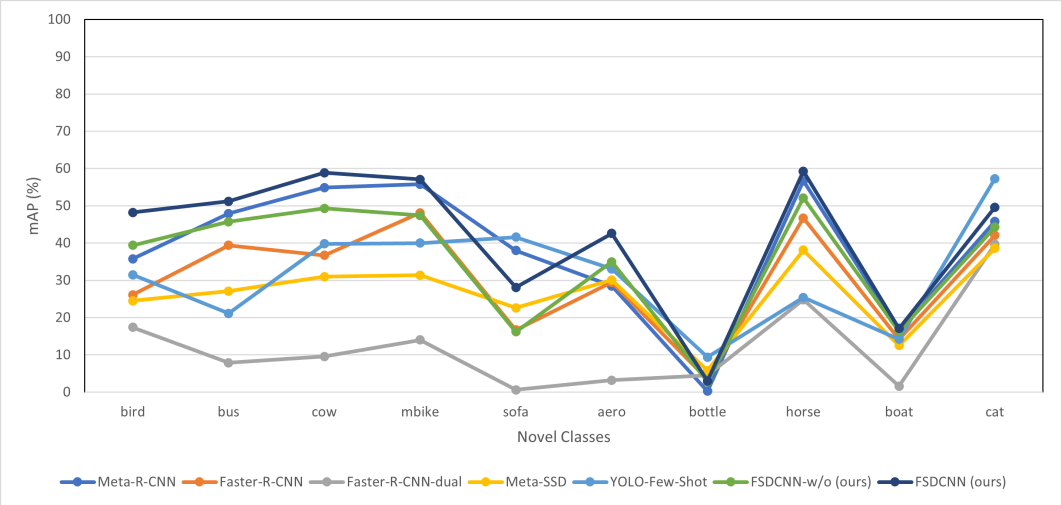

Figure 5.21
The mAP achieved for novel classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 5-shot setting.

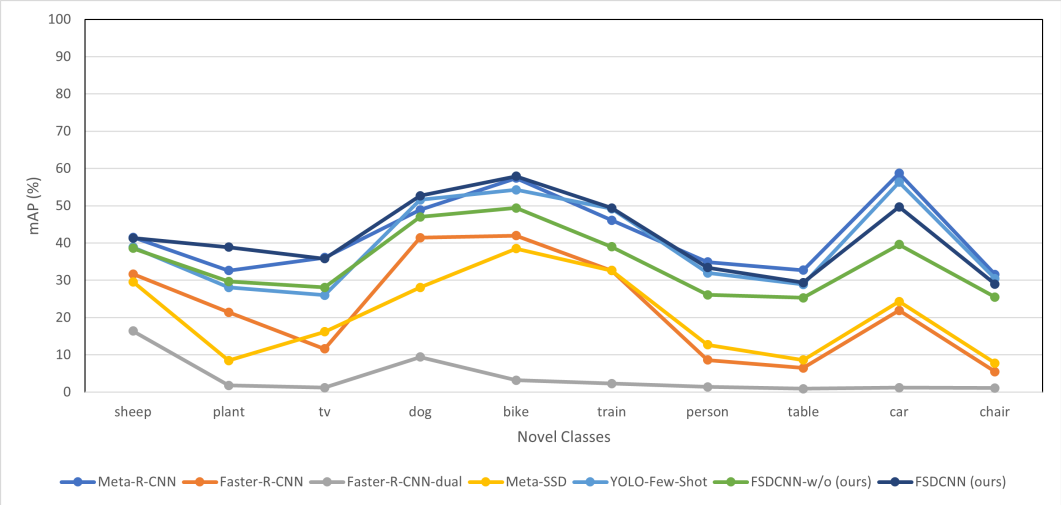

Figure 5.22
The mAP achieved for novel classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 5-shot setting.

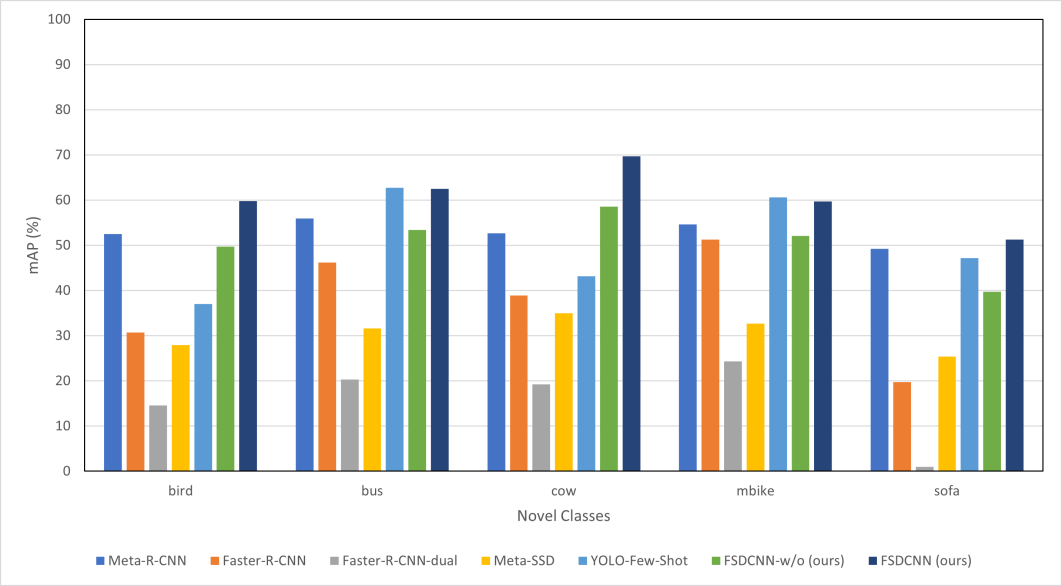

Figure 5.23
The mAP achieved for novel classes bird, bus, cow, mbike and sofa over thousand runs in 10-shot setting.

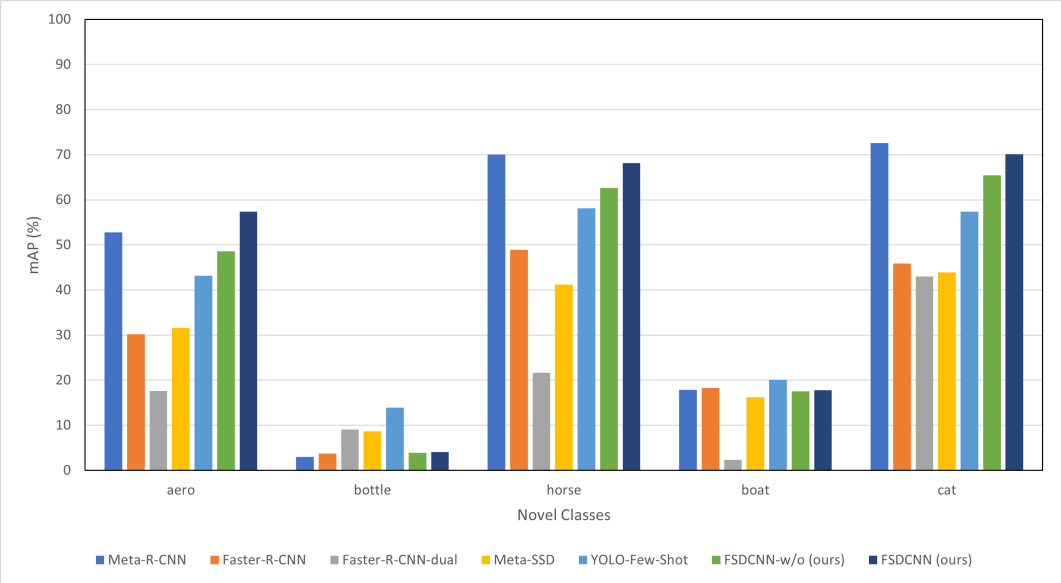

Figure 5.24
The mAP achieved for novel classes aero, bottle, horse, boat and cat over thousand runs in 10-shot setting.

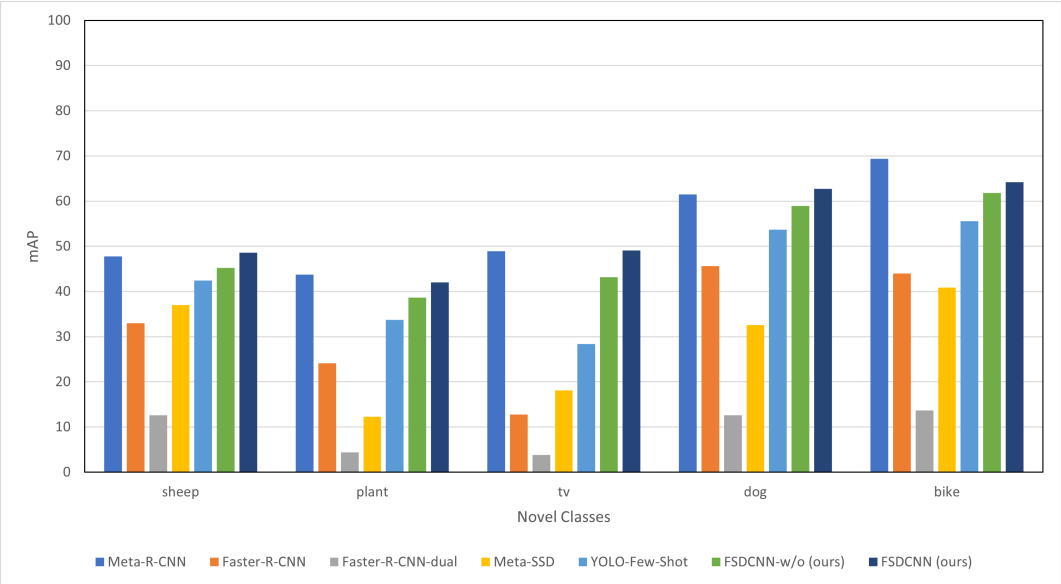

Figure 5.25
The mAP achieved for novel classes sheep, plant, tv, dog and bike over thousand runs in 10-shot setting.

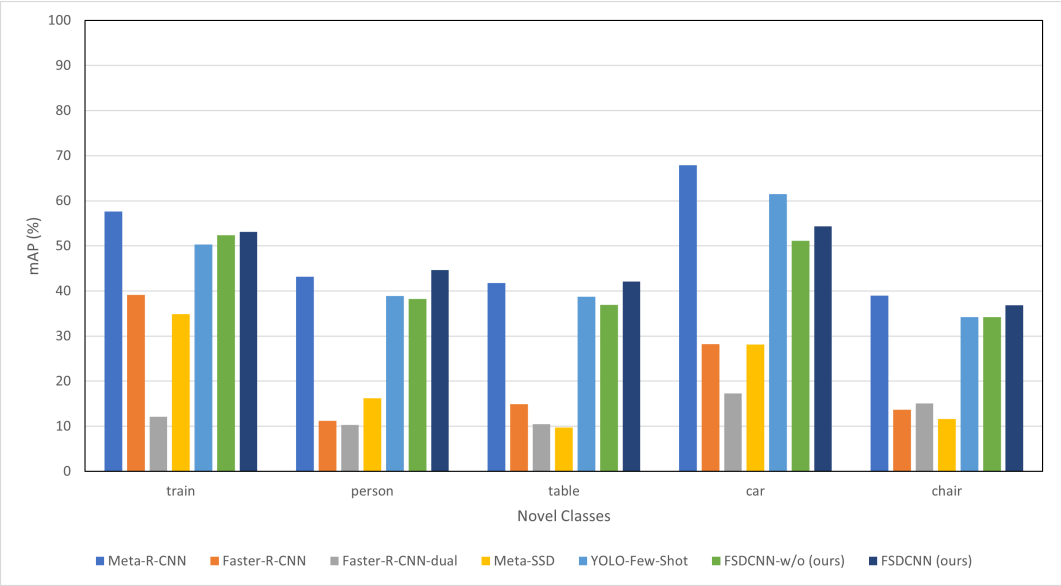

Figure 5.26
The mAP achieved for novel classes train, person, table, car and chair over thousand runs in 10-shot setting.

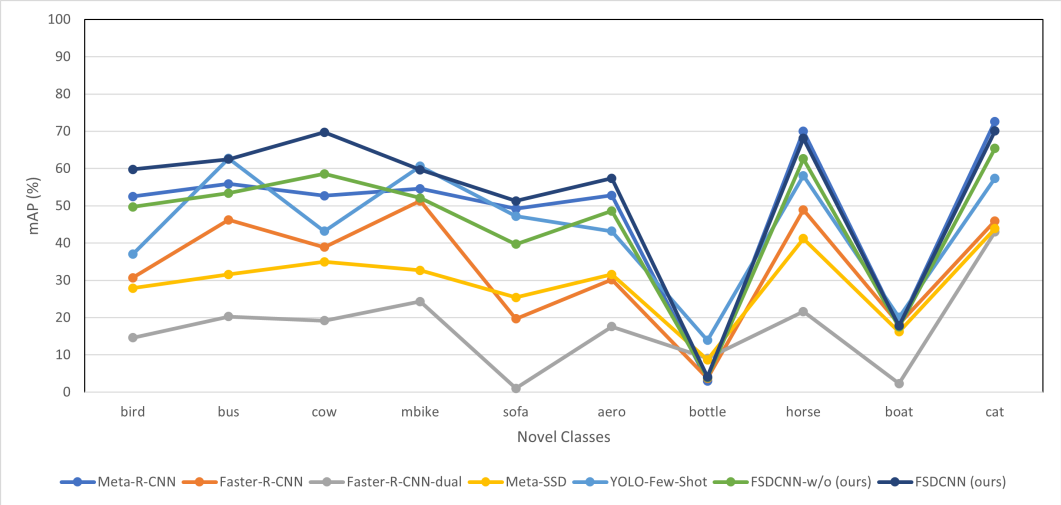

Figure 5.27
The mAP achieved for novel classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 10-shot setting.

K-Shot	Method	Split 1	Split 2	Split 3	Split 4	Mean
1-shot	Meta R-CNN [1]	19.9	14.3	15.1	11.1	15.1
1-shot	Faster R-CNN [10]	14.2	12.4	13.7	14.1	13.6
1-shot	Faster R-CNN-dual	2.4	3.1	2.6	3.5	2.9
1-shot	Meta-SSD [14]	**23.1**	**23.4**	**21.6**	**20.7**	**22.2**
1-shot	YOLO-Few-Shot [42]	17.3	16.4	15.2	15.5	16.1
1-shot	FSDCNN w/o (ours)	12.4	11.8	10.9	10.5	11.4
1-shot	FSDCNN (ours)	14.6	15.1	14.4	15.1	14.8
3-shot	Meta R-CNN [1]	**34.6**	27.4	**30.3**	**28.9**	**30.3**
3-shot	Faster R-CNN [10]	20.1	19.8	21.2	18.1	19.8
3-shot	Faster R-CNN-dual	7.2	6.4	6.9	6.3	6.7
3-shot	Meta-SSD [14]	23.6	22.7	24.2	23.1	23.4
3-shot	YOLO-Few-Shot [42]	26.1	25.6	25.1	26.8	25.9
3-shot	FSDCNN w/o (ours)	28.1	27.9	26.8	27.6	27.6
3-shot	FSDCNN (ours)	31.2	**30.5**	28.7	28.4	29.7
5-shot	Meta R-CNN [1]	41.2	**43.6**	36.4	**41.5**	40.7
5-shot	Faster R-CNN [10]	26.1	24.9	28.2	24.4	25.9
5-shot	Faster R-CNN-dual	7.9	8.3	7.6	8.6	8.1
5-shot	Meta-SSD [14]	25.1	28.3	26.1	18.9	24.6
5-shot	YOLO-Few-Shot [42]	40.2	36.3	31.2	32.7	35.1
5-shot	FSDCNN w/o (ours)	35.2	33.9	38.4	31.3	34.7
5-shot	FSDCNN (ours)	**43.6**	39.7	**40.2**	41.3	**41.2**
10-shot	Meta R-CNN [1]	52.5	47.7	49.2	**48.6**	49.5
10-shot	Faster R-CNN [10]	31.9	27.3	29.6	32.8	30.4
10-shot	Faster R-CNN-dual	15.2	12.4	13.5	16.1	14.3
10-shot	Meta-SSD [14]	27.1	24.2	26.7	29.2	26.8
10-shot	YOLO-Few-Shot [42]	44.3	42.1	39.9	48.1	43.6
10-shot	FSDCNN w/o (ours)	49.2	42.0	46.5	43.8	45.4
10-shot	FSDCNN (ours)	**54.2**	**49.3**	**50.1**	47.6	**50.3**

Table 5.1
Performance comparison of the different models in the experimented 4 Novel/Base split settings for 1-shot, 3-shot, 5-shot and 10-shot. Our model performs really well in 10-shot. Furthermore, the deviation of mAP for different settings is less.

5.2 displays the mAP achieved by the models for all the classes over thousand runs. It is also seen that the performance is not the same across all the classes.

Figures 5.11- 5.16 display the performances of the models on the 20 classes for 3-shot setting.

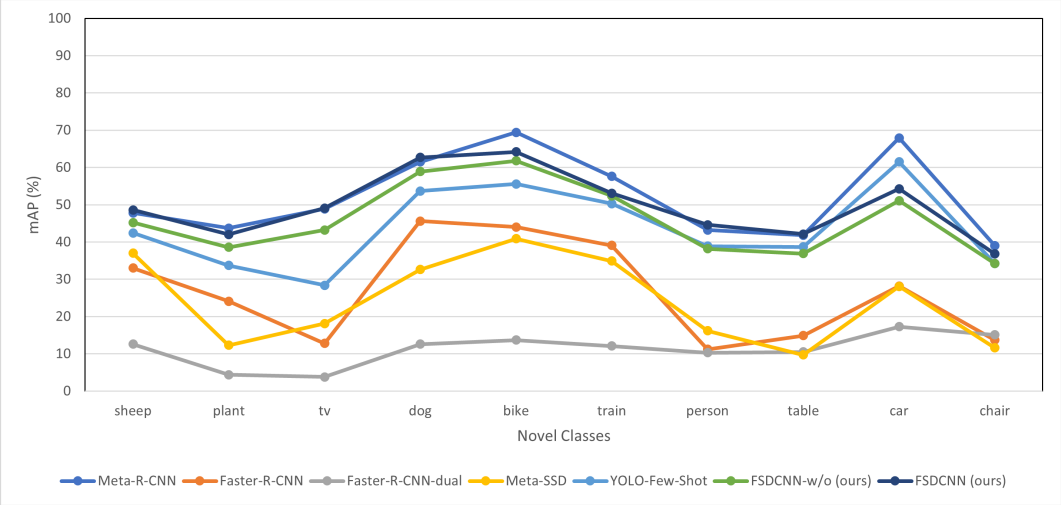

Figure 5.28
The mAP achieved for novel classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 10-shot setting.

Meta R-CNN along with our model FSDCNN give the top two performance levels in 9 out of the 20 classes (bus, cow, mbike, horse, boat plant, tv, person and table). Among the 9, FSDCNN outperforms Meta R-CNN for two classes. Out of the remaining 11 classes, FSDCNN performs the best in classifying bird and chair. Overall, in 3-shot setting, FSDCNN is a close second to Meta R-CNN over the twenty classes.

Figures 5.17- 5.22 display the performances of the models on the 20 classes for 5-shot setting. FSDCNN outperforms the other models in 10 out of the 20 classes. In the remaining ten, FSDCNN has the second best performance in five. When the mean of all the mAP over all the classes was taken, it was seen that FSDCNN outperforms all the other models , closely followed by Meta R-CNN.

Figures 5.23- 5.28 display the performances of the models on the 20 classes for 10-shot setting. FSDCNN outperforms the other models in 9 out of the 20 classes. In the remaining eleven, FSDCNN has the second best performance in eight. When the mean of all the mAP over all the classes was taken, it was seen that FSDCNN narrowly outperforms Meta R-CNN.

K-shot	Method	bird	bus	cow	mbike	sofa	aero	bottle	horse	boat	cat	sheep	plant	tv	dog	bike	train	person	table	car	chair
1-shot	Meta R-CNN [1]	6.1	32.8	15	35.4	0.2	23.9	0.8	3.1	0.6	31.1	11	6.3	3.4	32.6	38.2	26.8	11.6	3.4	21.6	5.9
1-shot	Faster R-CNN [10]	4.3	27.6	12.9	30.7	0.8	21.5	0.9	2.4	0.4	27.6	12.8	5.3	2.7	29.7	36.1	22.4	7.1	1.9	16.4	2.7
1-shot	Faster R-CNN-dual	9.7	0	1.5	0.5	1.8	1.6	0.3	3.6	0.2	21.9	1.1	0.2	0.1	8.7	4.1	2.6	0.5	0.1	0.5	0.2
1-shot	Meta-SSD [14]	21.6	27	30.1	30.6	20.9	29.1	5.7	33.9	12.4	38.8	27.6	6.4	11.5	26.4	38.7	30.1	12	8.5	23.6	6.5
1-shot	YOLO-Few-Shot [42]	13.5	10.6	31.5	13.8	4.3	11.8	9.1	23.7	10.8	44	18.1	1.2	1.8	38.1	34.1	29.4	5.9	1.4	17.6	4.1
1-shot	FSDCNN-w/o (ours)	6.1	16	12.4	24.1	0.1	17.5	0.4	1.4	0.4	21.1	6.7	4.2	2.1	29.4	26.3	21.1	12.3	2.3	19.3	5.2
1-shot	FSDCNN (ours)	6.5	19.3	15.4	28.6	0.7	21.3	0.5	3.8	0.8	32.3	7.9	5.9	3.1	35.1	34.6	28.6	16.2	3.3	26.7	4.2
3-shot	Meta R-CNN [1]	30.1	44.6	50.8	38.8	10.7	25.2	0.1	53.2	16.3	39.7	38.8	21.4	24.4	40.9	48.6	34.1	21.6	18.3	43.8	19.6
3-shot	Faster R-CNN [10]	14.3	31.6	21.9	36.7	8.8	26.5	1.9	24.4	9.4	32.6	16.8	15.3	8.7	38.7	38.1	29.4	7.9	5.9	19.4	4.1
3-shot	Faster R-CNN-dual	13.7	0.4	6.4	0.8	0.2	16.7	0.2	15.7	0.2	37.2	17.2	2.2	1.1	9.7	4.4	3.6	1.5	0.8	1.5	1.5
3-shot	Meta-SSD [14]	24.6	28	31.1	31.8	23.9	29.2	5.9	37.9	12.6	39.8	27.9	8.4	16.5	27.4	38.9	32.1	12.4	9.1	23.9	7.8
3-shot	YOLO-Few-Shot [42]	26.1	19.1	40.7	20.4	27.1	29.4	4.6	6.8	11.2	39.8	23.7	19.2	14.9	48.1	46.8	41.4	21.1	17.5	43.4	17.9
3-shot	FSDCNN-w/o (ours)	31.6	36.4	44.6	38.1	9.1	26.4	1.6	46.2	14.1	34.9	24.2	18.1	19.5	38.6	41.9	30.2	19.4	16.4	35.9	21.4
3-shot	FSDCNN (ours)	38.6	38.9	48.1	42.6	10.9	27.3	2.1	51	16.6	36.6	25.9	19.8	20.8	40.3	43.2	33.6	21.5	17.6	36.3	24.3
5-shot	Meta R-CNN [1]	35.8	47.9	54.9	55.8	38	28.5	0.3	56.7	16.6	45.8	41.5	32.6	36.1	48.9	57.4	46.1	34.9	32.7	58.7	31.6
5-shot	Faster R-CNN [10]	26.1	39.4	36.7	48.1	16.7	29.5	3.4	46.7	14.2	42.1	31.7	21.4	11.6	41.4	42	32.6	8.6	6.5	21.9	5.5
5-shot	Faster R-CNN-dual	17.4	7.9	9.6	14	0.6	3.2	4.5	24.8	1.6	39.7	16.4	1.8	1.2	9.4	3.2	2.3	1.4	0.9	1.2	1.1
5-shot	Meta-SSD [14]	24.5	27.1	31	31.4	22.6	30.1	5.8	38.1	12.5	38.6	29.6	8.5	16.2	28.1	38.5	32.6	12.7	8.6	24.3	7.8
5-shot	YOLO-Few-Shot [42]	31.5	21.1	39.8	40	41.6	33.1	9.4	25.4	14.2	57.3	38.9	28.1	26	51.6	54.3	49.2	32	28.9	56.3	30.5
5-shot	FSDCNN-w/o (ours)	39.4	45.7	49.3	47.4	16.2	35	2.8	52.1	16.3	44.3	38.6	29.7	28.1	47	49.4	39	26.1	25.3	39.6	25.5
5-shot	FSDCNN (ours)	48.2	51.2	58.9	57.1	28.1	42.6	3	59.3	17.1	49.6	41.3	38.9	35.8	52.7	57.9	49.4	33.4	29.4	49.7	29
10-shot	Meta R-CNN [1]	52.5	55.9	52.7	54.6	49.2	52.8	3	70	17.9	72.6	47.8	43.7	48.9	61.5	69.4	57.6	43.2	41.8	67.9	39
10-shot	Faster R-CNN [10]	30.7	46.2	38.9	51.3	19.7	30.2	3.7	48.9	18.3	45.9	33	24.1	12.8	45.6	44	39.1	11.2	14.9	28.2	13.7
10-shot	Faster R-CNN-dual	14.6	20.3	19.2	24.3	1	17.6	9.1	21.6	2.3	43	12.6	4.4	3.8	12.6	13.7	12.1	10.3	10.5	17.3	15.1
10-shot	Meta-SSD [14]	27.9	31.6	35	32.7	25.4	31.6	8.7	41.2	16.2	43.9	37	12.3	18.1	32.6	40.9	34.9	16.2	9.7	28.1	11.6
10-shot	YOLO-Few-Shot [42]	37	62.7	43.2	60.6	47.2	43.2	13.9	58.1	20.1	57.4	42.4	33.7	28.4	53.7	55.6	50.3	38.9	38.7	61.5	34.2
10-shot	FSDCNN-w/o (ours)	49.7	53.4	58.6	52.1	39.7	48.6	3.9	62.6	17.5	65.4	45.2	38.6	43.2	58.9	61.8	52.4	38.2	36.9	51.1	34.2
10-shot	FSDCNN (ours)	59.8	62.5	69.7	59.7	51.3	57.4	4.1	68.1	17.8	70.1	48.6	42	49.1	62.7	64.2	53.1	44.6	42.1	54.3	36.8

Table 5.2
Performance comparison of the different models in classifying each of the 20 unseen classes over a thousand runs for 1-shot, 3-shot, 5-shot and 10-shot. (Color guide: red is for the best performance, green is for second best and blue is for third best.)

As the performances increased in varied rates for all the models with increase in number of samples, we decided to plot and compare the performances achieved by all the models for different K-shot settings, similar to how we had plotted for FSDCNN in Figure 5.4.

We can see in Figure 5.29 that though Meta-SSD has the best performance for 1-shot setting, it doesn't improve much with increase in number of samples. In contrast, Faster R-CNN-dual shows the best improvement after 15 samples. It can also be seen that FSDCNN and Meta R-CNN have somewhat similar trajectory. The same can be said for FSDCNN w/o and YOLO-Few-Shot. In order to have a better comprehension of their performances, we plotted them as bars.

As seen in Figure 5.30, FSDCNN consistently outperforms Meta R-CNN when the number of samples are four or more. This could be due to the way meta-learning has been applied to our model in contrast to Meta R-CNN. Though the difference in the performance levels is very narrow for 4-shot, 5-shot, 10-shot, 15-shot, 20-shot and 25-shot, it has a greater difference in 30-shot.

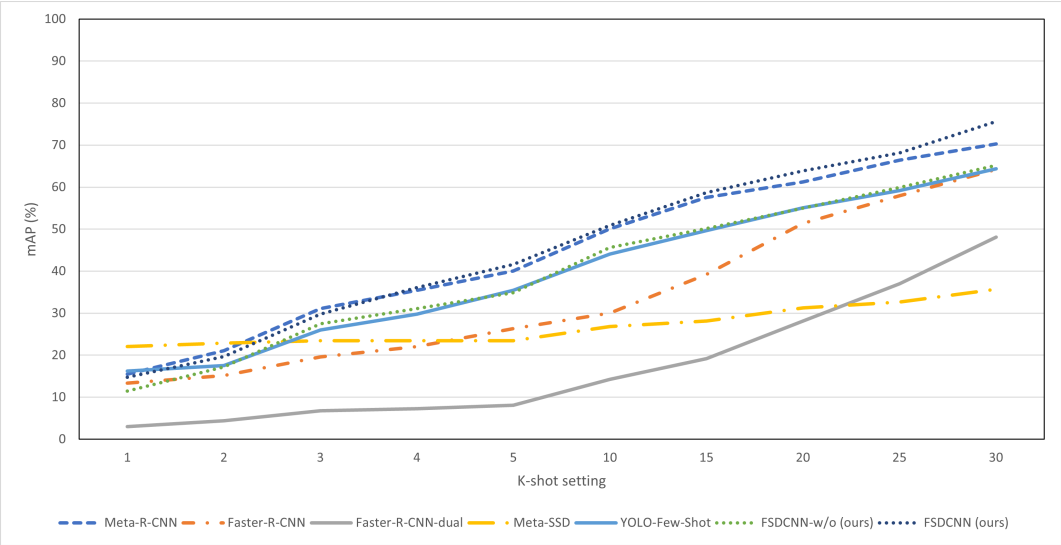

Figure 5.29
Comparison of mAP achieved by the models over different k-shot settings.

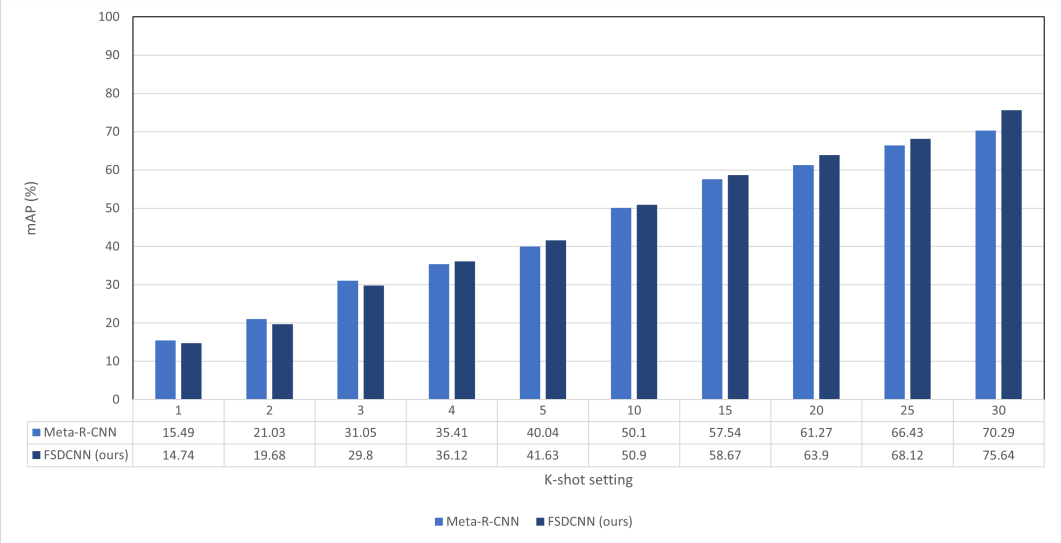

Figure 5.30
Comparison of mAP achieved by Meta R-CNN and FSDCNN over different k-shot settings.

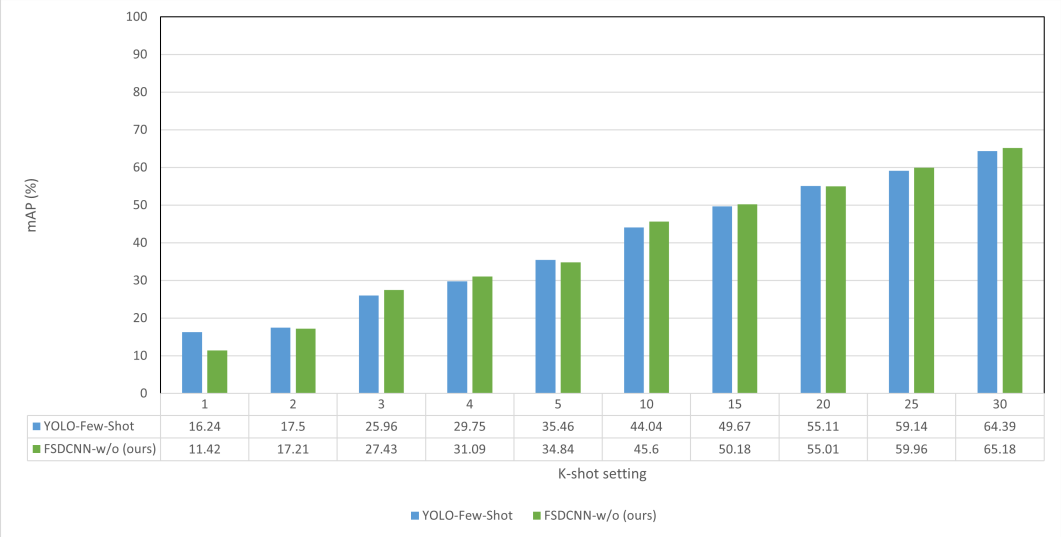

Figure 5.31
Comparison of mAP achieved by YOLO-Few-Shot and FSDCNN w/o over different k-shot settings.

As seen in Figure 5.31, FSDCNN w/o is consistently close to YOLO-Few-Shot when the number of samples are two or more. FSDCNN w/o doesn't have a meta-learner in contrast to FSDCNN. We plotted the performance levels of FSDCNN and FSDCNN w/o side by side in order to further emphasize the significance of meta-learner. It can be seen in Figure 5.32. FSDCNN consistently performs better than FSDCNN w/o in every K-shot setting. The architecture of both the models is similar but for the use of meta-learner in FSDCNN. This shows that the application of meta-learner in FSDCNN has helped in achieving a significant performance boost.

The performances of the models over the base classes were recorded and plotted to see if there was a significant drop in performance levels. Table 5.3 shows the mAP values for each base class obtained every model over a thousand runs in different K-shot settings. In Figures 5.33 through 5.56, we can see that Faster R-CNN-dual consistently outperforms the other models. Faster R-CNN-dual is the state-of-the-art supervised model Faster R-CNN which is trained on both novel and base classes. Our model, FSDCNN, is second only to Faster R-CNN-dual over the thousand

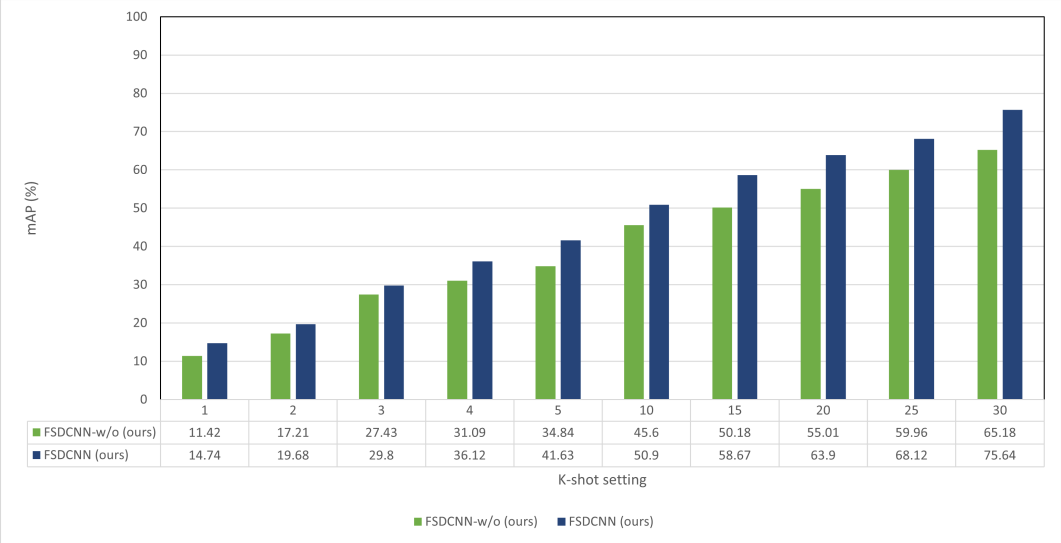

Figure 5.32
Comparison of mAP achieved by FSDCNN w/o and FSDCNN over different k-shot settings. This shows that the meta-learner in FSDCNN helps in achieving better performance.

runs in classifying detecting the base classes. However, when we take the performance of novel classes into comparison, the drop in performance level in base classes can be considered as a small compromise. It is able to remember the base classes whereas the same cannot be said about Meta-SSD. The performance levels of Meta-SSD on novel and base classes are very similar. It could be due to either of two reasons. One is that Single Shot Detector, which is the base model for Meta-SSD, isn't quite good in object classification and detection. The other reason would be the model's inability to remember the base classes, which is more likely as the few-shot models have been known to show a tendency to forget the base classes.

We had also recorded the mAP over the novel/base split detection tasks in each of the thousand runs for every model. The recorded mAP values were then plotted in a violin plot to see how spread or concentrated they were. Figure 5.57 displays the distribution of mAP values for 1-shot setting over 1000 runs for every model. It can be seen from the figure that the mAP values achieved by Meta R-CNN were spread between a range of 11-20 whereas the mAP values for FSDCNN were

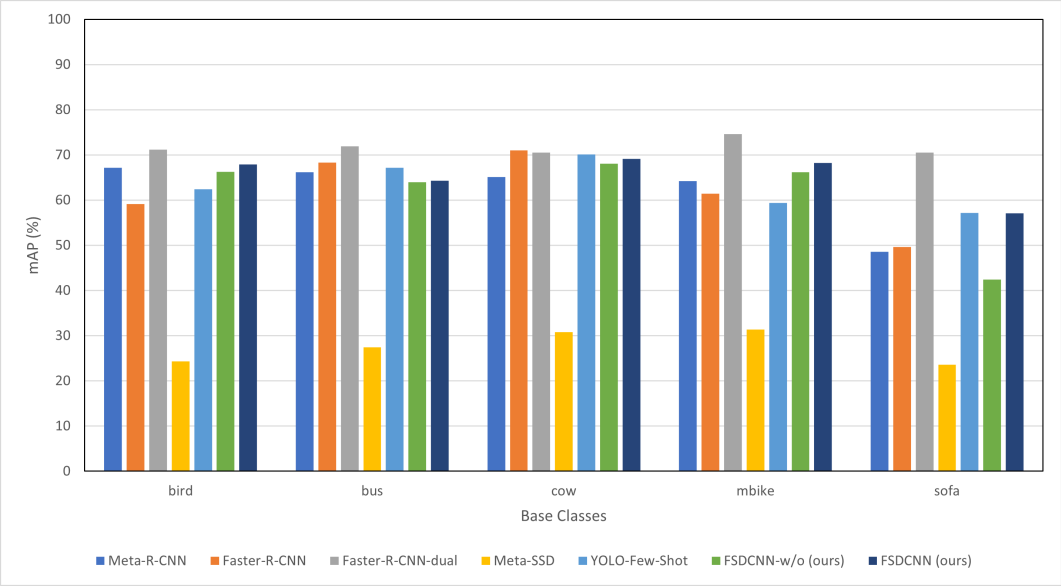

Figure 5.33
The mAP achieved for base classes bird, bus, cow, mbike and sofa over thousand runs in 1-shot setting.

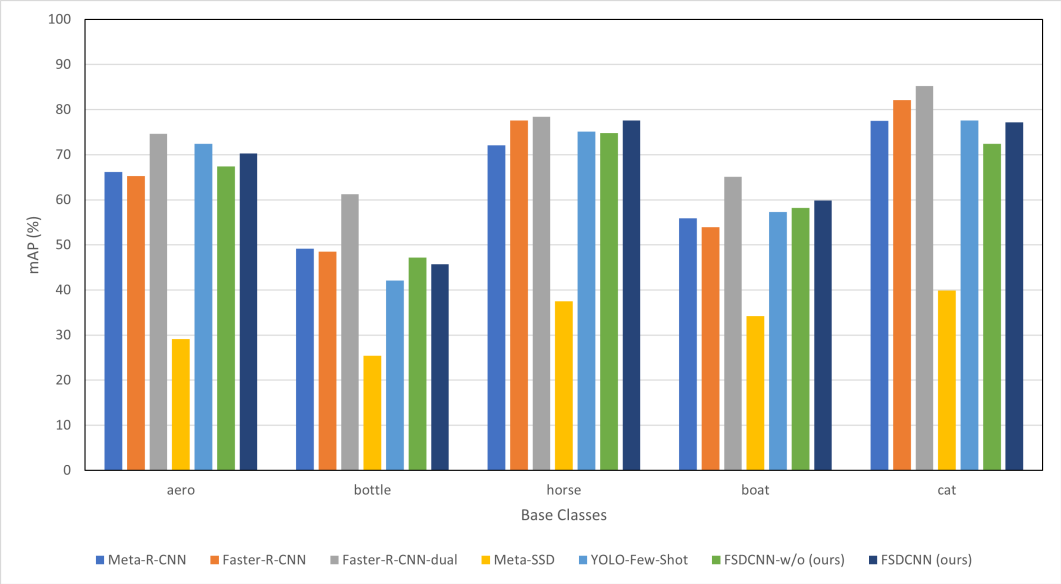

Figure 5.34
The mAP achieved for base classes aero, bottle, horse, boat and cat over thousand runs in 1-shot setting.

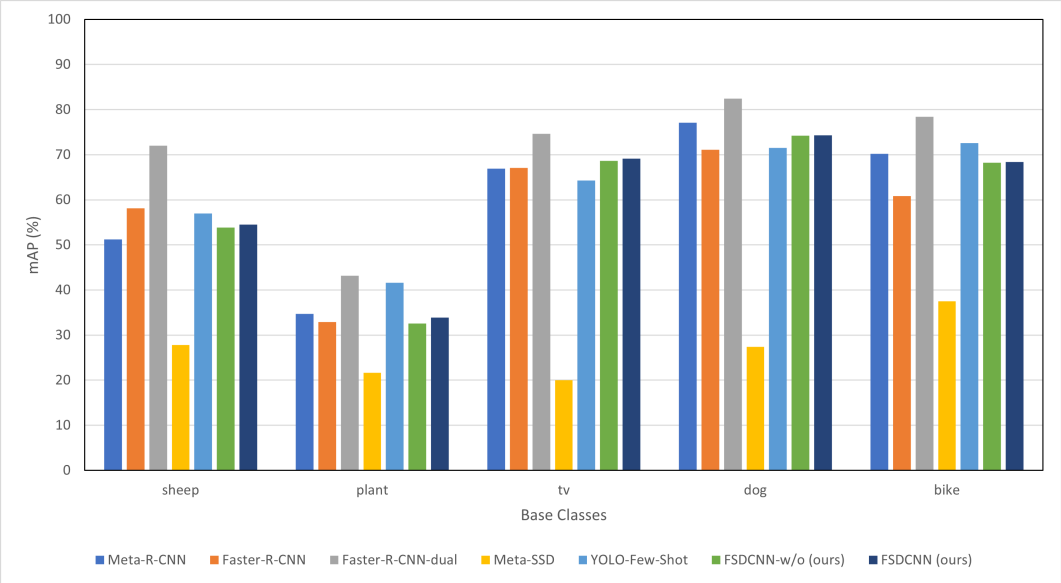

Figure 5.35
The mAP achieved for base classes sheep, plant, tv, dog and bike over thousand runs in 1-shot setting.

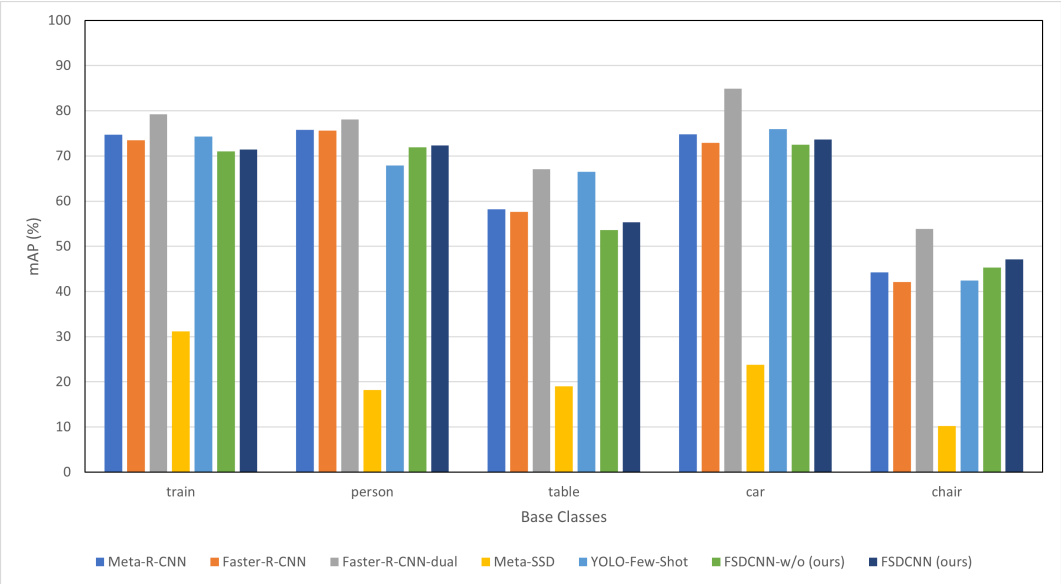

Figure 5.36
The mAP achieved for base classes train, person, table, car and chair over thousand runs in 1-shot setting.

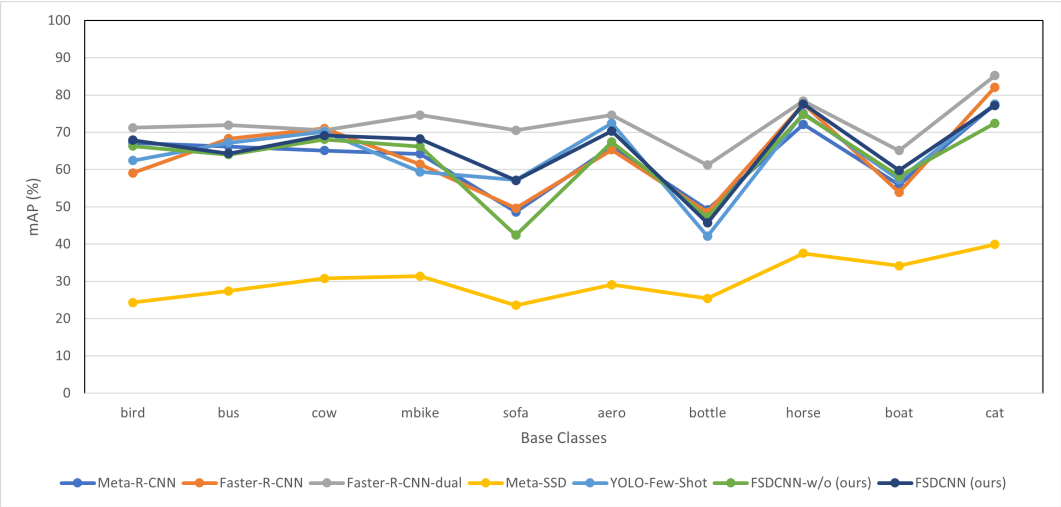

Figure 5.37
The mAP achieved for base classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 1-shot setting.

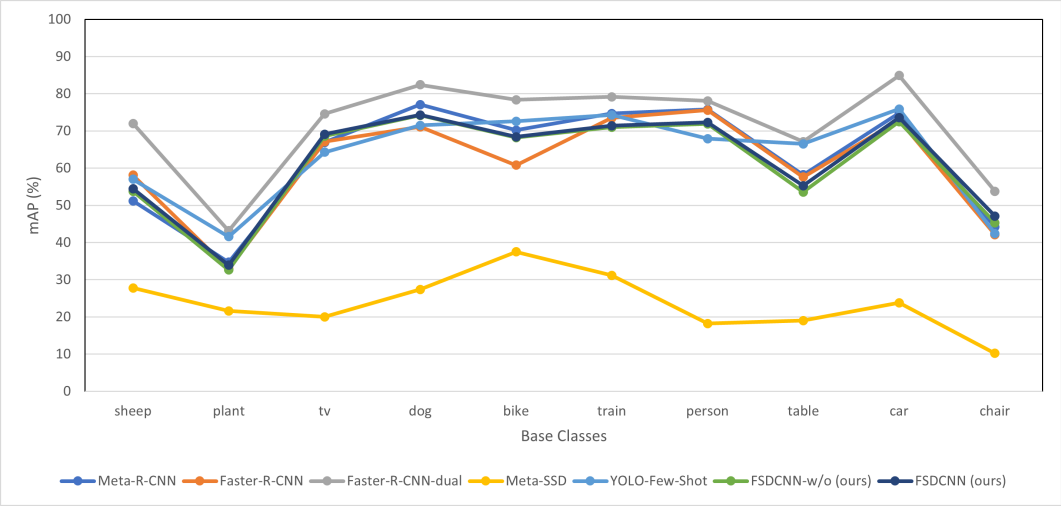

Figure 5.38
The mAP achieved for base classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 1-shot setting.

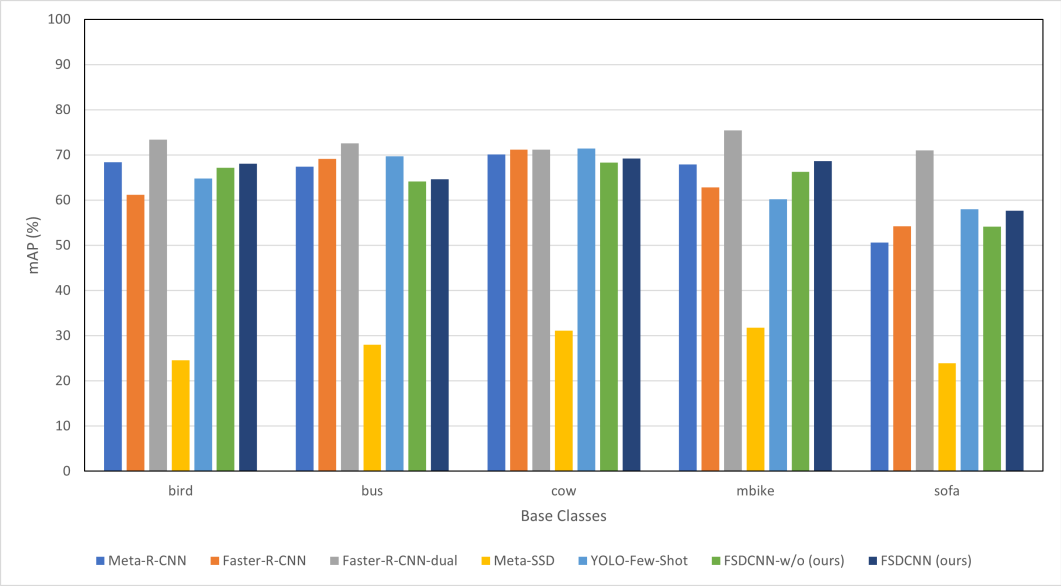

Figure 5.39
The mAP achieved for base classes bird, bus, cow, mbike and sofa over thousand runs in 3-shot setting.

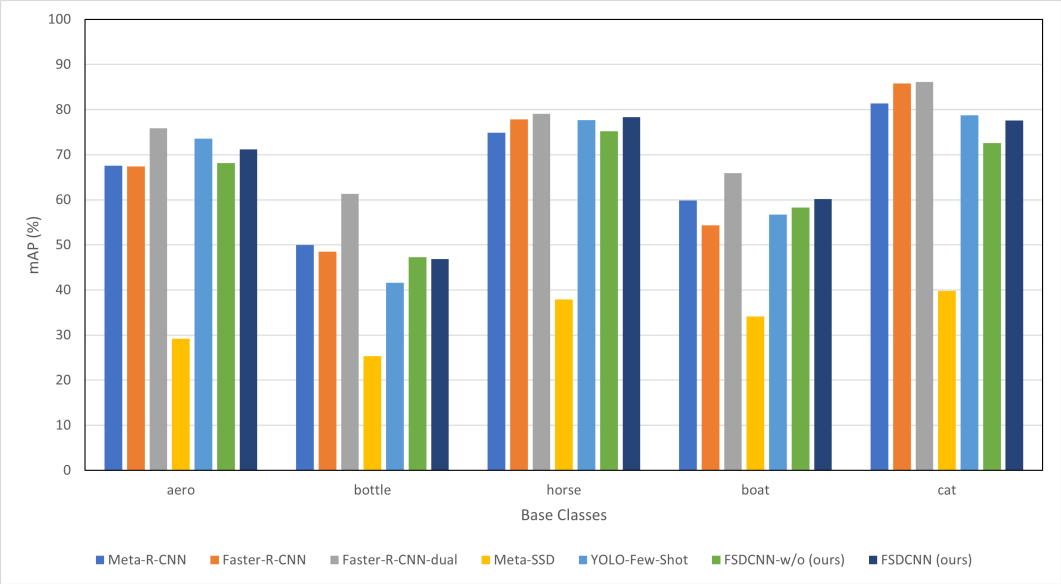

Figure 5.40
The mAP achieved for base classes aero, bottle, horse, boat and cat over thousand runs in 3-shot setting.

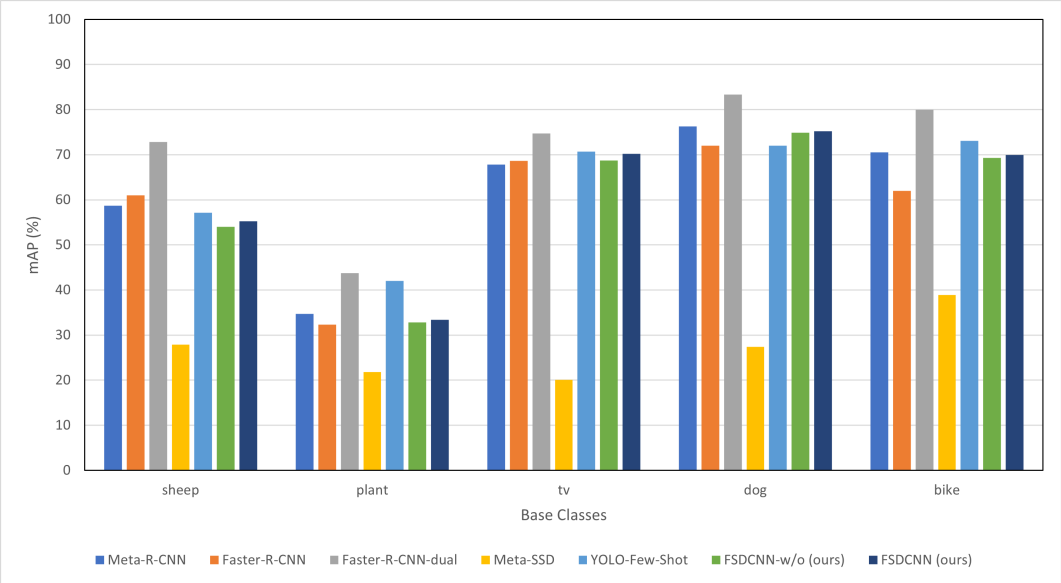

Figure 5.41
The mAP achieved for base classes sheep, plant, tv, dog and bike over thousand runs in 3-shot setting.

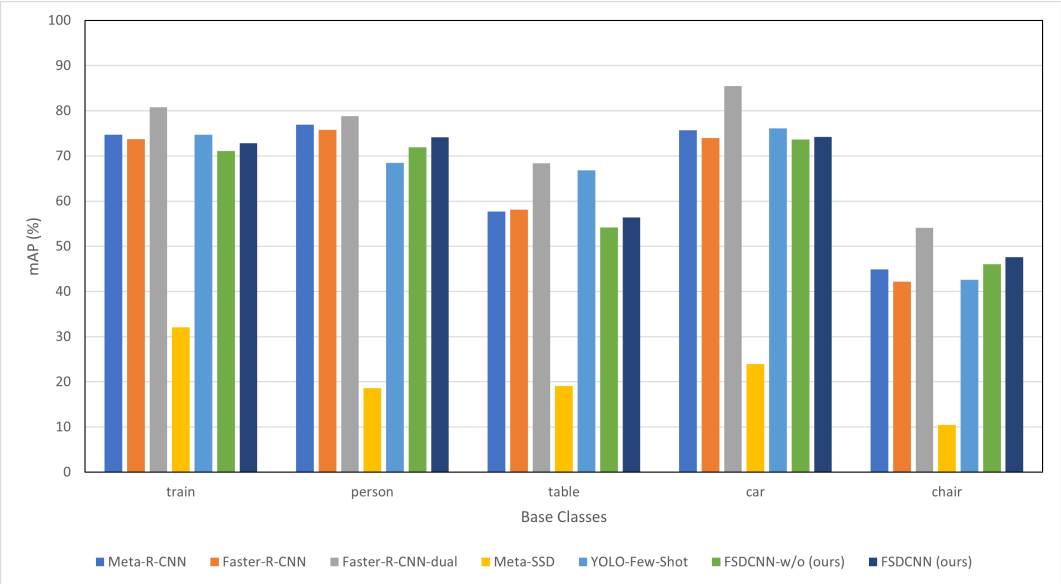

Figure 5.42
The mAP achieved for base classes train, person, table, car and chair over thousand runs in 3-shot setting.

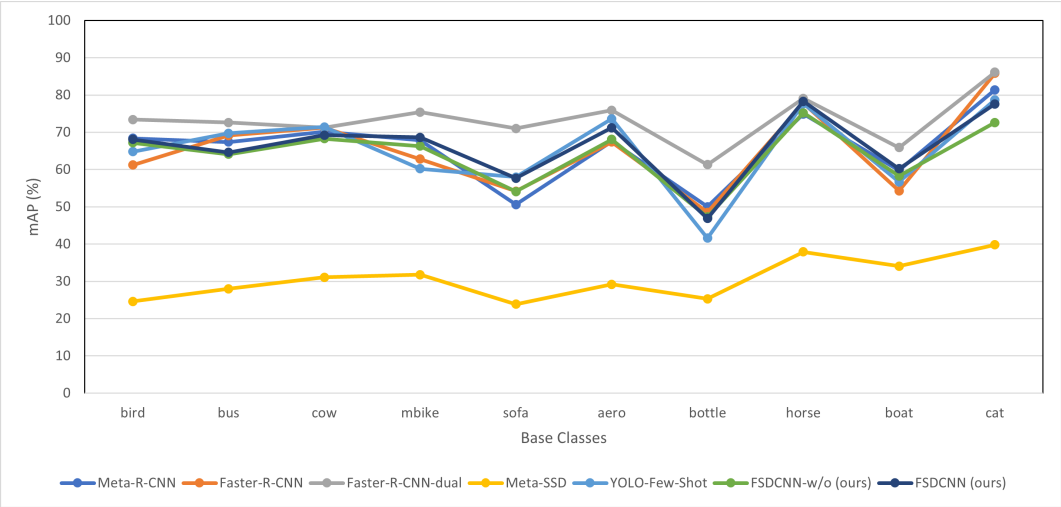

Figure 5.43
The mAP achieved for base classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 3-shot setting.

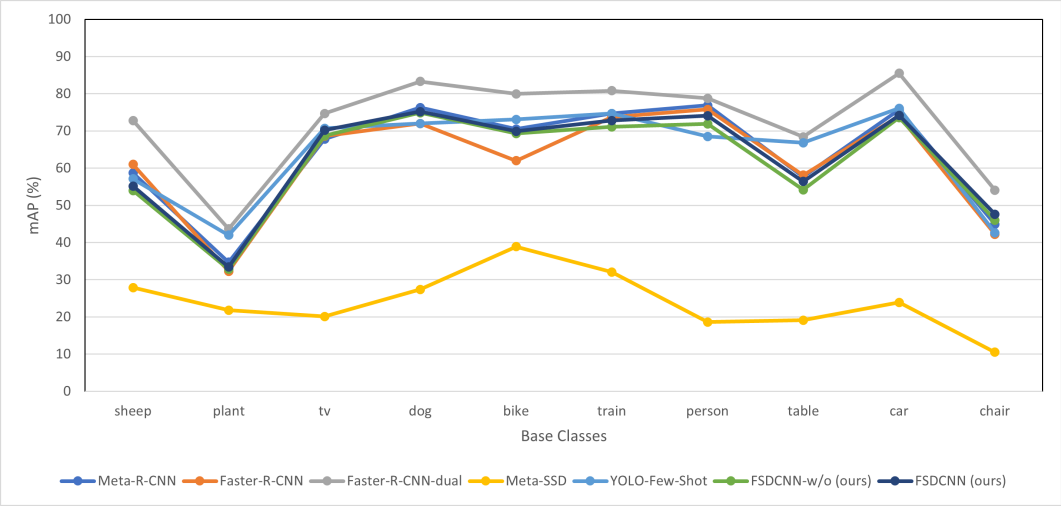

Figure 5.44
The mAP achieved for base classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 3-shot setting.

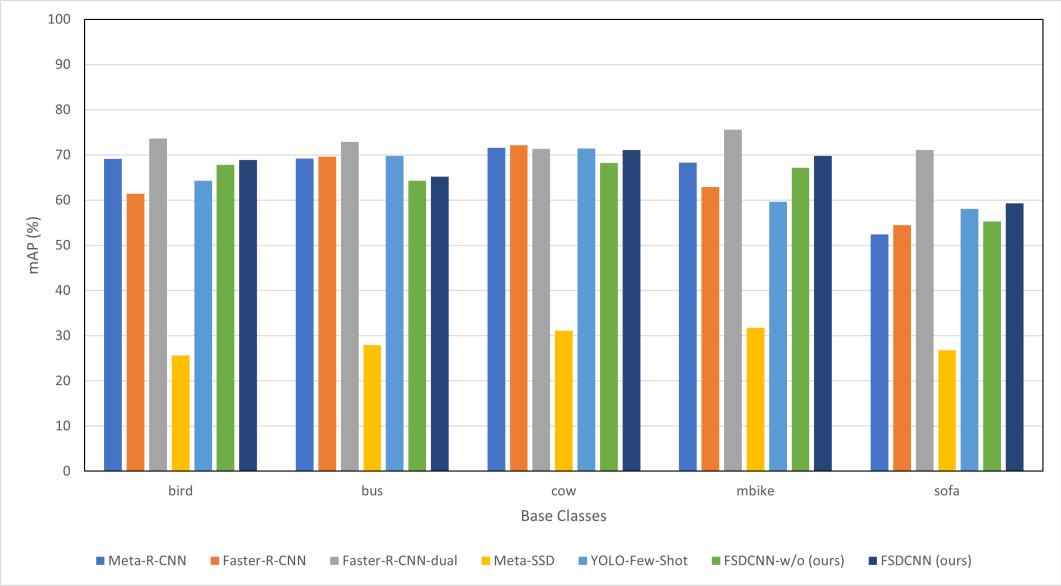

Figure 5.45
The mAP achieved for base classes bird, bus, cow, mbike and sofa over thousand runs in 5-shot setting.

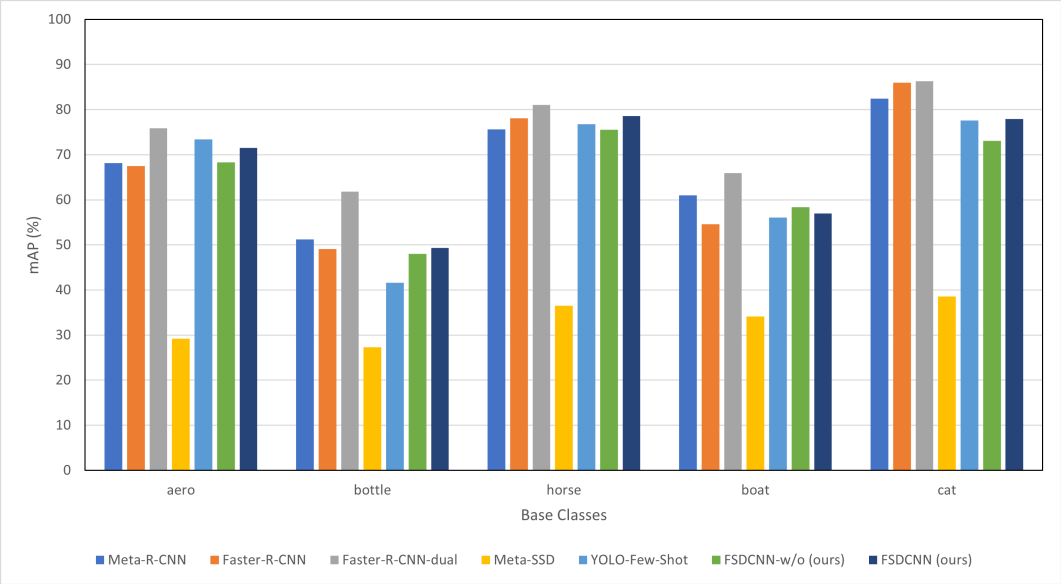

Figure 5.46
The mAP achieved for base classes aero, bottle, horse, boat and cat over thousand runs in 5-shot setting.

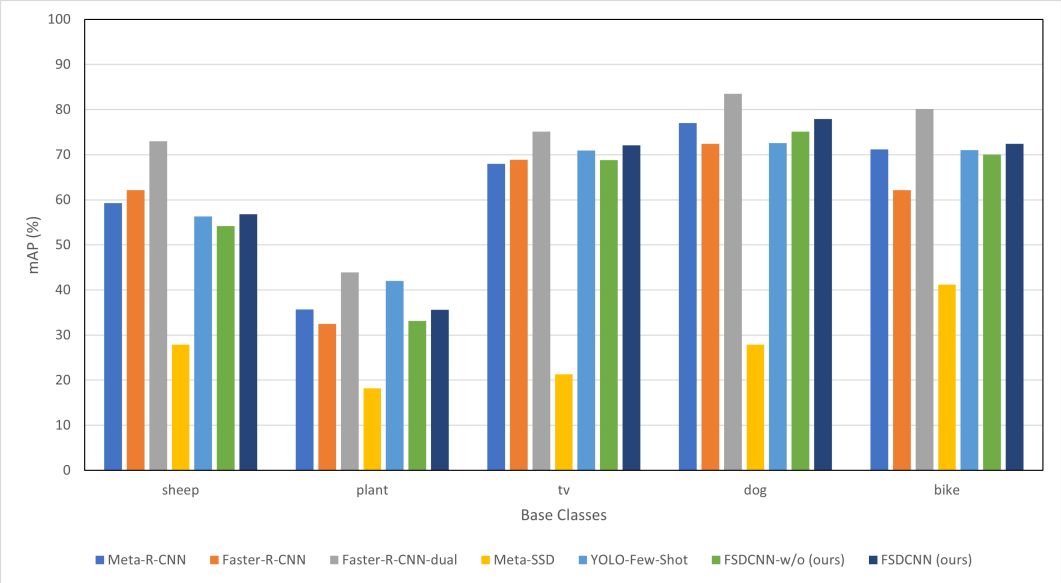

Figure 5.47
The mAP achieved for base classes sheep, plant, tv, dog and bike over thousand runs in 5-shot setting.

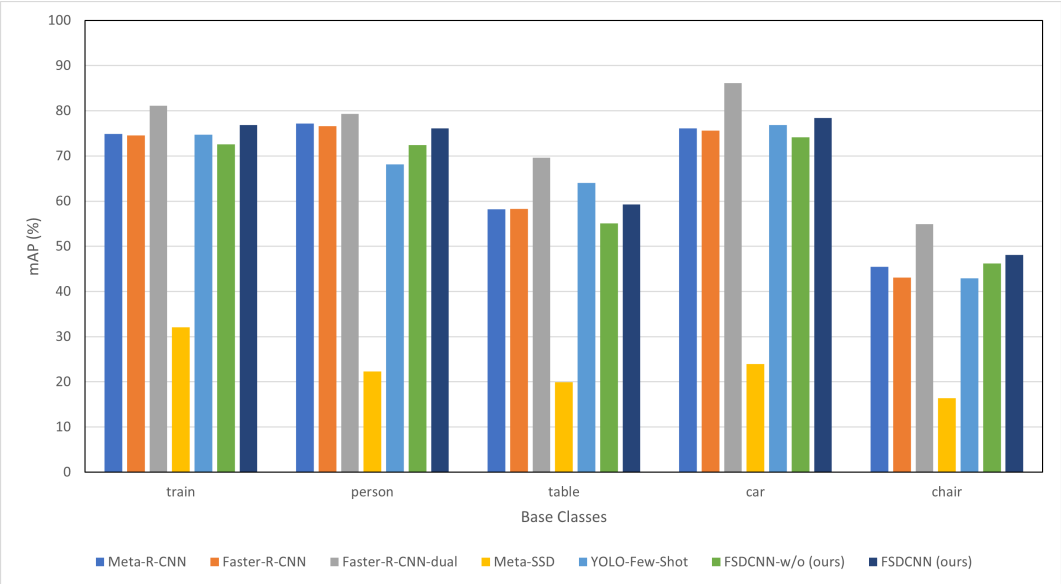

Figure 5.48
The mAP achieved for base classes train, person, table, car and chair over thousand runs in 5-shot setting.

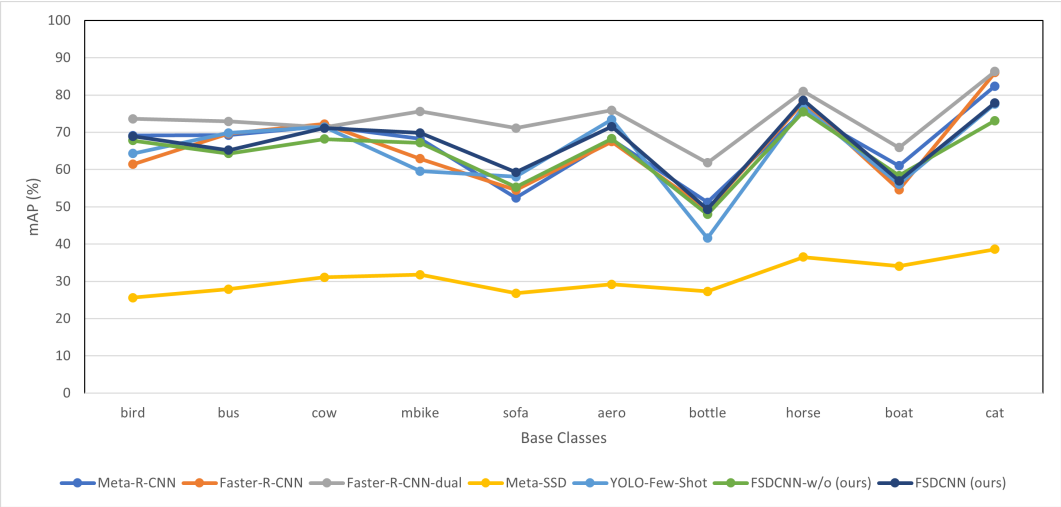

Figure 5.49
The mAP achieved for base classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 5-shot setting.

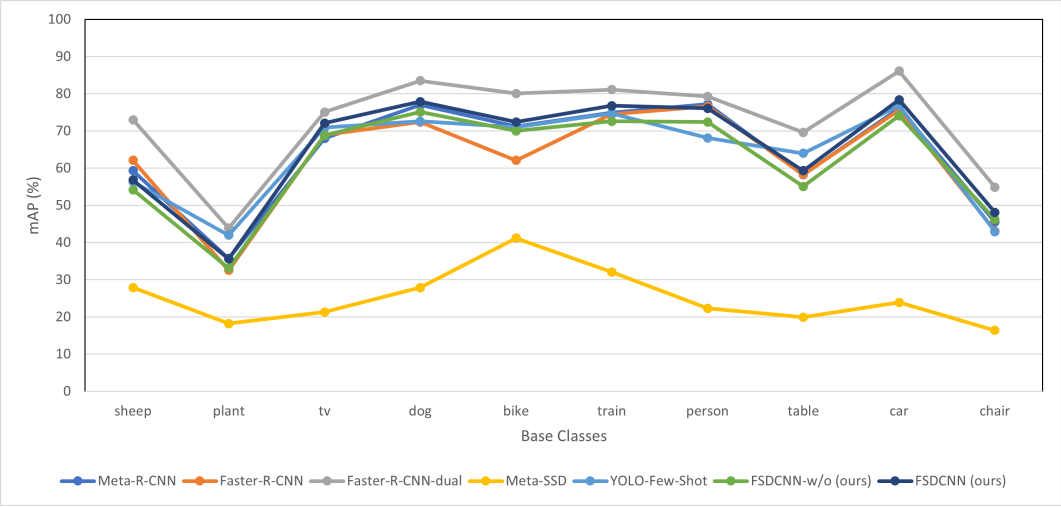

Figure 5.50
The mAP achieved for base classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 5-shot setting.

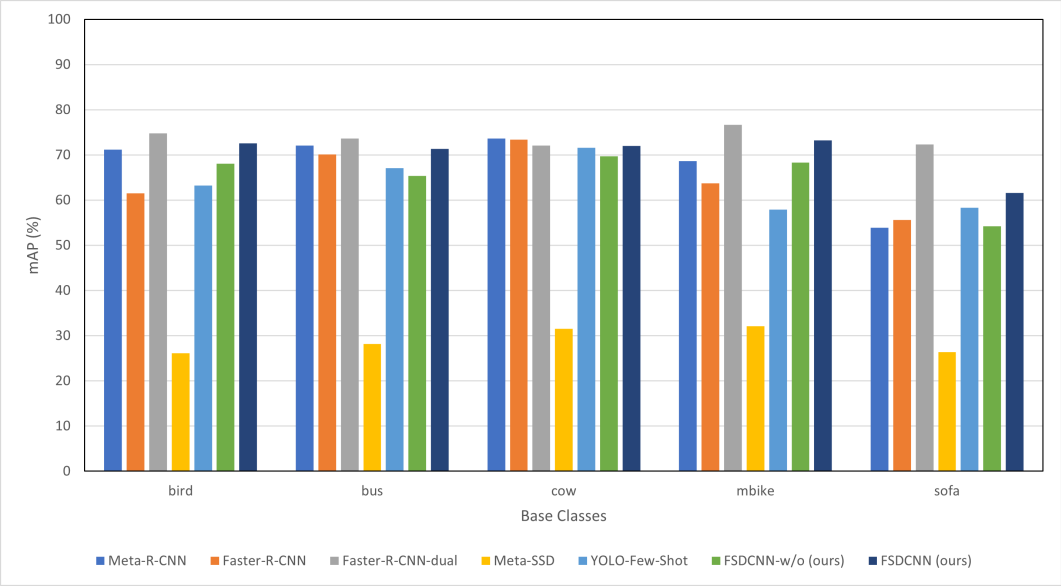

Figure 5.51
The mAP achieved for base classes bird, bus, cow, mbike and sofa over thousand runs in 10-shot setting.

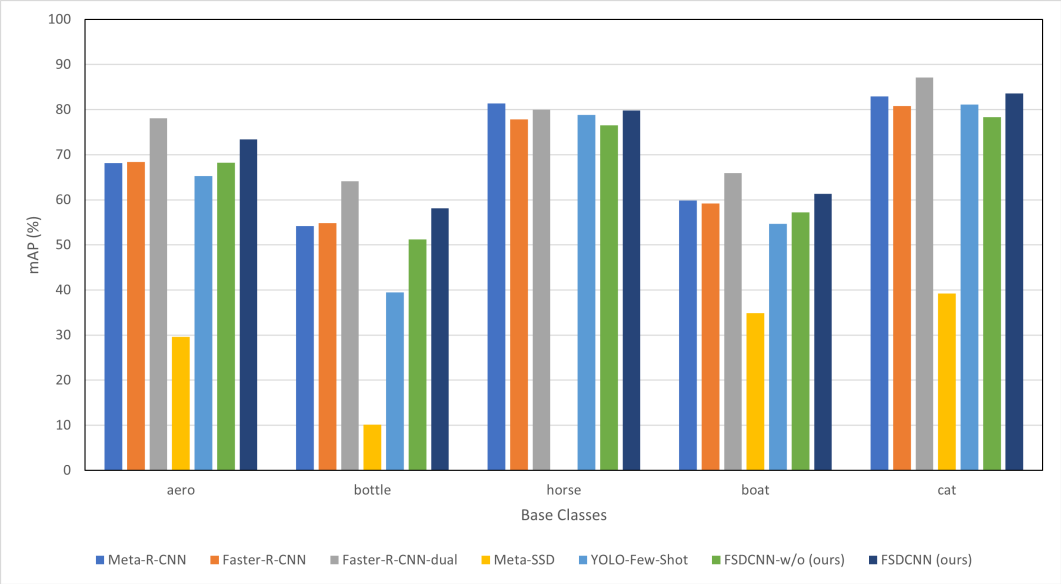

Figure 5.52
The mAP achieved for base classes aero, bottle, horse, boat and cat over thousand runs in 10-shot setting.

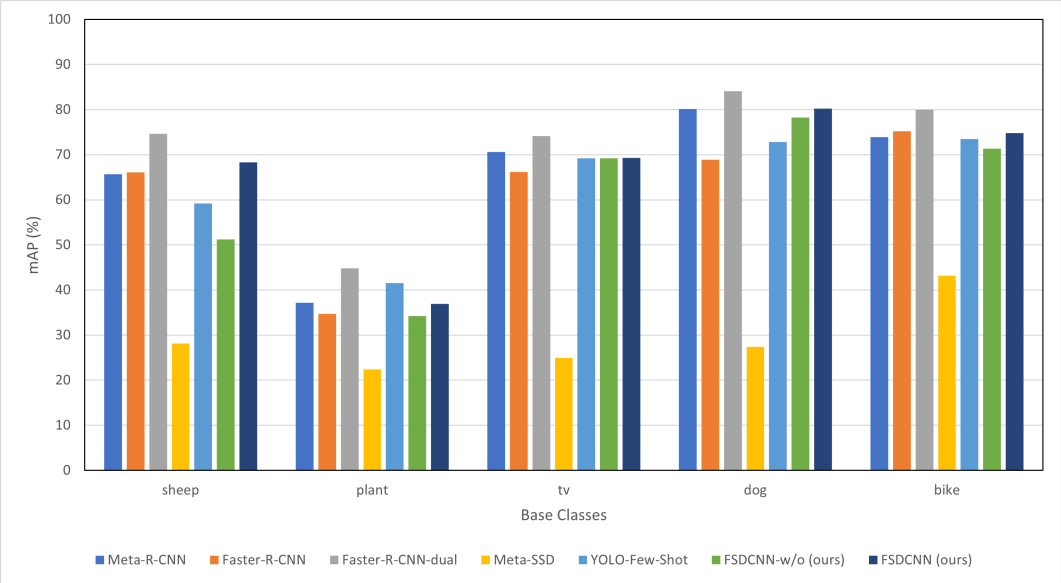

Figure 5.53
The mAP achieved for base classes sheep, plant, tv, dog and bike over thousand runs in 10-shot setting.

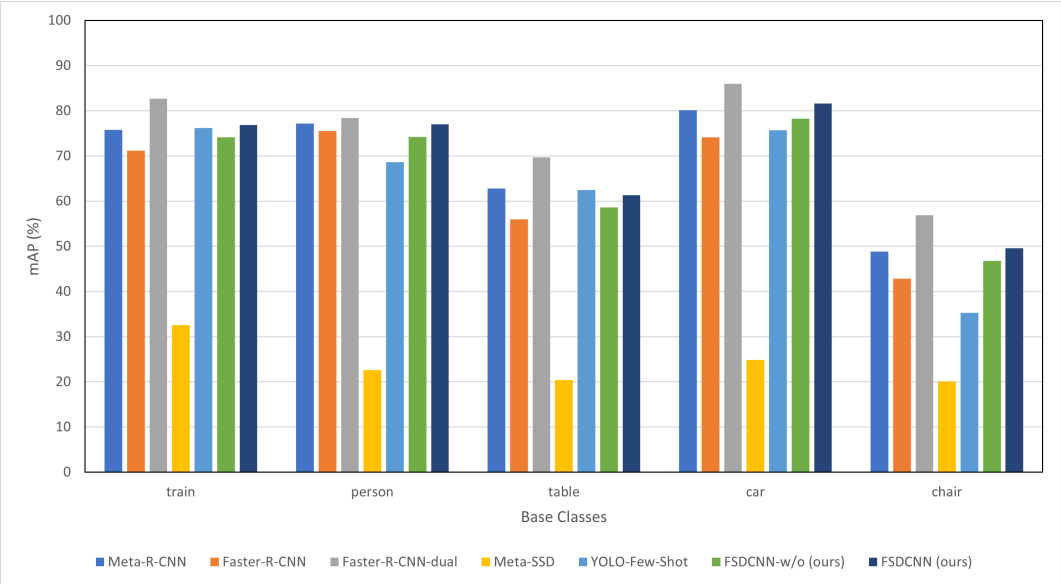

Figure 5.54
The mAP achieved for base classes train, person, table, car and chair over thousand runs in 10-shot setting.

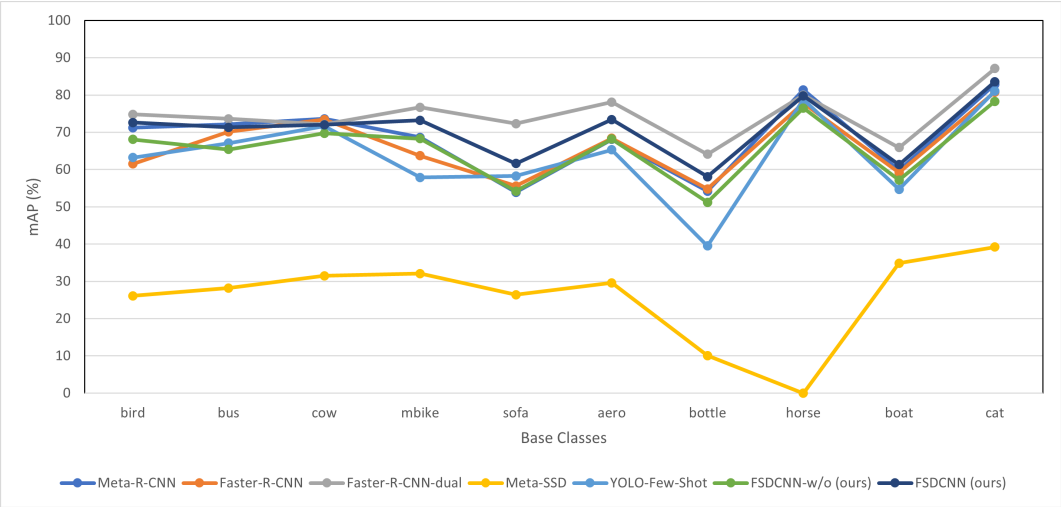

Figure 5.55
The mAP achieved for base classes bird, bus, cow, mbike, sofa, aero, bottle, horse, boat and cat over thousand runs in 10-shot setting.

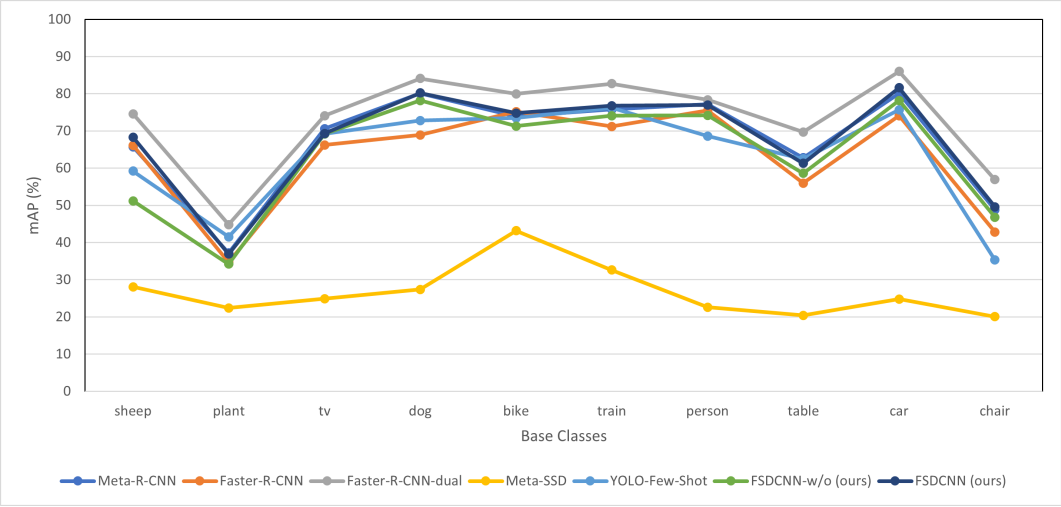

Figure 5.56
The mAP achieved for base classes sheep, plant, tv, dog, bike, train, person, table, car and chair over thousand runs in 10-shot setting.

K-shot	Method	bird	bus	cow	mbike	sofa	aero	bottle	horse	boat	cat	sheep	plant	tv	dog	bike	train	person	table	car	chair
1-shot	Meta R-CNN [1]	67.2	66.2	65.1	64.2	48.6	66.2	49.2	72.1	55.9	77.5	51.2	34.7	66.9	77.1	70.2	74.7	75.8	58.2	74.8	44.2
1-shot	Faster R-CNN [10]	59.1	68.3	71	61.4	49.6	65.3	48.5	77.6	53.9	82.1	58.1	32.9	67.1	71.1	60.8	73.5	75.6	57.6	72.9	42.1
1-shot	Faster R-CNN-dual	71.2	71.9	70.5	74.6	70.5	74.6	61.2	78.4	65.1	85.2	72	43.2	74.6	82.4	78.4	79.2	78.1	67.1	84.9	53.8
1-shot	Meta-SSD [14]	24.3	27.4	30.8	31.4	23.6	29.1	25.4	37.5	34.2	39.9	27.8	21.6	20	27.4	37.5	31.2	18.2	19	23.8	10.2
1-shot	YOLO-Few-Shot [42]	62.4	67.2	70.1	59.4	57.2	72.4	42.1	75.1	57.3	77.6	57	41.6	64.3	71.5	72.6	74.3	67.9	66.5	75.9	42.4
1-shot	FSDCNN-w/o (ours)	66.3	64	68.1	66.2	42.4	67.4	47.2	74.8	58.2	72.4	53.8	32.6	68.6	74.2	68.2	71	71.9	53.6	72.5	45.3
1-shot	FSDCNN (ours)	67.9	64.3	69.1	68.2	57.1	70.3	45.7	77.6	59.8	77.2	54.5	33.9	69.1	74.3	68.4	71.4	72.3	55.3	73.6	47.1
3-shot	Meta R-CNN [1]	68.4	67.4	70.1	67.9	50.6	67.6	50	74.9	59.8	81.4	58.7	34.7	67.8	76.3	70.5	74.7	76.9	57.7	75.7	44.9
3-shot	Faster R-CNN [10]	61.2	69.1	71.2	62.8	54.2	67.4	48.5	77.8	54.3	85.8	61	32.3	68.6	72	62	73.7	75.8	58.1	74	42.2
3-shot	Faster R-CNN-dual	73.4	72.6	71.2	75.4	71	75.9	61.3	79.1	65.9	86.1	72.8	43.7	74.7	83.3	80	80.8	78.8	68.4	85.5	54.1
3-shot	Meta-SSD [14]	24.6	28	31.1	31.8	23.9	29.2	25.3	37.9	34.1	39.8	27.9	21.8	20.1	27.4	38.9	32.1	18.6	19.1	23.9	10.5
3-shot	YOLO-Few-Shot [42]	64.8	69.7	71.4	60.2	58	73.6	41.6	77.7	56.7	78.7	57.1	42	70.7	72	73.1	74.7	68.5	66.8	76.1	42.6
3-shot	FSDCNN-w/o (ours)	67.2	64.1	68.3	66.3	54.1	68.1	47.3	75.2	58.3	72.6	54	32.8	68.7	74.9	69.3	71.1	71.9	54.2	73.6	46
3-shot	FSDCNN (ours)	68.1	64.6	69.2	68.6	57.7	71.2	46.9	78.3	60.2	77.6	55.2	33.4	70.2	75.2	69.9	72.8	74.1	56.4	74.2	47.6
5-shot	Meta R-CNN [1]	69.1	69.2	71.6	68.3	52.4	68.1	51.2	75.6	61	82.4	59.3	35.7	68	77	71.2	74.9	77.2	58.2	76.1	45.5
5-shot	Faster R-CNN [10]	61.4	69.6	72.2	62.9	54.5	67.5	49.1	78.1	54.6	86	62.1	32.5	68.9	72.4	62.1	74.5	76.6	58.3	75.6	43.1
5-shot	Faster R-CNN-dual	73.6	72.9	71.3	75.6	71.1	75.9	61.8	81	65.9	86.3	73	43.9	75.1	83.5	80.1	81.1	79.3	69.6	86.1	54.9
5-shot	Meta-SSD [14]	25.6	27.9	31.1	31.8	26.8	29.2	27.3	36.5	34.1	38.6	27.9	18.2	21.3	27.9	41.2	32.1	22.3	19.9	23.9	16.4
5-shot	YOLO-Few-Shot [42]	64.3	69.8	71.4	59.6	58.1	73.4	41.6	76.8	56.1	77.6	56.3	42	70.9	72.6	71	74.7	68.1	64	76.8	42.9
5-shot	FSDCNN-w/o (ours)	67.8	64.3	68.2	67.2	55.3	68.3	48	75.5	58.4	73.1	54.2	33.1	68.8	75.1	70	72.6	72.4	55.1	74.1	46.2
5-shot	FSDCNN (ours)	68.9	65.2	71.1	69.8	59.3	71.5	49.3	78.6	57	77.9	56.8	35.6	72.1	77.9	72.4	76.8	76.1	59.3	78.4	48.1
10-shot	Meta R-CNN [1]	71.2	72.1	73.6	68.6	53.9	68.1	54.2	81.4	59.8	82.9	65.7	37.2	70.6	80.1	73.9	75.8	77.2	62.8	80.1	48.8
10-shot	Faster R-CNN [10]	61.5	70.1	73.4	63.7	55.6	68.4	54.8	77.8	59.2	80.8	66.1	34.7	66.2	68.9	75.2	71.2	75.5	56	74.1	42.8
10-shot	Faster R-CNN-dual	74.8	73.6	72.1	76.7	72.3	78.1	64.1	80	65.9	87.1	74.6	44.8	74.1	84.1	80	82.7	78.4	69.7	86	56.9
10-shot	Meta-SSD [14]	26.1	28.2	31.5	32.1	26.4	29.6	10.1	36.8	34.9	39.2	28.1	22.4	24.9	27.4	43.2	32.6	22.6	20.4	24.8	20.1
10-shot	YOLO-Few-Shot [42]	63.2	67.1	71.6	57.9	58.3	65.3	39.5	78.8	54.7	81.1	59.2	41.5	69.2	72.8	73.5	76.2	68.6	62.5	75.7	35.3
10-shot	FSDCNN-w/o (ours)	68.1	65.4	69.7	68.3	54.2	68.2	51.2	76.5	57.2	78.3	51.2	34.2	69.2	78.2	71.3	74.1	74.2	58.6	78.2	46.8
10-shot	FSDCNN (ours)	72.6	71.3	72	73.2	61.6	73.4	58.1	79.8	61.3	83.6	68.3	36.9	69.3	80.2	74.8	76.8	77	61.3	81.6	49.6

Table 5.3
Performance comparison of the different models in classifying each of the 20 base classes over a thousand runs for 1-shot, 3-shot, 5-shot and 10-shot. (Color guide: red is for the best performance, green is for second best and blue is for third best.)

closely concentrated near 15. It also shows that the mAP values are very close to the median value in case of FSDCNN.

Similarly, Figures 5.58, 5.59 and 5.60 show the distribution of mAP values for 3-shot, 5-shot and 10-shot settings respectively. It can be seen in the figures that mAP values for FSDCNN are very close to the median in each of the settings. When compared to that of Meta R-CNN, the mAP values of FSDCNN are more spread out from the median in 10-shot setting and less spread out in 1-shot, 3-shot and 5-shot settings. This shows the consistency in performance achieved by FSDCNN.

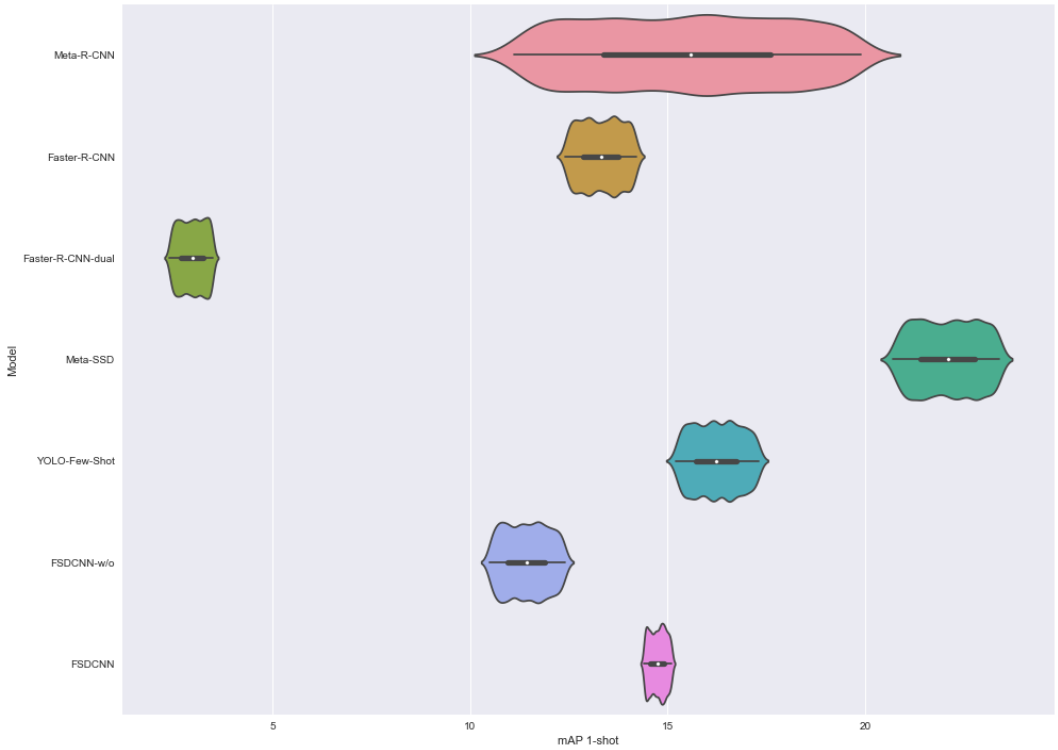

Figure 5.57
Violin plot showing the distribution of mAP values for 1-shot setting over 1000 runs.

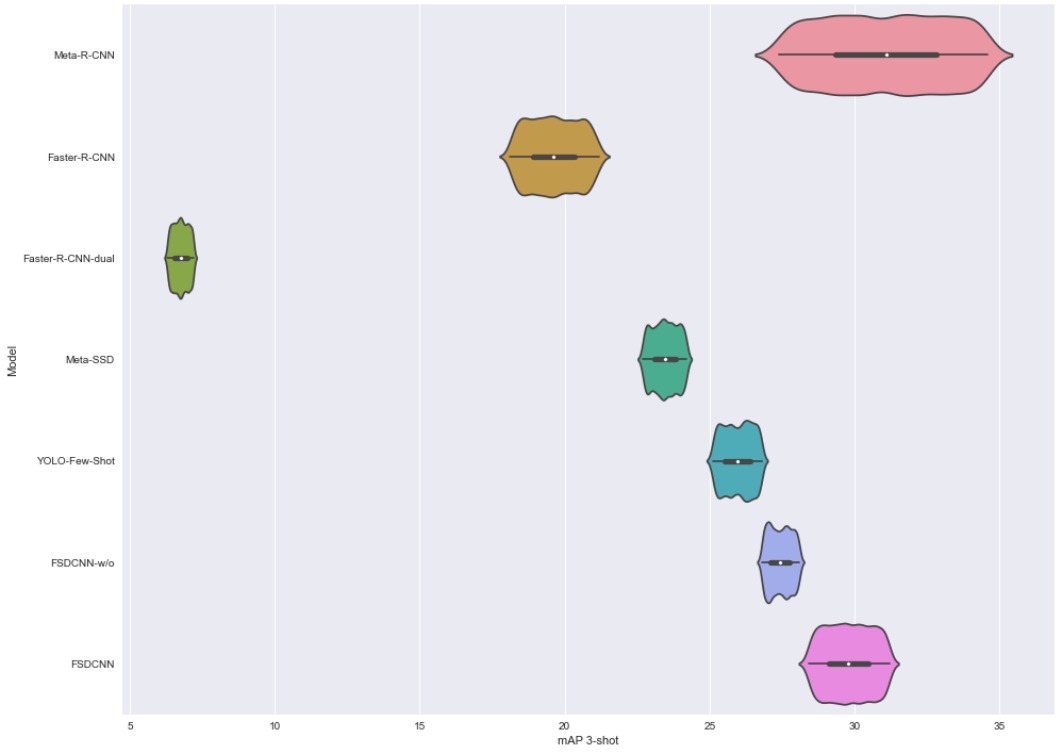

Figure 5.58
Violin plot showing the distribution of mAP values for 3-shot setting over 1000 runs.

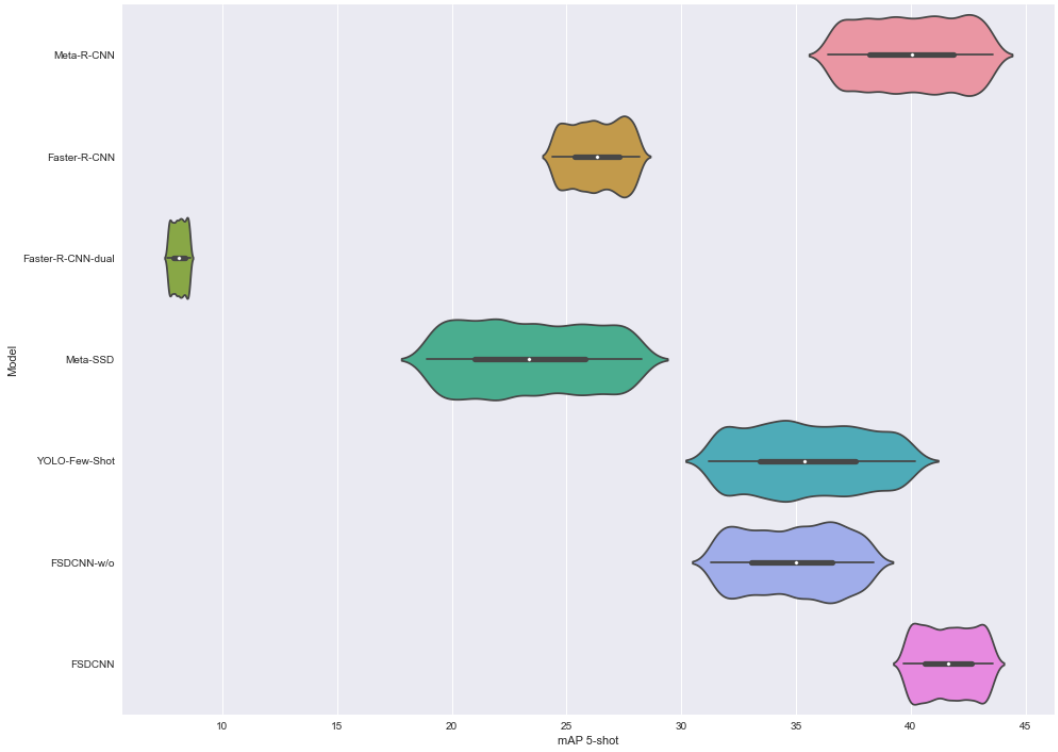

Figure 5.59
Violin plot showing the distribution of mAP values for 5-shot setting over 1000 runs.

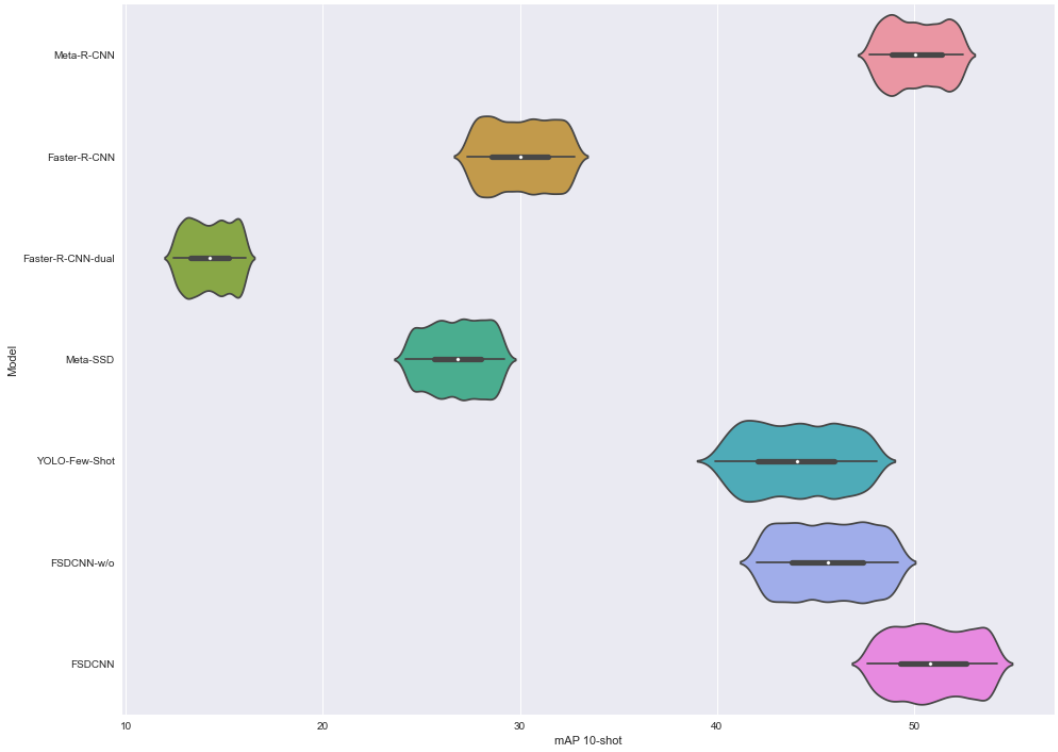

Figure 5.60
Violin plot showing the distribution of mAP values for 10-shot setting over 1000 runs.

CHAPTER VI: SUMMARY AND CONCLUSIONS

Object detectors, since their first implementation, have come a very long way. There have been wide and deep explorations into them but progress on designing detectors that learn from a low-training regime has been slow. In this thesis, we addressed the problem of few-shot detection with the help of meta-learning. We reviewed the existing solutions to the few-shot problem. In our review, we came across solutions such as Meta-SSD [14] and Meta R-CNN [1], who have addressed the problem of few-shot learning from the perspective of meta-learning.

We proposed FSDCNN that combines R-CNN and class matching networks. We update the feature map generator using meta-learning along with the few-shot system for the region classifier stage. The meta-learner also learned the region proposal network. We also preserved the supervised nature of the model without compromising the performance of the model on novel classes.

In the preliminary experiments, our model was able to beat the performance of the state-of-the art models for 3 out of 4 splits for 10-shot and 2 out of 4 for 5-shot, and has the best mAP across all splits. For 3-shot our performance is very similar to that of Meta R-CNN. Our 1-shot is comparable to all models except Meta-SSD which has good performance on 1-shot, but does not see much increase with more samples of the novel classes. We recorded the performance of our model with different K-shot settings where K=1,2,3,4,5,10,15,20,25,30. We observed that the mAP of our model increased when the number of samples were increased. To compare our model's performance against the other models on each class, we plotted and tabulated the performance for each class. Our model outperformed the state-of-the-art models for 10 classes in 5-shot setting and 9 classes in 10-shot setting. For 3-shot our model was consistently in the top two performers along with Meta R-CNN.

The performances of the models on base classes were also plotted and tabulated. Our model was second to Faster R-CNN-dual in detecting and classifying base classes in 1-shot, 5-shot and

10-shot setting. In 3-shot setting, our model was fourth after Faster R-CNN-dual, Meta R-CNN and YOLO-Few-Shot. Meta R-CNN and YOLO-Few-Shot had similar mAP scores and FSDCNN was close behind. Faster R-CNN-dual is a variant of the state-of-the-art model Faster R-CNN trained on both base and novel classes. This shows a slight dip in performance levels on base classes by FSDCNN. However, given the strong performance on novel classes, the slight drop in performance level on base classes can be considered a small compromise.

We had also recorded the performances over a thousand runs and plotted a violin chart to visualize the distribution. The distribution of the mAP scores of our model, in comparison to Meta R-CNN, which is the state-of-the-art few-shot detection mechanism, was closer to the median for 1-shot, 3-shot and 5-shot. It shows the consistency in our performance levels. However, for 10-shot setting, our distribution was a bit far-spread that Meta R-CNN.

Our preliminary results are quite promising, but there is room for improvement. For instance, we are actively looking at techniques to increase our performance for novel classes after just a single example. We are also looking to improve our model over base classes and increase the consistency of mAP scores for 10-shot setting. It does take time to train and test all of these models and we have ongoing experiments that we would like to complete as well. We would also like to improve our model to avoid even the slight drop in performance level on base classes.

Additionally, we would like to use explainable AI techniques to visualize and understand our models better. Neural networks are basically a black box. Using explainable AI techniques, we would like to understand the inner workings of our model. For example, we would like to see if the network is using similar features to predict novel classes as the base classes. During our experiments, we had seen that all the models had a very poor average precision for bottles when bottle was a novel class compared to the high levels when it was a base class. Sofas, boats and plants had similar stories. The performance was poor as a novel class. As every novel class will be a base class in another split, a comparison can be drawn. This will give us insight into understanding what changes can be made to improve our model's performance on such classes.

Another thing we would like to visualize is the features selected for novel classes for different K-shot settings. We observed that the performance of our model increases with the increase in number of samples. The visualization of features over the multiple k-shot settings will probably help us making alterations to our model so that it chooses better features even when the number of samples is low.

We would also like to compare the features selected by our model for classification in one-shot setting against that of Meta-SSD. We observed that even though our model had comparable performances to other models in 1-shot, Meta-SSD outperformed our model. Visualizing the features will help us understand where our model is being outperformed and help us close down the gap to Meta-SSD. These insights may help us to extend the model to improve performance. We would like to visualize the features selected for base classes by our model and Faster-R-CNN-dual. Even though the drop in performance level is small, we would like to see if we can prevent that to preserve the model's performance over base classes.

NOTES

REFERENCES

[1] 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 (ICCV)*, October 2019.

[2] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, jun 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, jun 2016.

[5] Y. Qu, R. K. Baghbaderani, and H. Qi, "Few-shot hyperspectral image classification through multitask transfer learning," in *2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, IEEE, sep 2019.

[6] B. Liu, X. Yu, A. Yu, P. Zhang, G. Wan, and R. Wang, "Deep few-shot learning for hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, pp. 2290–2304, apr 2019.

[7] S.-Y. Chou, K.-H. Cheng, J.-S. R. Jang, and Y.-H. Yang, "Learning to match transient sound events using attentional similarity for few-shot sound recognition," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, may 2019.

[8] K. Saenko, B. Kulis, M. Fritz, and T. Darrell, "Adapting visual category models to new domains," in *Proceedings of the 11th European Conference on Computer Vision: Part IV*, ECCV'10, (Berlin, Heidelberg), p. 213–226, Springer-Verlag, 2010.

[9] K. Saito, Y. Ushiku, T. Harada, and K. Saenko, "Strong-weak distribution alignment for adaptive object detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, jun 2019.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, jun 2017.

[11] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 594–611, apr 2006.

[12] M. Fink, "Object classification from a single example utilizing class relevance metrics," *Advances in neural information processing systems*, vol. 17, pp. 449–456, 2005.

[13] Q. Sun, Y. Liu, T.-S. Chua, and B. Schiele, "Meta-transfer learning for few-shot learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, jun 2019.

[14] K. Fu, T. Zhang, Y. Zhang, M. Yan, Z. Chang, Z. Zhang, and X. Sun, "Meta-SSD: Towards fast adaptation for few-shot object detection with meta-learning," *IEEE Access*, vol. 7, pp. 77597–77606, 2019.

[15] A. Ye, R. Wang, X. Luo, and R. Lan, "Meta-relation networks for few shot learning," in *2019 Eleventh International Conference on Advanced Computational Intelligence (ICACI)*, IEEE, jun 2019.

[16] S. Gidaris and N. Komodakis, "Dynamic few-shot visual learning without forgetting," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[17] A. Mishra, V. K. Verma, M. S. K. Reddy, A. S., P. Rai, and A. Mittal, "A generative approach to zero-shot and few-shot action recognition," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, mar 2018.

[18] J. Xu, S. Ramos, D. Vazquez, and A. M. Lopez, "Domain adaptation of deformable part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, pp. 2367–2380, dec 2014.

[19] A. Raj, V. P. Namboodiri, and T. Tuytelaars, "Subspace alignment based domain adaptation for rcnn detector," 2015.

[20] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[21] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. V. Gool, "Domain adaptive faster r-CNN for object detection in the wild," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[22] J. Lu, Z. Cao, K. Wu, G. Zhang, and C. Zhang, "Boosting few-shot image recognition via domain alignment prototypical networks," in *2018 IEEE 30th International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, nov 2018.

[23] W.-H. Chu and Y.-C. F. Wang, "Learning semantics-guided visual attention for few-shot image classification," in *2018 25th IEEE International Conference on Image Processing (ICIP)*, IEEE, oct 2018.

[24] S. Rahman, S. Khan, and F. Porikli, "A unified approach for conventional zero-shot, generalized zero-shot, and few-shot learning," *IEEE Transactions on Image Processing*, vol. 27, pp. 5652–5667, nov 2018.

[25] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, (Red Hook, NY, USA), p. 3637–3645, Curran Associates Inc., 2016.

[26] G. R. Koch, "Siamese neural networks for one-shot image recognition," 2015.

[27] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, (Red Hook, NY, USA), p. 4080–4090, Curran Associates Inc., 2017.

[28] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, p. 1126–1135, JMLR.org, 2017.

[29] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," in *International Conference on Learning Representations*, 2018.

[30] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[31] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *Proceedings of the 33rd International Conference*

on International Conference on Machine Learning - Volume 48, ICML'16, p. 1842–1850, JMLR.org, 2016.

[32] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[33] S. Qiao, C. Liu, W. Shen, and A. Yuille, "Few-shot image recognition by predicting parameters from activations," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[34] Y. Lifchitz, Y. Avrithis, S. Picard, and A. Bursuc, "Dense classification and implanting for few-shot learning," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, jun 2019.

[35] H. Qi, M. Brown, and D. G. Lowe, "Low-shot learning with imprinted weights," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[36] Y. Zheng, D. K. Pal, and M. Savvides, "Ring loss: Convex feature normalization for face recognition," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[37] W. Wan, Y. Zhong, T. Li, and J. Chen, "Rethinking feature distribution for loss functions in image classification," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, jun 2018.

[38] B. Hariharan and R. Girshick, "Low-shot visual recognition by shrinking and hallucinating features," in *2017 IEEE International Conference on Computer Vision (ICCV)*, IEEE, oct 2017.

[39] W. Liu, Y. Wen, Z. Yu, and M. Yang, "Large-margin softmax loss for convolutional neural networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 507–516, JMLR.org, 2016.

[40] L. Karlinsky, J. Shtok, S. Harary, E. Schwartz, A. Aides, R. Feris, R. Giryes, and A. M. Bronstein, "RepMet: Representative-based metric learning for classification and few-shot object detection," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, jun 2019.

[41] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, sep 2009.

[42] B. Kang, Z. Liu, X. Wang, F. Yu, J. Feng, and T. Darrell, "Few-shot object detection via feature reweighting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.