Few-Shot Object Detection using Transfer Learning

Aleksander Alan Prymek

Dept. of Electronic Computer Engineering

Faculty of Science and Engineering

University of Limerick

Submitted to the University of Limerick for the degree of

Master of Science in Artificial Intelligence 2024

Supervisor: Dr. Tony Scanlan
Department of Electronic & Computer Engineering
University *of* Limerick
Ireland

Abstract

By its nature, deep learning can not generalise well from a few examples. This is why the detection of rare objects is a difficult problem. As industrial demand for inexpensive learning on limited data grows, many new sophisticated few-shot learning methods appeared, albeit few, if any, can challenge a simple transfer learning strategy. In order to present and validate results from this murky area of machine learning, a state-of-the-art method called the Two-stage Fine-tuning Approach is implemented and tested. Moreover, two improvement areas — model fine-tuning adaptation and data augmentation — are identified and shown to even double the performance compared to the baseline on low-shot tasks. The source code is available at: `https://github.com/prymeka/low-shot-object-detection`.

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other Irish or foreign examination board.

The thesis work was conducted from 2023 to 2024 under the supervision of Dr. Tony Scanlan at University of Limerick.

Limerick, 2024

Acknowledgements

I would like to thank my supervisor for continuous support during the project.

Page intentionally left blank.

Contents

List of Tables

LIST OF TABLES

List of Figures

LIST OF FIGURES

1

Introduction

1.1 Overview

Object detection is one of the most fundamental problems in the area of Computer Vision, as the research on this topic can be traced back half a century to the early 1970s (Fischler and Elschlager, 1973). It is only in the last decade, however, that this field has experienced rapid growth and has been effectively applied to real-world problems on a wide scale thanks to the advancements in deep learning (LeCun et al., 2015; Liu et al., 2019; Zhao et al., 2019) abetted by rapid growth in computational power.

The progress in data-abundant problems seems to have stalled after the development of the current state-of-the-art (SOTA) model architectures: Faster R-CNN (Region-based Convolutional Neural Network) (Girshick, 2015), SSD (Single Shot Multibox Detector) (Liu et al., 2016), and YOLO (You Only Look Once) (Redmon et al., 2016). The focus has shifted to expanding the usability of object detection to more problems like those where gargantuan amounts of annotated samples are not available, for example, the detection of rare animal species (Mannocci et al., 2022) or various medical applications like the detection of bone fractures or other rare diseases (Yahalomi et al., 2019).

In the last few years, research in the area of few-shot learning (FSL), where prior knowledge is used to compensate for the lack of comprehensive supervised information, has been picking up pace. Many large and small annotated datasets

have been made public, allowing for the extraction and generalisation of knowledge learned on these datasets. Additionally, popular development frameworks (Tensorflow, Pytorch) have made pre-trained models readily available, allowing almost anyone to fine-tune the models on new task-specific datasets.

Howbeit, typical deep learning techniques can not generalize from few examples. They levy heavy training costs due to their almost unquenchable thirst for data and typically slow training. Hence, as the number of samples decreases, the efficacy of the FSL methods becomes increasingly important.

A lot of effort has been put into developing sophisticated ways designed to overcome these limitations; nonetheless, few, if any, claim supremacy over the simplest method centred on Transfer Learning presented by Wang, Huang, Darrell, Gonzalez and Yu (2020) and called the Two-stage Fine-tuning Approach (TFA).

1.2 Motivation

Object detection often serves as a basis for addressing many intricate visual tasks such as image segmentation, object tracking, event detection, and activity recognition. Practical applications for this field are vast and ever-growing, including, *inter alia*, robot vision, autonomous vision, and human-computer interactions. But, to come close to fully solving these problems and to even dream of approaching human level in problem solving on multiple tasks, it is crucial to solve the FSL problem.

Humans and animals have inspired many developments in Artificial Intelligence (AI) even since before the inception of the discipline as we know it today. FSL aims to bridge the gap between the way AI and humans learn. We are naturally capable of learning to recognise new categories of objects from just a few examples, sometimes even after seeing them once. It is no surprise then that FSL is becoming an increasingly popular area of research.

Industrial demand for inexpensive learning, often on rather poor (qualitatively and quantitatively) datasets, induced interest in FSL. Object detection, being one of the central AI problems, naturally exalts few-shot object detection (FSOD) to

an exceedingly influential field of study. It should now be obvious why more research is required in this area.

1.3 Research Question and Contributions

The main goal of this work is to reproduce in detail the methodology taken by Wang, Huang, Darrell, Gonzalez and Yu (2020) and validate their findings. To do so, I will first introduce the basic concepts and definitions related to FSOD as well as the most critical advancements present in the literature related to this area. Then, I will implement the TFA and compare several modifications to find ways of improving its performance without incurring significant computational costs or overly complicating the methodology.

The main contributions of this work are as follows: I propose appending new dense layers on top of the model instead of replacing them when modifying the model for fine-tuning; and, show that the method for novel weight initialisation and the type of classifier used loses importance when proper data augmentation is used as optimal data augmentation has the most potential for improving the final performance metrics.

1.4 Thesis Outline

In chapter 2, basic concepts related to FSOD are introduced and a review of the most important developments is given. Chapter 3 presents the methodology for the comparative experiments done to find out potential pathways for further enhancing the model's performance. Chapter 4, describes and rationalises the outcomes of the experiments as well as provides error analysis of the models trained in Chapter 3. Chapter 5 concludes the paper by stating the limitations of the results and proposes directions for further research.

2

Literature Review

2.1 Object Detection

One of the most rudimentary Computer Vision tasks is that of image classification: categorising images into a set of finite classes (Rawat and Wang, 2017). In these tasks, images contain a single object that is centred and scaled to take up most of the image. Object localisation is a step up in difficulty from image classification as it deals with neither centred nor scaled images. The goal is then to determine the smallest possible bounding box (usually defined by four values: x and y coordinates as well as the height and width of the box) around the object and assign it an appropriate label. Object detection is the generalisation to many objects of potentially various classes in a single image.

Object detection has benefited greatly from the development of deep learning applied to data-intensive applications. The first breakthrough defining the current SOTA trends happened in 2014 with the introduction of Region-based CNN (R-CNN) (Girshick et al., 2014), which used Selective Search (Uijlings et al., 2013) for region proposal, a deep neural network to detect objects in each region, and a Support Vector Machine to classify detections. An improved version, Fast R-CNN (Girshick, 2015), was presented a year later, but it still suffered from the handicap that was the separation of the region proposal module and the actual regression network.

One of the solutions was achieved by unifying the whole model into a single deep neural network. Faster R-CNN (Ren et al., 2015) achieved this through

the introduction of a Region Proposal Network (RPN) that shared full-image convolutional features with the detection network.

Notably, at roughly the same time, Redmon et al. (2016) introduced the You Only Look Once (YOLO) algorithm that took a completely different approach by redefining the detection task as a single regression problem. YOLO divides an image into an $S \times S$ grid where each cell predicts whether it contains an object and, if so, also its bounding box. The model had a lower rate of false positives on the background compared to Faster-RCNN as it considers the image globally when detecting objects; however, it also had a higher localisation error rate. Moreover, YOLO struggles with small objects, as each cell could predict only one bounding box.

Due to its small size and simple forward pass, YOLO achieved a remarkable detection speed of 45 frames per second, meaning it could detect objects in real-time. Because of its speed, researchers happily worked to improve the precision of the model, resulting in a total of nine canonical versions of YOLO so far (Redmon et al., 2016; Wang et al., 2024).

Hereunder, we will entirely focus on Faster R-CNN architecture since, it generally achieves higher precision on most benchmarks, and longer inference speed is not an issue in many few-shot tasks.

2.2 Few-Shot Learning (FSL)

Few-shot learning is a new machine learning paradigm (Fink, 2004; Li et al., 2006; Wang, Yao, Kwok and Ni, 2020). Following Wang, Yao, Kwok and Ni (2020), we define FSL to be a type of machine learning problem where only a limited number of examples with supervised information for the target task is available.

Effective FSL methodologies reduce the need for potentially costly or impossibly large amounts of data and can generalise only from few examples. They can also allow for rapid prototyping and testing therefore reducing a project's risk. There are three mostly independent strategies that one can use when faced with an FSL problem: enriching data, optimising model selection, and optimising the algorithm.

2.2.1 Enriching Data with Data Augmentation

Methods that enrich the data use prior knowledge to augment the existing data and increase the number of virtual samples as well as their diversity. Standard machine learning models can then be used on the augmented data with a lower risk of overfitting and better accuracy (Krizhevsky et al., 2012).

Usually, augmentation policies are manually designed in an *ad hoc* manner with a specific dataset in mind and are not necessarily optimal for other tasks. For example, horizontal flipping might work well for the Fashion-MNIST dataset (images of garments) but not necessarily for the original MNIST dataset (images of digits).

Cubuk et al. (2019) presented a solution to this problem called AutoAugment, which uses a Reinforcement Learning search algorithm (Zoph and Le, 2016) to find the augmentation policy that maximises the validation accuracy of a given model on a specified dataset. The search space consists of 5 sub-policies, each consisting of 2 image operations that are applied one after another. Moreover, each operation has two hyperparameters, probability and magnitude, discretised into 11 and 10 values. The huge downside of this method is the prohibitively large hyperparameter search space combined with the timely training of a child network for each policy. Thus, to fully harness its power, this algorithm requires thousands of GPU hours, even for relatively small datasets. Luckily, it is possible to transfer augmentation policies from one dataset to another with relatively decent effectiveness.

Lim et al. (2019) proposed a different search strategy implemented in their Fast AutoAugment method that does not require repeated training of the child model and instead searches for a match between the distributions of the augmented subset and unaugmented subset of the dataset using a single model. Instead of the usual view of data augmentation as a way of encoding invariances in the data, here the goal is to "fill out" the dataset with "missing" points, hence it is essential to maintain the original data distribution. This approach, while only slightly worse, reduced the GPU hours required to produce an augmentation policy for ImageNet over 30-fold. Hataya et al. (2020) developed an even faster method, aptly called Faster AutoAugment, based on the same principle

that reduced the computational cost by another factor of 200 (without changing the search space) using a differentiable policy search pipeline.

Cubuk et al. (2020) showed that the reduction of the search space is much more significant than improved search algorithms and, if done correctly, allows for a more direct solution that does not require a smaller proxy task as was employed in the previous AutoAugment modifications. RandAugment randomly selects N from the total of K operations, each with equal likelihood and the same global magnitude schedule, resulting in the K^N complexity of the search space. It is then possible for an institutional actor to find optimal augmentation policy using a naive grid search. The results proved to be comparable to the AutoAugment and minimally better than the Fast(er) AutoAugment.

Lamentably, the main focus of these methods is image classification, which, when it comes to data augmentation, does not always translate well to object detection. An example of an object-detection-specific augmentation policy investigation was presented in the YOLOv4 paper (Bochkovskiy et al., 2020). The most effective operations found were CutOut (DeVries and Taylor, 2017), label smoothing (Szegedy et al., 2016) and a new augmentation method called Mosaic, where four images are combined into a square grid and then randomly cropped to size. This new augmentation method forces the model to learn to detect objects outside of their usual context. Also, when using Batch Normalisation (Ioffe and Szegedy, 2015), it reduces the need for large batch size as each sample contributes four images to the calculation of the activation statistics.

Zoph et al. (2020) presented an AutoAugment-like approach for object detection achieving SOTA results. Aside of standard operations, they included bounding box-only transformations, where only a selected bounding box region is transformed. Authors found that the learned augmentation brings the most improvements on small training datasets and, in some situations, can effectively double the size of a dataset. This observation underlines the importance of good data augmentation for few-shot tasks. Their results show greater relative improvement on AP75 than AP50 as well as in locating small objects, implying that the primary benefit of a good augmentation policy is better fine spatial detail understanding and bounding box positioning.

2.2.2 Model Selection

Methods that simplify the model selection use prior knowledge to constrain the complexity of the hypothesis space to a space small enough such that the given data is enough to explore it and learn a reliable hypothesis. This can be accomplished by Metric Learning (Kaya and Bilge, 2019), that is, by learning to project data samples to a smaller embedding space to simplify the task and, thus, by extension, reduce the hypothesis space. The main goal is then to preserve the class neighbourhood structure; or, in other words, to embed similar samples or samples belonging to the same class closer together while dissimilar samples or samples from other classes further apart so that a simpler model can be used such that the data gathered is sufficient.

2.2.3 Algorithm

Algorithm optimisation methods use prior knowledge to search for the parameters defining the best hypothesis by improving the search strategy by either learning the optimal initialisation of the model's parameters or directly learning an optimiser.

Many algorithm optimisation methods exist, the simplest of which is training a model on a related data-abundant task and then initialising the actual model for the proper tasks with the learned parameters. It is assumed that parameters from the related task will capture the generic features and the overall structure of the large-scale data such that it can be easily adapted to the proper task even with a small dataset (Wang, Yao, Kwok and Ni, 2020). This is a widely used approach for any machine learning task called Transfer Learning.

Another example that fits into this category is to use the Model-Agnostic Meta Learner, which learns to initialise a model on a sequence of few-shot tasks using gradient descent (Finn et al., 2017).

2.3 Few-Shot Object Detection (FSOD)

As stated above, object detection models tend to be relatively large and data-thirsty. To effectively learn novel data-scarce classes, they require data-abundant

base categories to learn generic feature representation. For some problems, it is advantageous to not only learn the novel classes but also to retain the knowledge of the base classes (properly called Generalised FSOD), while for others, only novel classes are of interest.

Hereunder, generalised FSOD will be the main focus and will be simply referred to as FSOD.

2.3.1 Taxonomy

Köhler et al. (2023) divided FSOD into two main branches: Transfer Learning and Meta Learning, which is further divided into Single- and Dual-branch approaches. Dual-branch architectures are composed of a query network and a support network that process inputs separately. Single-branch architectures, on the other hand, have structures similar to that of a generic detector and often either directly reduce the number of learnable parameters or utilise metric learning.

2.3.2 Problem Definition

Notation similar to that of Köhler et al. (2023) will be used.

Let $\mathcal{D} = \mathcal{D}_{\text{base}} \cup \mathcal{D}_{\text{novel}}$ be the training dataset separated into two datasets: $\mathcal{D}_{\text{base}}$ containing the set of data-abundant base categories, $\mathcal{C}_{\text{base}}$, and $\mathcal{D}_{\text{novel}}$ containing the set of data-scarce novel categories, $\mathcal{C}_{\text{novel}}$, such that $\mathcal{C}_{\text{base}} \cap \mathcal{C}_{\text{novel}} = \emptyset$.

Let $\mathcal{D}$ consist of tuples $(I_i, \hat{y}_{o_1}, \ldots, \hat{y}_{o_M})$ where each tuple has an image $I_i = \{o_1, \ldots, o_M\}$ containing M objects $o_1, \ldots, o_M$ and their corresponding labels $\hat{y}_{o_i} = \{c_{o_i}, b_{o_i}\}$ where c_{o_i} is the category and b_{o_i} the bounding box $\{x_{o_i}, y_{o_i}, w_{o_i}, h_{o_i}\}$ defined by its top-left corner's coordinates (x_{o_i}, y_{o_i}), width w_{o_i}, and height h_{o_i}.

Generally, few-shot problems are called N-way K-shot where $N \leq |\mathcal{C}_{\text{novel}}|$ and K is the number of annotated object instances available for each category $\mathcal{C}_{\text{novel}}$.

(The number of annotated object instances does not necessarily correspond to the number of images, as one image may contain multiple instances.)

2.3.3 Meta Learning

2.3.3.1 Dual-Branch Meta Learning

A typical dual-branch model is composed of a support network which takes a set of K support images I_i^S from each of N categories with exactly one object o_j and its label $\hat{y}_{o_j}$: $\mathcal{D}^S = \left\{\left(I_i^S, \hat{y}_{o_j}\right)\right\}_{i=1}^{K \cdot N}$, as well as a query network which takes an image I^Q which contains objects to be detected.

Usually, a shared backbone is used to extract the support and query features, $f^{S,B}$ and $f^{Q,B}$, respectively. $f^{S,B}$ are then globally averaged to create f^S support vectors that will be aggregated with the query features before the Region Proposal Network stage and again before the final bounding box regression and classification.

Training is split into E episodes. During each episode the model is trained on an N-way-K-shot task sampled from the $\mathcal{D}_{\text{base}}$. Finally, the model is fine-tuned on a set of K images from each category. If retention of knowledge of the base categories is not needed, full $\mathcal{D}_{\text{novel}}$ is used; otherwise, a balanced dataset from both base and novel classes is used.

2.3.3.2 Single-Branch Meta Learning

Single-branch approaches are more varied and follow a number of themes. The model architectures, unlike in dual-branch meta learning, are similar to a generic object detectors such as Faster R-CNN.

2.4 Transfer Learning

Transfer learning is a learning framework that aims at transferring knowledge between domains so as to reduce the required sample complexity, increase learning speed, and/or improve the overall performance of a model. It tends to follow a two-phase training scheme on single-branch architectures.

In the first phase the model is trained on $\mathcal{D}_{\text{base}}$. In the second, a fraction of parameters is frozen, layers are replaced, extra layers added and/or learning rate is reduced, and the model is fine-tuned using a balanced combination of $\mathcal{D}_{\text{base}}$ and $\mathcal{D}_{\text{novel}}$ (or just $\mathcal{D}_{\text{novel}}$).

Pre-training models with ImageNet (for image classification) (Deng et al., 2009) or MS COCO (for object detection) (Lin et al., 2014) have become the standard procedure for most computer vision tasks, often regardless of the target task or the size of the dataset available. The main benefit of this paradigm is faster convergence (and, by extension, reduced computational cost) since, as He et al. (2019) showed, transfer learning does not produce higher accuracy if enough data is present and models are allowed to train for more epochs.

As Neyshabur et al. (2020) argued, transfer learning guides a model to a flat basin in the loss landscape, making optimisation smoother and quicker; random initialisation, on the other hand, is more chaotic, and a model during training may be forced to overcome performance barriers. Thus, one can consider the transfered weights as a form of regularisation of the model.

It is important to remember that transfer learning can be a double-edged sword: the model locked in the flat basin may be incapable of finding lower minima outside of it. In fact, models fine-tuned on pre-trained weights tend to have similar weights (i.e., they tend to find the same minima) (Neyshabur et al., 2020). This implies that if the target domain is significantly different from the source domain (large domain shift), the transfer will be "negative" unless there is enough target data to overcome "bad" initialisation.

An important insight into the theoretical workings of neural networks that came from the area of transfer learning is that features learned through the lower layers are more generic, whereas the high layers tend to learn more task-specific features (Yosinski et al., 2014). This observation rationalises the typical transfer learning methodology where only the last layer of a detector is fine-tuned on a novel task.

2.4.1 Transfor Learning for FSOD

In FSOD, transfer learning is the go-to method despite the huge amount of effort put into many, much more sophisticated Meta Learning techniques (Wang, Huang, Darrell, Gonzalez and Yu, 2020). The methodology is no different from that for image classification tasks, save for the necessity of fine-tuning the bounding box regressor aside from the classifier.

The majority of the ideas used in the area of FSOD that will be considered were first introduced in few-shot image classification, most importantly, by Gidaris and Komodakis (2018) and Qi et al. (2018). But, as Wang, Huang, Darrell, Gonzalez and Yu (2020) showed these methods work well in object detection too.

2.4.2 TFA & Cosine Similarity

Wang, Huang, Darrell, Gonzalez and Yu (2020) presented the two-stage fine-tuning approach (TFA) for FSOD. In the first stage of fine-tuning a model, it is trained on the data-abundant base dataset. In the second stage, the model is trained on a balanced dataset of K examples from all classes in base and novel datasets. To adapt the model to the new classes, the last dense layers of the bounding box regressor and classifier were extended, and the novel weight connections were randomly initialised. During the fine-tuning stage, the learning rate was reduced by a factor of 20 and all but the last layers of the regressor and classifier were frozen.

The authors have also confirmed that using a classifier based on cosine similarity with instance-level feature normalisation (l_2 normalisation) in the fine-tuning stage reduces the intra-class variance, resulting in higher accuracy for the novel classes and a smaller drop in the accuracy for the base classes compared to the standard dot product classifier.

Since the classifier, in the fine-tuning stage, holds an amalgamation of the base and novel weights, which will be susceptible to different input features (and in some implementations may have different learning rates), the weights and the classification scores may have drastically different values if the standard dot product operation is used. This can hinder the model's learning. Hence, the intra-class variance was suggested by Chen et al. (2019) to be one of the key hurdles in FSL, as proved by the efficacy of cosine similarity classifier.

The cosine similarity classifier comes from a convergence of metric learning and softmax activation (Chen et al., 2019). The weight vectors can be thought of as learned prototypes for each class; the classification label is proposed then based on the distance of the embedded features from these class prototypes. The feature extractor — if trained from scratch — is forced to learn the feature embedding

that forms compact category-wise clusters mimicking the behaviour of a typical metric learner (Gidaris and Komodakis, 2018).

2.4.3 Weight Initialisation

To further improve model training, it is important to initialise the new regressor and classifier layers well. For this purpose, the weight imprinting method for few-shot image classification was proposed by Qi et al. (2018) (also called support-based initialisation by Dhillon et al. (2019)). In this method, normalised logits produced by the backbone are taken as the initial weight values of the classifier layer. This is done to maximise cosine similarity and avoid the computational cost of training a proxy model or applying a meta learner, as in Gidaris and Komodakis (2018).

When $K > 1$, the weights are set to the mean of embeddings of all the images in the class. In practice, for $K = 1$, additional samples can be generated by augmenting the image; however, Qi et al. (2018) have found that it does not produce significant improvements, most likely because the embedding extractor (i.e., the backbone) has already learned to account for such transformations during the pre-training phase. As noted by the authors, the averaging approach may potentially have a significant drawback when a novel class has a multimodal distribution. However, they also suggest that when the embedding extractor is pre-trained on the base classes, the lower layers might learn to transform the samples into a standard distribution for the final layers. Thus, the embedding space should be normally distributed, and the average will reduce the noise present in the samples.

Weight imprinting can not be trivially applied to object detection, as a single image may contain multiple objects from different classes. Moreover, there is no guarantee that RPN will produce appropriate region proposals for the classifier to produce logits of the actual region of interest.

Wang, Huang, Darrell, Gonzalez and Yu (2020) used a similar approach based on weight imprinting principles that work with object detectors. Namely, they used weights of a detector fine-tuned on the novel dataset alone. Obviously, such a method is more computationally expensive than the weight imprinting, but it

may be worth it in some cases. In comparison with random initialisation, Wang, Huang, Darrell, Gonzalez and Yu (2020) found that it was ineffective in their tests on PASCAL VOC (Everingham, 2007) but did improve accuracy on MS COCO. The authors hypothesised that the benefit on the COCO dataset was due to the larger number of classes and general higher complexity of the dataset.

2.4.4 Adapting Model to Fine-Tuning

As mentioned above, TFA replaces the last layers of the bounding box regressor and classifier with new dense layers with reused weight connections corresponding to the base classes. This is not the only way of adapting the model for the fine-tuning stage.

Dhillon et al. (2019), in their study of few-shot image classification, appended a new dense layer on top of the model. This was justified by the findings of Frosst et al. (2019). In a trained backbone, the outputs of the layers are highly entangled and tend to identify class-independent similarity structures; whereas the logits of the final layer are well clustered. This uncoupling of logits, according to the authors, as corroborated by their empirical results in the few-shot image classification task, makes the logits a better input for a final classifier than the embedded features.

3

Methodology

3.1 Model & Datasets

The primary intention of this project was to reproduce in detail the methodology taken by Wang, Huang, Darrell, Gonzalez and Yu (2020); however, due to a different Machine Learning development framework being used, this proved to be troublesome, and a different approach had to be taken. Moreover, the authors took the base and novel classes from the same dataset, which makes the few-shot problem unrealistic and likely makes it much easier to achieve high scores due to the lack of domain shift between the datasets.

In the end, the model used was Faster R-CNN (Ren et al., 2015) with ResNet-50 with Feature Pyramid Network (FPN) (Lin et al., 2017) as the backbone. The base dataset on which the model was pre-trained was MS COCO (Lin et al., 2014) since it is one of the largest and most popular vision datasets.

For the novel dataset, CPPE-5 (Dagli and Shaikh, 2021) was chosen. It con-

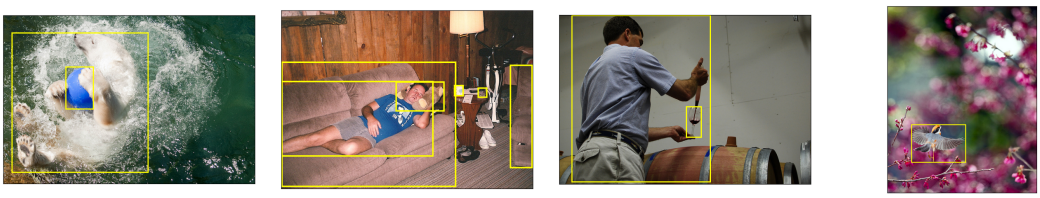

Figure 3.1: Examples of images with bounding boxes indicated from the MS COCO dataset (Lin et al., 2014).

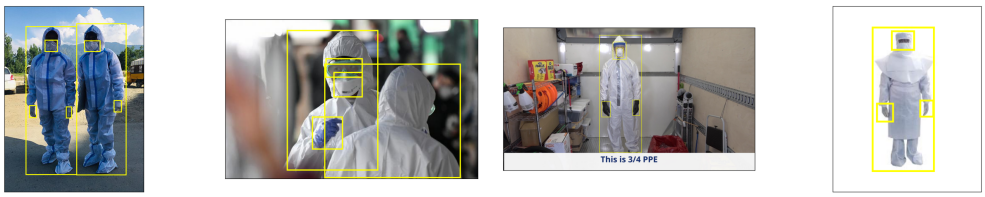

Figure 3.2: Examples of images with bounding boxes indicated from the CPPE-5 dataset (Dagli and Shaikh, 2021).

tains images of people wearing 5 classes of medical personal protective equipment in real life scenarios with $\sim$ 4.6 bounding boxes per image on average. This dataset contains pictures mostly from real-life with objects of relatively medium to large scale, which should make the COCO dataset a reasonable choice for pre-training. At the same time, the categories are wholly different, which makes the task challenging and ensures that no semantic label overlap will occur. This setup should be more representative of actual use cases of FSOD: the datasets are not too dissimilar to produce negative transfer but neither too similar to make the task easy; also, the base dataset is more generic, while the novel is more specialised as is typically the case when using transfer learning.

Few-shot tasks were artificially generated by randomly selecting $K \in \{1, 2, 3\}$ samples from each class of the novel dataset. This was repeated for 3 random seeds in order to account for the inherently large levels of noise in such small datasets. Ideally, much larger number of seeds should have been used, but due to the time limitations, it was not possible.

The model was fine-tuned on a balanced dataset of both base and novel classes using SGD optimiser. The model was fine-tuned for $2,000$ epochs with a batch size 16, a learning rate 0.002, a weight decay 0.0001, and a momentum 0.9. In preliminary experiments, this combination of hyper-parameters provided a good middle-ground between time efficiency and model precision. Fine-tuning for more epochs could provide better precision at the novel dataset, which usually comes at the cost of base classes' performance.

Extending the domain of a model will inevitably lead to worse performance on the original dataset. Combined with the large disparity between the number of classes of the CPPE-5 and COCO datasets, it makes the overall mAP a bad

metric for deciding the training duration, as it will favour the performance on COCO. It is important then to track both novel classes mAP and base classes mAP. One must also weight the importance of the old and novel information and make the decision based on the task at hand.

All experiments were run on a single RTX4090 GPU.

3.2 Experiments

3.2.1 Weight Initialisation

Since new weight connections are added to the model, it creates a question of how to best initialise the novel weights. In few-shot image classification, the most optimal approach is weight imprinting (Qi et al., 2018), but such a method is impossible to implement for object detection. Thus, Wang, Huang, Darrell, Gonzalez and Yu (2020) approach was used. Namely, a proxy model was trained on only the novel dataset for $1,000$ epochs (without any data augmentation) to obtain values for the novel weights.

To validate the efficacy of the initialisation method, it was compared to random weights generated using Xavier uniform initialiser (Glorot and Bengio, 2010). For the comparison purpose, a model with cosine similarity predictor was used just like in Wang, Huang, Darrell, Gonzalez and Yu (2020).

3.2.2 Cosine Similarity & Adapting the Model

Two methods of adapting the model to fine-tuning were compared, namely, that of Wang, Huang, Darrell, Gonzalez and Yu (2020) (replacing the last layers) and that of Dhillon et al. (2019) (appending new layers). Both papers closely follow the developments in the metric learning area and implement cosine similarity classifier. Versions with the standard dot product predictors were also implemented and used for benchmarking.

In preliminary experiments, it was found that when appending a layer, the model performs much better when no activation is added right before it. Thus, none was used.

Figure 3.3: Examples of mosaic augmentation technique implementation on the COCO (top row) and CPPE-5 (bottom row) datasets, with bounding boxes indicated.

3.2.3 Data Augmentation

Two augmentation schemes were compared to the baseline. The first scheme consisted of mosaic data augmentation with a probability of occurring of 50%. Mosaic augmentation had to be implemented from scratch as no compatible version existed.

There were a couple of choices that had to be taken in the implementation of the mosaic augmentation that were not addressed by Bochkovskiy et al. (2020) related to the construction of the initial grid. The wide range of possible aspect ratios of the images makes the construction of the mosaic non-trivial. First, to avoid constant cropping of images and reducing the amount of information each provides, if an image's ratio of larger to smaller side was less than 1.2, the image

was padded to the outside of the grid to match their grid square (i.e., images in the top-left corner were padded to the top and left, those in the top-right were padded to the top and right, etc.). Otherwise, the image was randomly cropped to square.

Second, the size of the initial grid was decided based on a minimal area of any image visible in the final crop. For example, if the desired fraction of the area was 0.25 and the final mosaic size was 1000×1000, a grid of 1500×1500 was first constructed and then cropped to 1000×1000. This would result in all images contributing at least an area of 375×375 of the final image. Cropping was not done in a purely random manner; rather, a random bounding box was selected, and then the crop area was decided such that the bounding box would definitely be present after cropping. This was done to ensure at least one bounding box in the final image.

The second augmentation scheme included an additional independent augmentation policy composed of colour operations such as `Equalize`, `Polarize`, `Invert`, etc. The policy used was an adapted version of the ImageNet Autoaugment policy with sub-policies containing geometric operations (`Rotate` and `ShearX`) removed as their implementation is troublesome with bounding boxes.

4

Results

4.1 Weight Initialisation

Novel weight initialisation using fine-tuned weights was compared to random initialisation using a Xavier uniform initialiser (see Table 4.1). A model with the cosine similarity classifier was used for this purpose, as in Wang, Huang, Darrell, Gonzalez and Yu (2020). There was little to no change in the performance for the base classes, which was expected as only novel connections were changed.

Weight initialisation made little difference for the novel classes. Due to large variances, partially caused by the small sample size, it is difficult to indisputably establish whether the novel method has the potential to improve performance. The biggest improvement can be seen for the AP75 metric, especially for the 3-shot task. This indicates that the bounding box regressor benefited the most. Using data augmentation for weight generation could produce a greater contrast between the methods.

Nevertheless, weight initialisation using fine-tuned weights bears a relatively small one-off computational cost that could improve model training; however, if any benefits can be materialised, they are moderate at best, especially if the cosine similarity classifier is used, which, thanks to its instance-level normalisation, reduces the discrepancy between the two sets of weights allowing for smoother fine-tuning.

This result matches observations by Wang, Huang, Darrell, Gonzalez and Yu (2020) where simple random initialisation could outperform the novel method on

the PASCAL VOC dataset but was slightly worse on the COCO dataset.

Method/Shot	Novel		
	1 shot	2 shot	3 shot
	mAP		
Random	3.9 ± 1.5	$\mathbf{4.7 \pm 1.1}$	$\mathbf{6.8 \pm 1.3}$
Imprinting	$\mathbf{4.1 \pm 1.8}$	4.1 ± 0.7	6.6 ± 1.2
	AP50		
Random	$\mathbf{10.7 \pm 3.2}$	$\mathbf{13.4 \pm 2.3}$	$\mathbf{21.6 \pm 2.4}$
Imprinting	10.5 ± 3.6	12.4 ± 1.5	21.1 ± 2.1
	AP75		
Random	1.5 ± 1.3	$\mathbf{1.5 \pm 0.9}$	1.3 ± 0.2
Imprinting	$\mathbf{1.7 \pm 1.2}$	1.4 ± 1.0	$\mathbf{2.2 \pm 1.4}$

Table 4.1: Comparison of weight initialisation methods when using TFA. Random weight initialisation using Xavier uniform initialiser (`Random`) was compared to weight initialisation using fine-tuned weights (`Imprinting`). Faster-RCNN with cosine similarity classifier was used as the object detection model. Taking into consideration the error range, the fine-tuned weights did not provide a significant improvement, but the effect of data augmentation should be further investigated.

4.2 Cosine Similarity and Fine-Tuning Method

The cosine similarity classifier was compared to the standard dot product classifier using two fine-tuning methods: replace and append approaches (see Table 4.2). In the case of the replace approach, there was a significant improvement in the performance — much more consequential for lower shot tasks. On average, cosine similarity gave 44%, 9%, 15% improvement in mAP metric for 1, 2, and 3 shot tasks, respectively. Surprisingly, neither AP50 nor AP75 metrics have seen such drastic changes. For 3 shot AP75, there was even a degradation in performance, but this metric also experienced a large variance, so it's impossible to provide a conclusive opinion.

A clear benefit of the cosine similarity for both fine-tuning methods is lower degradation of performance on the base classes. The difference in most cases is ~ 2 percentage points.

In the case of the append approach, the results were opposite: the standard dot product predictor achieved 5%, 13%, 2% higher mAP for 1, 2, and 3 shot

tasks, respectively.

Method/Shot	Novel			Base		
	1 shot	2 shot	3 shot	1 shot	2 shot	3 shot
	mAP					
Dot-product (replace)	2.7 ± 1.0	4.3 ± 0.2	5.9 ± 1.0	25.2 ± 0.9	26.1 ± 0.4	26.2 ± 0.3
Cosine (replace)	3.9 ± 1.5	4.7 ± 1.1	6.8 ± 1.3	$\mathbf{27.1 \pm 0.4}$	$\mathbf{27.7 \pm 0.1}$	$\mathbf{27.9 \pm 0.1}$
Dot-product (append)	$\mathbf{6.1 \pm 1.4}$	$\mathbf{7.1 \pm 0.5}$	$\mathbf{9.7 \pm 0.7}$	19.5 ± 1.4	24.4 ± 0.2	26.1 ± 0.5
Cosine (append)	5.8 ± 2.5	6.3 ± 1.0	9.5 ± 0.7	21.1 ± 1.5	26.0 ± 0.5	$\mathbf{27.9 \pm 0.6}$
	AP50					
Dot-product (replace)	8.7 ± 3.2	13.4 ± 1.8	17.5 ± 2.6	45.0 ± 1.2	46.4 ± 0.4	46.5 ± 0.8
Cosine (replace)	10.7 ± 3.2	13.4 ± 2.3	$\mathbf{21.6 \pm 2.4}$	$\mathbf{48.3 \pm 0.3}$	$\mathbf{49.1 \pm 0.1}$	$\mathbf{49.0 \pm 0.4}$
Dot-product (append)	$\mathbf{14.9 \pm 3.7}$	$\mathbf{14.7 \pm 1.7}$	21.4 ± 0.3	33.9 ± 1.9	41.3 ± 0.4	43.9 ± 0.9
Cosine (append)	11.8 ± 4.2	13.7 ± 1.8	19.9 ± 0.9	36.9 ± 2.2	44.4 ± 1.0	47.2 ± 1.1
	AP75					
Dot-product (replace)	0.8 ± 0.6	1.7 ± 1.0	2.4 ± 2.1	25.4 ± 1.4	26.5 ± 0.6	26.6 ± 0.2
Cosine (replace)	1.5 ± 1.3	1.5 ± 0.9	1.3 ± 0.2	$\mathbf{26.9 \pm 1.0}$	$\mathbf{28.2 \pm 0.4}$	28.5 ± 0.2
Dot-product (append)	2.8 ± 1.2	$\mathbf{5.5 \pm 0.6}$	7.5 ± 2.3	20.3 ± 1.6	26.0 ± 0.2	27.9 ± 0.5
Cosine (append)	$\mathbf{4.5 \pm 3.3}$	4.9 ± 1.5	$\mathbf{7.6 \pm 1.0}$	21.6 ± 1.6	27.6 ± 0.7	$\mathbf{29.6 \pm 0.5}$

Table 4.2: Comparison of two fine-tuning methodologies: replace and append, and two types of classifiers: standard dot product classifier and cosine similarity classifier. No data augmentation was used. The highest precision on the novel classes was achieved using the append approach with the standard dot product classifier. Whereas, for the base classes, the best model was one using the replace approach with the cosine similarity.

Unequivocally, the append fine-tuning approach performed much better in all metrics on the novel dataset. When comparing `Cosine (replace)` with `Dot-product (append)`, the latter achieves higher mAP by more than 40%. The greatest benefit is seen in AP75, where the scores increased at least two-fold.

Generally, the higher the performance on the novel classes, the lower it is on the base classes; thus, `Dot-product (append)`, the best method for the novel classes performs worse than all other methods on the base classes metrics. Notably, the difference in the base classes' performance reduces for higher shots.

4.3 Data Augmentation

Data augmentation proved to be crucial for FSL as shown in Tables 4.3 and 4.4.

Universally, the mosaic data augmentation technique significantly improves the performance of the models on the novel classes, but it comes at the price

of degraded precision for the base classes. Moreover, the difference between the two kinds of predictors greatly diminished. In the case of the append fine-tuning method, the cosine similarity classifier becomes a better option as the gap to the dot product classifier shrunk, but still degrades the base class performance less.

The addition of colour augmentation further enhanced the performance of the append method. For the replace method, the results vary: for 1 shot task the precision dropped, for 2 shot it rose, and for 3 shot there was no difference. While it's inconclusive whether colour jittering is always applicable, it underlines the need for careful selection of augmentation policy.

Method/Shot	Novel			Base		
	1 shot	2 shot	3 shot	1 shot	2 shot	3 shot
	mAP					
Dot-product (none)	2.7 ± 1.0	4.3 ± 0.2	5.9 ± 1.0	25.2 ± 0.9	26.1 ± 0.4	26.2 ± 0.3
Dot-product (mos)	5.5 ± 1.2	6.9 ± 0.3	8.8 ± 0.7	22.7 ± 0.5	24.2 ± 0.3	25.1 ± 0.4
Dot-product (mos+clr)	4.4 ± 0.9	7.5 ± 0.7	8.6 ± 0.8	22.2 ± 0.3	23.9 ± 0.1	24.9 ± 0.3
Cosine (none)	3.9 ± 1.5	4.7 ± 1.1	6.8 ± 1.3	$\mathbf{27.1 \pm 0.4}$	$\mathbf{27.7 \pm 0.1}$	$\mathbf{27.9 \pm 0.1}$
Cosine (mos)	$\mathbf{5.8 \pm 2.5}$	6.3 ± 1.7	$\mathbf{9.1 \pm 1.1}$	24.6 ± 0.1	25.7 ± 0.3	26.5 ± 0.2
Cosine (mos+clr)	4.9 ± 1.7	$\mathbf{8.4 \pm 1.4}$	$\mathbf{9.1 \pm 0.7}$	24.5 ± 0.2	25.6 ± 0.2	26.3 ± 0.2
	AP50					
Dot-product (none)	8.7 ± 3.2	13.4 ± 1.8	17.5 ± 2.6	45.0 ± 1.2	46.4 ± 0.4	46.5 ± 0.8
Dot-product (mos)	$\mathbf{15.0 \pm 2.3}$	17.7 ± 1.8	24.5 ± 1.2	41.3 ± 1.3	43.3 ± 0.5	44.5 ± 0.7
Dot-product (mos+clr)	12.0 ± 2.3	$\mathbf{20.6 \pm 2.5}$	24.8 ± 1.1	40.8 ± 0.9	42.7 ± 0.3	44.0 ± 0.6
Cosine (none)	10.7 ± 3.2	13.4 ± 2.3	21.6 ± 2.4	$\mathbf{48.3 \pm 0.3}$	$\mathbf{49.1 \pm 0.1}$	$\mathbf{49.0 \pm 0.4}$
Cosine (mos)	14.5 ± 6.0	18.3 ± 1.5	25.1 ± 0.4	45.0 ± 0.4	46.0 ± 0.4	46.7 ± 0.4
Cosine (mos+clr)	12.8 ± 3.2	20.5 ± 2.5	$\mathbf{25.2 \pm 1.1}$	45.0 ± 0.3	45.7 ± 0.4	46.2 ± 0.3
	AP75					
Dot-product (none)	0.8 ± 0.6	1.7 ± 1.0	2.4 ± 2.1	25.4 ± 1.4	26.5 ± 0.6	26.6 ± 0.2
Dot-product (mos)	2.6 ± 1.5	3.0 ± 1.6	4.0 ± 0.7	22.5 ± 0.4	24.5 ± 0.2	25.6 ± 0.3
Dot-product (mos+clr)	1.9 ± 1.1	3.3 ± 2.0	4.1 ± 1.5	21.9 ± 0.3	24.3 ± 0.2	25.5 ± 0.3
Cosine (none)	1.5 ± 1.3	1.5 ± 0.9	1.3 ± 0.2	$\mathbf{26.9 \pm 1.0}$	$\mathbf{28.2 \pm 0.4}$	$\mathbf{28.5 \pm 0.2}$
Cosine (mos)	$\mathbf{3.8 \pm 2.9}$	3.4 ± 3.3	$\mathbf{4.3 \pm 2.1}$	24.1 ± 0.3	26.0 ± 0.3	27.1 ± 0.2
Cosine (mos+clr)	1.5 ± 1.4	$\mathbf{5.4 \pm 2.8}$	3.4 ± 1.5	24.0 ± 0.5	25.8 ± 0.2	27.1 ± 0.2

Table 4.3: Comparison of data augmentation strategies: no augmentation (`(none)`), mosaic augmentation (`(mos)`), mosaic augmentation and colour jittering (`(mos+col)`) using the replace fine-tuning approach.

Overall, by using the append method for fine-tuning and data augmentation, the improvement over the base case is drastic: 222%, 137%, 92%, for the 1, 2, and 3 shot tasks, respectively.

Method/Shot	Novel			Base		
	1 shot	2 shot	3 shot	1 shot	2 shot	3 shot
	mAP					
Dot-product (none)	6.1 ± 1.4	7.1 ± 0.5	9.7 ± 0.7	19.5 ± 1.4	24.4 ± 0.2	26.1 ± 0.5
Dot-product (mos)	6.7 ± 1.1	8.5 ± 0.4	10.9 ± 0.8	18.0 ± 1.4	22.9 ± 0.4	25.1 ± 0.8
Dot-product (mos+col)	7.0 ± 0.8	$\mathbf{10.3 \pm 1.1}$	$\mathbf{11.5 \pm 0.6}$	17.3 ± 1.2	22.7 ± 0.2	24.9 ± 0.6
Cosine (none)	5.8 ± 2.5	6.3 ± 1.0	9.5 ± 0.7	21.1 ± 1.5	$\mathbf{26.0 \pm 0.5}$	$\mathbf{27.9 \pm 0.6}$
Cosine (mos)	7.6 ± 1.2	8.2 ± 0.5	10.8 ± 0.1	21.1 ± 1.2	25.8 ± 0.4	27.4 ± 0.8
Cosine (mos+col)	$\mathbf{8.7 \pm 3.6}$	10.2 ± 1.0	11.3 ± 0.6	$\mathbf{22.9 \pm 3.5}$	24.9 ± 0.4	26.9 ± 0.7
	AP50					
Dot-product (none)	14.9 ± 3.7	14.7 ± 1.7	21.4 ± 0.3	33.9 ± 1.9	41.3 ± 0.4	43.9 ± 0.9
Dot-product (mos)	15.7 ± 1.4	17.3 ± 1.8	22.9 ± 0.6	31.6 ± 2.2	39.0 ± 0.7	42.3 ± 1.2
Dot-product (mos+col)	15.4 ± 1.5	$\mathbf{21.1 \pm 2.8}$	$\mathbf{23.7 \pm 2.4}$	30.1 ± 1.5	38.6 ± 0.7	42.1 ± 1.0
Cosine (none)	11.8 ± 4.2	13.7 ± 1.8	19.9 ± 0.9	36.9 ± 2.2	$\mathbf{44.4 \pm 1.0}$	$\mathbf{47.2 \pm 1.1}$
Cosine (mos)	16.7 ± 2.3	16.4 ± 0.7	22.7 ± 0.7	37.0 ± 1.8	44.2 ± 0.9	46.7 ± 1.2
Cosine (mos+col)	$\mathbf{18.4 \pm 7.3}$	19.9 ± 2.4	23.0 ± 1.0	$\mathbf{39.7 \pm 5.8}$	42.8 ± 0.9	45.9 ± 1.1
	AP75					
Dot-product (none)	2.8 ± 1.2	5.5 ± 0.6	7.5 ± 2.3	20.3 ± 1.6	26.0 ± 0.2	27.9 ± 0.5
Dot-product (mos)	4.4 ± 1.4	6.5 ± 2.2	7.9 ± 1.2	18.4 ± 1.8	24.3 ± 0.5	26.6 ± 0.8
Dot-product (mos+col)	5.3 ± 1.0	$\mathbf{8.6 \pm 0.5}$	$\mathbf{9.2 \pm 1.2}$	18.1 ± 1.6	24.1 ± 0.2	26.7 ± 0.5
Cosine (none)	4.5 ± 3.3	4.9 ± 1.5	7.6 ± 1.0	21.6 ± 1.6	$\mathbf{27.6 \pm 0.7}$	$\mathbf{29.6 \pm 0.5}$
Cosine (mos)	5.0 ± 1.5	6.8 ± 1.7	8.5 ± 1.3	21.8 ± 1.6	27.0 ± 0.2	28.8 ± 0.7
Cosine (mos+col)	$\mathbf{7.1 \pm 3.2}$	8.4 ± 1.1	9.1 ± 0.8	$\mathbf{24.2 \pm 4.1}$	26.3 ± 0.3	28.3 ± 0.7

Table 4.4: Comparison of data augmentation strategies: no augmentation (`(none)`), mosaic augmentation (`(mos)`), mosaic augmentation and colour jittering (`(mos+col)`) using the append fine-tuning approach.

4.4 Error Analysis

Errors of models trained in previous sections were analysed and compared to better understand the qualities of the methods used. The analysis was based on work by Hoiem et al. (2012). Two main categories of errors made by object detectors are false positives and false negatives.

4.4.1 False Positives

False positives occur when a model produces a prediction that does not correspond to the target. Following Hoiem et al. (2012), these errors were further subdivided into four sub-categories based on the intersection-over-union (IoU) with the ground truth and the class label of the model's prediction:

- classification error occur when the model outputs correct bounding box

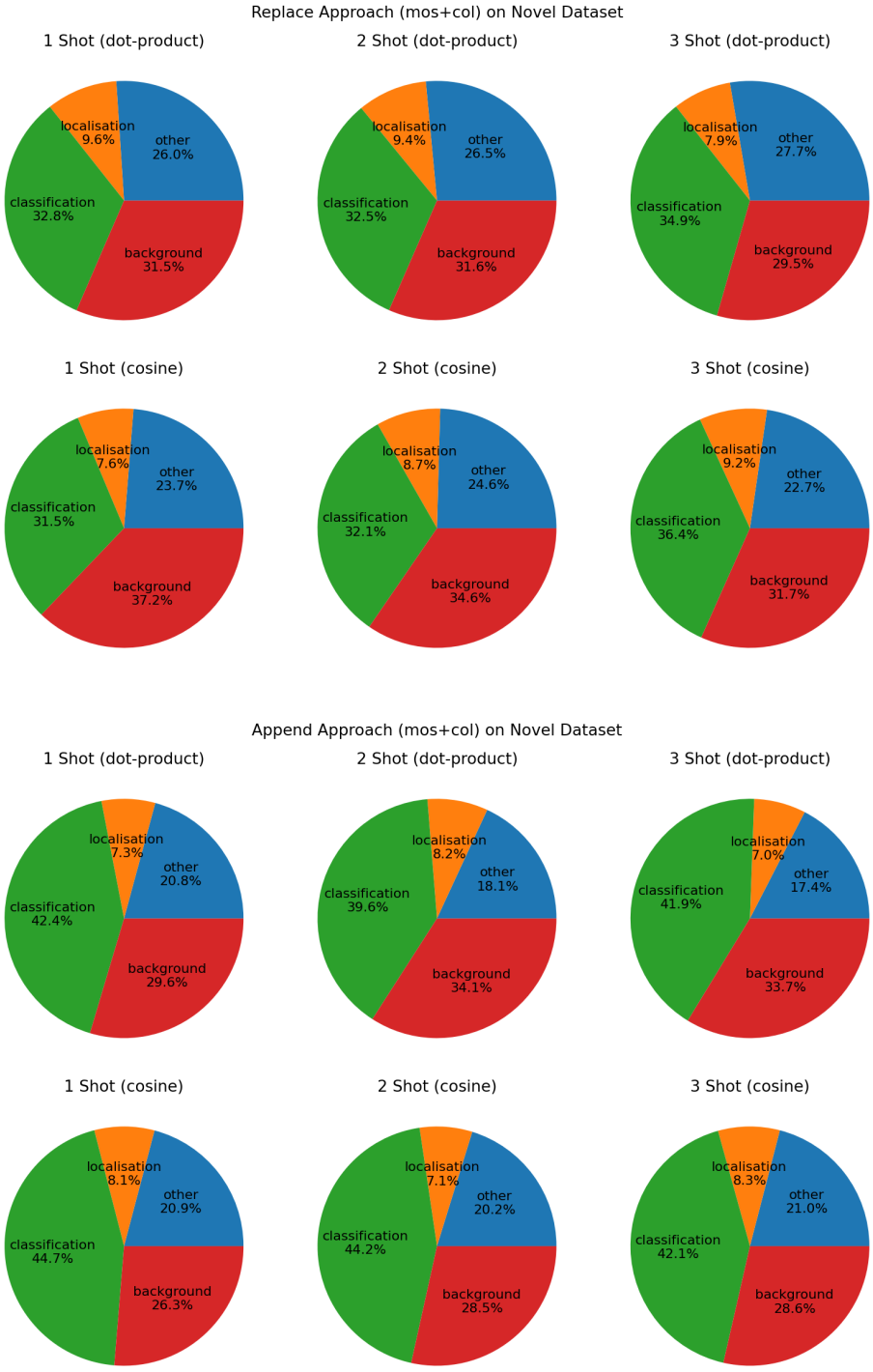

Figure 4.1: False positive rates divided into four categories: classification, localisation, confusion with other, and background on the novel dataset for the replace (top two rows) and append (bottom two rows) fine-tuning approaches. Both dot-product and cosine similarity classifiers are shown. Mosaic and colour jittering were used as data augmentation.

(≥ 0.5), but incorrect label;

- localisation errors occur when correctly labelled bounding box has IoU between 0.1 and 0.5;
- confusion with other errors occur when prediction has incorrect label and IoU between 0.1 and 0.5;
- background errors occur when IoU with any target is less than 0.1.

The most common types of false positives were classification and background errors, with an approximate rate of $30 - 45\%$ and $25 - 35\%$ of all false positive errors, respectively. The next most common category was confusion with other responsible for roughly $20 - 30\%$ of errors; and, lastly, localisation errors, with a single-digit percentage.

Regarding the classification errors, novel classes were uniquely misclassified as other novel classes, and similarly, the base classes were only misclassified as other base classes.

Weight initialisation did not change the rate of different false positive errors.

The addition of the cosine similarly classifier produced little change on the base classes; however, some trends can be identified on the novel classes (see Figure 4.1). Background errors, on average, increased by 3.7 percentage points when using the replace approach, and decreased by 4.7 percentage points when using the append method. The other three categories changed in the opposite direction to the background category.

When comparing the replace and append approaches, it is clear that the append method has a higher rate of classification errors and lower rates for all the other categories.

Data augmentation significantly increased the fraction of false positives on the background. It also reduced the fraction of the classification errors, but most drastically, it decreased the confusion with other object categories (see Figure 4.2). This change can be rationalised by the fact that mosaic augmentation changes the surroundings of objects, forcing the model to better learn the unique characteristics of objects themselves rather than relying on the environment they are present in, thus lowering confusion with other classes.

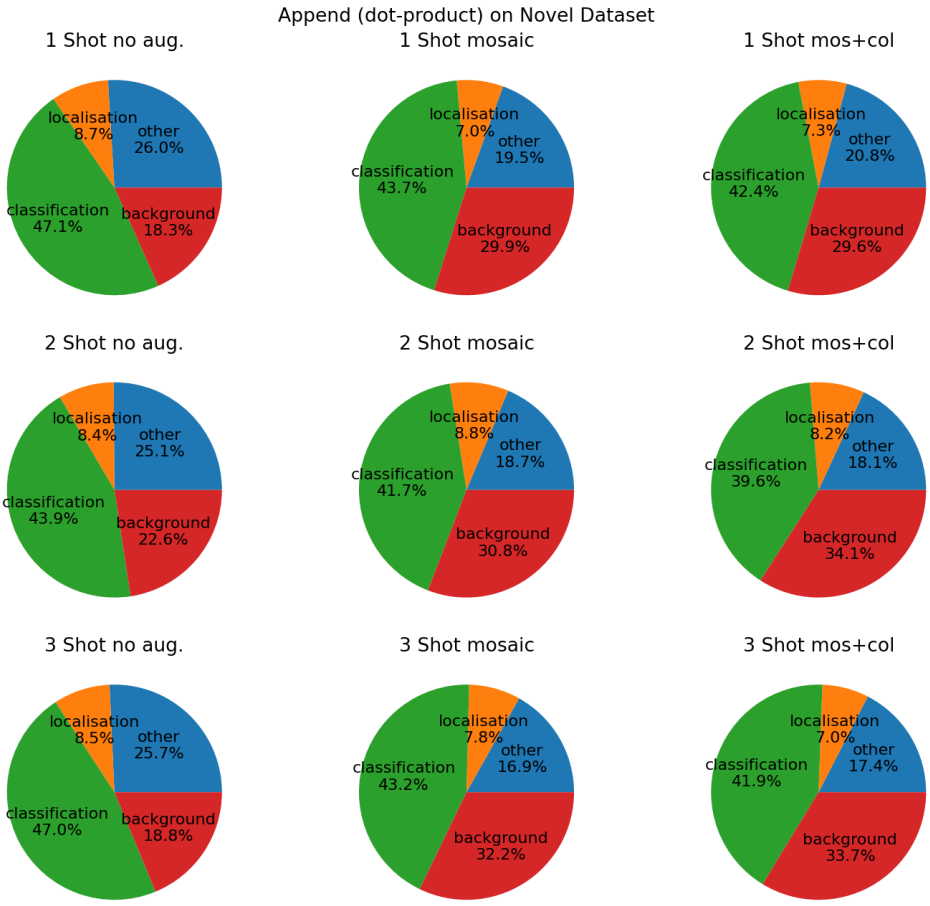

Figure 4.2: False positive rates divided into four categories: classification, localisation, confusion with other, and background on the novel dataset for the append fine-tuning approach with the standard dot-product classifiers.

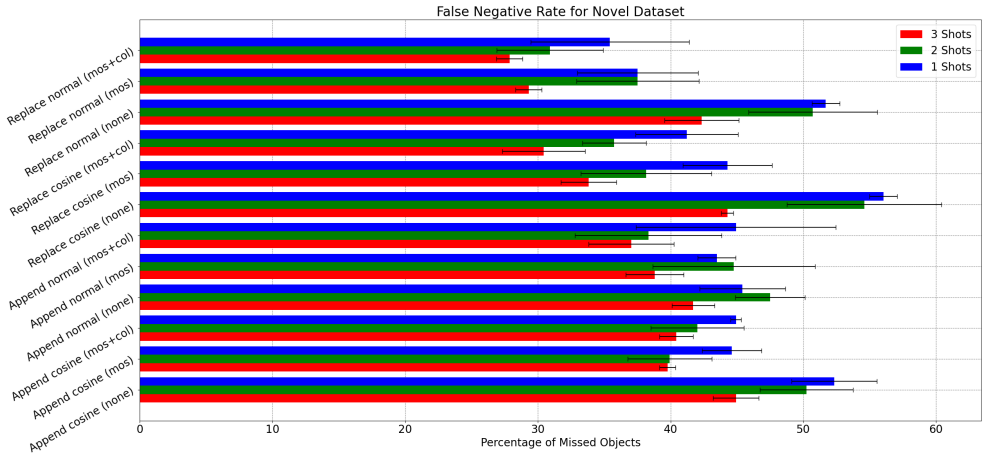

Figure 4.3: The percentage of objects belonging to novel classes missed by the various fine-tuned models with standard deviations shown.

The higher fraction of background errors does not necessarily imply worse background detection since the novel mAP increased after the addition of data augmentation, as was shown in the sections above. To better understand the source of the change in the error rates, absolute quantities should be compared, preferably on a dataset with a larger test set than CPPE-5, to ensure numerous samples for each class.

4.4.2 False Negatives

False negative errors occur when the model fails to detect target objects in the image or assigns low confidence to an object. To analyse this kind of error, the rate of missed objects was calculated. To be exact, missed objects were defined as those without corresponding predicted bounding box or if the IoU of a predicted bounding box was lesser than 0.1 with any ground truth. The percentage of missed objects for each method is presented in Figures 4.3 and 4.4.

Unsurprisingly, false negative errors are more frequent on the novel classes. The append fine-tuning approach has a lower rate of false negative errors than the replace approach when no augmentation is used but higher when data augmentation is applied.

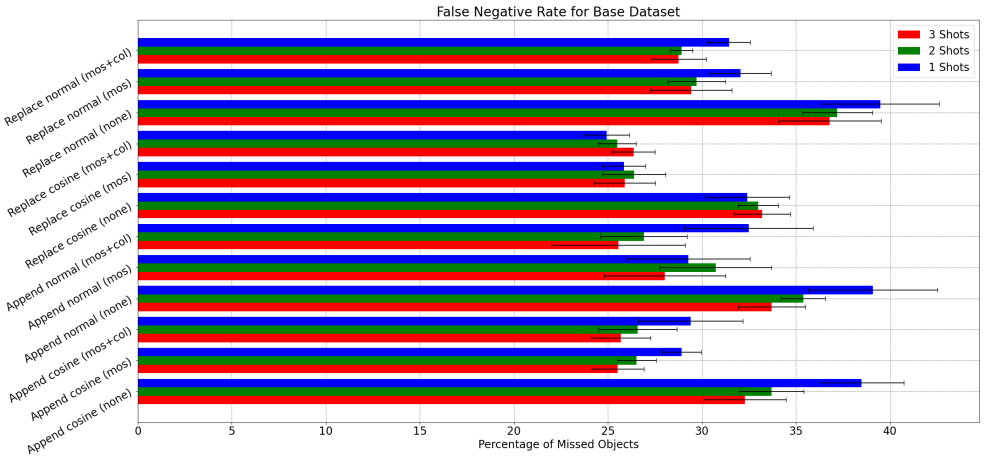

Figure 4.4: The percentage of objects belonging to base classes missed by the various fine-tuned models with standard deviations shown.

While it is understandable from the previous results that the cosine similarity classifier reduces the rate of false negatives on the base classes, it is interesting that novel classes see a rise in fraction of objects missed as compared to the dot-product classifier.

Generally, data augmentation reduces this kind of error. On average, false negatives dropped by 28% (15p.p.) for the novel classes and 23% (8p.p.) for the base classes.

5

Conclusions & Discussion

5.1 Summary

An adapted version of the methodology presented in Wang, Huang, Darrell, Gonzalez and Yu (2020) was successfully implemented and used to carry out multiple experiments on low-shot tasks. Two potential improvements the authors presented (weight initialisation and cosine similarity classifier) were further investigated, and their results were validated.

Additionally, two extra improvement areas were identified and investigated, namely, data augmentation and fine-tuning model adaptation. Both of these have shown great potential for enhancing few-shot models' performance, especially on very low shot tasks, providing, in total, two- to three-fold improvement on the mAP metric.

5.2 Conclusions

It was found that weight initialisation using fine-tuned weights, at best, provides little to no benefit when the cosine similarity classifier is used as well. When using TFA, the cosine similarity classifier was found to improve the novel class performance and degrade less the base class performance compared to the dot product classifier, as stated by Wang, Huang, Darrell, Gonzalez and Yu (2020). The most significant improvements were found to be in the strictest AP75 metric.

However, unlike Wang, Huang, Darrell, Gonzalez and Yu (2020), it was found that appending a new layer on top of the model as opposed to replacing the last layers greatly improved the model's performance. It was also found that this approach does not combine well with the cosine similarity classifier when no data augmentation is used.

Data augmentation policy selection proved to be crucial in the few-shot tasks. Mosaic data augmentation greatly enhanced the model's performance on all metrics, especially on 1 shot task.

Even though learned augmentation policies are said to be transferable to other datasets, it may not always be so. It was found that using colour jittering augmentation based on ImageNet-learned AutoAugment startegy did not clearly improve performance when added on top of the mosaic augmentation using the replace fine-tuning method. This may be explained by the low number of random seeds tested. This was not the case for the append approach, where the benefit was obvious. This observation showcases the need for careful data augmentation strategy selection on low-shot tasks. With only one or a few samples, seemingly benign augmentation techniques may add damaging noise to data.

5.3 Future Work

Due to limited resources, the experiments were repeated for a small number of seeds (3), causing the large variance in the results. It is difficult then to draw decisive conclusions, nevertheless, a general direction for further research can be outlined. Ideally, the above experiments should be repeated for a meaningful number of seeds for a wider number of shots to provide clear guidance on the optimal fine-tuning strategy.

When appending a new last layer, the final classifier depends on the well-clustered logits produced by the backbone and the "old" classifier trained on the base dataset. This method, thus, may be more sensitive to a domain shift than the typical replace approach, which only depends on generic class-agnostic features. For future research, it is proposed to test the methods on several novel datasets with varying similarity to the base dataset to investigate the method's resistance to the domain shift.

Although the TFA-like approaches presuppose the similarity of the datasets as they are based on transfer learning, knowledge of methods' sensitivity to domain shift may still be crucial for real-world applications, as it may not always be possible to predict final data distribution from the few samples available. Hence, in some applications, it may be difficult or impossible to know with certainty a priori which base dataset to use.

For the same reason, future work should investigate procedures for establishing optimal augmentation strategies from the limited data. Generally, strategies good for data-abundant dataset will be appropriate for the data-scarce dataset; however, to make the training more robust, procedure improvements would be suitable. For example, the data augmentation strategy need not be set in stone and could be guided by the model's training performance (Huang et al., 2024).

References

Bochkovskiy, A., Wang, C.-Y. and Liao, H.-Y. M. (2020), 'Yolov4: Optimal speed and accuracy of object detection', *arXiv preprint arXiv:2004.10934* . 8, 20

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C. F. and Huang, J.-B. (2019), 'A closer look at few-shot classification', *arXiv preprint arXiv:1904.04232* . 13

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V. and Le, Q. V. (2019), Autoaugment: Learning augmentation strategies from data, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition', pp. 113–123. 7

Cubuk, E. D., Zoph, B., Shlens, J. and Le, Q. V. (2020), Randaugment: Practical automated data augmentation with a reduced search space, *in* 'Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops', pp. 702–703. 8

Dagli, R. and Shaikh, A. M. (2021), 'Cppe-5: Medical personal protective equipment dataset'. vii, 17, 18

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. and Fei-Fei, L. (2009), Imagenet: A large-scale hierarchical image database, *in* '2009 IEEE conference on computer vision and pattern recognition', Ieee, pp. 248–255. 12

DeVries, T. and Taylor, G. W. (2017), 'Improved regularization of convolutional neural networks with cutout', *arXiv preprint arXiv:1708.04552* . 8

Dhillon, G. S., Chaudhari, P., Ravichandran, A. and Soatto, S. (2019), 'A baseline for few-shot image classification', *arXiv preprint arXiv:1909.02729* . 14, 15, 19

Everingham, M. (2007), 'The pascal visual object classes challenge,(voc2007) results', *http://pascallin. ecs. soton. ac. uk/challenges/VOC/voc2007/index. html.* . 15

REFERENCES

Fink, M. (2004), 'Object classification from a single example utilizing class relevance metrics', *Advances in neural information processing systems* **17**. 6

Finn, C., Abbeel, P. and Levine, S. (2017), Model-agnostic meta-learning for fast adaptation of deep networks, *in* 'International conference on machine learning', PMLR, pp. 1126–1135. 9

Fischler, M. and Elschlager, R. (1973), 'The representation and matching of pictorial structures', *IEEE Transactions on Computers* **C-22**(1), 67–92. 1

Frosst, N., Papernot, N. and Hinton, G. (2019), Analyzing and improving representations with the soft nearest neighbor loss, *in* 'International conference on machine learning', PMLR, pp. 2012–2020. 15

Gidaris, S. and Komodakis, N. (2018), Dynamic few-shot visual learning without forgetting, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 4367–4375. 13, 14

Girshick, R. (2015), Fast r-cnn, *in* 'Proceedings of the IEEE international conference on computer vision', pp. 1440–1448. 1, 5

Girshick, R., Donahue, J., Darrell, T. and Malik, J. (2014), Rich feature hierarchies for accurate object detection and semantic segmentation, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 580–587. 5

Glorot, X. and Bengio, Y. (2010), Understanding the difficulty of training deep feedforward neural networks, *in* 'Proceedings of the thirteenth international conference on artificial intelligence and statistics', JMLR Workshop and Conference Proceedings, pp. 249–256. 19

Hataya, R., Zdenek, J., Yoshizoe, K. and Nakayama, H. (2020), Faster autoaugment: Learning augmentation strategies using backpropagation, *in* 'Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16', Springer, pp. 1–16. 7

He, K., Girshick, R. and Dollár, P. (2019), Rethinking imagenet pre-training, *in* 'Proceedings of the IEEE/CVF International Conference on Computer Vision', pp. 4918–4927. 12

Hoiem, D., Chodpathumwan, Y. and Dai, Q. (2012), Diagnosing error in object detectors, *in* 'European conference on computer vision', Springer, pp. 340–353. 27

Huang, T., Liu, J., You, S. and Xu, C. (2024), 'Active generation for image classification', *arXiv preprint arXiv:2403.06517* . 35

Ioffe, S. and Szegedy, C. (2015), Batch normalization: Accelerating deep network training by reducing internal covariate shift, *in* 'International conference on machine learning', pmlr, pp. 448–456. 8

Kaya, M. and Bilge, H. Ş. (2019), 'Deep metric learning: A survey', *Symmetry* **11**(9), 1066. 9

Köhler, M., Eisenbach, M. and Gross, H.-M. (2023), 'Few-shot object detection: a comprehensive survey', *IEEE Transactions on Neural Networks and Learning Systems* . 10

Krizhevsky, A., Sutskever, I. and Hinton, G. E. (2012), 'Imagenet classification with deep convolutional neural networks', *Advances in neural information processing systems* **25**. 7

LeCun, Y., Bengio, Y. and Hinton, G. (2015), 'Deep learning', *nature* **521**(7553), 436–444. 1

Li, F.-F., Fergus, R. and Perona, P. (2006), 'One-shot learning of object categories', *IEEE transactions on pattern analysis and machine intelligence* **28**(4), 594–611. 6

Lim, S., Kim, I., Kim, T., Kim, C. and Kim, S. (2019), 'Fast autoaugment', *Advances in Neural Information Processing Systems* **32**. 7

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. (2017), Feature pyramid networks for object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2117–2125. 17

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C. L. (2014), Microsoft coco: Common objects in context, *in* 'Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13', Springer, pp. 740–755. vii, 12, 17

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X. and Pietikäinen, M. (2019), 'Deep learning for generic object detection: A survey', *International Journal of Computer Vision* **128**(2), 261–318. 1

REFERENCES

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C. (2016), Ssd: Single shot multibox detector, *in* 'Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14', Springer, pp. 21–37. 1

Mannocci, L., Villon, S., Chaumont, M., Guellati, N., Mouquet, N., Iovan, C., Vigliola, L. and Mouillot, D. (2022), 'Leveraging social media and deep learning to detect rare megafauna in video surveys', *Conservation Biology* **36**(1), e13798. 1

Neyshabur, B., Sedghi, H. and Zhang, C. (2020), 'What is being transferred in transfer learning?', *Advances in neural information processing systems* **33**, 512–523. 12

Qi, H., Brown, M. and Lowe, D. G. (2018), Low-shot learning with imprinted weights, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 5822–5830. 13, 14, 19

Rawat, W. and Wang, Z. (2017), 'Deep convolutional neural networks for image classification: A comprehensive review', *Neural computation* **29**(9), 2352–2449. 5

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016), You only look once: Unified, real-time object detection, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 779–788. 1, 6

Ren, S., He, K., Girshick, R. and Sun, J. (2015), 'Faster r-cnn: Towards real-time object detection with region proposal networks', *Advances in neural information processing systems* **28**. 5, 17

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016), Rethinking the inception architecture for computer vision, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 2818–2826. 8

Uijlings, J. R., Van De Sande, K. E., Gevers, T. and Smeulders, A. W. (2013), 'Selective search for object recognition', *International journal of computer vision* **104**, 154–171. 5

Wang, C.-Y., Yeh, I.-H. and Liao, H.-Y. M. (2024), 'Yolov9: Learning what you want to learn using programmable gradient information', *arXiv preprint arXiv:2402.13616* . 6

Wang, X., Huang, T. E., Darrell, T., Gonzalez, J. E. and Yu, F. (2020), 'Frustratingly simple few-shot object detection', *arXiv preprint arXiv:2003.06957* . 2, 3, 12, 13, 14, 15, 17, 19, 23, 33, 34

Wang, Y., Yao, Q., Kwok, J. T. and Ni, L. M. (2020), 'Generalizing from a few examples: A survey on few-shot learning', *ACM computing surveys (csur)* **53**(3), 1–34. 6, 9

Yahalomi, E., Chernofsky, M. and Werman, M. (2019), Detection of distal radius fractures trained by a small set of x-ray images and faster r-cnn, *in* 'Intelligent Computing: Proceedings of the 2019 Computing Conference, Volume 1', Springer, pp. 971–981. 1

Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. (2014), 'How transferable are features in deep neural networks?', *Advances in neural information processing systems* **27**. 12

Zhao, Z.-Q., Zheng, P., Xu, S.-T. and Wu, X. (2019), 'Object detection with deep learning: A review', *IEEE Transactions on Neural Networks and Learning Systems* **30**(11), 3212–3232. 1

Zoph, B., Cubuk, E. D., Ghiasi, G., Lin, T.-Y., Shlens, J. and Le, Q. V. (2020), Learning data augmentation strategies for object detection, *in* 'Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16', Springer, pp. 566–583. 8

Zoph, B. and Le, Q. V. (2016), 'Neural architecture search with reinforcement learning', *arXiv preprint arXiv:1611.01578* . 7