


# Uses of Continual Learning Techniques in Generalised Few-Shot Object Detection

MPhil Thesis

Alessandro Lekkas Department of Computer and Information Sciences
University of Strathclyde, Glasgow

June 4, 2025

# Abstract

The best large-scale deep learning models require massive amounts of training data. For some tasks, collecting such data may be unfeasible, due to logistical or legal reasons. Few-Shot Learning has emerged as a field of study to maximise performance based on very few samples. In Generalised Few-Shot Learning, a model has to learn new few-shot classes while recalling earlier large-scale training classes. In this work, we review the tools and techniques used in Few-Shot Learning before exploring the parallels between Generalised Few-Shot Object Detection (G-FSOD) and Continual Learning (CL) methods. We focus on the manipulation of gradient descent since it has been recently proposed for G-FSOD. We show that gradient methods appear to be no better than existing techniques, and point out that potentially beneficial insights on sampling from the Continual Learning world have yet to be employed. We hope this work will provide a blueprint for further study of both G-FSOD and CL as interconnected fields.

# Contents

# List of Figures

List of Figures

# List of Tables

# Acknowledgements

I would like to thank my first supervisor, Prof Marc Roper for guiding me through the program, the experimental side and regularly reviewing my thesis. I am also very grateful to my second supervisor, Dr Andrew Abel for helping me with the thesis and working on the submission of the insights from this work for publication. Special thanks to my family and friends for their moral support.

In addition, this project would not have been possible in its current form without the computing resources provided by the ARCHIE-WeST project and funded by the University of Strathclyde.

# Chapter 1

# Introduction

## Research Problem

The most popular Deep Learning models to solve image tasks are normally trained on millions of data samples. However, while general image recognition models such as ResNet [He et al., 2016] and DenseNet [Huang et al., 2016] can be trained on datasets containing thousands or millions of images, some problems relating to specific domains only provide low amounts of data for training.

This lack of data can be due to logistical reasons making the collection of a large number of samples impossible, or legal reasons such as anonymity requirements. The application of the usual architectures is thus hampered, since the resulting model may not converge at all, and the distribution of a small dataset may not represent real-world conditions.

As the size of a training set increases, the probability of training a bad classifier decreases. This was mathematically proven by Anselm Blumer [Blumer et al., 1987]. His work states the probability that a poorly trained classifier, whose accuracy is the same as random chance, can be correct with $m$ training samples is less than $r(1-\epsilon)^m$ . In this formula $\epsilon$ denotes the true error rate, $r$ the number of hypotheses and $m$ the number of data points. It follows that when $m$ is a very low number, this theoretical bound becomes relatively high, making the achievement of good results in a test setting particularly challenging.

Additionally, the bound mentioned above is not a guarantee [Domingos, 2012], since the probability of the classifier itself being a bad one depends on the number of dimensions, which is often doubly exponential. This phenomenon was named the "curse of dimensionality" [Richard Bellman, 1957]. This is more so true for image processing, where the number of pixels can be considered the number of dimensions [Domingos, 2012], similarly to how columns are considered dimensions in tabular data. Compressing those dimensions into a space where it's computationally feasible to learn high-level feature representations is vital for a learner, but it relies on correlations between dimensions (or pixels) which are harder to find with few samples.

On the empirical side, a study by Luo et al. [Luo et al., 2019] quantified the impact of the size of generic datasets on the performance of ResNet, showing a sharp accuracy drop when dataset size was reduced by a factor of 5. For an applicative example, Dawson et al. [Dawson et al., 2023] used different geological datasets for a similar experiment across popular CNN architectures, confirming that smaller datasets perform worse in a test setting due to overfitting.



### Few Shot Learning

The field known as Few-Shot Learning (FSL) seeks to train models on a very modest number of samples. While a few-shot model's performance may not match the one of a model trained on massive amounts of data, it is possible to significantly improve their performance with respect to standard baselines.

One of the main approaches taken in FSL is to adapt common information from similar domains. Transfer Learning is practice of pre-training a model on a large standard dataset, and fine-tuning it on a novel one in a shorter training cycle. While transfer learning by itself may not be enough to solve the lack of novel data [Bernico et al., 2019], given it requires a slightly similar pre-training domain, it is usually applied in conjunction with other techniques.

A well known way to make the best out of existing data is Meta-Learning, or learning how to learn [Vilalta and Drissi, 2002]. It aims to train a "meta" model that

observes the training of downstream models on unrelated, data-abundant tasks. The meta model then employs that knowledge to guide the training of a few-shot model, by having the starting weights be the reusable functions it learned, and picking appropriate hyper-parameters.

Lastly, overfitting is an ever-present concern when data is scarce. Some methods focus on the use of distance metrics to separate class representations instead of just linear layers. This is known as Metric learning [Suárez et al., 2021].

Few-Shot Learning experiments make use of little data, so they're often described as $N$-way, $K$-shot tasks [Vinyals et al., 2016], where $N$ is the number of classes and $K$ the number of samples of class. For object recognition and detection tasks, common values of $K$ are 1, 5 and 10. More information on few-shot classification is available in Section 2.7: Benchmarks.

It is also possible to perform classification when $K = 0$: this is known as Zero-Shot Learning. In the absence of examples, classification requires additional semantic content, and is thus considered out of scope for this paper.

In a Generalised Few-Shot setting, the model has to learn a new few-shot dataset without decreasing performance on the original large-scale data. This can be of use in a number of practical applications, such as wanting to detect specific vehicles while still being able to detect a car. This is the specific case we'll be focusing on in the context of FSL.



### Continual Learning

Another problem that standard models and optimisers cannot cope with is that some real-life settings continuously generate data points belonging to new classes, such as user recommendations [Portugal et al., 2018] or continuous remote sensing [Li et al., 2020a]. This task is known as Continual Learning (CL). Running simple fine-tuning on the new data will give rise to a problem known as **catastrophic forgetting**, where the model will perform worse on older data to improve on the latest data. This is usually caused by overfitting on novel data, or interference between novel and previous data.

Catastrophic Forgetting relates to a problem known as the stability-plasticity dilemma,

wherein prioritising the stability of performance on base data will harm its performance on novel data, and vice versa.

The task of generalised fine-tuning is close to Continual Learning, as old data may not be available, but it is different from CL as there are only two sets of classes: the base ones with abundant data and the novel few-shot ones.

### Objectives

This project explores machine learning methods that work in a few shot learning setting, and investigates the integration of Continual Learning techniques used to prevent overfitting to improve transfer for Generalized Few-Shot Learning methods.

We ask the following research questions:

- Do methods from the field of Continual Learning improve fine-tuning performance in the Generalised Few Shot Learning setting?
- Can we perform such an improvement in a way that's easily adaptable and model-agnostic, by focusing on methods that manipulate gradient descent?

We constrain ourselves to enhancing training regimes, avoiding additional resource requirements during deployment.

### Contributions

Our work contains several key contributions. First, we provide a comprehensive overview of the literature in the field of Few-Shot Learning, and discuss how some Generalised FSL methods relate to Continual Learning principles.

Next, we show that gradient correction methods do not improve performance, and can harm it unless the gradients are averaged due to the learning objectives not being separate.

We then perform various experiments on base set sampling, including the removal of the sample limit and mini-batch balancing. We identify that selecting base samples depending on their absolute distance to their respective prototype yields a base AP

improvement, but using the ratio of distances to the correct prototype can yield a further improvement.

We conclude with the recommendation that G-FSOD research should further investigate sampling strategies from the Continual Learning field.

### Structure

**Chapter 1 - Introduction** currently providing a short summary of the problem, related tasks and research objectives.

**- Part I**. This part of the thesis is a survey of ongoing development of methods in the field of Few-Shot Learning, starting from classification methods, relating their advances to the FSOD task, and discussing the relationship between Continual Learning methods and G-FSOD methods. We provide some suggestions for future research.

**Chapter 2 - Classification** focuses on Few-Shot Classification. This is because many methods which we explore later were introduced in this context. We also detail how some domain-agnostic methods have been used in a few-shot setting.

**Chapter 3 - Few-Shot Object Detection** is an overview of Object Detection techniques and the adaptations which have been investigated for a few-shot setting.

**Chapter 4 - Continual Learning** outlines various continual learning tasks and details the various methods proposed to solve them, with a focus on gradient methods. It also shows how Class-Incremental Learning benchmarks share many similarities with standard fine-tuning tasks.

**Chapter 5 - Commonalities Between Fields** outlines some parallels between advances in Continual Learning and Generalised Few-Shot Object Detection which we consider important.

**- Part II**. In this part we discuss our rationale and experimental plan. After conducting experiments on gradient correction and sampling methods, We display our results, analyse them and draw overall conclusions.

**Chapter 6 - Methodology** outlines the methods chosen to fulfil the objectives mentioned above: evaluate the effectiveness of known gradient manipulation methods and sampling strategies as well as our own variations.

**Chapter 7 - Experiments** lists the conditions and the results of our experiments, then discusses why some of the results contradict an earlier study while reinforcing another.

**Chapter 8 - Conclusions** summarises our findings and contributions to the field, while charting some possible avenues to explore.

# Part I

# Chapter 2

# Few-Shot Classification

In this chapter we provide an overview of related research from the Few-Shot Classification setting. While the main task explored in this work is Few-Shot Object Detection (Chapter 3), many of the methods applied to FSOD were originally introduced for the task of image classification. This chapter explores the insights behind them, as well as potential advantages and disadvantages.

We start by exploring the most intuitive option: creating new samples for the network to train on in Dataset Augmentation. We cover relatively simple general-purpose methods in Classic Methods. The way we categorise most recent research is similar to the one used by Antonelli et al. [Antonelli et al., 2022], which groups approaches based on the distinction between metric based methods and meta-learning based methods. The former deals with overfitting by learning how to cluster samples based on a certain metric, and the latter learns a general-purpose classifier that can be easily fine-tuned. Finally, we investigate some field-agnostic methods that can be used to modify all existing architectures and evaluate the impact of self-supervision.

## 2.1 Dataset Augmentation

Several methods have been devised to deal with the issue of low dataset size. One of the most intuitive ones is to simply introduce more data by modifying the training samples, also known as data augmentation. Algorithmic methods such as rotation and

cropping have been used for large-scale architectures since AlexNet [Krizhevsky et al., 2012]. They improve networks' ability to generalise object detection with regards to simple image transforms, such as changes in position, rotation and colour patterns, but are not enough to solve data scarcity on their own. The downside is high correlation between training data samples does not help the model learn a broad distribution, and can in fact lead to overfitting on small sample sizes [Shorten and Khoshgoftaar, 2019]. Furthermore, the training time usually increases with each augmentation method employed, and it may provide diminishing returns.

### 2.1.1 Generative Adversarial Networks

Another data augmentation approach is to train a Generative Adversarial Network (GAN) [Goodfellow et al., 2014a] to generate extra samples for the training set. While GANs are generally used on large datasets, there have been a few advances towards applying them to a few-shot scenario, generating new samples from a few input images.



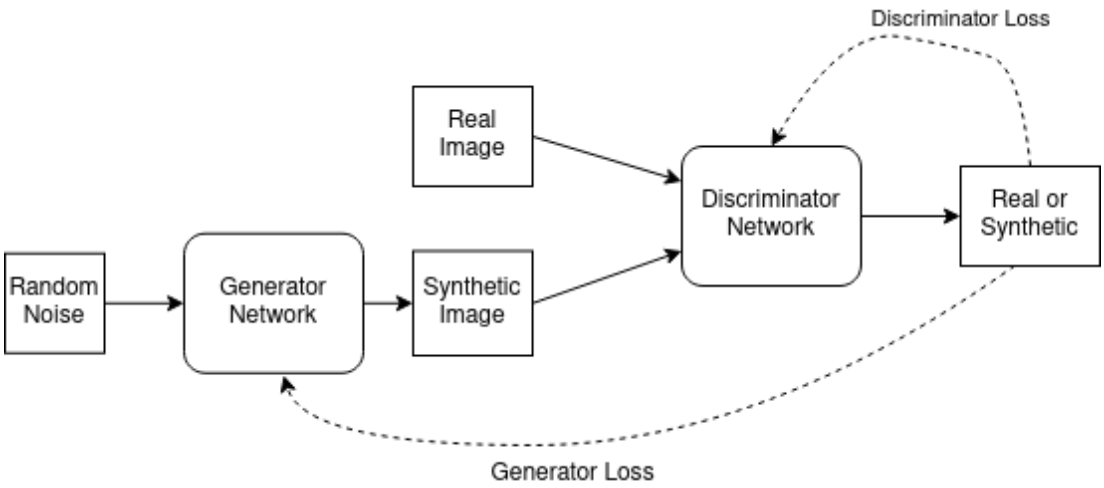


Figure 2.1: Summarised GAN architecture diagram

Figure 2.1 shows the functioning of the original GAN architecture, whereby a discriminator network outputs whether an image created by a generator network is real or synthetic. Springenberg [Springenberg, 2015] improved upon this architecture by

having the discriminator act as a classifier: instead of just outputting a truth score it would also output the category a sample belonged to.

Antoniou et al. [Antoniou et al., 2017] were the first to train a GAN to augment data for another network, and explored its use in a few-shot setting. Ali-Gombe et al. [Ali-Gombe et al., 2018] built on some of these insights to train a few-shot fine-grained classifier. Few-Shot GAN [Robb et al., 2020] achieved convergence by using pre-training and restricting fine-tuning to a small number of trainable parameters representing independent features. FIGR [Clouâtre and Demers, 2019] instead applied a paradigm called Reptile to the task, which will be later discussed in the Meta-Learning section (2.4) together with MetaGAN [ZHANG et al., 2018].

However, GANs can easily run into problems when generating new training data. A GAN may simply fail to converge and produce useless data. Another common failure mode is a GAN producing the same kind of output instead of properly modelling the distribution of existing data. This is known as mode collapse. A GAN will have difficulty interpolating relevant latent features based on just a few shots, and it may instead memorise the samples, causing it to display abrupt transitions, as explored by Radford et al. [Radford et al., 2015] in their seminal paper on GAN-based unsupervised learning. Current research often focuses on trying to transfer relationships between features across domains to avoid this problem.

Even if a GAN manages to overcome mode collapse, it still does not necessarily solve the problem of the downstream classification model not generalising outside the training and testing sets, since such extra samples generated from a small dataset are unlikely to match test conditions. Finally, most of the papers cited in this section have been tested on fairly uniform datasets such as MNIST [LeCun et al., 1998] and SVHN [Netzer et al., 2011] instead of more varied ImageNet-derived ones which would be closer to a real-world use case. So far, methods other than GANs may be better suited for few-shot learning, which will be explored in the following sections.

### 2.1.2 Variational Autoencoders

An alternative to GANs for sample generation is a Variational Autoencoder (VAE). An autoencoder is a simple approach that consists of two sequences of network layers: an encoder, and a decoder. The encoder shaped like a bottleneck, and it filters the input down to a few significant variables in a low-dimensional space. The decoder is tasked with reconstructing the output from those variables. The output is then compared to the original input, and usually the Mean Square Error between them will inform the weight update step. See Figure 2.2 for an illustration.

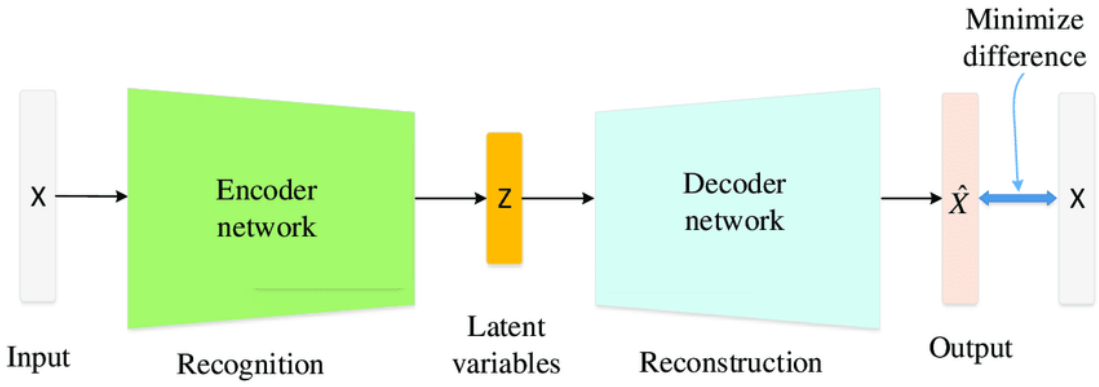


Figure 2.2: A basic Autoencoder architecture, adapted from Yang et al. 2019 [Yang et al., 2019]

This architecture is very popular for compressing data [Berahmand et al., 2024] since it can return a simplified version of an input, but it is not good enough for data generation, since it is simply attempting to replicate the original input, which will lead to overfitting.

By contrast, a Variational Autoencoder saves statistics about the input distribution such as mean and standard deviation to regularise the latent space, and randomly samples latent variables to recreate the output. This adds a degree of diversity in the output which helps generate extra data samples. Refer to Figure 2.3 to see how a VAE network operates.

The few-shot setting makes data generation with VAEs still a challenge, since there are so few samples to establish a distribution from, and generating new images might create spurious features for the classifier to train on. Schonfel et al. [Schonfeld et al.,

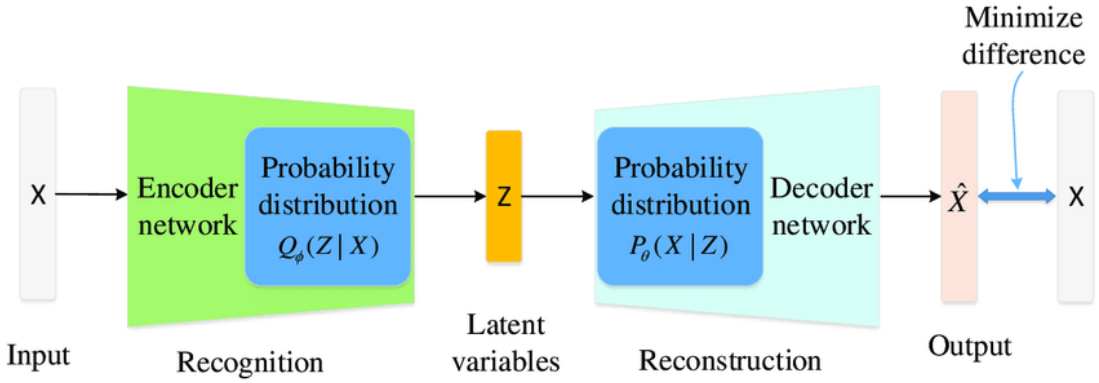


Figure 2.3: A Variational Autoencoder architecture, from Yang et al. 2019 [Yang et al., 2019]

2019] tackled this by generating low-dimensional features instead. Some of the inputs were from a large base training set to generate those features for novel ones. They tested their results in Generalised Zero-Shot and Few-Shot Learning, surpassing previous GAN-based approaches. Xu et al. [Xu and Le, 2022] improved upon their work by picking the most representative samples from base classes, before generating the new features. This was done by selecting the samples that generated the most common features, assuming such features followed a Gaussian distribution.

Part of the reason why VAEs are generally superior to GANs in a few-shot feature augmentation context is likely due to their different objectives: a GAN has to reduce loss for the discriminator to create realistic images, while a VAE has to reconstruct them by modeling data distribution, and its resulting features are interpolated. Thus, while images or features generated by a VAE may not match the expectations of a human eye, they are less likely to cause the mode collapse mentioned in the previous section 2.1.1.

Variational autoencoders are a valuable tool in FSL, although they require time for training the autoencoder and generating the extra samples, and the generated samples may not always be representative of their classes when some of the base features are from a different domain.

## 2.2 Classic Methods

In this section we discuss some more traditional methods. They have not been devised for FSL but we find they can be a good starting point for the FSL problem.

### 2.2.1 Boosting

Another way to improve generalisation is to use an ensemble of "weak learners" creating different feature maps from different subsets of the training data, whose combined predictions tend to prove more accurate than those of a single model. Combining such learners in parallel and merging their predictions as the final step is known as bagging [Breiman, 1996], while combining learners in a sequential manner, with each subsequent "weak" predictor correcting the previous one's output is known as boosting [Freund et al., 1999].

While ensemble learning is often used with traditional analysis methods, there have been studies to integrate it with Convolutional Neural Networks. This technique can make use of transfer learning as well, since the underlying individual CNNs can be initialised with pre-trained weights. For example, Taherkhan et al. [Taherkhani et al., 2020] and Ren et al. [Xudie Ren et al., 2017] combined popular boosting algorithms to perform classification tasks. The method itself is quite versatile, having been used for counting objects [Walach and Wolf, 2016], and for few-shot instance segmentation [Nguyen and Todorovic, 2019] with good results in their respective fields.

However, some of these papers used their own CNN [Taherkhani et al., 2020] or similar works [Xudie Ren et al., 2017] as a baseline, to show any CNN-based method can be improved by boosting, rather than trying to compete with the state of the art in large models. Additionally, surveys in the field, while very useful, often focus on large-scale rather than few-shot settings [Rahman et al., 2021].

One of the exceptions to this is Zhang et al. [Zhang et al., 2021], who in 2021 used boosting in a few-shot learning context on challenging datasets, albeit their work has flown under the radar at the time of this writing. They combined a CNN with two CV algorithms, Histogram of Oriented Gradient [Dalal and Triggs, 2005] and Local Binary

Pattern [Ojala et al., 2002] for boosting. Their intuition was that, although CNNs have trouble learning features from a handful of examples, algorithmic methods can still be relied upon to improve feature detection. They achieved impressive performance, demonstrated with various baselines from the FSL field.

Boosting methods require more computational resources, not just at training time but in deployment as well. If inference time is to be kept constant, then the individual component models have to run in parallel, increasing memory and computation requirements. Conversely, running the models sequentially and holding the results for each stage will require less memory but increase inference time.

When boosting provides better results its usage should be evaluated for a cost-benefit analysis. It can be a useful paradigm but we did not make use of it, since one of the aims of this project is to avoid additional resource requirements in deployment.

### 2.2.2 Transfer Learning

Transfer Learning involves two phases: pre-training and fine-tuning. First, in the pre-training phase, the model is trained on a massive generic dataset such as ImageNet [Jia Deng et al., 2009]. Then, in the fine-tuning phase, the last layer which determines the output is removed or modified, some of the initial layers may be frozen to preserve low-level features, and the model is trained on the actual dataset for the task at hand.

Its helpfulness in the context of few-shot learning derives from the original model having already learned low-level features such as edges, but it will not incorporate domain-specific data, and high-level features will not be transferable unless the domain of the original model was to some degree similar to the target one. More information about what constitutes "similarity" is available in Section 2.7 - Benchmarks.

The work by Dawson et al [Dawson et al., 2023], previously mentioned in the Introduction (Chapter 1), found that transfer learning methods are still very prone to overfitting on small datasets (1-10k data points) when results were compared with a larger dataset of 104k data points.

While transfer learning by itself is not enough to solve the FSL research problem, it can be combined with other techniques to speed up training. A review paper [Song

et al., 2022] mentions the limitations of domain transfer but points out its usefulness when integrated with other methods.

Using pretrained weights in FSL is a common practice, so many FSL works use well-known backbones such as ResNet [He et al., 2016] and VGGNet [Simonyan and Zisserman, 2014] When the novelty of a work is not just predicated on a custom CNN structure, some authors [Tian et al., 2022] [Nguyen and Todorovic, 2019] run their experiments on top of different pre-trained convolutional backbones, to include fair comparisons with previous works.

## 2.3 Metric Learning

Metric learning aims to find distance metrics to separate instances of different classes and aggregate instances of the same class, to enhance the performance of similarity-based algorithms [Suárez et al., 2021] . This section explores metric learning methods that have been used in a few-shot settings.

### 2.3.1 Siamese Networks

In a Siamese Network, introduced by Bromley et al. [Bromley et al., 1993] in 1993: two sub-networks are trained with different inputs but share their weights, meaning they have fewer parameters to learn. It is quite proficient at identifying class similarities from a low number of samples, though it generally requires more training time than a single network. The reason Siamese Networks qualify as metric learning is that they use loss functions tailored for them, most commonly contrastive loss [Hadsell et al., 2006] [Chopra et al., 2005] and triplet loss. [Schroff et al., 2015]

**Contrastive loss** uses two image inputs and calculates their distance, if it is smaller than a certain margin images are treated as belonging to the same class, otherwise as different classes and the prediction is penalised.

**Triplet loss** uses three images: an "anchor" (the input), a "positive" one (labelled as belonging to the same class) and a "negative" one (labelled as belonging to a different class). The function minimises the difference between the input and positive one, while

maximising the difference between the input and the negative one.

The usage of a Siamese Network in the context of few-shot learning was investigated by Koch et al. [Koch, 2015], reporting a significant performance boost, albeit their experiments were run on relatively simple datasets such as MNIST [LeCun et al., 1998] and OmniGlot [Lake et al., 2015]. Further discussion of datasets is available in Section 2.7 - Benchmarks.

Bootstrap Your Own Latent (BYOL) [Grill et al., 2020] used Siamese Networks and metric learning to adapt learned features without using positive-negative pairs. It managed to achieve good results by only using a small subset of labeled data for ImageNet, and without having to retrain the network for new data. Their approach used one branch of the network as a predictor for the other.

Siamese Networks can develop an undesirable behaviour: when two images fed to the network at the same time are too similar, the network may learn the same exact representation for both. This "collapse" was investigated by Chen et al. [Chen and He, 2020], who developed a Siamese model simpler than BYOL which it shares some similarities with, such as not using positive-negative pairs and having a target and prediction branch, but it relies on a stop-gradient operation to prevent the prediction target branch from adopting the same solution as the predictor.

However, it is important to remember while the two works above provide a useful way of learning with relatively few labeled samples, their experiments used subsets of data which are still in the order of hundreds or thousands of images, rather than ten or less.



### 2.3.2 Contrastive Learning

Contrastive Learning is a field that focuses on the creation and application of proper contrastive loss functions. As mentioned in section 2.3.1, contrastive loss relies on maximising intra-class distance and minimising inter-class distance. Since Contrastive Learning networks rely on a distance metric, they are classified as a form of metric learning.

One of the earliest contrastive functions is **Noise Contrastive Estimation** (NCE)

[Gutmann and Hyvärinen, 2010]. It attempts to solve the task of estimating the probability a sample is a "positive" one by presenting it as a linear regression task, where the goal is to estimate the parameters of the probability distribution function. It functions effectively as a binary classifier to establish whether a chosen sample is "real" (positive) or "noise".

It was originally devised for language processing. Since comparing a sample against all possible words in the vocabulary was not feasible, it uses random negative sampling, which provides a good enough approximation of the training set's properties.

**InfoNCE** [van den Oord et al., 2018] is a newer loss function for contrastive learning. While similar to NCE, in that it tries to find a "real" sample in a pool of "noise", it generalises previous insights by turning it into a custom cross-entropy loss whose value depends on the similarities between data.

This makes InfoNCE particularly apt for unsupervised computer vision problems, as demonstrated by He et al. [He et al., 2020]. Building on the insight of other unsupervised methods, they treat images as a dictionary and train an encoder network to perform image based look-ups without supervision by minimising InfoNCE loss. To stop the encoder from changing too quickly and failing, they use a momentum function, which works well in smoothing encoder outputs.

**SimCLR** [Chen et al., 2020] adopted a simpler approach, as per the name (CLR stands for Contrastive Learning of visual Representations). The authors have the network look at randomly augmented views of the same images and minimising a loss function similar to InfoNCE (NT-Xent) which includes a temperature hyperparameter. The positive pairs derived from the same images are pulled closer in the representation space and the negative ones pushed further away. SimCLR achieved very good results against ImageNet with only 1% of labeled data, but it was not tested in a stricter few-shot setting with 5 or 10 samples.

One drawback of NCE is it considers all of the negative samples drawn to be the same "noise", without considering any relationships they might have. Recent research [Kalantidis et al., 2020] has proven that investigating strategies to pick "hard" negative samples improves learning and requires less memory for sample dictionaries.

Contrastive Learning methods are used in tasks other than classification, which is why we mention them again in Chapter 3. While contrastive learning may not suffice to adapt known methods to solve FSL, since it usually changes just the loss formulation, it can be used together with other approaches.

### 2.3.3 Matching Networks

Proposed by Vinyals et al. [Vinyals et al., 2016], this method is based on the idea of a weighted, fully differentiable nearest-neighbours classifier. The motivation is to solve the need for large numbers of parameters, and thus training data, by integrating non-parametric methods, in this case a kNN-like algorithm.

During training, the network learns through an LSTM the functions to retrieve the most important embeddings, as a form of attention. During inference, an input sample is compared with every sample in the training set. The functions learned by the neural network create final embeddings, the embeddings are compared using cosine distance to check their similarity, and the labels are predicted via SoftMax. According to the authors, part of the importance of their method to few-shot learning is that this process can be applied to unseen classes without changes to the network itself.

One downside is that, since it needs to compare every input to all support samples, it does not scale well with dataset size. Although few-shot learning datasets are relatively small, this would be a constraint in deployments where computing power is limited.

The paper also helped introduce the concept of episodic training in FSL. In each episode, a subset of classes is picked from the training set, each with a handful of examples split into training (support) and test (query) sets. The network then learns to solve a large number of these sub-problems or "episodes". This could be seen as a form of Meta Learning, or learning how to learn, more information on which is available in Section 2.4 - Meta Learning.

### 2.3.4 Prototypical Networks

Prototypical Networks [Snell et al., 2017] compute a centroid vector for each class, called a "prototype", that represents a cluster of samples. Incoming input data is classified

based on the distance between prototypes. While this may be reminiscent of traditional clustering algorithms, there are some key differences. The prototypes are more than just a "mean" vector, such as in k-means. Rather, they are learned through a deep neural network, which learns to map points to their corresponding prototypes. To learn the distance functions, networks use differentiable, convex functions (Bregman divergences) for regular exponential families to estimate distance between data points and centroids. Additionally, the input for learning the prototypes is provided via random sub-sampled batches, which means the samples don't have to be present in memory like in k-means clustering.

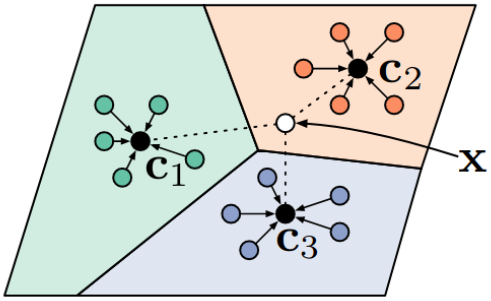


Figure 2.4: Prototypical networks in few-shot learning, using an existing support set. A sample $x$'s class is determined by its proximity to learned centroids $C_i$. [Snell et al., 2017]

According to the authors, when each class' train set contains only one shot (one-shot learning), Prototypical Networks and Matching Networks become equivalent, since there is only one support sample per class. However, with multiple data points per class (i.e. few-shot rather than one-shot) prototypical networks are computationally much more efficient than matching networks, since the input sample is only compared to the centroids rather than every sample in the support set. A good visualisation of this was provided by the authors of the paper in Figure 2.4. The method can be used for supervised as well as unsupervised learning (see Section 2.6: Unsupervised Learning).

Prototypical learning can also be combined with contrastive learning, as shown by Li et al. in their 2020 paper, [Li et al., 2020b], albeit in a large-scale learning context. Each image was assigned to different centroids, and a contrastive loss function

(ProtoNCE) employed to minimise distance to the most representative centroid while maximising distance with the others.

While the performance of prototypical networks on their own has been surpassed, the integration of prototypes is still a good addition to the toolkit of few-shot learning practitioners, and it has been employed in one of the baselines this work compares against, as explained in Chapter 3 - Few Shot Object Detection.

### 2.3.5 DeepEMD

Earth Mover's Distance [Rubner et al., 2000] is a general similarity metric built in the context of image processing research. It treats the matching of different distributions with signatures of varying sizes as a transportation problem, using the analogy of a shoveler moving dirt from piles to holes. The DeepEMD network introduced by Zhang et al. [Zhang et al., 2020] makes use of EMD as an image similarity metric, rather than cosine or euclidean distance.

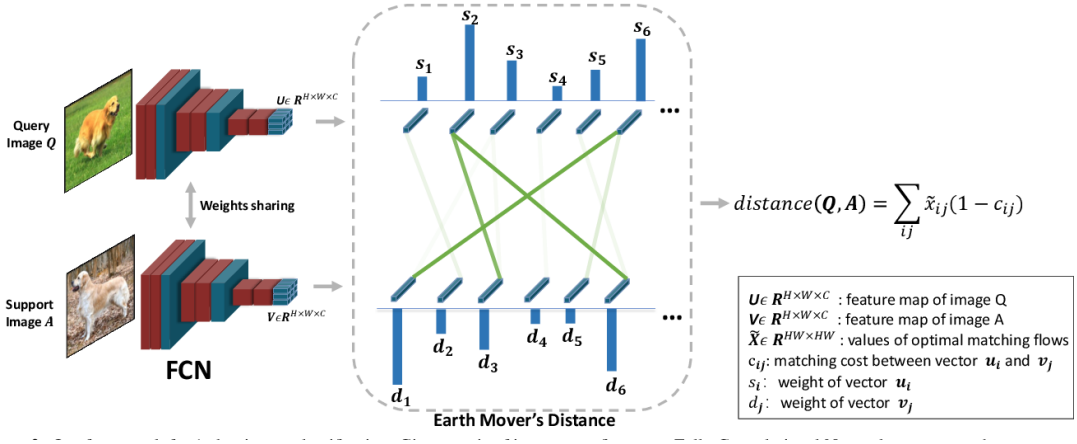


Figure 2.5: DeepEMD network diagram from Zhang et al. [Zhang et al., 2020]

While the DeepEMD backbone in Figure 2.5 above is reminiscent of Siamese Networks, it is not the same since, rather than just duplicating weights, it uses a cross-reference mechanism that takes statistics of weights into account to devalue high-variance background regions or non-correlated object parts. Their accuracy was improved by the usage of a "Structured" FC layer with normalised weights, which learns vector prototypes for each class to be later compared by EMD.

DeepEMD achieved competitive results in the context of few shot classification, making it another baseline for few-shot learning methods that make use of metrics. It uses Meta-Learning as part of its training loop, which we discuss in Section 2.4.

## 2.4 Meta-Learning

A branch of data science that is promising in the field of FSL is meta-learning: a model can learn how to solve specific tasks by solving a generic set of training tasks as the first step, then applying that knowledge to solve previously unseen tasks with a test set. It usually operates on two levels, a "base" learner which is simply trained on actual data, and a "meta" learner that holds general knowledge about learning derived from the "base" one [Vilalta and Drissi, 2002].

Meta-learning is different from transfer learning. Instead of reusing a previously trained network to learn different weights, it aims to train a completely new network with parameters learned from training for other tasks, such as weight initialisations, learning rate and gradient updates. In addition, while pre-training focuses on transferring features across domains for a single task such as classification, meta-learning networks train the "base" learner on a number of different pretext tasks. [Song et al., 2022]. In the rest of this section, we offer a short history of meta-learning and how it has been applied to the few-shot setting.



### 2.4.1 Optimisation Methods

Learning how to learn, how it was first described, was first investigated in the 1990s [Vilalta and Drissi, 2002], but meta-learning with image processing techniques including CNNs scaled poorly until the use of GPUs for experiments became widespread and supporting infrastructure matured.

One of the first modern meta-learning works used Neural Turing Machines, a recurrent architecture for Turing Machines, for retrieving general knowledge information [Santoro et al., 2016]. Training a meta-learner via a conceptually simpler Long Short-Term Memory (LSTM) network was explored by Andrychowicz et al. [Andrychowicz

et al., 2016]. The authors used the parameters of the base learner such as the gradient as the input for the LSTM cell, with the base learner's updated state as the output. By training the LSTM's update function while training a base model on multiple tasks, they were able to create a meta-learner that could generalise for unseen tasks.

The first work to adapt the method for a few-shot setting was by Ravi et al. in 2017 [Ravi and Larochelle, 2017]. In this work they also observed that standard gradient-based descent resembles the cell state update in LSTM by changing the form of some parameters. This proved useful as it made it possible to envision meta-learning without LSTM.

Model Agnostic Meta Learning (MAML) [Finn et al., 2017] aims to learn the initial weights for a base model by training on a random set of meta-tasks, so it can be easily fine-tuned with a relatively small number of gradient updates (See Figure 2.6). As described, the test error from the meta-tasks serves as the training error for the meta-learner.



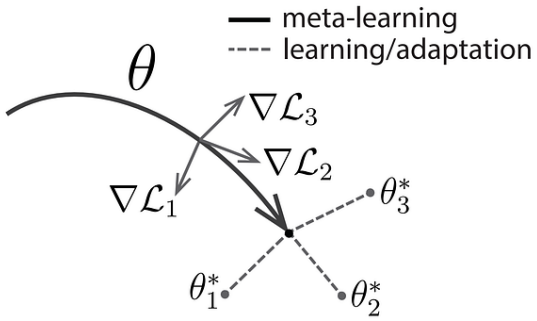


Figure 2.6: The working of Model-Agnostic Meta Learning (MAML), illustration from the original paper [Finn et al., 2017]. $\theta$ denotes the initial parameters, and $L_T$ the loss gathered over testing of random meta-training tasks $T_i$

Model-Agnostic refers to it not being dependent on any particular network architecture, since the meta-optimisation uses Stochastic Gradient Descent. This approach makes it malleable for learning on small data sets without overfitting, and it has proven better than regular transfer learning on benchmark datasets.

It was suggested by Raghu et al. [Raghu et al., 2019] that MAML's effectiveness lies in feature reuse, rather than just rapid learning. According to their findings, the outer optimisation loop of the meta-learner does not necessarily find settings that make

specialisation easier, rather it leads to parameter values that correspond to reusable features. They proved this empirically by freezing the last layers of the sub-network that learns the specific task, and observing that performance was almost unaffected: the meta-initialisation provided good features already.

Since then, a number of improvements have emerged over MAML. Meta-SGD [Li et al., 2017] uses SGD just like MAML, but learns the update direction and learning rate as well, rather than just the parameter initialisation. In the original MAML paper, they mentioned that doing backpropagation twice across the base and meta network is slow. Reptile [Nichol et al., 2018] is a simpler algorithm that uses a first-order approximation instead, by just moving the inputs for the meta-learner slightly closer to the minima obtained across various meta-tasks. While MAML typically uses shallow networks to generalise better, thus limiting its effectiveness, [Sun et al., 2019] employed deeper neural networks by having the meta-network learn how to scale and shift model weights for different tasks.

Additionally, MAML evaluates multiple test images in a batch sharing a few statistics, which makes a comparison with the other methods not entirely fair, given the standard problem is to predict labels for one image at a time. More information on this distinction is available in Section 2.6.1 - Transductive Learning.



### 2.4.2 Relation Networks and Meta-Learning

The idea of relation networks was first proposed by [Santoro et al., 2017]. The network learned a function $f\theta$ to encode image feature maps and a function $g\theta$ which would produce a relation score from the concatenated feature maps. Unlike older pipeline methods, the network is trained fully end-to-end.

While this intuition was applied to visual reasoning in a question-answer architecture for relating objects in the same image, which included a LSTM language processing module, the image processing step was vital to some subsequent works in the field of FSL.

In Sung et al.'s work [Sung et al., 2018], this relational image processing architecture was applied to few-shot image classification (see Figure 2.7). One possible downside of

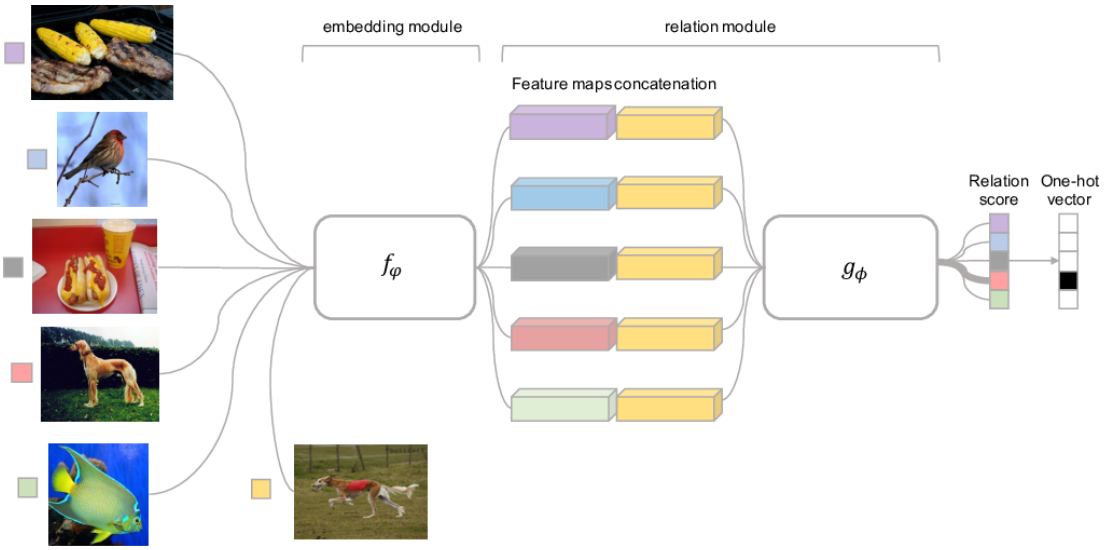


Figure 2.7: Diagram of Relation Network for FSL image classification from [Sung et al., 2018]

this architecture is the added complexity during training as opposed to simple metric-based methods and thus the possibility of diminishing returns compared to simpler methods (see Section 2.8 for a performance comparison).

In the Latent Embedding Optimization method [Rusu et al., 2019], the authors focused on the meta-learning optimisation of the network, by training it to generate parameters via an encoder-decoder architecture combined with a relational network. Meta-learning on these compressed features achieved better results than both MAML and Relation Networks.

### 2.4.3 Other Combinations

Since Meta-learning is more of a paradigm rather than a specific optimisation or architecture, it can be utilised with other specialised approaches. A non-typical combination involves training **GANs with meta-learning** to generate more training data. One such work is Data Augmentation GAN (DAGAN) [Antoniou et al., 2017], although a subsequent paper named MetaGAN [Zhang et al., 2018] claims that by treating generated data the same as the original training one it risks mode collapse (see 2.1.1). The two papers are hard to compare as MetaGAN evaluated performance on top of differ-

ent base methods, while DAGAN did not test on miniImageNet. As for the method, MetaGAN trained a discriminator to recognise the class of a sample as well as its realness, treating generated data separately from genuine data. Additionally, their method allowed sampling of unsupervised tasks for meta-training. Both meta-learning GANs mentioned here can be integrated with any other FSL classification approaches such as Matching Networks or Relation Networks.

Meta-learning has been used with Boosting (see Section 2.2.1) as well. Diversity with Cooperation [Dvornik et al., 2019] combined meta-learning and prototypes with an ensemble strategy of deep networks. At the end classifier outputs were averaged, which outperformed the simple voting typical of boosting methods.

Although combining methods often improves performance, the caveats associated with their component methods remain the same: boosting methods require extra resources, and generative methods require extra training time, which may provide diminishing returns.



## 2.5 Network Architecture Enhancements

In this section, we discuss extended training methods that rely on modifying the architecture of the network itself. These have the advantage of being independently applicable to any method. Reviewing such enhancements will help us determine their impact in the works we'll be comparing, and whether our original view on deployment resources may be too constraining.

### 2.5.1 Network Architecture Search (NAS)

While the standard way to create networks is to hand-code the layers, in NAS, an algorithm or model tries to pick the optimal layer sizes and various other configurations based on a search strategy, the network is trained and evaluated and its results fed back into the searcher, as shown in Figure 2.8.

However, searching for an optimal architecture from scratch for any task is resource-intensive. To help solve this, meta-learning has been applied to NAS as well. In M-

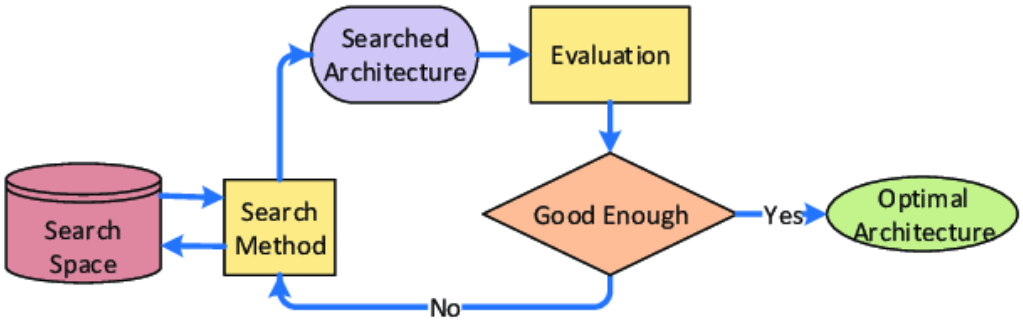


Figure 2.8: Network Architecture Search diagram from Santra et al. [Santra et al., 2021]

NAS [Wang et al., 2020a] and Meta-NAS [Elsken et al., 2019], a network performs meta-training on a number of dummy tasks and learns meta-weights that adapt to a new task in just a few steps to obtain an optimal architecture. Both works achieved better results than baselines such as MAML. However, it can take a long time to find the best parameter combinations compared to other approaches.

Furthermore, according to Li et al. [Li and Talwalkar, 2020], random search with early stopping can perform better than neural architecture search.

### 2.5.2 Knowledge Distillation

Knowledge Distillation (KD) is a training method where a "student" network tries to keep its outputs as close as possible to the outputs of a "teacher" network. This is done by adding a new term to the loss, often the MSE or the KL divergence between their outputs [Kim et al., 2021]. This method was originally devised with the goal of model compression by Bucila et al. in 2006. [Bucilă et al., 2006], to shrink the parameters of a larger model (the teacher) into a smaller (student) network. It was formalized as an approach by Hinton et al. in 2015 [Hinton et al., 2015]

An overview of the process is shown in Figure 2.9, whose authors [Feng et al., 2023] used KD to facilitate deployment on low-resource embedded platforms. We can see from the diagram that the use of two models during training is similar to Siamese Networks (previously discussed in Section 2.3.1). However, while SiamNets use shared

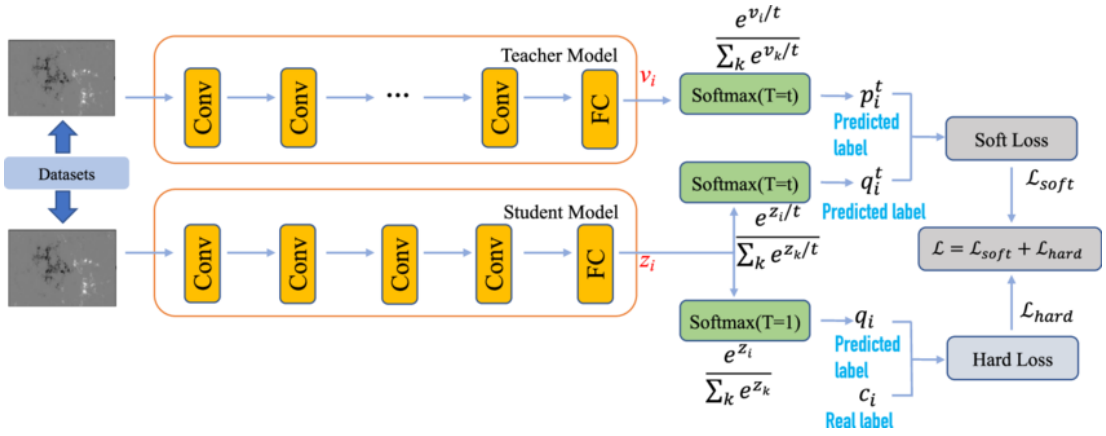


Figure 2.9: Knowledge Distillation diagram from Feng et al. [Feng et al., 2023]

weights and different inputs to achieve close outputs, Knowledge Distillation has a student network learn from the teacher using the same inputs.

When the teacher and student network architectures are the same, it is known as Self-Distillation (SD). In this case distillation is not used for shrinking the network but to improve baseline performance. While this process seems rather counter-intuitive, Allen-Zhu and Li [Allen-Zhu and Li, 2020] conclude that the SD's peformance boost is due to the network learning different representations thanks to random initialisation and consolidating them into a single model. SD has been frequently used in Few Shot Learning, and has seen use in Few-Shot Object Detection as well, as we'll discuss in Chapter 3.

A successful application of Self-Distillation was in the popular meta-learning few-shot classification baseline "A Good Embedding Is All You Need" [Tian et al., 2020]. At first, the authors used the union of all possible meta-training tasks during the pre-training phase, instead of sampling them, and learned feature embeddings via ResNet. They proceeded to use those embeddings to train a linear classifier. Surprisingly, they found that using logistic regression as a final (base) learner was highly competitive with state of the art meta-learning algorithms. When combined with Self-Distillation over a number of iterations (termed "generations"), their method outperformed all prior works, demonstrating how easily it can improve such a simple baseline.

Self-Distillation can form a baseline or be employed with other methods. While it

requires lengthier training, it does not require additional deployment resources. Given our time constraints we will not make use of it, although experiments in this direction might be a good extension to this work.

## 2.6 Reducing Supervision

Most of the methods described so far were primarily conceived in a supervised learning setting, albeit some such as ProtoNets [Snell et al., 2017] can be applied without supervision. In **Unsupervised Learning**, there is no ground truth fed to the model: the model has to learn to cluster together representations of the data.

Fei-Fei et al. [Fei-Fei et al., 2003] is a notable early example of one-shot learning using unsupervised learning. They used algorithmic feature detection followed by a Variational Bayesian method for approximating a probability distribution function in order to classify object categories. Their experiments were applied to a small number of examples (1 to 5) as well as larger sets for completeness.

Probability function estimators and clustering algorithms are generally framed as unsupervised learning. As previously discussed in Chapter 1, dimensionality reduction methods are needed to compress representations into a manageable space. This begs the question of how to let a network learn high-level features without any human supervision. The usual technique for this is called **Self-Supervised Learning** (SSL) [Raina et al., 2007].

In SSL, training is performed via a neural network, and this network is pre-trained on ad-hoc pretext tasks before the main "downstream" training. These tasks can be applied to the original dataset or completely new data, and they don't require human annotation, just like the downstream task: the labels are generated together with transformations on the data. One example pretext task would be predicting the rotation of an image [Gidaris et al., 2018]. Another popular task is the jigsaw puzzle: the image's patches are scrambled, and the network has to put them back together, as shown in Figure 2.10 [Noroozi and Favaro, 2016]. These tasks help the network learn about general features such as rotation and shapes.

The use of pretext tasks may sound similar to meta-training, but it features a major

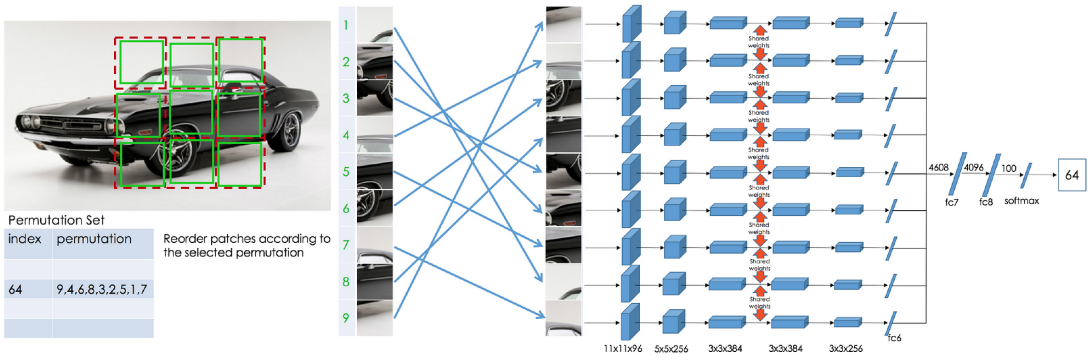


Figure 2.10: Diagram illustrating jigsaw pretext task from Norooz et al. [Noroozi and Favaro, 2016]

difference. While meta-training tasks are randomly sampled, their labels usually require human annotation. Some work has been proposed for making use of automatically constructed tasks [Hsu et al., 2019] to merge the two approaches. In contrast, labels for SSL tasks are always generated from transformations and such.

However, learned feature representations can be being overly dependent on the chosen pretext tasks. Misra I. and Maaten, L. [Misra and Maaten, 2020] proposed Pretext-Invariant Representation Learning (PIRL), which predicts image representations similar to the transformed ones but different from each other, in a fashion similar to contrastive learning.

The work of Gidaris et al. [Gidaris et al., 2019] concatenated a self-supervised learning stage with well-known metric learning methods, testing it on as ProtoNets, which use Euclidean distance, and cosine-distance classifiers. They also had their own take on the pre-training jigsaw task, introducing a task to predict the location of image patches only relative to each other, rather than in the whole image.

According to their experiments, chaining SSL with cosine classifiers beat most of the state of the art methods, save for A Good Embedding Is All You Need [Tian et al., 2020] and DeepEMD [Zhang et al., 2020].

Since Unsupervised Learning often requires the exploration of similarity metrics, and thus Contrastive Learning, it would be unfair not to mention UniSiam [Lu et al., 2022]. In this paper, the use of self-supervised contrastive learning is investigated for few-shot classification, aiming to maximise mutual information between augmented

views of the same image. Following previous work that finds InfoNCE is biased when it comes to estimating mutual information, they propose another estimator for pre-training, integrate Knowledge Distillation [Hinton et al., 2015] and the final classification result is found to be fairly competitive with state of the art methods.

Self-supervision entails longer training time, but such methods are generally superior in performance to traditional methods. What's more, it can be applied to few-shot transfer learning as shown by Yu et al. in TransMatch [Yu et al., 2020b]. Since it can be limited to the fine-tuning stage, a meta-learning approach is not required.

While method comparisons are not always straightforward, one of the most performant methods is a modular combination of the methods explored so far: "Ensemble Augmented-Shot Y-shaped Learning" [Bendou et al., 2022] makes use of a self-supervised pre-training task (predicting object rotation), as well as prototypes for data pre-processing and the classifier. The feature extractor can be optionally improved by building an ensemble of ResNet12 backbones and reducing their parameters by pooling to decrease resource requirements and avoid an unfair advantage over other methods. It is one of the best methods surveyed so far, although at a cost of higher training time.

A summary of the performance of each approach explored so far is available under Results (Section 2.8). The next sections explore why some historically included methods have been excluded from comparisons, as they use a different evaluation setting (Section 2.6.1), and how performance is compared for FSL methods (Section 2.7).



### 2.6.1 Transductive Learning

Most of the supervised and unsupervised approaches explored in the previous sections would fall under Inductive Learning: a model learns from a training set, then tries to predict labels on a test set. On the other hand, Transductive Learning makes use of an unlabeled test set as well to help model distribution. Although this deviates from the common practice of keeping test sets separate, it can work in scenarios where data to be tested is known in advance.

It is a reasonable compromise when the main obstacle to obtaining training data is the lack of labels, since it can leverage feature distribution across the entire set to

generate "**pseudo-labels**" on unseen data. These are intermediate labels generated by the network in a training cycle, which do not necessarily correspond to the final predictions and they must be considered separately from actual ground-truth labels.

One of the first Transductive Learning works for a few shot setting was by Liu et al. in 2019 [Liu et al., 2019]. In this paper, feature embeddings are constructed over the entire set (train and test), and a Graph Neural Network is used to propagate labels from the support samples to the unlabeled query samples.

An alternative to label propagation is embedding propagation, where the feature embeddings are interpolated across the training and unlabeled test set [Rodríguez et al., 2020]. In "A baseline for few-shot image classification" [Dhillon et al., 2019], The authors simply fine-tune a pre-trained network using features from both the support and query set to provide a simple baseline approach, and the results for this approach are not distant from the state of the art.

"An Embarrassingly Simple Approach to Semi-Supervised Few-Shot Learning" introduced MUSIC [Wei et al., 2022], a method based upon applying negative pseudo-labels to samples over a number of iterations, thus learning a classifier by exclusion. The number of iterations depends on the number of classes. The intuition behind it is that negative labels can be easier to learn than positive ones.

Another interesting development which starts from relatively simple components was contributed by Liu et al. [Liu Jinlu, 2020], who pointed out that when the few available examples have a narrow distribution, the prototypes in a ProtoNet tend to be heavily biased. They set about rectifying this bias by propagating pseudo-labels to unlabeled samples, and achieved state-of-the-art results for their time. Their work uses cosine rather than Euclidean similarity as a metric. Transductive Learning and label propagation can be complemented by a meta-learning approach to pre-initialise the weights, as shown by Dong et al. and Liu et al. [Dong et al., 2022] and [Liu et al., 2022].

One flaw of previously mentioned transductive learning research is that the samples are assumed to come from a **normal** distribution rather than a potentially skewed one. In "Realistic Evaluation of Transductive Few-Shot Learning" [Veilleux et al., 2021] ,

the authors use a Dirichlet distribution for sampling benchmark data and re-ran most previous authors' experiments. Their results show a significant drop in performance across transductive methods. What's more, in a useful contribution to the rest of the Few-Shot Classification field, it shows MAML and derivative works are impacted as well, because they expect test data to arrive in batches and perform normalisation on it.

It was for this very reason we chose not to include MAML and derived methods in the results table, as it would not be a comparison performed in the same setting. Inductive metric and meta-learning methods are immune to this effect, as shown by their tests on ProtoNet, since their networks were not led to make assumptions about the distribution of the data. The loss function used by the authors can improve results on DeepEMD as well.

While Transductive Learning is an interesting paradigm, it has the disadvantage of not being suited to continuous inference: since a transductive method requires the entire test set upfront, it has to be re-run from scratch upon addition of new test samples.



## 2.7 Benchmarks

Development in every field of Machine Learning is helped by a set of objective benchmarks for evaluation. While the usual large-scale benchmark datasets such as ImageNet or MS-COCO are too large for few-shot evaluation, there are options available that have fewer images per class without compromising class distribution.

Some of the traditional datasets such as MNIST (handwritten digits) [LeCun et al., 1998] and SVHN (street view house numbers) [Netzer et al., 2011] are far too specialised and easy to learn, so they are no longer in mainstream use by papers that attempt to challenge the current state-of-the-art. The same goes for OmniGlot [Lake et al., 2015], a dataset of handwritten characters from different alphabets. OmniGlot is not a challenging dataset given its limited scope [Triantafillou et al., 2020], with most methods achieving over 90 % accuracy, so it is not a particularly useful benchmark.

Most of the benchmarks currently in use are down-sampled versions of large-scale

datasets. During each training episode, 5 random classes are chosen from the dataset, and only a handful of images are available per class (usually 1, 5 or 10). The test results are averaged over 600 such splits of 5 classes each.

One of the most popular and challenging benchmarks is Mini-ImageNet [Vinyals et al., 2016], which is a much reduced version of ImageNet consisting of 50k training and 10k test images, divided into 100 classes. Tiered-ImageNet [Ren et al., 2018] is a different ImageNet split that groups classes together in a hierarchical manner.

Another classification benchmark, CIFAR-FS [Bertinetto et al., 2019] is a randomly sampled subset of the generic dataset CIFAR-100 [Krizhevsky et al., ]. Finally, CUB-200-2011 [Welinder et al., 2010] is a commonly down-sampled dataset used for fine-grained classification: the subjects of the photos are birds, and the benchmarked model has to learn to tell different species apart.

### 2.7.1 Cross-Domain Transfer

Some of the generic benchmark datasets mentioned in the previous section such as mini-ImageNet partly suffer from similarity between classes, which can hurt performance in a real-world domain-specific setting. Meta-Dataset [Triantafillou et al., 2020] samples data from previous datasets such as ImageNet, MS-COCO, CUB-200, OmniGlot and many other domain-specific ones such as the VGG Flower Dataset [Nilsback and Zisserman, 2008] to provide networks with a broad variety of samples across domains.

Meta-Dataset [Triantafillou et al., 2020] introduced a merged dataset drawn from multiple known benchmark datasets. Although a key step towards datasets that facilitate domain generalisation, some issues remained. An early study [Chen et al., 2019b] found that when performing domain transfer from miniImageNet to the CUB-200-2011 dataset, using simpler methods such as ProtoNets or their own baseline had an advantage over meta-learning optimisation methods. “A Broader Study of Cross-Domain Few-Shot Learning” [Guo et al., 2020a], is a work meant to discover why this is the case.

The authors of the study point out that the existing benchmark datasets mostly represent natural scenes and incorporate perspective, which may not be the case in a

real world domain-specific dataset. They proceed to rank dataset similarity based on three criteria: perspective distortion, semantic data content and colour depth. Their data comprises images from datasets where these aspects vary substantially, such as x-rays [Wang et al., 2017] and satellite data [Helber et al., 2019]. Their also work evaluates common FSL methods and confirms that ProtoNets [Snell et al., 2017] often outperform more complex methods such as Relation Networks when evaluated against domain-specific datasets.

Cross-Domain adaptation is still an area of current research, since the change in distribution affects most approaches investigated so far. This includes the pseudo-labeling methods explored in the previous section [Li et al., 2023a].

Authors of applicative papers will often make use of networks pre-trained on relevant datasets, such as large satellite imagery datasets [Sun et al., 2021b] for remote sensing applications, so while enhancing the potential for domain transfer is important, it may not be paramount in the real world depending on the task.

## 2.8 Summary of Results

We have condensed the accuracy of most methods summarised so far in Table 2.1 below. Only those that attempted the mini-ImageNet benchmark are included, as it is a reasonably hard problem and commonly used in the field.

Transductive methods have been excluded since they may not represent real-world conditions, as discussed in Section 2.6.1 - Transductive Learning. While some surveys [Song et al., 2022] include both inductive and transductive methods, we believe this is not a fair comparison, as inductive methods do not have access to the distribution statistics of test data.

When papers included test results using different backbones, we chose the ones with the closest depth to previous works, in order to close the gap between test settings.



| | Model | miniImageNet Backbone | miniImageNet 1-shot | 5-shot |
|---|---|---|---|---|
| Metric | Matching Networks [Vinyals et al., 2016] | 64-ch. conv x4 | 43.6 ± 0.8 | 55.3 ± 0.7 |
| | Prototypical Networks [Snell et al., 2017] | 64-ch. conv x4 | 49.4 ± 0.8 | 68.2 ± 0.7 |
| | Relation Networks [Sung et al., 2018] | custom | 50.4 ± 0.8 | 65.3 ± 0.70 |
| Generative | MetaGAN+RN [ZHANG et al., 2018] | custom | 52.7 ± 0.6 | 68.6 ± 0.7 |
| | R-SVAE [Xu and Le, 2022] | ResNet12 | **74.8 ± 0.2** | 83.3 ± 0.4 |
| NAS | M-NAS [Wang et al., 2020a] | Searched, 28K params | 51.4 ± 1.4 | - |
| | MetaNAS [Elsken et al., 2019] | Searched, 1.1M params | 61.7 ± 0.3 | 78.8 ± 0.2 |
| Miscellaneous | DiversityCooperation [Dvornik et al., 2019] | ResNet18 | 59.5 ± 0.6 | 76.9 ± 0.5 |
| | Meta-Transfer Learning [Sun et al., 2019] | ResNet12 | 61.2 ± 1.8 | 75.5 ± 0.8 |
| | LEO [Rusu et al., 2019] | WideResNet28 | 61.8 ± 0.1 | 77.6 ± 0.1 |
| | A Good Embedding (SD) [Tian et al., 2020] | ResNet12 | 64.8 ± 0.6 | 82.1 ± 0.4 |
| | UniSiam (SD) [Lu et al., 2022] | ResNet18 | 64.1 ± 0.4 | 82.3 ± 0.2 |
| | TransMatch [Yu et al., 2020b] | ResNet18 | 62.9 ± 1.1 | 82.2 ± 0.6 |
| | DeepEMD+Boosting [Zhang et al., 2021] | ResNet12 | 67.1 ± 0.8 | 83.1 ± 0.7 |
| | DeepEMD+Sampling [Zhang et al., 2020] | ResNet12 | 68.8 ± 0.3 | 84.1 ± 0.5 |
| | EASY [Bendou et al., 2022] | ResNet12 | 70.6 ± 0.2 | **85.7 ± 0.1** |

Table 2.1: Accuracy table of classification methods surveyed in this chapter, some of whom from Tian et al. [Tian et al., 2020]

We chose to separate the results of generative methods and Neural Architecture Search based methods, since the former rely on data generation and the latter generate custom backbones. However, it is clear that both are not intrinsically superior to other few-shot classification works.

Looking at the Backbone column, it is clear that deeper architectures are not necessarily the main factor in performance: the best methods, R-SVAE [Xu and Le, 2022] and EASY [Bendou et al., 2022], both relied on a simple ResNet12 backbone. We also note that, although the best methods for 1 and 5 shots are different, good performance with a number of shots correlates with good performance on the others.

### 2.8.1 Discussion

Since Few Shot Learning is a relatively novel field, a wide array of approaches have been investigated to solve the problem. That said, when researcher follow complex threads and discard simple baselines it may cause them to overlook more efficient methods [Liao et al., 2021]. One example was "A Good Embedding Is All You Need" [Tian et al., 2020], in which using a merged set of meta-training tasks and a simple linear classifier proved better than much more complex methods.

The usage of SSL pre-training could be instrumental in improving the performance of existing FSL classification methods, as shown by the results for UniSiam [Lu et al., 2022], TransMatch [Yu et al., 2020b] and EASY [Bendou et al., 2022] in Table 2.1.

However, it is important to keep the compared settings as close as possible. As an example, in Su et al. [Su et al., 2020], SSL was prospected as an improvement for ProtoNets. Their work found significant improvements in downstream classification task accuracy, but their work used an image size of 224x224 pixels while the original ProtoNets used smaller inputs (84x84). A reproducibility report [Ashok and Aekula, 2022] revealed that using the same image size as the original did not help boosting the accuracy of the existing method.

The image sizes for the results we chose in Table 2.1 are 84x84, same as ProtoNets, or very close to that.

While some of the methods mentioned in this chapter are no longer state of the art,

they still inform newer methods, and when employed in conjunction can improve them.

## 2.9 Conclusions

We have reviewed the most common approaches to few-shot learning invented in the classification context. Our findings may be summarised as follows:

- The most complex methods and architectures do not necessarily translate into better results.
- The earliest Metric Learning methods such as ProtoNets still play a part in newer methods such as EASY.
- Meta-Learning is a valuable tool and the most common in the field of FSC, but combining different methods yields better results. As shown by TransMatch and EASY, Transfer Learning and Unsupervised Learning are equally important.
- Generative methods can be helpful, but Variational AutoEncoders are cheaper and better performing than Generative Adversarial Networks since they're not restrained by the needs of human consumption.
- Some of the previous methods such as MAML should be considered separately due to the assumption that test data will be presented in batches (known as Transductive Learning).

Some of the methods mentioned in this chapter will resurface in Chapter 3, Few Shot Object Detection.

# Chapter 3

# Few Shot Object Detection

Many of the previously discussed methods in the field of Few Shot Classification, such as meta-learning [Kang et al., 2019] and distillation [Ma et al., 2023], have also been employed in Few Shot Object Detection. However, there are important differences dictated by the end goal.

In object detection, there may be more than one class instance for each image, and the need for localisation precludes simply treating the entire image as a vector without storing any additional information. Additionally, given the possibility of multiple objects in the same image, a model should be able to produce variable length output. [Liu et al., 2023].

While these requirements make the task more challenging than classification, object detection opens up new possibilities for real-world applications.

This chapter aims to provide an overview of Object Detection benchmarks and techniques, and how they have been adapted to solve Few-Shot Object Detection tasks.

## 3.1 Benchmarks

This section details the most common benchmarks for Few-Shot Object Detection. This is important as not only the methods, but the benchmarks themselves have evolved over the past decade, both in large scale and few-shot settings, so this might shed some clarity on the performance measures cited in this chapter.

### 3.1.1 Datasets

One of the earliest benchmark datasets for object detection was PASCAL Visual Object Classes (VOC) [Everingham et al., 2010]. Its associated research challenge ran from 2005 to 2012, during which the dataset and annotations underwent refinement. The last iteration (2012) consists of 11.5k images with annotations for 20 classes.

For few-shot applications, the standard way to use VOC is to split it into 15 "base" classes with abundant data and 5 few-shot "novel" ones. To avoid bias against certain classes, FSOD method are usually evaluated against 3 different base/novel class splits.

The VOC dataset is still widely used, although new and more challenging datasets have been built since then. The most widely known one is Microsoft Common Objects in Context (COCO), created for both classification and object detection. Containing 200k annotated images, it is a more extensive and diverse dataset than VOC. It contains 80 classes representing "things" with defined shapes and 11 background classes such as "sky".

FSOD works evaluating against COCO use the 80 classes with defined objects, split into 60 base classes and 20 novel ones. Since the number of classes and labeled samples is higher than VOC, no other class subdivisions are used.

As shown in Figure 3.1, COCO has another advantage over VOC: while the VOC dataset contains many images with a single object, making it less resilient to interference from the background or objects belonging to other classes, COCO tends to contain more than one type of object per image, making for a more challenging yet far more realistic setting.

A recent paper [Tong and Wu, 2023] analysed both benchmark datasets in the context of small object detection. They found that both benchmark datasets include labeling errors, such as missing annotations or wrong ones, which can have a noticeable impact on performance. However, to ensure consistency with previous works, we used the same benchmarks.

In the field of Few-Shot Classification, hundreds of tests on different splits are performed and their results averaged, in order to avoid new methods overfitting to the specifics of a given data split. While Few-Shot Object Detection is a more complex

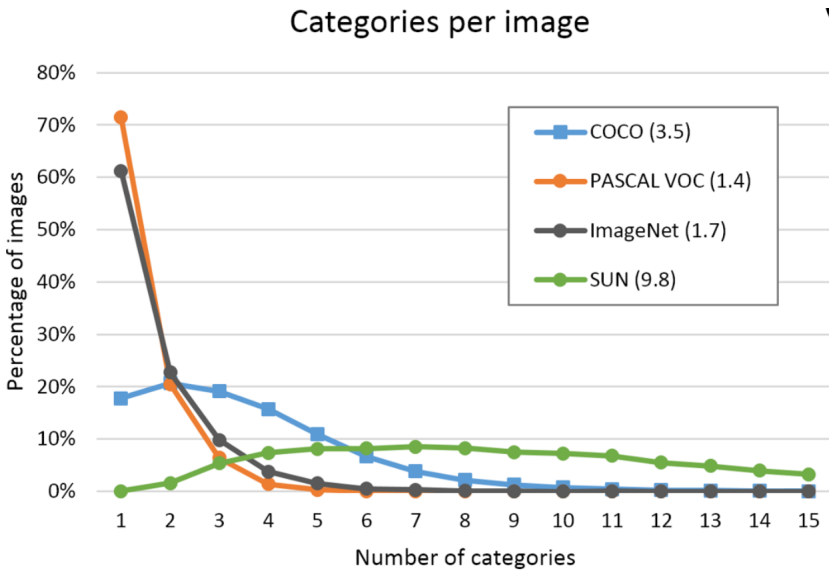


Figure 3.1: Number of categories per image in COCO vs in other datasets. Diagram from the original COCO paper [Lin et al., 2014]. SUN [Xiao et al., 2010] and ImageNet are classification datasets, which were included simply because COCO could be used for classification as well as detection.



task with multiple outputs, where picking split classes at random would cause too much variance [Sun et al., 2021a], when the evaluation subset is fixed there is still a risk of rewarding adaptations that suit a specific subset. In the first FSOD works, models were evaluated on the same data split [Kang et al., 2019], but the work of Wang et al. [Wang et al., 2020b] introduced a number of new data splits. Today, methods are evaluated on 10 different fixed splits from that work on both COCO and VOC. VOC is still evaluated on 3 different class splits as well, meaning that for each number of shots 30 experiments are performed on VOC and 10 on COCO.

### 3.1.2 Metrics

Since the output of an object detection network is not a single class, but a variable length array of regions with respective classes, accuracy is not a viable metric. **Intersection over Union** (IoU) is a fundamental localisation metric, which simply denotes the area of a proposal which intersects the true labeled box, divided by the area union of both boxes. This rewards a network for reaching closer to the ground truth and avoids excessive penalties for having slightly different boundaries. The area under the

precision-recall curve for the task is evaluated at different IoU thresholds, since they will impact the shape of the curve: a threshold of 0.5 will find more positives but also generate more false positives when compared to a IoU of 0.95. This concept is illustrated in Figure 3.2: the prediction overlaps only 53% of the combined ground truth and prediction boxes, so while an IoU threshold of 50% will ensure it is considered in the evaluation, a threshold of 60% will exclude it.

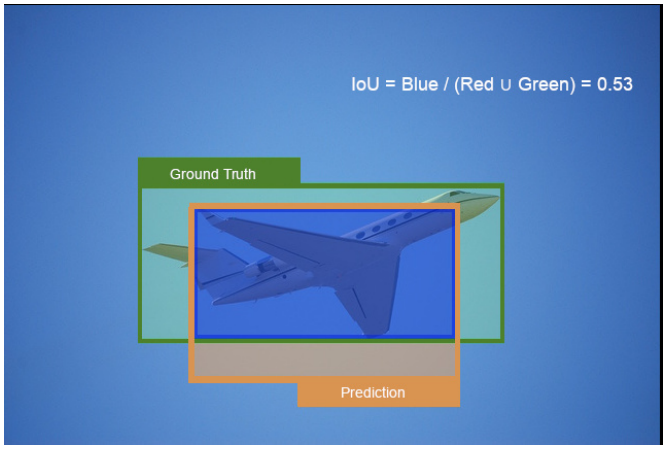


Figure 3.2: IoU illustrated. The green area is the ground truth, the orange area the prediction, the blue one the intersection.

In the series of **Pascal VOC** challenges [Everingham et al., 2010], average precision was evaluated over 11 recall thresholds in increments of 0.1, from 0 up to and including 1. The IoU threshold was fixed at 0.5 yielding what was termed Average Precision (AP). The mean of AP across all classes is known as **mean Average Precision** (mAP). However, this metric was already considered dated by 2018 [Redmon and Farhadi, 2018], due to the fact a fixed IoU threshold of 0.5 tends to reward improper placement of boxes.

This flaw led to a more refined metric for the **COCO** challenge [Lin et al., 2014]. For COCO, the old VOC mAP metric was modified to account for the intersection area of predicted and labeled boxes (IoU). Instead of being evaluated at 0.5 IoU, Average Precision is evaluated across 10 IoU thresholds, from 0.5 to 0.95 with an interval of 0.05. According to the authors, this rewards models with better localisation capabilities.

Practically, the mAP of a method is usually reported, calculated according to the

challenge, together with the basic VOC AP at 0.5 IoU and 0.75 IoU, to help readers visualise how well a network can cope with localisation requirements. In the case of Generalised Object Detection, the results usually include a breakdown of the metrics mentioned (mAP, AP0.50, AP0.75) for both base classes and novel classes to show how well a model can retain performance on base classes while learning new ones.

## 3.2 Architectures

Few-Shot Object Detection works usually start from the same network architectures as standard Object Detection [Liu et al., 2023] and modify them for the task at hand. For the purposes of background research, this section will explore the two most commonly adapted architectures: R-CNN [Girshick et al., 2014] and YOLO [Redmon et al., 2016].

### 3.2.1 R-CNN

Object detection as a task comprises two tasks: **object localisation** within an image and **object classification**. The traditional way is to solve it in two stages: filtering regions which are likely contain an object, and classifying them. That is the approach taken by the original **R-CNN** [Girshick et al., 2014]. It first extracted regions with a selective search algorithm, like in previous works [Uijlings et al., 2013], used a CNN to extract features from each region, then had Support Vector Machines [Burges, 1998] classify them. This approach yielded better results than previous algorithm-based methods [Lowe, 2004] but it was still quite slow, taking 13s per test image on their original setup.

Fast R-CNN, building upon R-CNN, did not run the CNN on every region, but across all regions, pooling them. It also used a SoftMax classification layer instead of SVMs. It was a substantial improvement in terms of speed and average precision. The original Fast R-CNN architecture is shown in Figure 3.3.

Faster R-CNN [Ren et al., 2017] (**FRCN**) used a CNN to generate the Regions of Interest, termed a Region Proposal Network (RPN), eliminating the efficiencies of Selective Search. Most of the convolutional weights of the feature extractor are shared

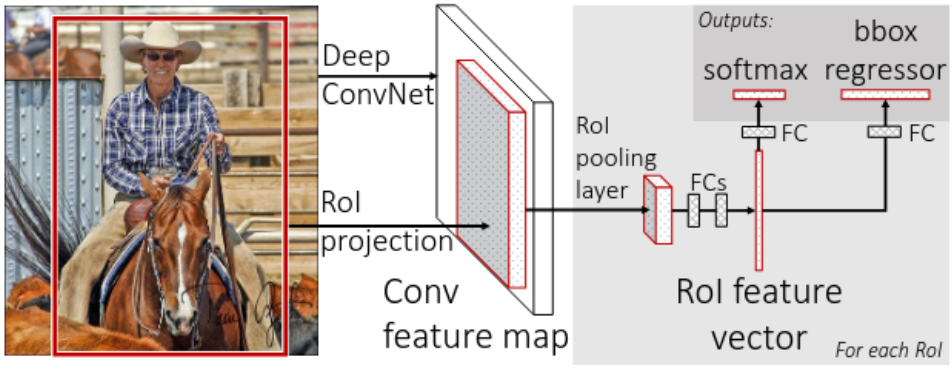


Figure 3.3: Fast R-CNN by R. Girshick (2014) [Girshick et al., 2014]

with the RPN. This architecture is still often used as a baseline. Another important contribution was Non-Maximum Suppression [Hosang et al., 2017], a method which discards overlapping predictions based on their "confidence" score, or how certain the network is of a prediction. Doing this step through a neural network rather than algorithmically increased performance further.

However, despite the high average precision of methods in this family, it is less common when speed is of the essence: since the feature extraction backbone and the RPN have to run sequentially, even Faster R-CNN is quite slow compared to other approaches.

### 3.2.2 YOLO

Unlike the R-CNN family of models, the You Only Look Once (YOLO) [Redmon et al., 2016] method is a single stage method: it combines both region proposal and classification into a single network, processing the entire image in a single step. The conceptual diagram in Figure 3.4 shows the bounding box regression and classification being performed at the same time.

While much faster, single-stage methods used to be less accurate than R-CNN and its enhancements. This changed over time as new versions of YOLO were researched. YOLOv2 [Redmon and Farhadi, 2017] used an entirely convolutional network, instead

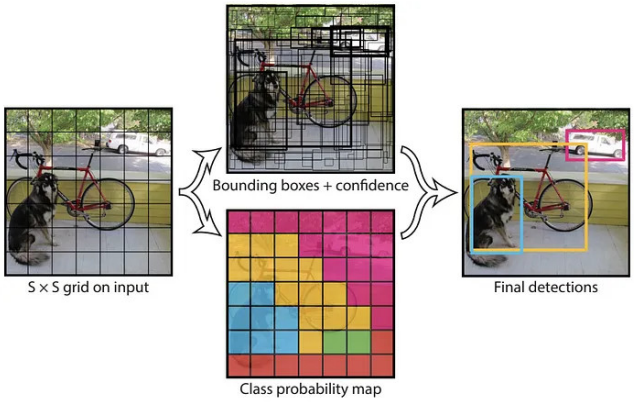


Figure 3.4: You Only Look Once by J. Redmon [Redmon et al., 2016]

of fully connected layers, which helped preserve spatial information. YOLOv3 [Redmon and Farhadi, 2018] had three separate convolutional branches after the initial backbone, called a Feature Pyramid Network (FPN). The convolutions of those branches had different filter sizes to better detect objects of different scales.

Research in the YOLO family of networks also spurred improvements in augmentation strategies, one of the most notable ones being "mosaic" augmentation, where four different images are stitched together, which reduces the risk of overfitting when performing the task of background separation. After many iterations, the current YOLOv7 [Wang et al., 2023] can match the performance of two-stage R-CNN-like methods while achieving a lower inference time. However, adoption has been slow in the few-shot line of research, for reasons we'll explore in the next section.

### 3.2.3 Few-Shot Architectures

In one of the latest surveys on FSOD [Liu et al., 2023], most of the papers cited rely on Faster R-CNN, with just a few using YOLO, even though YOLO-based methods are fairly common in the wider field of Object Detection. During this research, we had to take notice of the fact that recent YOLO-based works often lack few-shot benchmark comparisons or publicly available code, while comparisons between open-source Faster R-CNN based methods are easier to perform. Most YOLO-based works in this field have relied on an early meta-learning work [Kang et al., 2019] which used YOLOv2. One

likely reason is that the pressure to publish leads researchers towards existing few-shot codebases and frameworks, which are skewed towards Faster R-CNN. Furthermore, the separation between region proposal and detection has likely helped researchers engineer targeted few-shot improvements. One such improvement is discussed in Section 3.4.3. The survey mentioned [Liu et al., 2023] also makes the point that adapting architectures from large-scale models may not be enough, and thus building an architecture meant specifically for few-shot scenarios could be a helpful avenue of future research.

## 3.3 Meta-Learning

Meta-Learning, having been developed and proven popular on few-shot classification tasks (Section 2.4), has been adopted in the field of Few-Shot Object Detection as well. While our work is not based on a meta-learning approach, it is worth discussing some related developments since they will be part of the final performance comparisons.



### 3.3.1 Few-Shot Feature Reweighting

The first meta-learning method for FSOD was "Few-shot Feature Reweighting" (FSRW) by Kang et al. [Kang et al., 2019]. It relies on two components: a meta-feature learner and a reweighting module. During the meta-training phase, base classes are divided into multiple pre-training sets of support and query images, according to the episodic meta-learning paradigm we mentioned in Section 2.3.3. The network learns to extract generic features and then re-weigh their importance based on a few support samples and a query sample. Finally, in the meta-testing phase, the network is trained this way on the real novel few-shot data. The architecture is shown in Figure 3.5.

This work was a marked improvement over naïve fine-tuning, and it still remains of the few to be based on YOLO. It also introduced the very first few-shot data splits for VOC and COCO, used for meta-testing. So, while since superseded, this work left a blueprint for subsequent meta-learning methods to follow.

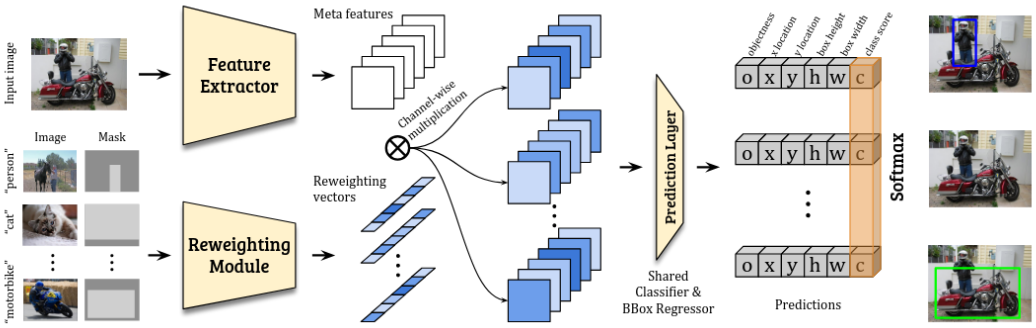


Figure 3.5: FSRW diagram by Kang et al. [Kang et al., 2019]. Note the shared classifier-regressor from YOLO and the additional reweighting module.

### 3.3.2 Subsequent works

Soon after FSRW, a meta-learning work with a similar setup was "Meta R-CNN" by Yan et al. [Yan et al., 2019]. They recognised that the detection setting presented an additional challenge due to the need to discard background features, so instead of performing meta-learning over full image features, they performed it over Region of Interest features, which in Faster R-CNN are extracted by the Region Proposal Network. This achieved a substantial improvement over FSRW.

Most works still tend to limit changes to the original Faster R-CNN architecture. One of the exceptions was "Meta Faster R-CNN" [Han et al., 2022a], not to be confused with Meta R-CNN. They tackled the RPN, which outputs region proposal coordinates as well as an "objectness" score or likelihood that they contain an object. They changed it to use prototype-based matching instead of just a linear layer for the "objectness" score. They also trained two detection heads, one for the base classes and one for the novel ones, to keep base class from interfering with the novel ones in the classifier. They improved upon both FSRW and Meta R-CNN.

Variational Feature Aggregation (VFA) [Han et al., 2023] reprises the same concept of image feature aggregation as FSRW from a different perspective. VFA aggregates features without distinguishing between classes, then uses a Variational Auto-Encoder pre-trained on base classes to pick important features for novel classes. The use of VAEs in this context has been covered in Chapter 2.1.2. When compared to the state

of the art methods, including fine-tuning methods, VFA achieved the best performance on VOC but weaker performance on COCO.

While meta-learning has been an important avenue of research in the field of FSOD, it has not produced the best performing methods. A direct comparison is available in Section 3.8 - Results. The next sections will discuss alternative methods.

## 3.4 Fine-tuning

From what we've seen in Chapter 2 - Classification, network fine-tuning by itself is often not enough to overcome data scarcity, and it is usually paired with an additional pre-training phase involving Meta-Learning or Self-Supervised Learning. In the Few Shot Object Detection field, localising objects requires contextual and spatial information which is not necessarily bound to specific classes: this is particularly evident in the Faster R-CNN architecture we summarised in Section 3.2, which uses a class-agnostic Region Proposal Network. Therefore, there exists a large body of work dedicated solely to improving fine-tuning performance for a single comprehensive training episode as usual, without the use of Meta-Learning.

### 3.4.1 Early Methods

One of the first works which focused on improving transfer learning in FSOD without making use of meta-learning was "LSTD: a Low-Shot Transfer Detector for object detection" [Chen et al., 2018]. Starting from Faster R-CNN, they made various changes to the architecture. One of their improvements was to share the weights for localisation regression across all classes, a strategy employed by an older family of detection networks called Single Shot Detector [Liu et al., 2016]. Starting from the same pre-initialised weights for novel classes meant they were less likely to overfit the feature output of the few samples. They also changed the classifier to be entirely convolutional, running over Region of Interest features instead of a fully connected layer as in Faster R-CNN.

However, while the performance of LSTD was better than simply fine-tuning Faster R-CNN, the changes the authors made to the architecture were not picked up by later

works, which preferred to start from a clean slate and examine each performance factor separately. Additionally, since their evaluation protocol was not used by later works, we will not include it in the final comparison of results.

In "Multi-scale Positive Sample Refinement" (MSPR) [Wu et al., 2020] the authors focused on the difference in the distribution of object scales between models trained on large datasets and few-shot datasets. Detection networks are likely to miss samples which appear at different scales than the novel few-shot set, so MSPR used branches with different convolution sizes to process features at multiple scales. The work was based on Faster R-CNN and achieved an improvement over FSRW, which was based on YOLOv2, but the idea of a branching Feature Pyramid Network (FPN) with different convolution sizes also appeared in YOLOv3 in a large-scale standard context (see Section 3.2.2). Works based on later Yolo versions [Ouyang et al., 2023] incorporate it by default. However, when comparing any proposed method to previous Faster R-CNN based works, a FPN is often considered an extension rather than a constituent component [Qiao et al., 2021].



### 3.4.2 Two-stage Finetuning Approach

In "Frustratingly Simple Few-Shot Object Detection" [Wang et al., 2020b], the authors created a baseline called Two-stage Finetuning Approach (TFA). The network is first trained on a large dataset consisting of base classes, then most of the network including the feature extractor and RPN is frozen. The last parts of the RCNN network, which are the box regressor and classifier, are fine-tuned on a smaller few-shot dataset consisting of novel samples mixed with base ones.

This simple baseline out-performed many contemporary meta-learning methods as well as LSTD, and while the method has since been superseded, it provided a good baseline and yet another reminder that the most complex methods aren't necessarily the most effective. The freezing of the feature-extracting backbone limited the network's potential to learn novel classes, and this was addressed in a later work named DeFRCN [Qiao et al., 2021], which we'll discuss in Section 3.4.3.

TFA also improved the existing evaluation protocol, introducing more few-shot data

splits to reduce variance between runs and thus making comparisons more meaningful. TFA also started the practice of reporting base class AP instead of just few-shot novel class AP, thus paving the way for Generalised Few-Shot Detection.

### 3.4.3 DeFRCN

The authors of Decoupled Faster R-CNN (DeFRCN) [Qiao et al., 2021] examined the usual Faster R-CNN architecture and pointed out that the RPN is class-agnostic, as opposed to the classifier, but while their tasks are different both their gradients flow into the convolutional backbone, which creates a conflict. They decoupled the RPN (hence the name) by stopping RPN gradients from being propagated to the backbone. This improved classifier regularisation by decoupling classification and box prediction. The backbone and RPN are trainable, instead of being frozen as in TFA.

The authors also employed a prototype-based method as an add-on called Prototypical Calibration Block (PCB): the existing novel samples were passed through a ResNet backbone, and the resulting features combined with the predicted ones at inference time, which slightly increased the mAP of novel classes in all scenarios. The changes to the network are shown in Figure 3.6.

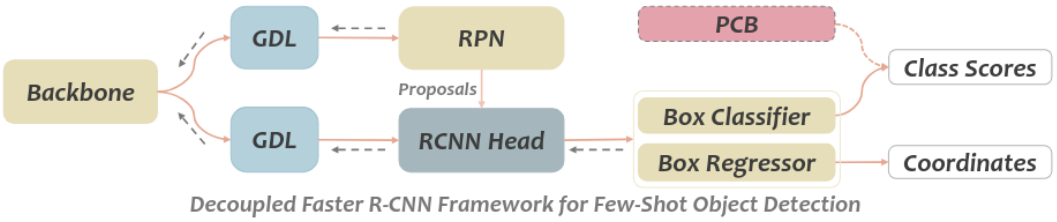


Figure 3.6: DeFRCN diagram by Qiao et al. [Qiao et al., 2021], with GDL denoting gradient-decoupling layers, PCB the prototypical calibration block, and yellow boxes denoting trainable modules.

As a note, even though PCBs are created from all the novel training samples, they only need one test sample and are thus not considered transductive learning.

DeFRCN has proven to be a versatile baseline for other works [Ma et al., 2023] [Guirguis et al., 2022], since it doesn't affect the loss function or training regime, leaving

room for further improvements. It inherits the same fine-tuning process as TFA of mixing base and novel samples in the last stage, as well as the same evaluation protocol over at least 10 data subsets, which makes comparisons reliable. These reasons are why we've decided to use DeFRCN as a starting point for our work on CL training regimes.

One of the works which used DeFRCN as a baseline was by Lin et al. [Lin et al., 2023]. It's a data augmentation approach which crops novel instances and pastes them onto base images. It was inspired by other augmentation methods such as YOLOv4's mosaic, but it requires multiple stages. Since some base images might contain unlabeled instances, they are sent by a vision-language model for checking, and the space in which the crops might be pasted is taken into account. The result is a net improvement over the baseline.

DeFRCN was also used as a baseline for Constraint-based Finetuning Approach (CFA) [Guirguis et al., 2022], a gradient manipulation method devised to avoid catastrophic forgetting of base classes in a Generalised FSOD setting. We will discuss gradient correction methods extensively in Chapter 4 - Continual Learning.



## 3.5 Other Approaches

This section discusses various methods adopted in FSOD which can be seen as orthogonal to the previously discussed fine-tuning or meta-learning approaches.

### 3.5.1 Contrastive and Class Separation Methods

Margins are known as the distance between a class decision boundary and the closest data points. This is a concept which we've touched upon in Section 2.3.2 - Contrastive Learning. The aim behind increasing margins is to keep class representations as far apart from each other as possible to improve performance on unseen samples, which are unlikely to follow the exact same distribution as the seen ones.

An important work in this part of the field was "Few-Shot Contrastive proposal Encoding" (FSCE) [Sun et al., 2021a]. They added another branch to the detector to measure the similarity of object proposal encodings. This then added a new loss

term to the network which encouraged keeping closer representations of objects from the same category within a mini-batch, while separating objects from other classes.

FSCE adapted the contrastive learning formulation to the detection setting by comparing proposals only if their IoU is greater than a certain threshold, to discard irrelevant comparisons. Although no longer competitive, this method achieved better results than TFA.

Another notable contribution to FSOD, and a contemporary to FSCE was "Class-Margin Equilibrium" (CME) [Li et al., 2021a]. This paper also uses an additional contrastive branch. It focused on both on the need for maximising margins, as well as the need for a minimum margin in the loss, to keep the class representations compact so they could make room for novel classes. Balancing these two loss objectives is the main part of their strategy. The contrast between these two objectives is intuitively shown in Figure 3.7.



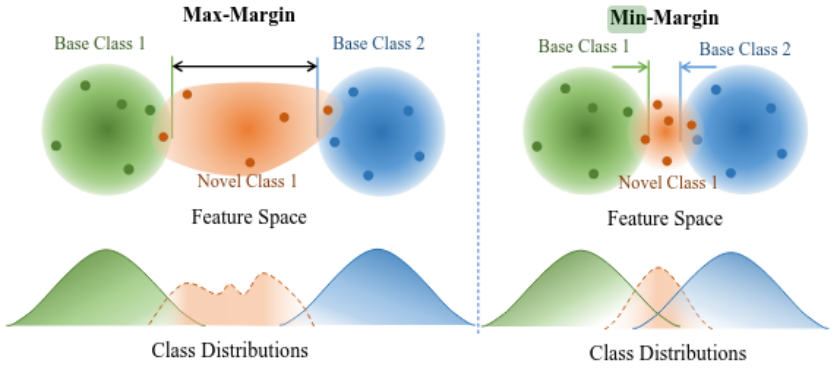


Figure 3.7: Min-max margin diagram by Li et al. [Li et al., 2021a]

CME training has another improvement to increase contrast between samples: since it is based on FSRW, it has access to a meta-learned support image "mask" which highlights relevant features. The top 15% of pixels in this mask are set to 0 to stop the network from relying too much on the same features, which can be seen as a form of dropout.

CME achieved better results than both FSCE and TFA on the COCO dataset, a benchmark more challenging than VOC. While CME is no longer state of the art, it performed better than its contemporaries and it could work with both YOLO and

Faster R-CNN architectures.

Adapting contrastive learning to FSOD is still an active area of research. "Positive Sample Improvement" (PSI) [Ouyang et al., 2023] adopts a number of data augmentations as well as what could be described as a contrastive loss to aggregate samples with same-class labels and separate differently labeled ones, including a hyperparameter to control the degree of separation. This work is also valuable since it adopted YOLOv4 as a baseline, showing the possibilities in focusing on improving a single-stage detection architecture.

"Swapping Assignments between Views" (SwAV) [Caron et al., 2020] uses the same contrastive learning principles as SimCLR (see Section 2.3.2) to reward consistency between multiple augmentations of the same image. Ultimately, the DeFRCN authors ran additional tests with various backbones including a SwAV-trained one. While SwAV requires lengthy self-supervised pre-training, they found it could be used with DeFRCN's Prototypical Calibration Block to improve performance further [Qiao et al., 2021].



### 3.5.2 Distillation methods

Knowledge Distillation is the practice of transferring knowledge from a teacher model to a student model, which we talked about in Section 2.5.2. We also mentioned Self-Distillation (SD), where student and teacher are the same model with different random initial weights.

Self-Distillation is one of the main components of "Discriminative Geometry-Aware Learning" (DiGeo) [Ma et al., 2023]. The first innovation of the paper was to use a property known as "Neural Collapse", which is the natural tendency of a fine-tuned network to form an Equiangular Tight Frame towards the end of training [Kothapalli, 2022]. This means all classes become equally distant prototype vectors in the last layer of a classifier, with the final output being determined by which is the closest vector. See Figure 3.8 for an example.

This trend has been exploited in recent works including DiGeo to keep class representations apart, usually through modified loss functions. The objective of Neural

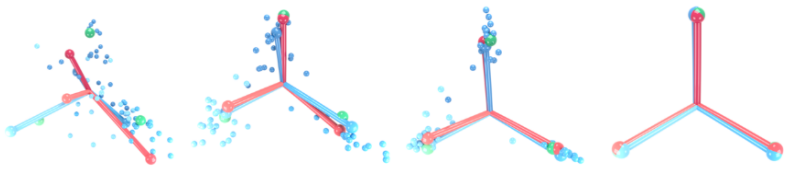

Figure 3.8: Neural Collapse diagram from Kothapali et al. [Kothapalli, 2022] The blue balls represent final layer activations, red vectors the final layer classifier.

Collapse is similar to that of contrastive learning (see Section 2.3.2), but less computationally expensive since it does not have to rely on pair-wise comparisons of samples, but rather samples to class prototypes, which will have to be equidistant.

Instead of having a base-training and novel fine-tuning step, DiGeo trains the network from scratch on the full set, while increasing the probability of picking novel samples to avoid forgetting them. The class margins for the ETF are adjusted through self-distillation to reduce variance. When applied on top of DeFRCN, the DiGeo strategy manages to raise base class performance without hurting novel class performance, earning it a top spot in the task of Generalised FSOD. While applicative works often choose methods that only require changes in the fine-tuning step to speed up training, this full-step method is certainly worth considering.

Another interesting application of the distillation principle was "Multi-Faceted Distillation of Base-Novel Commonality" [Wu et al., 2022a]. This method measures the similarity between novel samples and a shifting subset of base ones. Classification and localisation related similarities are stored separately, as well as the variance of class features. Just like classic distillation, a new term is added to the loss which depends on the distribution of the output logits, here compared between the novel classes and the most similar base ones.

Multi-faceted Distillation is based on DeFRCN but the same principle can be applied to other methods. It achieved an improvement in novel class results over the baseline, although we should note that the presence of a large domain shift between base and novel sets could hamper this strategy.

### 3.5.3 Retentive R-CNN

Retentive R-CNN [Fan et al., 2021] seeks to solve the Generalised FSOD problem, that is to keep base class performance intact while learning few-shot novel classes. The authors studied TFA and found that the RPN rather than being class-agnostic was biased towards base classes. The resulting method to fix this is essentially an ensemble method. The original network, trained on base classes is frozen. Another branch of the RPN is created to be fine-tuned on novel classes. The region proposals are combined during the Non-Maximum Suppression stage. The proposals are then passed to separate classification and regression heads, and outputs are jointly evaluated in the last layers of the network. The resulting architecture is shown in Figure 3.9.

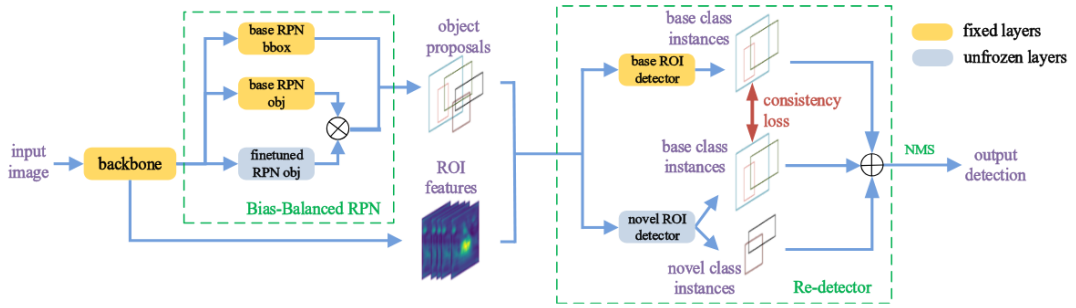


Figure 3.9: Retentive R-CNN diagram by Fan et al. [Fan et al., 2021]

This approach has been adopted as an optional add-on in various works such as DeFRCN [Qiao et al., 2021], yielding a consistent performance boost if the objective includes detecting base classes. However, the number of trainable parameters as well as the inference time increases substantially when compared to the usual single branch network approach. Since this clashes with one of our original objectives (see Section 1), we will not make use of ensemble methods such as this one.

## 3.6 Transformer-based methods

Transformers are an architecture consisting of an encoder-decoder network. The encoder typically processes the image to extract important features, compressing the

input into a more manageable representation, while the decoder turns them into a desired output, in this case spatial information. Transformers are usually pre-trained on a self-supervised task, such as the jigsaw puzzle task we encountered in Section 2.6.

### 3.6.1 Transformers and Object Detection

Transformers have been successfully adapted to the Object Detection task in "Detection Transformers" (DETR) [Carion et al., 2020]. Figure 3.10 provides a high-level overview of DETR. The initial features are extracted by a CNN backbone, as usual. However, the detector works differently to the approaches we've reviewed so far. The encoder turns features into "queries", which attempt to look for objects. An attention mechanism highlights the important embeddings in context, and the query results are matched with the labels. We remark that while region predictions are similar to the Faster R-CNN ones, they are treated differently. The network does not perform Non-Maximum Suppression to remove irrelevant matches: rather, it uses a the Hungarian cost-assignment algorithm to match the predictions to the labels. After they're matched, loss is calculated via IoU and L1 distance (sum of absolute differences) for classification and localisation.

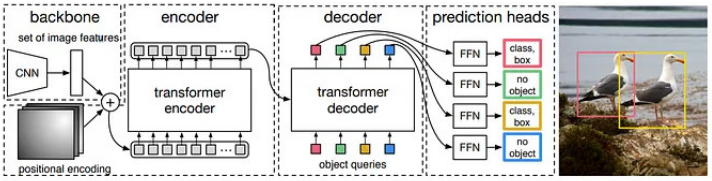


Figure 3.10: End-to-End Object Detection with Transformers diagram Carion et al. [Carion et al., 2020]

However, transformers need a extremely long training phase, and they tend to overfit on scarce data to their large number of parameters. Due to these factors, a number of hybrid methods have been developed which combine CNN feature extraction with transformer-based detection in few-shot learning.

### 3.6.2 Transformers and Few-Shot Object Detection

One of the first methods to adapt DETR for a low-data setting was "Unsupervised Pretraining DETR" [Dai et al., 2021], which used a CNN backbone to extract features, froze the backbone and trained transformers to turn the features into detection targets. They emphasized that for the transformers to work, the backbone must be frozen and image patches should be assigned to multiple queries, since multiple objects can be contained in an image. UP-DETR proved that less data-hungry transformers were possible in an Object Detection context, but it was not squarely aimed at a few-shot setting.

DETReg [Bar et al., 2022] followed up by using different unsupervised pre-training tasks: an object localisation task and one to predict object embeddings from another network. The authors also used a shallower backbone compared to CNN methods such as TFA or DeFRCN to minimize overfitting. While the performance on 1-5 shots was somewhat worse than previous fine-tuning methods, it performed much better on 30 shots.

One of the advantages of transformers is their versatile embeddings, which means they do not necessarily require fine-tuning (see Fully Cross Transformer for an example) [Han et al., 2022b], albeit regular fine-tuning methods such as DeFRCN still yield better performance.

We will not be using transformer-based backbones, in order to ensure fair comparisons with previous fully CNN based methods. It will also help us stay well on track with one of our initial objectives: to minimise deployment requirements. Networks with CNN backbones tend to be smaller, and we believe that models which don't take up excessive resources can increase the potential of such models for edge deployment and large-scale adoption. We have also excluded large vision-transformer foundational models such as DINO [Caron et al., 2021] from our evaluation since we consider them out of scope for this work: their number of parameters is at least an order of magnitude higher than the methods we've discussed so far.

## 3.7 Reliability of Benchmark Data

The issue of mislabeling is well known, with vision datasets as far back as MNIST [LeCun et al., 1998] being affected [Zhang, 2017]. FSOD methods are usually evaluated against few-shot splits of the VOC and COCO datasets. Given both datasets also suffer from this problem, a number of methods have been devised to mitigate this problem. In this section, we also discuss the problem of pre-training leakage.

### 3.7.1 Label Quality

Labeling a detection dataset is a laborious task, so some objects might not be assigned a bounding box and classification label. While a minority of missing labels may not seem like a problem, it is harmful to the performance of a detection network: when a label is missing, the object might be sampled in a "negative" region proposal, as in part of the background, thus sending the wrong signal to the network.

To solve this, Gao et al. created an add-on method based on DeFRCN called Decoupling Classifier [Gao et al., 2022a]. The classifier is split into two parallel heads, one for positive samples and one for negative ones. Their method was successful in reducing the performance drop caused by missing labels. As pointed out in the open review of the paper [Gao et al., 2022b], the method deals with a data problem by changing the model, but it is a worthy contribution to this line of research.

In "Classification Refinement and Distractor Retreatment" (CRDR) [Li et al., 2021b], the authors use the pre-trained base class detector to find a subset of the base set that lacks "distractors", their term for unlabeled instances. The samples the detector is most confident about are retained and used to fine-tune the network. One of their other improvements is a "Few-Shot Correction Network" module that can make Faster R-CNN better at detecting edge objects, since CNN backbones are trained on ImageNet classification and thus biased towards objects being present in the center of the image [Szabo and Horvath, 2022]. It would be interesting to see how data augmentations such as mosaic compare to CRDR method with respect to correcting the issue.

### 3.7.2 Information Leakage

Data leakage is commonly defined as the presence of training data samples in the test set, which deviates from a real-world setting and erroneously increases performance. A less studied issue is the leakage of information between tasks: FSOD backbones are always pre-trained on ImageNet, but some of the few-shot novel classes in VOC and COCO are also present in ImageNet. This means the backbone is already biased towards collecting their features (such as in Figure 3.11), which is hardly representative of a real-world setting when it comes to specialised domains.

Figure 3.11: Example of 'redshank' from ImageNet vs 'bird' from the VOC novel training set (first subset)

The only work which tackles this in FSOD is "Semantic Relation Reasoning" [Zhu et al., 2021]. The authors re-trained the ImageNet backbone from scratch with some of the similar classes removed, such as "ox" due to its closeness to VOC's "cow" and "pelican" for VOC's "bird". This caused a performance drop in those classes. Their method creates class embeddings from a large corpus of text, then trains the object detector to match those embeddings. This work would be classed as multi-modal, given it requires complex text, which is why we haven't included it in the results, but we are glad to see this interest in more robust assumptions.

## 3.8 Results

We have collated the authors' published results from the most significant papers mentioned in this review of Few-Shot Object Detection.

The first set of results collated from this review is in Table 3.1. This uses the first

data split developed by the FSRW authors. It is known as "seed 0" in later works which incorporated it as part of the wider range of comparisons. We use these results for comparisons with legacy methods, since this single data split is affected by variance.

We have excluded methods that rely on extracting novel instances from the base dataset [Cao et al., 2022], since this means they cannot work in some real-world settings where there is no overlap between the base and novel set. We also excluded methods that rely on multi-modal information such as text, since that information may not always be available, as well as methods that deal with issues in the benchmarks themselves [Gao et al., 2022a] to err on the side of caution.

We did not include a column for the architecture, since in the vast majority of works a Faster R-CNN network with a pre-trained ResNet-101 backbone was used. We indicate any works that differ in this respect through the Model Name.

| Model Name | 10-shot | 30-shot | Method Type |
|---|---|---|---|
| Faster R-CNN (FT-basic) [Yan et al., 2019] | 6.5 | 11.1 | - |
| Faster R-CNN (FT-full) [Wang et al., 2020b] | 9.2 | 12.5 | - |
| YOLOv4 (FT-full) [Ouyang et al., 2023] | 10.2 | 15.6 | - |
| FSRW (YOLOv2) [Kang et al., 2019] | 5.6 | 9.1 | ML |
| Meta R-CNN [Yan et al., 2019] | 8.7 | 12.4 | ML |
| CME [Li et al., 2021a] | 15.1 | 16.9 | ML |
| VFA [Han et al., 2023] | 16.2 | 18.9 | ML |
| MSPR [Wu et al., 2020] | 9.8 | 14.1 | FT |
| TFA w/cos [Wang et al., 2020b] | 10.0 | 13.7 | FT |
| FSCE [Sun et al., 2021a] | 11.9 | 16.4 | FT |
| PSI (YOLOv4) [Ouyang et al., 2023] | 13.4 | 18.8 | FT |
| DeFRCN [Qiao et al., 2021] | 18.5 | 22.6 | FT |
| DeFRCN+CropPaste [Lin et al., 2023] | **20.3** | **23.1** | FT |
| DETRreg [Bar et al., 2022] | 13.7 | 22.6 | TH |
| FCT [Han et al., 2022b] | 17.1 | 21.4 | TH |

Table 3.1: Novel Average Precision on COCO, FSRW data split. Method types are FT=FineTuned CNN, ML=Meta-Learned CNN, TH=Transformer Hybrid

It is rarer to find works that deal with the Generalised FSOD setting, which is concerned with both base and novel class performance. Since most of them used the 10-split setting, we picked those results as a reference in Table 3.2.

| | 5-shot | | 10-shot | | 30-shot | |
|---|---|---|---|---|---|---|
| Model | bAP | nAP | bAP | nAP | bAP | nAP |
| Faster R-CNN [Yan et al., 2019] | 17.6 ± 0.9 | 4.6 ± 0.5 | 16.1 ± 1.0 | 5.5 ± 0.9 | 15.6 ± 1.0 | 7.4 ± 1.1 |
| TFA w/cos [Wang et al., 2020b] | 32.3 ± 0.6 | 7.0 ± 0.7 | 32.4 ± 0.6 | 9.1 ± 0.5 | 4.2 ± 0.4 | 12.1 ± 0.4 |
| TFA+R. R-CNN [Fan et al., 2021] | **39.3** ± ? | 7.7 ± ? | **39.2** ± ? | 9.5 ± ? | 39.3 ± ? | 12.4 ± ? |
| VFA [Han et al., 2023] | - | - | 30.9 ± ?* | 16.8 ± ?* | - | 18.4 ± ?* |
| DeFRCN [Qiao et al., 2021] | 32.6 ± 0.3 | 13.6 ± 0.7 | 34.0 ± 0.2 | 16.8 ± 0.6 | 34.8 ± 0.1 | 21.2 ± 0.4 |
| DeFRCN+CFA [Guirguis et al., 2022] | 32.8 ± 0.2 | **15.2** ± 0.5 | 34.0 ± 0.2 | **18.8** ± 0.4 | 34.6 ± 0.1 | **23.0** ± 0.3 |
| DiGeo [Wang et al., 2020b] | - | - | **39.2** ± ?* | 10.3 ± ?* | **39.4** ± ?* | 14.2 ± ?* |
| DeFRCN+DiGeo [Wang et al., 2020b] | - | - | 35.1 ± ?* | 17.1 ± ?* | - | - |

Table 3.2: Generalised FSOD results on 10 COCO data splits: base/novel Average Precision. Asterisk (*) indicates experiment was only performed on split 0, while question mark (?) indicates no Confidence Interval was published. Any confidence intervals are calculated at 95%.



We only reported results on COCO since it's the most challenging benchmark, and because in the case of VOC some papers only reported selective metrics from their results such as nAP50 or AP50, making head-on comparisons impossible. The final results in Chapter 7 include some additional metrics as reproduced by us.

## 3.9 Discussion

While compiling this table, we encountered a few challenges in making sure FSOD methods received a fair comparison. Even recent works still sometimes publish only the results they obtained from the old FSRW split, making it hard to switch to the more robust protocol introduced by TFA. While some authors ran the seed 0 benchmarks multiple times to reduce variance across runs, they did not include the standard 95% confidence interval. Additionally, while repeating the same experiment reduces variance caused by random initialisation it does not account for the quirks of the data split: the FSRW one is generally more conducive to better performance, likely owing to the

representativeness of its samples [Wang et al., 2020b]. To better understand how results can be significantly swayed by protocol as well as other factors, we included a study in Chapter 7 - Experiments.

Hyperparameters also play a factor, since better tuning may lead to better results: Table 3.1 shows a remarkable difference in basic Faster R-CNN performance can be achieved by training the network until the loss asymptote is reached (FT-basic vs FT-full).

In the same table, we can see that while methods based on meta-learning have come a long way, they still do not perform as well as the top fine-tuning based method, DeFRCN. At the same time, transformer-hybrid methods, which use a convolutional backbone with a transformer detector, still have a way to go but research in that direction is ongoing.

The sub-field of Generalised FSOD is narrower, with some methods focusing primarily on maintaining base class performance and mostly succeeding, and others providing better novel class performance. The choice of method for an applicative work would be determined by the balance of priorities, as well as how willing a team would be to adopt more complex training strategies such as DiGeo, which requires full training, the combination of Retentive R-CNN with other works, or CFA which doubles the training time for DeFRCN.



### 3.9.1 Architecture

We can see that almost all CNN methods have used Faster R-CNN as a base to modify for their work, while YOLO has fallen by the wayside. There appear to be a variety of reasons: first is the speed vs performance tradeoff long associated with that family of networks, even though that may no longer be the case with the latest versions [Wang et al., 2023]. Additionally, since time is often short in academia, authors may be tempted to use the released source code of existing influential works (such as TFA) when it is available, which will create a network effect. Further investigation on which methods can work with single-shot architectures would be useful, in order to achieve real-time processing speeds: it is very rare for a FSOD paper to include inference time

among their results.

## 3.10 Conclusions

After reviewing various approaches to FSOD, and discussing some of the task's peculiarities, here are some of the most important insights from this chapter:

- Most works use Faster R-CNN as a basis and add their own adaptations, but some (e.g. Crop-Paste [Lin et al., 2023]) have been inspired by later innovations in the field of detection.
- Methods that focus on fine-tuning achieve better performance than meta-learning, but given the leap in performance compared to early methods there seems to be room for improvement in meta-learning as well.
- Transformer-based and hybrid methods are worse than methods that allow fine-tuning, if you exclude foundational models which should be in their own league owing to their requirements
- Benchmarks have become more robust since the early days of the field, but adoption is still an issue. Research would also benefit from benchmarks which focus on classes which are very rare in real life as well, to avoid inadvertedly relying on information leakage.

The next chapter discusses various continual learning methods, and how they relate to the Generalised FSOD setting we mentioned.

# Chapter 4

# Continual Learning

Incremental or Continual Learning is a branch of machine learning tasked with learning from data which is not static, but can be expanded over time in batches. When data is ingested one by one rather than in batches, it is referred to as online learning rather than incremental learning. We are not going to consider that specific sub-field, since our work deals with batched training.

Van de Ven et al. [van de Ven et al., 2022] describe different types of incremental learning objectives as task-incremental, domain-incremental and class-incremental. Task-incremental learning is about designing models that adapt to novel tasks, domain-incremental is for adapting for different domains, and class-incremental is about adding novel classes after it was trained.

The key difference between task-incremental and class-incremental learning lies in how tasks are separated. As an example, in a task-incremental setting on the MNIST [LeCun et al., 1998] dataset the network has to distinguish between pairs of digits such as 1/2, 3/4 until 9. On the contrary, in a class-incremental setting the network learns to discriminate between digits 1 to 10.

Following this designation, the focus of our work falls under "class-incremental" learning, where new classes are added after the first training episode, but the network has to be able to discern between all of them without additional hints by the end of training. This is similar to a Generalised Fine-tuning setting from Chapter 3 where both novel and base classes have to be detected. In this chapter, we will focus on

methods which have either been devised with the class-incremental setting in mind, or have been compared in such a setting.

We will not delve into methods which modify the hyper-parameters of the optimiser such as Stable-SGD [Mirzadeh et al., 2020], since following the second research question from Chapter 1 - Introduction we aim to study the purported benefits of gradient manipulation methods under equal conditions, and in a model-agnostic way.

## 4.1 Benchmarks

Continual Learning datasets are generally split into multiple "tasks", with each task containing a subset of classes from the main set. In class-incremental batch learning, tasks are seen by the network sequentially in a succession of training sessions.

### 4.1.1 Datasets

One of the most widely used benchmarks is Permuted MNIST [Goodfellow et al., 2014b] an incremental MNIST split in which a fixed random proportion of pixels is changed. This benchmark can still be included for comparison to previous works, but it is regarded as an ineffective one due to the simplicity of the underlying dataset [Schwarz et al., 2018].

Other common benchmarks include splitting the generic CIFAR10 dataset incrementally, or CIFAR100 which includes the same data but uses finer-grained class annotations [Krizhevsky et al., ].

Later works [Saha et al., 2021] [Chaudhry et al., 2019b] make use of incremental splits of mainstream datasets such as Caltech-UCSD Birds (CUB) [Welinder et al., 2010] and mini-ImageNet [Vinyals et al., 2016]. We already introduced most of the datasets in this section including CUB and mini-ImageNet in Chapter 2 - Few Shot Classification.

However, there are problems when adapting a static dataset into an incremental one. Particularly since the classes in subsequent tasks don't necessarily have anything in common, it is possible to obtain competitive performance by simply training a network

from scratch on chunks of data from all classes, as shown by Prabhu et al. [Prabhu et al., 2020]. Datasets which map classes to multiple attributes which overlap between one another are more resilient to such trivial solutions. An example of such a dataset is Split AWA, an incremental version of the early "Animals with Attributes" [Lampert et al., 2009] dataset. A more recent example is Lin et al.'s "Continual LEArning on Real-World Imagery" benchmark [Lin et al., 2021]. This benchmark is better at testing a network's Forward Transfer and Backward Transfer, since data samples change gradually and they were captured at different points in time.

The uptake of such benchmarks with smoother transitions has been inconsistent across the field, with most works focusing on performance on Permuted MNIST and CIFAR100 [van de Ven et al., 2022]. Some new works have adopted incremental versions of miniImageNet and CUB [Masana et al., 2022] since they have higher complexity.

In our view the previous CL benchmarks, due to aforementioned abrupt class transitions [Lin et al., 2021], are quite similar to a Generalised Fine-Tuning setting, with a key difference: fine-tuning is composed of a base and a novel task, rather than an arbitrary number of tasks such as 5, 10 or 20. This is why when discussing the importance and effectiveness of CL methods, we will prioritise the previous well-known benchmarks such as CIFAR and mini-ImageNet.



### 4.1.2 Metrics

The set of metrics agreed for Continual Learning is generally consistent across works. The most important metric is **Average Accuracy (AA)**. This has been defined in two ways [Zhou et al., 2022]:

- Final/Last-Task Average Accuracy ($A_B$) is simply the average of all task accuracies at the end of the last training session.
- The mean of the Average Accuracy captured after every task, to capture performance across all tasks rather than only after the final one.

Other metrics include:

- **Backwards Transfer (BWT)** is the effect learning a new task has on older tasks. Negative backwards transfer is also known as Forgetting and fairly common in evaluations.

- **Forward Transfer (FWT)** is how well a network can perform future tasks based on the previously learned ones, also known as zero-shot learning.

When using integer task descriptors (i.e. 1 to 10) rather than more complex structures that allow for linking information across abrupt class transitions, forward transfer is impossible for class-incremental learning [Lopez-Paz and Ranzato, 2017] but useful in other areas of CL such as task-incremental learning or online learning.

We will be focusing on Average Accuracy (last task) and Backwards Transfer for the purposes of this research, since we do not make use of additional information in fine-tuning targets.



## 4.2 Network Expansion

When a network has to learn new tasks using the same weights, some of those weights are going to be inevitably overwritten. One solution is to allocate new weights every time a task has to be learned to minimize interference between tasks. This was leveraged by Rusu et al. in "Progressive Neural Networks" (ProgNN) [Rusu et al., 2016]. A new sub-module is allocated for each task, and the weights of previous tasks are frozen. However, the features learned from previous tasks are still accessible via new weight connections to maximise feature transfer between tasks. See Figure 4.1 for an illustration.

Since the principle behind ProgNN is quite versatile, they have been applied to reinforcement learning tasks as well as incremental classification tasks [Rusu et al., 2016]. That said, a drawback is the vastly increased network size, for both the new modules and lateral intra-module connections.

A proposed improvement was "Dynamically Expandable Networks" [Yoon et al., 2017], which selectively retrains important neurons and prunes redundant weights. It also adds new neurons to the network after a task only when loss falls under a certain

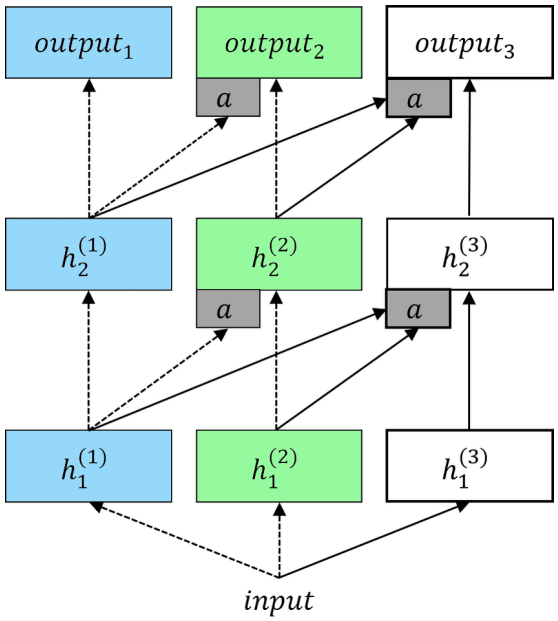


Figure 4.1: An illustration of a Progressive Neural Network [Rusu et al., 2016], where $h$ columns are added to the right for each new task and $a$ is a new set of weights for lateral connections between tasks. Diagram by Luo et al. [Luo et al., 2020]

threshold. While network capacity is still 50% more than the base network, the authors improved performance while reducing overhead compared to ProgNN.

Other approaches to restraining network module size include "PackNet" [Mallya and Lazebnik, 2018], which focused on pruning old weights before retraining to keep the same capacity, and later "Compacting, Picking and Growing" [Hung et al., 2019], which learned a mask for important weights and used iterative pruning after each task instead of weight sharing. "Feature Boosting" (FOSTER) used Knowledge Distillation after the expansion phase, a technique we already mentioned in Section 2.5.2.

A related approach is to train an ensemble of networks and specialising each one to avoid wasting computational power. This entails solving a similar problem of bounding the size of the combined networks. As an example, BatchEnsemble [Wen et al., 2020] was an interesting contribution, where the combined ensemble weights were expressed as the naive multiplication of a slow-changing matrix W and a fast-changing rank-one matrix F. The predictions of all ensemble members are averaged during inference.

In "Class-Incremental Learning with Strong Pre-trained Models" [Wu et al., 2022b],

the authors use an evaluation protocol which assumes most of the data is in the initial task, which is often the case in the real world. This protocol where a higher proportion of classes are initially learned has already been explored in model expansion methods [Zhou et al., 2022]. The authors hypothesize that a strong initial model can be adapted to other tasks with few changes and for each task, part of the backbone's initial weights are copied to a new model and fine-tuned on a new task. The classifiers are merged using confidence scores as well as concatenation. Their model performs quite well compared to other approaches in that setting, especially when there isn't a large domain shift between tasks.

A recent method, "Ensemble of Selectively Trained Experts" (SEED) [Rypeść et al., 2024] creates an ensemble of networks, but rather than training them all at once, it selects only one model to fine-tune on each task. The resulting model is an "expert" model since it focuses on a single task. The selection step is done in an initial training step for each task, and the chosen expert is the one where the new classes overlap the least with the old ones, measured with KL divergence. It achieved good performance on existing benchmarks while keeping the parameter count low by using a shallow feature extractor and sharing some of its layers between models. As acknowledged by the authors, shared parameters potentially hamper generalisation ability if the new classes are markedly different from the previous ones.

The network expansion area of Continual Learning is fairly complex, with research ongoing on how to best model a network architecture that can expand while keeping the memory cost reasonable. It should be noted that methods such as Progressive Networks do not seem to perform well on tasks without abundant samples [Chaudhry et al., 2019a].

## 4.3 Weight Regularisation

A way to stop catastrophic forgetting is to protect weights learned on previous tasks from major changes. As we've seen in Section 4.2, this can be done by freezing the important ones and pruning the rest. This kind of training is time and resource intensive, so regularisation methods seek to achieve the same aim by regularising weights during

training.

Elastic Weight Consolidation (EWC) [Kirkpatrick et al., 2017] protects previous task weights by issuing on each iteration a loss penalty, which is proportional to the deviation from previously learned task weights. This method was an important first step in this research direction. In Synaptic Intelligence (SI) [Zenke et al., 2017], the network's weights are not just scalar values, but they also track past values and keep a running estimate of their importance. The importance is computed during training as opposed to after each task as in EWC. Despite using a different strategy, this method performed similarly to EWC.

Hard Attention to the Task (HAT) [Serra et al., 2018] learns attention masks for each layer and applies them during training to protect previous task weights. It's a more direct approach than EWC, which uses a loss penalty. It might hamper parameter sharing between tasks, but it performed better than EWC in a task-incremental setting.

"Learning without Forgetting" (LwF) [Li and Hoiem, 2017] trains a classifier for each task and uses Knowledge Distillation: while learning a new task, it adds a loss term to keep the output logits related to old tasks close to what they were before learning a new task.

We note that most of these importance estimation methods introduce new hyper-parameters: for example, EWC has a $\lambda$ parameter that regulates the importance of weight deviations on loss, and LwF has a $\lambda$ parameter to balance logit distillation loss and new-task loss.

### 4.3.1 Gradient Manipulation for Weight Regularisation

Various regularisation methods have been studied which impact the direction of gradient descent rather than directly affecting the weights or changing the loss terms.

One of the first was Orthogonal Gradient Descent [Farajtabar et al., 2020], which stored gradient directions after learning every task, and projected new task gradients orthogonally to the previous ones. This method had the drawback of excessive memory requirements.

Gradient Projection Memory (GPM) [Saha et al., 2021] uses Scalar Value Decom-

position (SVD) to determine the most significant network activations and stores them in memory at the start. Then, on every training iteration, the gradient is projected orthogonally to those to reduce interference. This change made it much more efficient in terms of computation and memory management than OGD.

While GPM divides categorizes the subspace of previous tasks as important vs non-important and always projects orthogonally to that subspace, a newer work by Yang et al. [Yang et al., 2023b] stores multiple subspaces with varying degrees of importance to relax that constraint and improve learning on new tasks. This means the method does not necessarily perform a strict orthogonal projection on previous spaces, making it possible to better learn new tasks.

While regularisation-based methods have proven valuable in task-incremental learning, class-incremental learning is more complex than task-incremental learning due to abrupt class transitions. Regularisation-based methods usually struggle or fail on class-incremental tasks [van de Ven et al., 2022], so their usefulness might be limited for our broader fine-tuning objective. We come back to this point in Chapter 7.



## 4.4 Replay Methods

In task and class sequential CL benchmarks, the network is trained in succession on each task or class subset, and the previous dataset may be unavailable. All replay methods have one thing in common: they store a small subset of samples from previous tasks in an "episodic memory" and feed them into the network alongside new ones. This class of methods tends to perform better in class-incremental learning than regularisation methods.

### 4.4.1 Experience Replay

The first CL method to use replay was "Incremental Classifier And Representation Learning" (iCarl) [Rebuffi et al., 2017]. iCarl employs a special strategy for selecting samples to store: it creates a pool of samples per class, averages their features into a prototype and selects samples based on their similarity to this average. This ensures the

examples in memory are representative of their class. The loss is calculated on a mix of novel and memory samples, with the total number of samples being distributed among all known classes. The output logits are saved at the end of each task. Another loss term is added for Knowledge Distillation, penalising drift from the old logits to preserve their distribution. iCarl's inference is also based on the nearest mean of exemplars seen, similarly to the Prototypical Networks [Snell et al., 2017] we explored in Chapter 2. This work was a key contribution and it is still included in class-incremental comparisons as a relatively strong baseline.

Riemer et al. [Riemer et al., 2019] combined plain sample replay with meta-learning using a meta-trainer similar to Reptile, which we discussed in Chapter 2 - Section 2.4, and achieved improved class-incremental performance over iCarl, albeit at the cost of a longer training procedure.

Tiny Episodic Memories (TEM) [Chaudhry et al., 2019b] found that decreasing the number of samples held in memory did not cause overfitting, but rather improved generalisation compared to previous baselines, which is relevant to our overarching few-shot use case. The authors also were the first to present previous literature in the context of Experience Replay as a learning paradigm for Continual Learning.

Dark Experience Replay (DER) [Buzzega et al., 2020] calculates loss on new data and adds a distillation component (Backwards Consistency), but the old logits are evaluated on every iteration instead of the end of every task like in iCarl. Their second proposed algorithm, named DER++, calculates loss on a batch of new data as well as a batch of memory data, with the distillation component still present. DER++ was one of the top performing algorithms in class-incremental learning.

Experience Replay has been successfully used in related fields as well, such as Reinforcement Learning: Rolnick et al. [Rolnick et al., 2019] paired simple experience replay with logit Knowledge Distillation on various game benchmarks. We discuss the similarities between the Experience Replay and Generalised Few-Shot Object Detection approaches in detail in Section 6.5.

### 4.4.2 Generative Replay

Storing previous task data can be cumbersome and a privacy risk, as we explore in Section 5.1. This is why a related area of investigation in CL is to to generate old data instead of storing it. The first work to propose this was Shin et al.'s Generative Replay [Shin et al., 2017], which trained a GAN on previous task samples. While a notable contribution, it was only tested on Permuted MNIST, and training a GAN is no easy task as discussed in Section 2.1.1, so most subsequent approaches focused on lighter methods. "Generative feature replay for class-incremental learning" [Liu et al., 2020] (GFR) trained a generator after every task to replay features rather than entire images, while confirming that GANs were not the best approach. Brain-Inspired Replay (BI-R) used a VAE (see Chapter 2 - Section 2.1.2) to replay features to the network, with one VAE head for each class. Another method proposed by the same authors as BI-R was Generative Classifiers [Van De Ven et al., 2021], which created a VAE for each class and had the network infer which model it came from.

While these methods are effective in a task-incremental setting, the picture is different for a class-incremental one. GFR is better than iCarl, which is promising, however the evaluation setting is different from the original. It would also be good to compare it directly to other ER methods. As for Brain-Inspired Replay, it is competitive with ER but only when combined with Synaptic Intelligence. Generative Classifiers is a contender for the state of the art in class-incremental learning, albeit at an increased computational cost due to the training of one VAE for each class and subsequent generation.

### 4.4.3 Sampling and Replay

Both G-FSOD and Experience Replay use a small, fixed subset of the original data for their memory set, but the memory set in G-FSOD is randomly drawn. In contrast, CL works often explore how to create the memory set in an optimal fashion. As we've mentioned, iCarl [Rebuffi et al., 2017] uses all images seen by the network to build prototypes by averaging their features, then uses the images which best approximate each prototype to build the memory set.

Later, in one of their experiments, Castro et al. [Castro et al., 2018] found no improvement when ranking the images according to prototype distance, similarly to iCarl, then sampling them equally from split histogram bins.

Tee et al. [Tee and Zhang, 2023] calculated a confidence score and a similarity matrix of all samples across all classes as measures of difficulty, then used those measures for ranking. They reported that an even spread of samples presented in order of easy to difficult significantly improves average accuracy on classification benchmarks.

The methods we focus on process data in batches, but it is worthy to note that Online Continual Learning can also make use of algorithms to determine the best replay set, a process known as known as Coreset Selection [Aljundi et al., 2019] [Yoon et al., 2022]. These methods usually rely on gradients calculated on previous samples. We delve into gradient manipulation in the section below.



### 4.4.4 Gradient Manipulation with Replay

Episodic Replay has been used in conjunction with gradient manipulation as well, trying to improve generalisation by steering the direction of the gradient during each training step. These methods hinge on two variables: the gradient derived from a batch of novel data and the one derived from a memory batch. The first method to set this paradigm was Gradient Episodic Memory (GEM) [Lopez-Paz and Ranzato, 2017]. The essence of GEM is an orthogonal constraint on the novel gradient: when the angle between the novel gradient and the base gradient is higher than 90°, the novel gradient is projected orthogonally to the base one. This operation is also known as a rejection. Since a set of data to replay is kept for every task, there are many base gradients, which means GEM has to solve a quadratic programming problem, The main drawback of this methods is the vastly increased computation time on each iteration.

Orthogonal constraints have been used in Multi-Task Learning as well. The multi-task setting entails learning two different tasks at the same time from scratch, as opposed to sequentially as in CL. As an example, the method known as “PCGrad” [Yu et al., 2020a] in that field uses the exact same projection as GEM, although it is performed between randomly ordered pairs of task-specific gradients, instead of just on

the current task. Not all gradient methods necessarily require orthogonal constraints. Another gradient method in Multi-Task Learning is Conflict-Averse Gradient Descent (CAG) [Liu et al., 2021a], which finds the best update vector within a hypersphere centered around the average of the gradients.

In the CL field, GEM is considered an unwieldy method due to computation time and memory usage, so Averaged Gradient Episodic Memory (A-GEM) stores a joint memory of all past tasks to create a single "base" gradient, then uses the same orthogonal constraint and projection as GEM. This solved the problem of computation while improving performance. An illustration is provided in Figure 4.2.

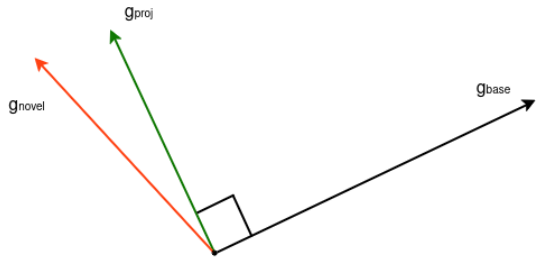


Figure 4.2: Simplified rendering of the A-GEM algorithm, with $g_{novel}$ as the current task gradient and $g_{base}$ as the gradient on samples from all previous tasks. The result is $g_{proj}$

Other proposed improvements [Hu et al., 2020] were SOFTGEM and Average A-GEM. Both use the same orthogonal constraint violation as A-GEM as the reason for correction. SOFTGEM adds a hyperparameter $\epsilon$ to control the contribution of the base gradient, while Average A-GEM averages the base and novel gradients instead of projecting the novel onto the base one. Both changes led to slight performance improvements over base A-GEM.

Guo et al. [Guo et al., 2020b] sought to provide a comprehensive view of replay-based gradient manipulation methods and provide their own alternatives in Improved Schemes for Memory-based Learning (MEGA). They use the same constraint with different correction schemes. Their first proposal, MEGA-I, prioritises base or novel gradient respectively based on the ratio of base and novel loss. With loss ratio $r = L_{base}/L_{novel}$, gradient $g_{novel}$ is multiplied by $r$ and $g_{base}$ is added onto it if the constraint is violated, otherwise the base gradient is ignored. Their second proposal, MEGA-II, uses the same

loss term to find an optimal angle that aligns with base novel and gradient, through an iteration-based approximation. Both methods perform better than A-GEM on various benchmarks including CIFAR and CUB.

Another interesting work that followed from A-GEM was Constraint-based Fine-tuning Approach (CFA) [Guirguis et al., 2022]. This work was concerned with the Generalised FSOD setting rather than CL. The constraint is the same as A-GEM, with the base training being treated as task 0: the novel gradient must not point more than 90 degrees away from the base one. To correct it, they average between the projection of the base gradient onto the novel one and novel onto base. When the A-GEM constraint is respected, this projection simply becomes an average of the novel and base one. We provide an illustration of this process in Figure 4.3 This method was used in the G-FSOD setting but it was only compared directly to A-GEM, and not evaluated in a CL setting as well. We will investigate this difference in setting in Chapter 7.

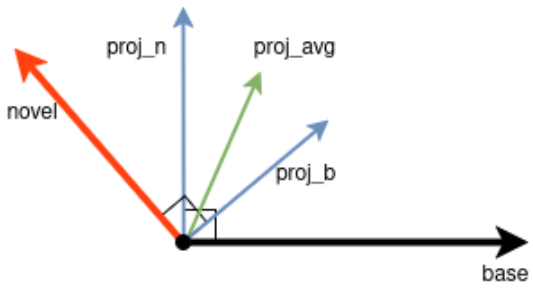


Figure 4.3: Illustration of Constrained Fine-tuning Approach by Guirguis et al. [Guirguis et al., 2022]. Novel gradient shown in red, base gradient in black.
If $g_{base} \perp g_{novel} > 0$, CFA will project $g_{base}$ onto $g_{novel}$ and vice versa, then average their projections. Otherwise it simply averages $g_{base}$ and $g_{novel}$.

We have to mention that gradient-based replay methods can suffer from slower convergence than simple replay methods, and that they have the same inconvenient requirement of storing earlier data as most of them.

## 4.5 Summary of Results

We focused on the published results for the class-incremental setting, since it is the most similar setting to generalised fine-tuning: new classes usually include stark changes in

extracted features, as opposed to task-incremental learning. We have included task-incremental results because they have been used to guide the adoption of gradient methods in a Generalised Few-Shot Detection setting, despite the most similar setting being class-incremental learning. This is because in a few-shot fine-tuning setting, a network has to detect all classes without knowing whether an image contains base or novel ones from the start.

While compiling this relatively brief chapter on Continual Learning, we discovered some real problems in collating performance metrics: the reported settings are often too different to be compared. For example, Dark Experience replay used CIFAR10 instead of CIFAR100 as Experience Replay and Tiny Episodic Memories did. Even when two methods are run on the same dataset, there are often significant differences in evaluation: for example, Brain-Inspired Replay was evaluated on a CIL setting with 10 subsets of CIFAR100 data, with the first task learning 10 classes just like the other ones. Generative Feature Replay also evaluated on CIFAR100, but the first task was included 50 classes. This setting, while being closer to the real world [Wu et al., 2022b], tends to yield stronger results and thus their performance metrics are not comparable.

This is why we limited ourselves to reporting VanDeVen's class-incremental learning Table 4.1, which provides a good enough overview of the performance of each method in a CIL setting and includes many replay methods. We consider adapting all the methods mentioned in this chapter to the same consistent conditions to be out of scope for this work.



### 4.5.1 Discussion

In the task-incremental setting, the best methods are LwF, a distillation-based method, and BI-R, a generative replay method, although the performance gap in the TIL setting is not very high. We can also see that A-GEM yields slightly lower performance than Experience Replay.

It is thus apparent that good performance in the task-incremental setting, which gradient methods are best known for, does not translate to a similar performance in the class-incremental setting. The best gradient replay methods yield inferior performance

| Method | Task-Incremental | Class-Incremental |
|---|---|---|
| Joint | 78.8 ± 0.2 | 49.8 ± 0.2 |
| A-GEM [Chaudhry et al., 2019a] | 73.3 ± 0.4 | 20.4 ± 1.4 |
| EWC [Kirkpatrick et al., 2017] | 76.3 ± 0.3 | 8.2 ± 0.2 |
| SI [Zenke et al., 2017] | 74.8 ± 0.4 | 8.1 ± 0.2 |
| LwF [Li and Hoiem, 2017] | **78.6 ± 0.2** | 25.6 ± 0.3 |
| DGR [Shin et al., 2017] | 71.4 ± 0.3 | 9.7 ± 0.2 |
| iCarl [Rebuffi et al., 2017] | - | **37.8 ± 0.2** |
| ER [Riemer et al., 2019] | 76.4 ± 0.2 | 37.6 ± 0.2 |
| BI-R [Van de Ven et al., 2020] | **79.1 ± 0.2** | 25.8 ± 0.4 |
| GC [Van De Ven et al., 2021] | - | **46.8 ± 0.2** |

Table 4.1: Final accuracy on 10-Task CIFAR100, collated by Van De Ven et al. [van de Ven et al., 2022]. Joint refers to training the model from scratch on data from all classes, for a simple baseline that is similar to Prabhu et al.'s [Prabhu et al., 2020]. We have highlighted the top 2 methods in each setting.



compared to replay methods in class-incremental learning. We consider this important as it suggests they should not be used in a fine-tuning setting. We provide a detailed comparison in the G-FSOD setting based on our own experiments in Chapter 7.

For the class-incremental setting, the Generative Classifier method performs the best, albeit at the cost of training a VAE for each class. By comparison, iCarl is a good compromise which only requires inference on a subset of data to build class prototypes. We also note that iCarl is an older, more established method with many available implementations. It does have the drawback of having to store previous samples.

It is also confirmed that regularisation methods are not as good as replay methods in class-incremental learning. While this was confirmed to be the case in G-FSOD as well, they do have the advantage of not requiring storage of previous data. We discuss the implications of data storage in detail in Section 5.1.

## 4.6 Conclusions

We have provided an overview of the different tasks that comprise the field of Continual Learning, and the most common methods to address them including the benchmarks.

After our investigation, we can state the following:

- The distinction between Task and Class Incremental learning matters very much: TIL's inclusion of task descriptors in the input allows for the optimisation of two separate tasks orthogonally, as opposed to CIL which has to treat all class as equally likely.
- Following up the above, regularisation methods used in Task-Incremental Learning do not work as well in Class-Incremental Learning and vice versa. This merits reflection, since the field of Generalised Few-Shot Learning we mentioned in the previous chapter is close to CIL.
- While network expansion is a useful technique for CIL, especially when combined with ensembles, it requires high memory capacity and computational resources. Pruning can alleviate that, but it increases training time.
- The best performing methods for CIL always involve some form of Experience Replay. Recently, new ways to improve basic ER have attracted the attention of researchers, such as distillation or generative replay. This looks like a very promising avenue of research.
- The current benchmarks in all areas of Continual Learning may not reflect real-world conditions due to sharp transitions between sub-tasks.

The next chapter deals with the parallels between methods developed for Continual Learning and ones developed for Generalised FSOD, showing they are often based on the same insights. We later perform a detailed study on the performance of CL methods in Generalised Few-Shot Object Detection in Chapter 7 - Experiments.

# Chapter 5

# Commonalities Between Fields

Here we comprehensively discuss the relationship between the fields of CL and G-FSOD which we have previously examined. We also acknowledge some of the potential issues associated with this research.



## 5.1 Considerations on Base Sample Storage

Storing base samples alongside the model to fine-tune can lead to privacy and copyright risks, since not every large base set might be composed of public-domain images. While it is possible to store CNN backbone features instead of images, which would also speed up training, feature storage requires higher storage costs as pixels have been converted into richer feature representations.

Even storing sample features only may not lay these concerns to rest. Reconstructing images from convolutions is an inverse problem, since it entails reconstructing inputs from observed outputs, and there can be multiple solutions. This problem has been studied extensively in recent years. He et al. [He et al., 2019] found that images can be reconstructed from intermediate layer features, albeit they are too noisy when the features are high-level ones from the last convolutional layers.

Efficiency and privacy are not the only concerns. One of the current lines of research [Hayes et al., 2021] focuses on bridging the gap between neuroscience and machine learning research. A point frequently raised is that storing raw inputs is not how our

own brain works: our brain remembers features rather than pixel-wise inputs, so feature replay is a closer approximation to how the hippocampus can "replay" previously seen patterns during sleep. We expand on this in Section 5.2.3 - Brain Inspired Replay.

## 5.2 Similarities between Generalised-FSOD and other Continual Learning approaches

While Generalised Few-Shot Learning is not as popular as plain Few-Shot Learning, there have been a few studies towards maintaining base performance in the classification context, such as Shi et al.'s graph-network based approach [Shi et al., 2020]. The link between Generalised FSL and Continual Learning has not gone entirely unnoticed: Kukleva et al. [Kukleva et al., 2022] drew from the similarities between the two fields to build a framework which works in both CL and Generalised Fine-Tuning on few-shot classification. Their method uses weight consolidation similar to EWC, penalising deviation from previous weights with a loss term, followed by experience replay in a separate phase.

One of the most immediate links between Continual Learning and Generalised Few-Shot Object Detection is CFA [Guirguis et al., 2022], which is derived from the CL method known as A-GEM [Chaudhry et al., 2019a]. However, its usefulness may lie in its inherent use of experience replay just as TFA/DeFRCN. Other methods have made use of very similar approaches to Continual Learning, or concepts agnostic to both fields, to achieve the same ends. We summarise the connections between those methods in Figure 5.1 below, and explain the common concepts in the next sections.

### 5.2.1 Network Expansion

Network expansion is a concept which has received attention in both GFSOD and CL. Specifically, Retentive R-CNN [Fan et al., 2021] builds on the same principle of creating another branch and freezing the previous one for a new set of classes to learn. This is directly related to the module-based network expansion approaches we encountered in Section 4.2 when discussing Continual Learning, such as Progressive Networks [Rusu

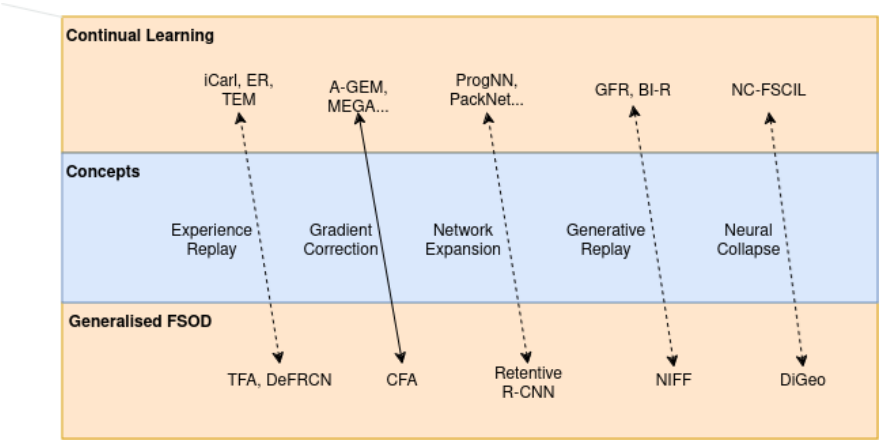


Figure 5.1: Usage of related concepts across the fields Continual Learning and G-FSOD. Solid lines indicate known connections between the fields at the time of writing.

et al., 2016]. The difference is that Retentive R-CNN's modules are connected at the ROI pooling stage, rather than having lateral connections on high-level feature layers.

That said, expansion approaches have a long way to go. While our own brains are able to produce new cells as we expand certain areas of knowledge [Shors et al., 2012], this biological process seems to be much more efficient than current CL methods that follow this approach. While network pruning methods have improved memory efficiency, the process of expanding and shrinking a network is still a very intensive process. We hope that studies in how the brain consolidates new knowledge may inform new advances in this field.

### 5.2.2 Neural Collapse

Another concept which has been applied to both CIL and G-FSOD is Neural Collapse (NC), which is the tendency of a network to reduce its last fully-connected classification layer to an Equiangular Tight Frame (ETF), that is a number of class prototypes equidistant from each other. When applied to tasks which require the addition of new classes to a model, this means the network can leave enough room for new embeddings during successive training phases. As mentioned in Section 3.5.2, DiGeo [Ma et al., 2023] uses a loss function that encourages NC as part of its solution to the G-FSOD problem. The same principle has been used in Class-Incremental Learning: Yang et

al. [Yang et al., 2023a] provide a fixed ETF to represent all classes, reserving space for future ones. Their method was tested specifically in a few-shot setting. NC can be applied in any field where data may not be equally available across classes. This is why it has been used in Federated Learning [Li et al., 2023b], where the model may be scattered across multiple devices.

Another common concept used by DiGeo is self-distillation, which we explained in Section 2.5.2 - Knowledge Distillation. This relies on preserving the model's output distribution by mimicking the logits on the old class set. Teacher-student distillation would be more complicated in Object Detection than it was for Dark Experience Replay [Buzzega et al., 2020], since the latter was grounded in the CL classification setting. A detection network produces a large number of logits since they often depend on region proposals, which makes representing their distribution more difficult. Distillation is an ongoing research topic for object detection networks.



### 5.2.3 Brain-Inspired Replay

Replaying a past memory seems to be the most efficient way to prevent forgetting, given how well replay methods hold up compared to regularisation ones in a class-incremental setting. However, storing samples is not the most efficient or safe method, as we've discussed in Section 5.1. This is why a number of generative approaches have emerged in the CL space to obviate the need for sample storage. Generating samples can also provide better performance, since they are less likely to be out-of-distribution ones compared to random sampling.

We already summarised some generative replay methods in Section 4.4, one of which was Brain-Inspired Replay. It trains a Variational AutoEncoder (see Section 2.1.2 for VAE details) on the seen samples instead of storing them. A new head in the VAE is allocated to each known class to better generate class-specific features by keeping some parameters separate. The features are then replayed through deep layers of the network, as opposed to "exact" or pixel-level replay which replays the entire inputs, which can decrease spurious correlations.

Another method recently reveloped in the G-FSOD space bears remarkable resem-

blance to Brain-Inspired Replay: "Neural Instance Feature Forging" (NIFF) [Guirguis et al., 2023]. In NIFF, a Variational AutoEncoder is trained with a separate head for each class, and features generated by the VAE are fed into the network when learning new classes. That said, the two methods differ in how they use replay. Feature replay is not the exact same since BI-R uses whole image features, while NIFF replays RoI features since it was built for the task of object detection. Additionally, while in BI-R the generator shares many layers with the classifier model, in NIFF the statistics of the RoI features are used as a target to train a separate generator model, which allows NIFF greater flexibility.

Unfortunately, NIFF was evaluated as an addition to CFA, whose theoretical justification is fairly questionable. In the methodology part of our work, we found their results to be not reproducible under the same conditions as DeFRCN (see Section 7.4). However, there is a strong case for feature replay yielding similar performance to exact replay with the right architecture, so we invite the authors to publish the results for NIFF without CFA, since adapting VAE generative replay for G-FSOD is an achievement in of itself.



### 5.2.4 Neuroscience and Neural Network Replay

When comparing hippocampal replay and neural network replay, there are many differences between their respective tasks which we have to take into account. As mentioned in Section 4.1.1, most CL benchmarks focus on classes whose features are fairly disjoint between one another and learned at the same time. This may not be too far from current industry use cases, but it is fairly limiting. Even in stream-based continual learning, the specific time at which a sample was observed is usually considered unimportant.

By contrast, time and order is far more important in mammal brains, and it drives replay processes as well. Reverse replay (starting from most recent experiences) is correlated with consolidation of memories, while forward-ordered replay is used in the awake state to sample possible outcomes [Wikenheiser and Redish, 2015] [Pfeiffer, 2020].

Probabilistic sampling of features for replay can be better than straightforward

sample replay, but it suffers from simplification derived by the tasks it solves, which means the order of features does not matter, as class samples are sourced indifferently to their time of appearance in the dataset in current benchmarks. We believe that in order to better exploit advances in neuroscience, we will need ambitious benchmarks more closely aligned to the real world. Right now, the most widely used CL benchmarks have no time encoding, except for specific applications such as video feed analysis, and a few seldom used like CLEAR [Lin et al., 2021] have concepts mutate over time.

It is entirely possible that current strides in the usual benchmarks do not represent improved generalisation ability: Prabhu et al.'s [Prabhu et al., 2020] work showed that CL benchmarks were not adequate in testing for future generalisation due to sharp transitions. Real-world generalisation is a known problem in standard large-scale learning as well: Fang et al.'s work [Fang et al., 2024] found just a weak correlation between performance of an architecture on ImageNet and domain-specific datasets, recommending broader benchmarks. Contemporary research should embrace recently developed benchmarks to make full use of insights derived from biological studies.



## 5.3 Conclusions

So far we have discussed the various aspects and implications of each technique with respect to their own field. In this section, we strive to provide a unified view of the insights gathered from every chapter and inform the direction our experiments should take in the next chapter.

**Meta-Learning**. One of the most important avenues of research in Few-Shot Learning is Meta-Learning. However, while Meta-Learning is a definite staple of Few-Shot Classification methods, with the vast majority of recent methods incorporating ML in some way. However, the results in the Few-Shot Object Detection field consistently underperform simpler fine-tuning based methods. The consensus appears to be that ML methods cannot provide a flexible enough starting point for fine-tuning algorithms in the relatively complex task of Object Detection [Huang et al., 2023]. However, other works have ascertained the presence of class information leakage in FSOD benchmarks (See section 3.7.2) which may affect both research paths differently and warrants further

investigation. Due to this under-performance, while a strand of ML research is present in Continual Learning, we chose not to pursue further investigation in the matter when building our FSOD experiments in the next Chapter 6 - Methodology.

**Metric Learning**. On the other hand, metric learning has been successfully employed in both FSC [Zhang et al., 2020] and FSOD [Wang et al., 2024]. We note that it is rare for FSOD works to experiment with explicit distance metrics, likely due to the complexity of the task to solve. Still, class prototypes have seen use in both fine-tuning and meta-learning based works, and prototypes are either learned via neural network [Karlinsky et al., 2019] or created from feature means [Qiao et al., 2021]. The latter, simpler kind of prototypes have also been employed in Continual Learning for recalling previously seen classes [Rebuffi et al., 2017].

**Self-Supervised Learning**. While more complex and time-consuming, SSLprovide a moderate performance boost across different tasks. This was the case in both FSC and FSOD: for example, DeFRCN saw an increase in AP when coupled with a SSL pre-trained backbone such as SWaV [Caron et al., 2020]. There is little use of self-supervised pre-training in the Continual Learning field due to catastrophic forgetting, albeit that might change soon [Liu et al., 2025].

**Architectures.** While there is no fixed architecture in FSC, most of the ones we reviewed are variants of ResNet. In the field of FSOD, the standard is Faster R-CNN thanks to a its good performance and extensibility, with just a few works opting for recent versions of YOLO. The field of FSOD might need more targeted architectures in the future, but the convenience of the previous body of work (and likely time-to-publish pressure upon researchers) has so far narrowed the choice between those Faster R-CNN and YOLO.

**Memory Replay**. We have expanded on the parallels between G-FSOD and Continual Learning, which can both make use of network expansion and memory replay approaches. In order to avoid excessive resource usage, we focus on replay methods to evaluate possible improvements. As we mentioned in Chapter 4 - Continual Learning, there have been innovations in the way past samples are incorporated. Dark Experience Replay uses distillation loss on previous logits, but this is hard to transfer to

a FSOD setting given the complexity of region proposals in object detection. Generating replay samples to ignore spurious features is an improvement which has been explored before [Guirguis et al., 2023] and requires further investigation, though it requires training of a generator network which clashes with our self-imposed resource constraints from Chapter 1 - Introduction. On the other hand, selecting which samples are replayed based on different criteria could be a useful low-cost enhancement for any G-FSOD network.

An even stronger continuity between Class-Incremental Learning and Generalised Fine-Tuning is apparent thanks to the sharp transitions between classes in common CIL benchmarks. After looking at the latest developments in Continual Learning and Neuroscience, we advocate for more widespread usage of alternative benchmarks such as CLEAR [Lin et al., 2021].

**Training Optimisations.** We have investigated techniques which can be applied orthogonally to any method: Network Architecture Search, Self-Distillation and Gradient Correction Methods. It is clear from our review that the utility of NAS in this setting is questionable and that it provides diminishing returns for its training cost. By contrast, Self-Distillation has seen successful usage in the FSOD setting, as evidenced by the works known as Multi-Faceted Distillation [Wu et al., 2022a] and DiGeo [Ma et al., 2023]. Self-Distillation is popular in the Continual Learning field as well, used by older methods such as LwF [Li and Hoiem, 2017] and more recent ones such as FOSTER [Wang et al., 2022]. We recommend that new methods seriously consider Self-Distillation as a natural part of their toolbox, as long as prolonged training time is not an issue.

Finally, with Gradient Correction Methods, we found consistent research in Task-Incremental Learning (TIL) but none of these methods are meant for Class-Incremental Learning, and in fact do not perform well in that setting [van de Ven et al., 2022]. The only use of gradient manipulation outside of TIL which we found was Constrained Fine-tuning Approach [Guirguis et al., 2022] (CFA). In the next part, we focus on the rationale for such methods in Generalised FSOD, attempt to reproduce the results of CFA and to see whether gradient correction methods can actually improve performance

in that setting.

# Part II

# Introduction

This part of the thesis covers our own research, as informed by the research survey conducted in Part I. Having discussed the relationship between Continual Learning and Generalised Few-Shot Object Detection, we investigate the integration of methods belonging to the former into the latter. We explain our focus on gradient correction methods and experience replay sampling, outlining the methods we are going to be testing. After laying out a plan and performing the experiments, we analyse the results and draw conclusions accordingly. We believe this part constitutes a contribution to current research in Generalised Few-Shot Object Detection, just as the previous literature survey contributes to the wider field of Few-Shot Learning.

# Chapter 6

# Methodology

In this chapter, we outline our research direction and empirical investigation of Continual Learning methods in G-FSOD. As stated in the previous chapter, what spurred our interest in gradient correction was the claim made by the authors of CFA [Guirguis et al., 2022] that base and novel sets could be treated as orthogonal tasks by virtue of the base classes having undergone a different, large-scale pre-training regime. We discuss the issue in detail, then provide an overview of the gradient correction methods we've chosen for comparison, focusing on their differences. We then discuss the rationale for a separate investigation concerned with sampling strategies.



## 6.1 Rationale for Gradient Correction

The reason we decided to test a number of gradient correction methods was to find out whether their benefits carried over into a class-incremental setting with a different objective from the original task-incremental one, a distinction which was not made by CFA. We already mentioned in Chapter 4 - Continual Learning this difference in objectives: the former includes task descriptors as part of the training and inference process and the latter does not.

Establishing a minimal case with two sets of classes, $C_b$ and $C_n$ The multi-task objective is to learn a cost minimisation function with input sample x and task descriptor $t = T_b|T_n$, as shown in 6.1:

$$f(x,t) \rightarrow y \in (C_b|C_n) \tag{6.1}$$

This means the classification head will also take the task descriptor as an input, resulting in two distinct paths depending on $t$.

Class-incremental learning, on the other hand, has the unified objective to infer over the whole set of potential output classes (Equation 6.2).

$$f(x) \rightarrow y \in C_b \cup C_n \tag{6.2}$$

Since the structure of the classification layer depends on the input, we posit there is no reason the whole input class sets $C_b$ and $C_n$ should be treated as contrasting objectives. Instead, it may be useful to separate individual classes, which was the approach of DiGeo [Ma et al., 2023]. We decide to test this by replicating the original CFA experiment as well as a selection of related gradient correction methods.



## 6.2 Training Procedure

In this section we show the overall training procedure for G-FSOD, so we may later discuss changes to it. As we mentioned in Chapter 3, two domains are available when fine-tuning a network in the Generalised FSOD setting: base and novel, which we refer to as $D_b and D_n$. In the majority of the G-FSOD methods we explored, the training procedure is the standard used for neural networks, where a mixed batch of items is drawn from both domains.

To be concise, we have split the training loop into two functions. The outer $Train$ function which takes the base memory set, novel set, number of epochs $E$ and learning rate $\eta$. The inner function *Step* takes base and novel sets and calculates the gradient. The *Step* function is a good injection point to change the usual implementation and implement any given gradient correction method.

**procedure** TRAIN($D_{base}, D_{novel}, E, \eta$)
  **for** epoch = 1 to $E$ **do**
    $\hat{g} \leftarrow Step(D_b, D_n)$

$\theta \leftarrow \theta - \eta * g$ ▷ Update weights

**end for**

**end procedure**

**function** STEP($D_{base}, D_{novel}$)

$D = D_b \cup D_n$

$x, y \leftarrow D$ ▷ Draw mixed batch

$l \leftarrow L(f_\theta(x), y)$ ▷ Loss

$g \leftarrow \Delta_\theta(l)$ ▷ Backpropagation

**return** $g$

**end function**

### 6.2.1 Experience Replay

As we mentioned in Chapter 5, the default G-FSOD setting is close to ER, which is why we'll use it as an additional baseline. The following is a basic Experience Replay algorithm very similar to Chaudhry et al.'s algorithm [Chaudhry et al., 2019b] in the Continual Learning space. No gradients are manipulated, losses are simply added together then used for backpropagation.

**function** STEP($D_{base}, D_{novel}$)

$x_n, y_n \leftarrow D_{novel}$

$x_b, y_b \leftarrow D_{base}$

$l_b \leftarrow L(f_\theta(x_b), y_b)$ ▷ Base loss

$l_n \leftarrow L(f_\theta(x_n), y_n)$ ▷ Novel loss

$g \leftarrow \Delta_\theta(l_b + l_n)$ ▷ Backpropagation

**return** $g$

**end function**

## 6.3 Summary of Gradient Correction Methods

The notation of all the following algorithms has been updated to reflect a fine-tuning setting. Since the base memory is built outside the training loop, and we only manipu-

late the gradient rather than any hyperparameters, we only included the training *step* function.

Since all gradient manipulation algorithms mentioned here require base and novel gradients, we omitted the loss and back-propagation steps after showing them once in A-GEM for brevity (marked with $< snip >$). All $g_b$, $g_n$, $l_b$, $l_n$ are calculated the same way as in A-GEM.

### 6.3.1 A-GEM

A-GEM [Chaudhry et al., 2019a] was the first gradient manipulation method in the field of Continual Learning. When the angle between base and novel gradient is wider than 90 degrees, the novel gradient is projected onto the base one and used as the new update direction.

**function** STEP($D_{base}, D_{novel}$)
  $x, y \leftarrow D_{novel}$
  $x_b, y_b \leftarrow D_{base}$
  $g_b \leftarrow \Delta_\theta(L(f_\theta(x_b), y_b)$ ▷ Backpropagation (base)
  $g_n \leftarrow \Delta_\theta(L(f_\theta(x_n), y_n)$ ▷ Backpropagation (novel)
  **if** $g_b \,^\perp\, g_n < 0$ **then** ▷ Angle check
    $\hat{g} \leftarrow g_n - \frac{g_b \,^\perp\, g_n}{g_b * g_b} g_b$ ▷ Projection
  **else**
    $\hat{g} \leftarrow g_n$
  **end if**
  **return** $\hat{g}$
**end function**

### 6.3.2 CFA

Constraint Fine-Tuning Approach [Guirguis et al., 2022] was built specifically in the context of G-FSOD but follows a similar pattern. When the angle between base and novel gradient is wider than 90 degrees, the novel gradient is projected onto the base one, the base one is projected onto the novel one, and their average is set as the

new update direction. When the angle is acute, base and novel gradients are simply averaged.

**function** STEP($D_{base}, D_{novel}$)

$< snip >$

**if** $g_b{}^{\perp} g_n < 0$ **then** ▷ Angle check

$\hat{g} \leftarrow \frac{1}{2}(1 - \frac{g_n{}^{\perp} g_b}{g_b * g_b} g_b) + \frac{1}{2}(1 - \frac{g_b{}^{\perp} g_n}{g_n * g_n} g_n)$ ▷ Projection

**else**

$\hat{g} \leftarrow \frac{g_b + g_n}{2}$

**end if**

**return** $\hat{g}$

**end function**

### 6.3.3 MEGA-I

This algorithm [Guo et al., 2020b] is a straightforward one, with one novelty over A-GEM: the use of ratio $r$ between base and novel losses when calculating the gradient. The same gradient angle check is performed, but with $\epsilon$ as a hyperparameter denoting sensitivity ($10^{-10}$ by default). When the angle between gradients is acute and no correction is needed, $\hat{g}$ is set to the novel gradient $g_n$. Otherwise $\hat{g}$ will become the $g_b + g_n * r$, so that the influence of the novel gradient is reduced by $r$ to avoid forgetting.

Since this method did not provide new insights, we did not repeat this test on the COCO dataset.

**function** STEP($D_{base}, D_{novel}$)

$< snip >$

**if** $l_t > \epsilon$ **then**

$\alpha_1 = 1, \alpha_2 = \frac{l_b}{l_n}$

**else**

$\alpha_1 = 0, \alpha_2 = 1$

**end if**

$\hat{g} \leftarrow \alpha_1 . g_n + \alpha_2 . g_b$

**return** $\hat{g}$

**end function**

### 6.3.4 MEGA-II

MEGA-II [Guo et al., 2020b] is a gradient correction method for Task-Incremental Learning, more complex than A-GEM. The algorithm is summarised below.

- $\hat{\iota}$ is the angle between base and novel gradient, $\beta$ is the angle between $g_n$ and the desired $\hat{g}$
- $\iota$ is an angle chosen from a random pool and iteratively refined to maximise $l_n cos(\beta) + l_b cos(\hat{\iota} - \beta)$
  The end result is a gradient vector with the same magnitude as $g_n$ but rotated to align with $g_b$ as well.
- If $\hat{\iota}$ is less than $\epsilon$ ($10^{-10}$) then no correction step is needed and $g_n$ is used as usual.

**function** STEP($D_{base}, D_{novel}$)

$< snip >$

$\hat{\iota} = \frac{gb^{\perp} g_n}{||g_b||.||g_n||}$

**if** $||g_b||.||g_n|| > \epsilon$ **then**

$\alpha_1 = \frac{||g_n||^2||g_b||^2 cos\iota - g_b{}^{\perp} g_n||g_n||||g_b||cos(\hat{\iota}-\iota)}{||g_n||^2||g_b||^2 - ||g_b{}^{\perp} g_n||^2}$

$\alpha_2 = \frac{-g_n{}^{\perp} g_b||g_n||^2 + ||g_n||^2||g_n||||g_b||cos(\hat{\iota}-\iota)}{||g_n||^2||g_b||^2 - ||g_b{}^{\perp} g_n||^2}$

$\hat{g} \leftarrow \alpha_1.g_b + \alpha_2.g_n$

**else**

$\hat{g} \leftarrow g_n$

**end if**

**return** $\hat{g}$

**end function**

### 6.3.5 CAG

Conflict-Averse Gradient descent [Liu et al., 2021a] was built in the context of Multi-Task learning, adapted here for Generalised FSOD. It was not intended for use in

a fine-tuning context. We have included it here to show that methods which tend to average the gradients always perform better than those that don't in the FSOD context, despite differences in the underlying objective.

**procedure** STEP($D_{base}, D_{novel}$)

$< snip >$

$g_a = \frac{g_b + g_n}{2}$

$\phi = c^2 ||g0||^2$

Minimise $g_w \perp g_a + \sqrt{\phi}||g_w||$ for $g_w = w_b g_b + w_n g_n$

**return** $g_a + \frac{\sqrt{phi}}{||g_w||} g_w$

**end procedure**

## 6.4 Other methods

### 6.4.1 CFA With Loss

In this experiment, we tried to use the loss term to steer the gradient correction. The ratio coefficient $r$ is calculated as $\frac{l_b}{l_n}$ just as $\alpha_2$ in MEGA-I, but is clamped because a coefficient too close to 0 or 1 leads to a decrease in performance. The clamping is performed by running $r$ through the following function:

$$\rho = tanh(a * (r - 0.5)) * b + 0.5$$

with $a = 2$ as a parameter that controls the steepness of the curve and $b = 0.1$ controlling the clamp limit, in this case 0.4 to 0.6.

Then, $\rho$ is applied to the base and projection cases as follows, instead of averaging them:

**function** STEP($D_{base}, D_{novel}$)

$< snip >$

**if** $g_b \perp g_n < 0$ **then** ▷ Angle check

$\hat{g} \leftarrow (1 - \rho)(1 - \frac{g_n \perp g_b}{g_b * g_b} g_b) + \rho(1 - \frac{g_b \perp g_n}{g_n * g_n} g_n)$ ▷ Projection

**else**

$\hat{g} \leftarrow (1 - \rho) g_b + \rho g_n$

**end if**

**return** $\hat{g}$

**end function**

### 6.4.2 Averaging

In this experiment, we simply averaged the base and novel gradients ($g_b$ and $g_n$) during each training iteration, without checking the angle.

**function** STEP($D_{base}, D_{novel}$)

$< snip >$

$\alpha_1 = 0.5$

$\alpha_2 = 0.5$

$\hat{g} \leftarrow \alpha_1.g_n + \alpha_2.g_b$

**return** $\hat{g}$

**end function**

This was intended as an ablation study, which proved very useful when interpreting the full results.

## 6.5 Rationale for Sampling: G-FSOD and Experience Replay

Base samples are crucial for some of the influential FSOD methods we explored before in Chapter 3. TFA, which uses frozen weights, exploits base samples both when considering novel classes (FSOD) and base/novel ones (G-FSOD). DeFRCN uses them in the G-FSOD setting, a practice adopted in subsequent works. We argue that these well-known fine-tuning methods employ a replay strategy similar to the existing CL method known as Experience Replay.

In ER, for a set of base classes $D_b$, a memory of samples $M_b$ is created. Samples drawn from novel task classes $D_n$ are sent through the network alongside samples drawn from $M_b$, and loss calculated on both ($L_n + L_b$).

ER uses reservoir sampling for creating and updating the buffer and A-GEM uses ring-buffer sampling, but in G-FSOD there is no need to update the memory buffer since

there are only two tasks. Both settings use a small, fixed subset of the original data for their memory when transitioning from Task 1 to Task 2. G-FSOD is not concerned with partitioning memory for subsequent tasks, since there is only one original task.

While G-FSOD benchmarks have a 75/25% base/novel class proportion, with mixed mini-batches entailing a 25/75% class probability for the samples drawn, ER methods often draw their samples separately, calculating two loss terms: $L_b$ for base sample loss and $L_n$ for the current task, with the two being equally weighted. Indeed, a viable alternative is to simply chain the memory samples to the novel ones in a batch for a ratio of 50/50% [Prabhu et al., 2023].

The G-FSOD setting likely benefits from not requiring storage of the whole base set, as images used for a detection setting tend to be higher resolution than those used for classification to improve localisation granularity, so the entire base set would never fit into cache memory. However, we noticed that the base samples for the 10-split fixed benchmarks introduced by TFA were drawn through random sampling without replacement. We asked the follow-up question: while gradient correction methods are not effective, is it possible to improve performance based on sampling alone? A small pool of fixed random samples may not be the best way to represent the underlying distribution of base classes. This spurred us to investigate the code which produced the original TFA data splits, and to explore sampling methods.



## 6.6 Summary of Sampling Strategies

We considered the default sampling strategy in the mainstream TFA benchmark: for the $k$-shot setting, $k$ samples are present in each novel class, and $k$ samples are selected for each base class to match them. Only one instance annotation is collected for each image, even if there are more present. While this makes sense to keep the number of novel instances limited in a few-shot setting, this limit has been applied to both base and novel instances, despite base annotations being abundant. Additionally, the base images are selected at random, despite the fact that in G-FSOD we have the benefit of starting from a large-scale dataset. We thus sought to select select samples that could help the network best remember its base task. The novel samples were not changed in

any of our experiments for a thorough comparison with the original DeFRCN work.

For each class, we create a pool of candidate images to draw statistics from, since running inference on the entire base training set would be very time consuming and beyond the scope of this research. The number of images in each pool is a hyperparameter we set to $p = 50$.

**Instance Limit**. Most of the fine-tuning works we have reviewed follow a balancing strategy that is open to question: they remove base instance annotations that exceed $k$ shots in both base and novel images. However this causes the network to treat extra base class instances as part of the background. We wanted to quantify how this would affect novel class performance, and so we removed the base instance limit.

When simply removing this limit, the number of instances for some classes can be much higher than others since they appear multiple times in the same image. This imbalance can worsen the performance of novel classes. To mitigate this, we populate our sample pool starting with the least frequent classes, whose images can already contain instances of common classes. Additionally, when selecting images for a pool, we discard those that contain extra instances for any class with an already full pool.



### 6.6.1 Prototype Distance

Our first method was to select the images with instances closest to their respective class prototypes. We collected image features from the network's backbone, then applied a RoI alignment operation with ground-truth base class labels. We then averaged the features of instances from every base class across the dataset to create prototypes, as shown in Equation 6.3. $\mu_c$ refers to the prototype for class $c$ and $x_i$ is each instance from candidate sample pool $P_c$.

$$\mu_c = \frac{1}{|P_c|} \sum_{x_i \in P_c} x_i \tag{6.3}$$

We ranked the images based on their euclidean distance to each prototype. For each class $c$, every sample $s$ containing that class was assigned a score $W_{s,c}$ based on its Euclidean distance from the class prototype as in Eq. 6.4. To obtain $k$ required shots

for the base set, we selected the samples with the lowest distance scores.

$$W_{s,c} = ||s - \mu||_2 \tag{6.4}$$

This yields the samples with the instances most similar to the prototypes to form the memory set.

While reviewing the literature, we found that this method of sample selection was also employed in iCarl [Rebuffi et al., 2017] ("mean of features") as well as the early CL work"End-To-End Incremental Learning" by Castro et al. [Castro et al., 2018], which called it "herding". This method is known to perform well in a classification setting.

As a note, Prototype sampling is not the same as DeFRCN's Prototype Calibration, since that entails combining novel class prototypes to the novel test samples during inference, while we operate on base samples before fine-tuning instead.



### 6.6.2 Prototype Distance Ratio

In this experiment, we calculated the between each class instance and all class prototypes, rather than just the one whose class it belongs to. We ranked the samples based on the ratio between the distance to the closest prototype and the distance to the correct prototype.

This strategy was meant to mitigate class confusion by selecting clearly defined samples, since we believed that relying on distance alone could cause the selection of ambiguous samples close to multiple classes. See Figure 6.1 for an illustration.

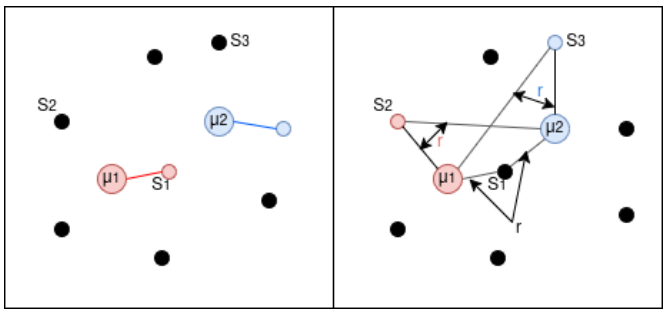


Figure 6.1: Illustration of sample ranking methods: Prototype Distance (left) versus Prototype Distance Ratio (right). $r$ denotes the ratio between prototype distances to a given sample.

Given a number of class prototypes $\mu_k$, we ranked the images for sampling based on the ratio between the distance to the correct prototype $\mu_y$ and their distance to the next closest prototype, as shown in Eq. 6.5.

$$W_{s,c} = \frac{||s - \mu_y||_2}{min_{k \neq y}(||s - \mu_k||_2)} \tag{6.5}$$

This implies that if $\mu_k = \mu_y$, i.e. the closest prototype is also the correct one, then $W_{s,c}$ will be lower than 1. The further away the next closest prototype is, the lower the score. We selected $k$ samples with the lowest scores. A practical example is shown in Figure 6.2.

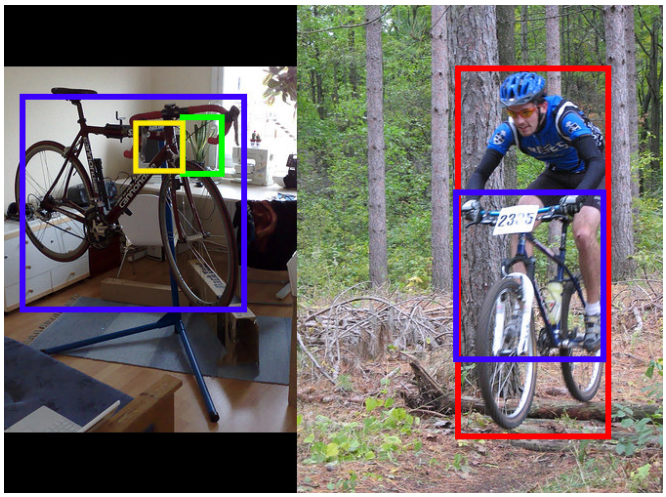

Figure 6.2: Most important instances for class "Bike" as captured by ProtoDist Ratio (Left) vs ProtoDist (right). Simple ProtoDist picked an instance whose features substantially overlap with "Person" due to the annotations, but is close to "Bike" as well. By contrast, ProtoDist Ratio picked an instance that does not lend itself to confusion: the overlapping classes ( "monitor" and "potted plant") have distant prototypes.

### 6.6.3 Histograms vs Top-K (Variation)

Incidentally, Castro et al. [Castro et al., 2018] also wondered if selecting the closest images to the prototypes might be too limiting, and set out to select images proportionally based on their distance to each class prototype, by dividing the image pool into histogram bins. While this did not yield good results for them, we were interested to see the impact this would have on few-shot learning since it provides a good alternative

to just selecting the closest prototypes.

Our implementation used 10 distance bins regardless of the number of samples and picks them in order starting from the first bin (closest to prototype). This meant that Bottom-K and Histogram sampling were equivalent under the 1-shot setting, as shown in Figure 6.3 . We saw histogram sampling as a possible enhancement of either ProtoDistance or ProtoDistanceRatio, as it could ensure feature diversity in higher shot settings.

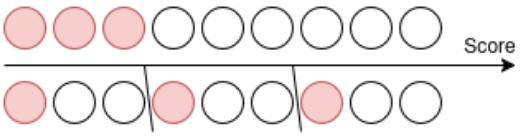


Figure 6.3: Bottom-K sampling (above) simply picks the samples which minimise the chosen metric, while Histogram sampling (below) takes one item from each bin, starting from the first and wrapping back around until the shot quota is reached.



### 6.6.4 Mini-Batch Distribution (Variation)

The standard G-FSOD training process involves drawing mixed mini-batches of base and novel images during training, although novel classes are only 25% of the total in the datasets we tested. This means the probability of drawing a novel sample is much lower than a base one. We attempt to redress this by drawing separate batches for base and novel samples, and calculating loss as the sum of their respective losses $L_{base} + L_{novel}$ as in ER [Chaudhry et al., 2019b]. This ensures the network sees equal numbers of base and novel images, although the number of instances may differ. We test this change on both the original sampling method as well as Prototype Distance Ratio, denoting it as ER.

# Chapter 7

# Experiments

In this chapter, we outline our experiments, present our results and interpret them, expanding upon the relationship between the Generalised Fine-Tuning and Class-Incremental Learning settings.



## 7.1 Experimental Plan

Our first batch of experiments aimed to answer the research question: do gradient correction methods improve G-FSOD performance, and if so, which approach works best? To see if the first hypothesis is true, we train multiple copies of a network with a constant architecture but different training procedures, with results averaged over different data splits to account for randomness inherent to neural networks. We outline the gradient correction methods tested in Section 6.3 below.

The second batch of experiments aims to answer the remaining question: is it possible to use Continual Learning methods to improve G-FSOD performance at all? After the negative results of the first batch, we adopted a different approach: since current methods use random sampling when selecting previously seen images that maintain performance on old classes, we focus on the selection of these base images that are used alongside the novel ones. As before, we keep the network architecture constant, and we use the same training procedure as well, only changing the subset of base class images for each trial. The reason we only modify the base subset instead of the novel one is

that the novel images are few-shot, and our results must remain directly comparable to other works.

Both batches of experiments are run on Pascal VOC as well as COCO: since the two datasets have fairly different characteristics as outlined in Chapter 3, it was important to know if the effects observed on one dataset would be present in the other.

The core of each experiment is the training of an DeFRCN-based Object Detection network with a ResNet-101 pre-trained backbone. To keep results consistent, we follow the class splits of others in the literature [Wang et al., 2020b] [Qiao et al., 2021]. More details are available in Section 7.2 - Experimental Setup.

## 7.2 Experimental Setup

We have run our experiments on both the VOC and the COCO datasets. The VOC dataset [Everingham et al., 2010], with 20 classes in total, is divided alternatively into 3 random class splits, with each split having 15 base and 5 novel classes. The COCO dataset is split once into 60 base classes and 20 novel classes, all belonging to the physical, well-defined object category known as "things" in the original specification [Lin et al., 2014]. Both VOC and COCO splits were defined in previous works [Kang et al., 2019]. COCO experiments were performed on 1, 5, 10 and 30 shots, while VOC experiments on 1, 5 and 10 shots. As a note, the highest number of samples for VOC has been 10 in related works as well, since performance increases tend to plateau quicker given the relatively simpler framing of the samples (See Section 3.1.1 for more dataset details).

We selected DeFRCN as the base implementation, using the same hyper-parameters and starting from the pre-trained weights to gauge the effectiveness of the methods discussed. We explain the importance of this in Section 7.3. We decided to use DeFRCN over TFA since TFA relies on freezing most of the network, which makes it less relevant for evaluating gradient correction methods. Additionally, most of the recent fine-tuning based literature uses DeFRCN as a baseline comparison or starting point [Gao et al., 2022a] [Ma et al., 2023] [Guirguis et al., 2023].

For evaluation, we use the standard approach of VOC-2007 for the VOC splits and

and 5k separate images for COCO. We average each result over 10 sets of samples according to the TFA protocol [Wang et al., 2020b]. We report base and novel Average Precision (bAP/nAP) calculated according to the specifics of both datasets, as we believe that AP without class set distinction can mask potential base/novel performance trade-offs in the G-FSOD setting.

All of our gradient correction method comparisons were thus implemented as custom trainer classes for DeFRCN, with the original DeFRCN implementation named 'Original' in our tables. We also used the same number of training iterations as DeFRCN for each $k$-shot experiment. We note that gradient correction methods take twice as long to converge compared to the base method, something that was noted by the authors of CFA-DeFRCN as well [Guirguis et al., 2022].

We used 4 A100 SXM4 GPUs for training the network, with 10 CPUs dedicated to each batch training job. The mini-batch size was 16, the same as previous works, resulting in a per-gpu mini-batch size of 4. We note the possible differences mini-batch size can cause in Section 7.3.

The VOC and COCO experiments were run 10 times with data derived from different random seeds, using the same data as TFA (and consequently DeFRCN) for a fair comparison [Wang et al., 2020b]. The experimental results are then averaged, and we report base class AP (bAP) and novel class AP (nAP). This is calculated in accordance to each dataset's definition of Average Precision, so mAP on COCO is calculated over the [0.5:0.05:0.95] IoU thresholds rather than just 0.5 as in VOC.



## 7.3 Experimental Notes

In this section we showcase some findings related to the reliability of the results themselves. We hope this will help reproduce results and hopefully inform the evaluation phase of future works.

### 7.3.1 Causes of Result Variance

Since a well-initialised base model will tend to be less affected by catastrophic forgetting, we believe that any method which uses the same basic framework for FSOD or G-FSOD should always start from previously published weights when possible to avoid any bias. This is not always possible if different frameworks are in use: for example, the MMFewShot framework [Chen et al., 2019a] uses another initialised base model, and while it was trained in a setting close to the original it achieves much better results on base classes.

Tweaking the implementation of the data loader can also produce unexpected results: creating separate base/novel splits and mixing them together instead of using the original one causes the base performance to decrease and novel performance to improve substantially on COCO, with novel results very similar to CFA's. Further investigation is required to ascertain the cause.

In the fine-tuning setting common to TFA, DeFRCN and related methods, an additional factor is the random initialisation of weights for the novel classifier, which can also sway results. This was acknowledged by the authors of TFA. Thankfully, the authors of DeFRCN have published the weights obtained by the "model surgery" step during which the classifier head is expanded to include novel classes.

The initial random seed plays a part as well, causing high variance as seen in Figure 7.1, which is quite apparent when using results from a single split, in the way of FSRW. We remark that the nAP50 metric is even more unreliable than nAP in this regard, since it selects results at a single IoU threshold of 0.5 instead of averaging them. The only way to counter this is to train and evaluate over multiple splits, average their results and estimate the associated confidence interval.

Part of the reason the FSRW split is still in use is that certain frameworks (such as MMFewShot) and repositories are older than TFA and do not include code for handling multiple splits. However, given the high variance caused by various factors we've shown in Figure 7.1, it affects the reliability of published results. Confidence intervals are key to quantifying the improvement of any method.

Additionally, FSRW results tend to be better than TFA (averaged) ones, which

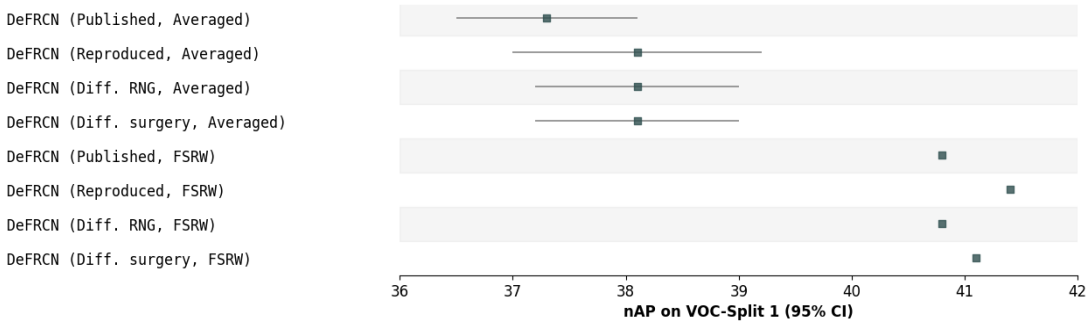


Figure 7.1: 5-shot results on VOC set 1, with Averaged referring to 10-split averaged metrics from TFA and FSRW referring to original Split 0 alone.

means they should not be directly compared. Why this happens is unclear: discarding the instance cap to bring sample selection in line with FSRW (see Section 6.6) does not improve average results for VOC, so the original FSRW samples for that one split likely happened to be representative ones.

We discovered that batch size is not the only hyperparameter that's important to reproduce previous papers' results: the effective mini-batch size of the data sent to all GPUs plays a role that has not been given due attention. While the nominal batch size has been 16 across many works, TFA and DeFRCN used 8 GPUs with 2 images each, and CFA 4 GPUs with 4 images each. We observed that sending more than 4 images per GPU causes performance to decrease across shots: see Table 7.1 for details.

| Shot No. | b=16 (1 GPU) | b=8 (2 GPUs) | b=4 (4 GPUs) |
|---|---|---|---|
| 1 | 41.2 ± 1.1 | 41.8 ± 1.1 | 42.5 ± 0.9 |
| 5 | 45.4 ± 0.7 | 46.4 ± 0.6 | 46.7 ± 0.5 |

Table 7.1: Effect of mini-batch size $b$ on VOC mAP (Class Split 1), where $b$ is the global batch size (16) divided by number of GPUs

### 7.3.2 Note on Confidence Intervals

The mean of multiple runs along with a confidence interval is the best way to estimate a method's effectiveness, all factors such as batch size being equal. However, we would like to add a minor correction to the calculation of the confidence interval, since the calculation has been adopted by other works using the same standard.

The confidence interval formula used by Wang et al. [Wang et al., 2020b] is meant for a normal distribution:

$$CI = 1.96 * \frac{\sigma}{\sqrt{n}} \tag{7.1}$$

where 1.96 is as the Z-value for a 95% confidence interval, $\sigma$ is the standard deviation, and $n$ is the number of repeated runs. $\sigma$ is commonly calculated as

$$\sigma = \sqrt{\frac{\Sigma(x_i - \mu)}{n}} \tag{7.2}$$

While the use of confidence intervals (Equation 7.1) in TFA was a massive improvement from previous work in terms of statistics, it may require a revision. For a small sample size such as 10 experiments (used for COCO in all such works), we cannot estimate the real mean of the possible set of experiments. Thus, the standard deviation from the mean of these 10 samples will inevitably be an underestimate of the standard deviation of the samples from the real population mean.

The first change we propose is the use of Bessel's Correction, named after the German scientist Friedrich Bessel. It's a well-known approach that replaces $n$ with $n - 1$ for the standard deviation's denominator, correcting some of the bias in the sample distribution.

$$\sigma = \sqrt{\frac{\Sigma(x_i - \mu)}{n - 1}} \tag{7.3}$$

Secondly, while a normal distribution is good fit for the mean with abundant samples (as stated by the Central Limit Theorem), this is not the case here. We believe a t-distribution is a much more cautious approach to modelling performance statistics since we only have 10 samples in the form of experiment results [Brereton, 2015]. Thus our confidence interval formula changes slightly:

$$CI = 2.262 * \frac{\sigma}{\sqrt{n}} \tag{7.4}$$

where 2.262 is the value of $t$ for a two-tailed confidence test of 95% drawn from the t-distribution table ($9_{dof}$), and $\sigma$ is standard deviation for the samples rather than the

assumed standard deviation of the population.

The original calculation of the confidence interval makes the assumption of a normal distribution. This might be justified if considering a large set of experiments as a fixed closed population in of itself, but this does not help direct comparisons with other works: TFA used 30 experiments for VOC and 10 for COCO while DeFRCN and others used 10 for both, to save on time. Furthermore, the experiment conditions may change substantially, as we did in Section 6.6 when randomly replacing base class data for an ablation experiment.

## 7.4 Performance of Gradient Correction Methods

We performed a number of experiments on both COCO and VOC in the G-FSOD setting to evaluate the effectiveness of gradient correction methods. *Original* refers to the original DeFRCN. A-GEM's implementation is derived directly from the original CL code [Chaudhry et al., 2019a], and $CFA$'s from the modifications to A-GEM detailed in the paper. $MEGA-II$ is the second CL method published by Guo et al [Guo et al., 2020b]. $CFA+Loss$ refers to an original experiment we performed adding a loss term to CFA, as detailed in Section 6.4.1. $CAG$ is the same multi-task learning method adapted for this fine-tuning setting [Liu et al., 2021a].

As shown by Table 7.2 and 7.3, no gradient manipulation procedure was better than the original training procedure of DeFRCN in a G-FSOD setting. MEGA-I, which only relies on a loss term, had even worse performance than A-GEM, and adding a loss term to CFA only manages to slightly decrease novel AP. See Figure 7.2 for a visual comparison.

Furthermore, based on the results of our proposed *Averaging* method, what makes other methods such as CFA outperform A-GEM and MEGA-II is the averaging of base and novel gradients, rather than any projection algorithm. This also explains why the CAG method achieves the same performance as the DeFRCN baseline, since CAG constrains the gradient to be within a certain user-parametrised distance from the average gradient. In contrast, A-GEM and MEGA-II focus on preserving the base gradient and have no averaging step. These results were not entirely unexpected, since

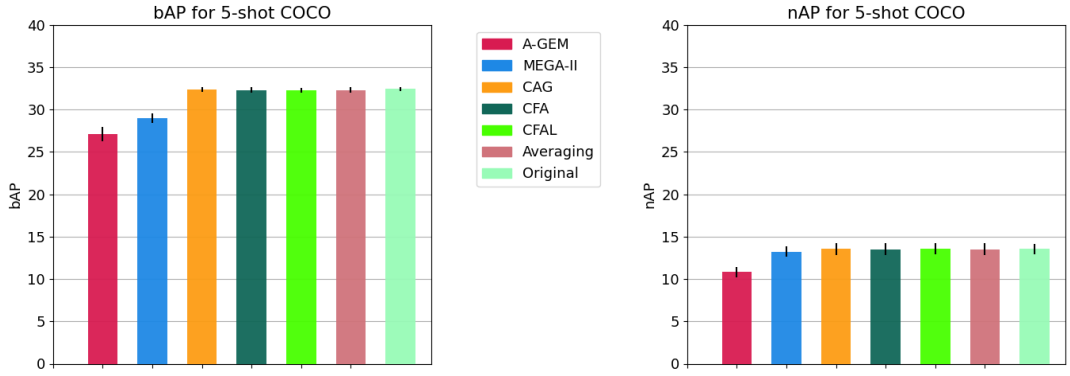


Figure 7.2: Gradient correction methods with memory replay: G-FSOD results on 10 COCO data splits. DeFRCN was used as the base method in every row.

CL methods devised in a Task-Incremental setting are known not to perform as well in a Class-Incremental [van de Ven et al., 2022] or Generalised setting such as in this case. The orthogonal constraint paradigm itself may be too restrictive when task descriptors are not provided and the model may predict from the full range of available classes.

Complete results are available in the appendix ( Section A.3).

| **Method** | **1-shot** | | **5-shot** | | **10-shot** | |
|---|---|---|---|---|---|---|
| | bAP | nAP | bAP | nAP | bAP | nAP |
| Original DeFRCN | 48.6 ± 0.8 | 24.1 ± 2.4 | 49.6 ± 0.4 | 38.1 ± 1.1 | 49.8 ± 0.3 | 40.0 ± 0.9 |
| A-GEM | 48.9 ± 0.7 | 25.4 ± 1.8 | 42.2 ± 0.8 | 36.6 ± 1.1 | 38.3 ± 1.9 | 38.7 ± 0.8 |
| CFA | 49.0 ± 0.8 | 25.8 ± 2.1 | 49.7 ± 0.5 | 38.5 ± 0.9 | 49.9 ± 0.2 | 40.5 ± 0.7 |
| CFA+Loss | 49.1 ± 0.8 | 25.5 ± 2.2 | 49.9 ± 0.4 | 37.6 ± 1.2 | 49.8 ± 0.2 | 39.1 ± 0.9 |
| MEGA-I | 41.6 ± 0.8 | 24.7 ± 1.9 | 40.4 ± 0.7 | 35.0 ± 2.0 | 37.7 ± 1.2 | 37.6 ± 1.7 |
| MEGA-II | 47.2 ± 1.1 | 24.0 ± 1.8 | 46.9 ± 0.7 | 32.7 ± 0.9 | 46.0 ± 0.5 | 34.1 ± 1.4 |
| CAG | 48.8 ± 0.8 | 25.5 ± 2.2 | 49.8 ± 0.4 | 38.5 ± 0.8 | 49.9 ± 0.2 | 40.5 ± 0.8 |
| Averaging | 49.1 ± 0.7 | 25.7 ± 2.1 | 49.8 ± 0.4 | 38.5 ± 0.9 | 49.9 ± 0.2 | 40.6 ± 0.8 |

Table 7.2: Gradient correction methods: G-FSOD base/novel AP (Average Precision) on 10 VOC data splits for class split VOC-1.

In the CL setting, we can see in Figure 7.3 that while A-GEM performs worse than both CFA and ER, CFA does not beat basic Experience Replay in a continual learning setting.

Guirguis et al. [Guirguis et al., 2022] postulated in CFA that the reason for A-GEM

| Method | 1-shot | | 5-shot | | 10-shot | |
|---|---|---|---|---|---|---|
| | bAP | nAP | bAP | nAP | bAP | nAP |
| Original DeFRCN | 30.3 ± 0.5 | 4.8 ± 0.6 | 32.4 ± 0.2 | 13.5 ± 0.6 | 33.9 ± 0.2 | 16.7 ± 0.6 |
| A-GEM | 28.2 ± 0.3 | 4.7 ± 0.6 | 27.1 ± 0.8 | 10.8 ± 0.6 | 27.7 ± 1.0 | 13.4 ± 0.4 |
| CFA | 30.1 ± 0.5 | 5.0 ± 0.5 | 32.3 ± 0.3 | 13.5 ± 0.6 | 33.7 ± 0.2 | 16.7 ± 0.5 |
| CFA+Loss | 30.2 ± 0.5 | 5.1 ± 0.6 | 32.3 ± 0.2 | 13.6 ± 0.6 | 33.7 ± 0.2 | 16.8 ± 0.5 |
| MEGA-II | 28.3 ± 0.7 | 5.1 ± 0.6 | 29.0 ± 0.5 | 13.2 ± 0.5 | 29.6 ± 0.3 | 16.4 ± 0.5 |
| CAG | 30.1 ± 0.5 | 4.9 ± 0.5 | 32.4 ± 0.3 | 13.5 ± 0.6 | 33.8 ± 0.2 | 16.7 ± 0.5 |
| Averaging | 30.1 ± 0.5 | 5.0 ± 0.6 | 32.3 ± 0.3 | 13.5 ± 0.6 | 33.7 ± 0.2 | 16.7 ± 0.5 |

Table 7.3: Gradient correction methods with memory replay: G-FSOD results on 10 COCO data splits. DeFRCN was used as the base method in every row.



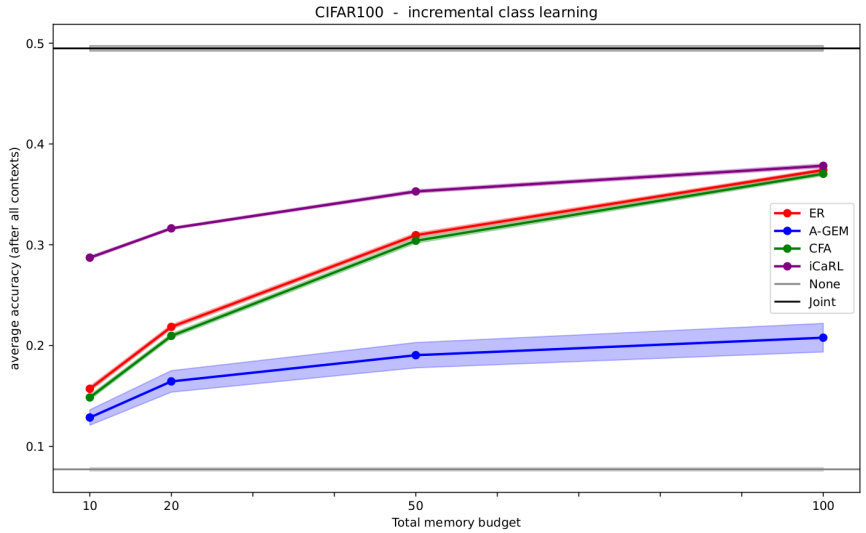


Figure 7.3: Average final accuracy across tasks for different CL methods on CIFAR100, with various memory budgets (logarithmic). We can see that while CFA improves upon A-GEM and does not suffer from high variance, it does not outperform simple ER in a class-incremental setting.

causing worse performance in FSOD was the constraint being too restrictive to learn base samples, while achieving good performance on old samples. Our experimental results contradict that line of reasoning in the FSOD setting (See Table 7.2), where a performance drop is observed on both base and novel classes. From a per-task breakdown in Table 7.4, we see that the A-GEM does not perform as well as the other methods on old tasks, despite reaching a similar accuracy on final tasks. This would match the conclusions drawn from an experiment performed by the Chaudhry et al. in Tiny Episodic Memories [Chaudhry et al., 2019b], where A-GEM was found to underfit memory samples, as its training accuracy on those in the MNIST never reached 100% while basic experience replay did so.

| Classes | A-GEM | CFA | ER | iCarl |
|---|---|---|---|---|
| 1-10 | 3.5 ± 2.2 | 3.3 ± 0.9 | 4.0 ± 1.1 | 15.0 ± 3.4 |
| 11-20 | 4.0 ± 1.3 | 4.6 ± 0.7 | 5.4 ± 0.6 | 15.6 ± 2.0 |
| 21-30 | 4.3 ± 2.8 | 5.3 ± 1.8 | 5.4 ± 1.9 | 20.6 ± 3.7 |
| 31-40 | 3.9 ± 1.7 | 5.7 ± 1.7 | 6.2 ± 2.1 | 19.2 ± 2.4 |
| 41-50 | 5.5 ± 3.2 | 6.1 ± 1.4 | 6.2 ± 1.4 | 20.1 ± 2.9 |
| 51-60 | 5.5 ± 2.6 | 6.9 ± 1.5 | 8.1 ± 1.9 | 25.0 ± 3.2 |
| 61-70 | 7.5 ± 2.6 | 11.6 ± 1.9 | 12.9 ± 2.1 | 31.9 ± 2.5 |
| 71-80 | 8.0 ± 2.4 | 13.4 ± 2.2 | 14.6 ± 2.3 | 39.5 ± 3.3 |
| 81-90 | 9.7 ± 4.2 | 14.7 ± 2.6 | 17.8 ± 2.1 | 46.5 ± 2.5 |
| 91-100 | 76.6 ± 4.4 | 76.9 ± 4.1 | 76.6 ± 4.1 | 53.8 ± 5.2 |

Table 7.4: Final class accuracy on 10-task CIFAR100 in a class-incremental setting, 20 samples per class, averaged over 10 seeds.



We performed a further experiment, changing the Continual Learning benchmark's original implementation by VanDeVen et al. to support few-shot fine-tuning, so we would have a confirmation on the behaviour of these methods in a fine-tuned classification setting. We only ran the model on 2 tasks, with the first one being the "base", large-scale one, and the second one the "novel" one. As shown by Table 7.5, when learning 10 shots per novel class, A-GEM completely forgot the 10 base shots stored in memory, while CFA and ER could cope with it. The method with the least forgetting was iCarl, although it also displayed the least plasticity when learning new class sets.

Interestingly, as shown by Table 7.6, neither EWC nor GPM achieved better per-

| Classes | A-GEM | CFA | ER | iCarl |
|---|---|---|---|---|
| 1-10 (Base) | 0.0 ± 0.0 | 40.7 ± 4.1 | 39.7 ± 4.4 | 69.7 ± 2.5 |
| 11-20 (Few-Shot) | 52.8 ± 3.1 | 46.4 ± 2.9 | 46.9 ± 2.8 | 31.9 ± 4.8 |

Table 7.5: Final class accuracy on few-shot fine-tuning scenario, with abundant Task-1 samples, 10 memory samples and 10 few-shot samples on Task-2. Results were averaged over 10 seeds.

formance than basic DeFRCN when base samples storage was unavailable. The chosen hyperparameters were lambda=0.1 for EWC, since the original lambda=0.4 led to too harsh modification penalties with stability issues, and t=0.97 for GPM, chosen arbitrarily. While the under-performance could be caused by the need to tune hyperparameters, these methods do not seem as effective with two tasks as they usually are in a continual learning setting.

| **Methods** | **1-shot** | | **5-shot** | |
|---|---|---|---|---|
| | bAP | nAP | bAP | nAP |
| EWC | 25.9 ± 0.9 | 9.2 ± 0.4 | 27.2 ± 1.1 | 11.9 ± 0.5 |
| GPM | 26.8 ± 0.9 | 10.9 ± 0.5 | 27.3 ± 1.0 | 13.4 ± 0.4 |

Table 7.6: Gradient correction methods with regularisation: G-FSOD results on 10 COCO data splits. DeFRCN was used as the base method in every row.



## 7.5 Performance of Sampling Strategies

We performed our experiments under the conditions described in Section 7.2 - Experimental Setup. We chose $k = 1, 5, 10$ shots for both VOC and COCO, without including 30 shots for COCO due to the already wide variety of experiments with respect to time constraints. For the same reason, we only performed the Experience Replay variant of the experiments on the "Original" baseline and Prototype Distance Ratio.

### 7.5.1 COCO

As we can see in Table 7.7, removing the class limit (Random) improves base AP substantially on when few instances are available (1, 5), but the impact is much lower

| Sampling Algorithm | 1-shot | | 5-shot | | 10-shot | |
|---|---|---|---|---|---|---|
| | bAP | nAP | bAP | nAP | bAP | nAP |
| Original (DeFRCN) | 30.3 ± 0.5 | 4.8 ± 0.6 | 32.4 ± 0.2 | 13.5 ± 0.6 | 33.9 ± 0.2 | 16.7 ± 0.6 |
| Original (DeFRCN) * | 30.4 ± 0.4 | 4.8 ± 0.6 | 32.6 ± 0.3 | 13.6 ± 0.7 | 34.0 ± 0.2 | 16.8 ± 0.6 |
| Original (DeFRCN) (ER) | 30.0 ± 0.3 | 4.4 ± 0.6 | 32.5 ± 0.3 | 13.6 ± 0.6 | 33.9 ± 0.3 | 16.7 ± 0.5 |
| Random (No Ranking) | 32.5 ± 0.3 | 5.5 ± 0.8 | 32.6 ± 0.3 | 14.6 ± 0.5 | 33.7 ± 0.4 | 18.2 ± 0.6 |
| ProtoDist (Bottom-K) | 33.2 ± 0.3 | 5.1 ± 0.8 | 32.1 ± 0.3 | 14.7 ± 0.7 | 33.4 ± 0.3 | 18.1 ± 0.5 |
| ProtoDist (Histogram) | 33.2 ± 0.3 | 5.3 ± 0.7 | 32.3 ± 0.3 | 14.7 ± 0.7 | 34.0 ± 0.2 | 18.1 ± 0.5 |
| ProtoDist Ratio (Bottom-K) | 33.1 ± 0.4 | 5.4 ± 0.7 | 33.0 ± 0.3 | 14.9 ± 0.5 | 34.3 ± 0.2 | 18.2 ± 0.5 |
| ProtoDist Ratio (Histogram) | 33.1 ± 0.4 | 5.4 ± 0.6 | 33.1 ± 0.2 | 14.8 ± 0.5 | 34.5 ± 0.2 | 18.1 ± 0.4 |
| ProtoDist Ratio (B-K) (ER) | 33.4 ± 0.4 | 5.4 ± 0.7 | 33.0 ± 0.3 | 14.9 ± 0.5 | 34.3 ± 0.2 | 18.3 ± 0.5 |

Table 7.7: **Replay Sampling** Strategies: G-FSOD **base/novel AP** on **COCO**, averaged over 10 data splits. The baseline labelled as "Original" uses the same random sampling and instance limit as TFA. (∗=paper's results)

in a 10-shot setting. See Figure 7.4 for an illustration.

Selecting images with instances closest to the class mean (ProtoDist Bottom-K) does not improve either bAP or nAP. Using histogram sampling can ensure a more even feature distribution on higher shots as seen in 10-shot ProtoDist Histogram.

The experiments show that the best performing strategy is to select images according to their Prototype Distance Ratio, since it is a clear winner on 5 and 10 shots. The use of bottom-k or histogram sampling does not influence this outcome, and neither does the use of separate batch loss emulating the original Experience Replay.

Interestingly, lifting the base instance cap improves performance on novel classes as well for COCO, despite the number of novel instance annotations remaining fixed. The novel classes with the highest AP gain were the ones with overlapping annotations such as "dining table", or easily confused ones such as "TV", which is often misclassified for the "microwave" base class. We believe this is the result of a relatively high number of classes (80) compared to the next dataset, VOC which only includes 20.

### 7.5.2 VOC

As shown in Table 7.8, lifitng the base cap without ranking provides a moderate improvement in bAP, but ranking the images greatly increases it, independently of whether Prototype Distance or Prototype Distance Ratio are used (Figure 7.5). This

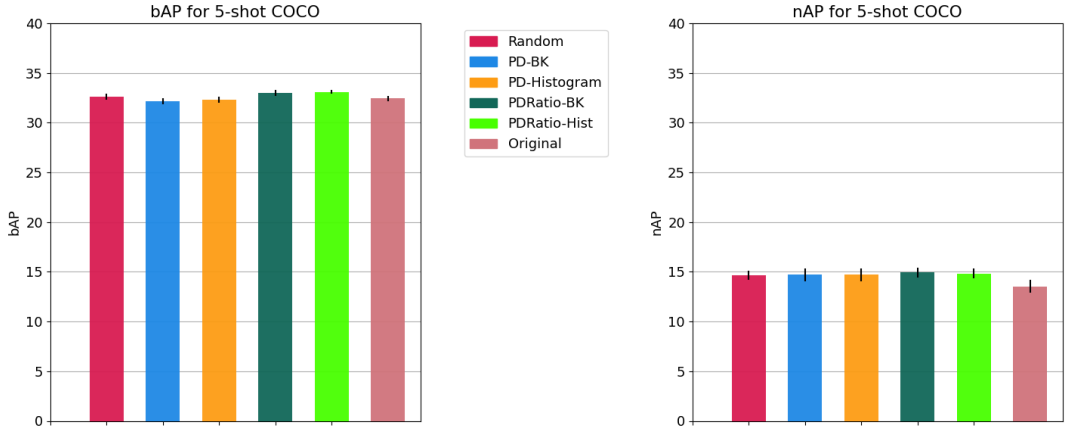


Figure 7.4: Sampling methods with memory replay: G-FSOD base/novel AP on 5-shot COCO, averaged over 10 data splits. DeFRCN was used as the base method in every row.The old sampling method is named "Original", unranked with class priority is "Random" and PD stands for Prototype Distance.



| **Sampling Algorithm** | **1-shot** | | **5-shot** | | **10-shot** | |
|---|---|---|---|---|---|---|
| | bAP | nAP | bAP | nAP | bAP | nAP |
| Original (DeFRCN) | 48.6 ± 1.0 | 24.1 ± 2.9 | 49.6 ± 0.5 | 38.1 ± 1.3 | 49.8 ± 0.3 | 40.0 ± 1.1 |
| Original (DeFRCN) * | 48.4 ± 0.4 | 22.5 ± 1.7 | 49.6 ± 0.3 | 37.3 ± 0.8 | 49.9 ± 0.2 | 39.8 ± 0.7 |
| Original (DeFRCN) (ER) | 49.0 ± 0.8 | 25.7 ± 2.1 | 49.8 ± 0.5 | 38.5 ± 0.9 | 50.0 ± 0.2 | 40.4 ± 0.9 |
| Random (No Ranking) | 50.3 ± 0.6 | 21.0 ± 3.1 | 50.5 ± 0.4 | 37.9 ± 1.2 | 50.4 ± 0.4 | 40.2 ± 1.2 |
| ProtoDist (Bottom-K) | 51.4 ± 0.4 | 22.7 ± 3.8 | 51.5 ± 0.3 | 37.7 ± 1.5 | 51.2 ± 0.1 | 39.4 ± 1.4 |
| ProtoDist (Histogram) | 51.4 ± 0.4 | 22.7 ± 3.8 | 51.1 ± 0.2 | 37.5 ± 1.2 | 51.0 ± 0.3 | 40.1 ± 0.8 |
| ProtoDist Ratio (Bottom-K) | 50.8 ± 0.3 | 21.7 ± 3.3 | 51.3 ± 0.3 | 37.9 ± 1.0 | 51.3 ± 0.2 | 39.9 ± 0.9 |
| ProtoDist Ratio (Histogram) | 50.8 ± 0.3 | 21.7 ± 3.3 | 51.1 ± 0.3 | 37.8 ± 1.2 | 51.1 ± 0.2 | 39.6 ± 0.9 |
| ProtoDist Ratio (Bottom-K) (ER) | 50.7 ± 0.4 | 24.8 ± 2.9 | 51.2 ± 0.3 | 38.4 ± 1.0 | 51.2 ± 0.2 | 40.4 ± 0.9 |

Table 7.8: Study of Replay Strategies: G-FSOD **base/novel AP** on VOC Split 1, averaged over 10 data splits. The DeFRCN baseline labelled as "Original" uses the same random sampling and instance limit as TFA. (∗=paper's results)

effect is maintained across 1, 5 and 10 shot settings, and Bottom-K vs Histogram sampling is unimportant.

Removing the instance cap causes novel AP to decrease in a 1-shot setting, albeit with a great overlap in confidence intervals, with the ranking strategies only partially restoring it. However, using the Experience Replay variant recovers and even improves novel AP, with the same effect observed in the "Original" setting. We attribute that to the balancing of base and novel inputs in their contribution to training loss.

Such a difference between the 1-shot setting and the others is not entirely surprising, considering the influence of spurious features is bound to decrease as more base and novel samples are employed in training, as we discussed in the Introduction (Chapter 1). It is also consistent with a previous experiment by Chaudhry et al. in Tiny Episodic Memories [Chaudhry et al., 2019b], which showed prototype-based sampling methods losing effectiveness over Reservoir sampling as the memory set per class increases in size. Full results are available in Appendix A.4.



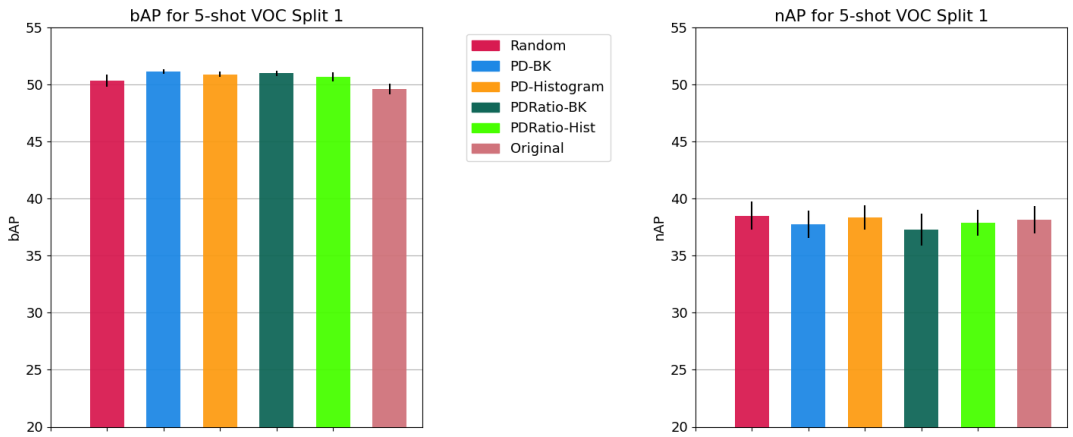


Figure 7.5: Sampling methods with memory replay: G-FSOD base/novel AP on 5-shot VOC-1, averaged over 10 data splits. DeFRCN was used as the base method in every row.

### 7.5.3 Ablation Study

The evaluations of the base set with the instance cap removed and different sampling techniques were reported separately in an ablation study to quantify the impact of both

| **Method** | Cls Priority | Limit | bAP | nAP |
|---|---|---|---|---|
| Original split | None | Yes | 49.6 ± 0.5 | 38.1 ± 1.3 |
| Original split | None | No | 51.5 ± 0.3 | 35.8 ± 1.2 |
| Random split (No ranking) | Rarest First | No | 50.5 ± 0.5 | 37.7 ± 1.5 |
| ProtoDist Ratio (Histogram), p=30 | Rarest First | No | 51.2 ± 0.3 | 38.1 ± 1.2 |
| ProtoDist Ratio (Histogram), p=100 | Rarest First | No | 51.1 ± 0.3 | 37.8 ± 1.2 |

Table 7.9: **Ablation Study**: G-FSOD **base/novel AP** on **VOC Split 1** with 5 shots, averaged over 10 data splits.

decisions. Additionally, we uncoupled the prioritisation of selecting rare classes from the removal of the base limit.

Furthermore, we chose to experiment on the size of the sample pool, changing it to to $p = 30$ and $p = 100$ when selecting images, to determine whether the same results could be achieved with a shorter inference time in the sampling phase before training, or if more images were needed for the prototypes. We provide results on 5-shot VOC Split 1 as it is faster to report results.

The results in Table 7.9 show that simply removing the base instance cap favours base AP but causes a drop in novel AP due to the excessive number of base instances. Creating a random split which prioritises rare classes helps address the imbalance, but loses some of the base AP gain. In contrast, our Prototype Distance Ratio method can improve base AP and keep novel AP stable. It is interesting to see that smaller instance pools can still create solid prototypes.

Overall, it is clear that softening the base instance limit and sampling images according to Prototype Distance Ratio is a viable method to increase base performance, and may also improve novel performance. However, in a one-shot setting, careful mini-batch balancing is crucial to avoid compromising novel class performance for base class gains.

# Chapter 8

# Conclusions

The two original aims of this work were:

- To evaluate the suitability of gradient correction methods for FSOD
- To explore the broader integration of CL methods into the field

After a literature survey, both points were investigated and, from the second one, additional research emerged into the effectiveness of herding sampling, as well as some observations on the effectiveness of current benchmarks.



## 8.1 Contributions

We provide a point-by-point summary of our contributions throughout this work, addressing the original objectives and further research.

### 8.1.1 Literature review

We provide a survey of the literature on few-shot learning, and how certain methods have developed across the fields of FSC and FSOD. We observe the streamlining of architectures and paradigms in both fields, as well as how some concepts such as meta-learning may not automatically carry over their benefits from one field to the other. We also provide an overview of Continual Learning methods and relate them to Generalised FSOD methods.

### 8.1.2 Parallels between Generalised Fine-Tuning and Class-Incremental Learning

We explain how experience replay is already part of the fine-tuning pipeline of popular FSOD methods in Section 6.5. We draw comparisons between methods that have made use of the same insights across the fields, such as ER and TFA for experience replay, Retentive R-CNN and ProgNN for expandable networks or BI-R and NIFF for VAE-based generative feature replay in Section 5.2. We hope that the fields can learn from each other's advancements given the similar challenges they face.

### 8.1.3 Gradient correction methods are ineffective

When the objective is to preserve previously learned features, gradient correction methods work in the Task-Incremental Learning setting, where the network knows which task a sample came from, but they fall flat in the Class-Incremental Learning setting where a sample could originate from any class. In CIL, respecting base gradient constraints is no better than replaying base data, as shown in Section 7.4, and it converges more slowly. We are reminded that the internal working of networks often differs from researchers' assumptions, exemplified by gradient correction methods not coping with relatively complex tasks.

### 8.1.4 Sampling strategies improve base performance on G-FSOD

Our proposed method (Prototype Distance Ratio) consistently obtained equal or better performance compared to the other methods we tested on the VOC and COCO datasets (Section 6.6). We can also infer that while Base/Novel set balance is important to a degree, reducing the underlying pool of base data down to the same number of instances should not be treated as a strict requirement, since mini-batch balancing is more important. Additionally, our methods do not require complete training of the network from the start and they do not need additional modules. They only run inference on $p$ samples for each class pool, thus demonstrating an efficient approach for performance improvement

### 8.1.5 Another look at current benchmarks

The current TFA-derived benchmark limits the number of base instances per sample instead of just number of base samples. While there was a concern about balancing, and later works have found RPN biased towards base samples, it is not enough to justify such a hard limit. While it is convenient to build upon other works, we should always look at older methods with a critical eye, and revise previous assumptions. Furthermore, the fields of CL and FSOD can take inspiration from neuroscience, since they have come to similar conclusions about the importance of replay in memory recall. However, current benchmarks aren't necessarily aligned to the goals of many realistic settings, since data is usually considered to be neatly separable into different categories which appear at the same time (See 5.2.4 and 4.1.1). This is partly the reason why the same methods can be applied across Class-Incremental Learning and Generalised Fine-Tuning.



## 8.2 Future Work

While we have provided new insights into the relationships between fine-tuning and continual learning methods, and their applicability to Generalised FSOD, our investigation was limited in scope due to time constraints. We propose some relevant areas where further analysis could lead to greater performance improvements.

### 8.2.1 Active Learning and G-FSOD

Methods of subset selection for base class samples have potential in a large-scale learning setting: Active Learning [Ren et al., 2022] is a field in machine learning which strives to maximise performance with a small portion of samples, often with the goal to shorten training time. In this context, sampling a subset is known as Core-set Selection [Aljundi et al., 2019].

Active Learning often uses on-line training metrics such as gradient statistics or the rate at which a network will forget a particular sample. This means that to apply Active Learning principles, base training should be started from scratch to collect relevant

statistics. This is different from our work, where we started from the perspective of an already trained model requiring few-shot fine-tuning. That said, since a few methods in G-FSOD such as DiGeo [Ma et al., 2023] already require full-scale training, it would be interesting to see how such works might benefit from Core-set Selection methods.

### 8.2.2 Sampling strategies

There are many sampling strategies which we did not test in this work, some of which may hold potential. A method known as Similarity-Weighted Interleaved Learning (SWIL) [Saxena et al., 2022] substantially reduced the amount of training data required by replaying past samples which had the highest cosine similarity to the novel classes after LDA. This allowed the network to make use of existing knowledge while focusing on discriminant features. However, we must mention that SWIL was used for classification on CIFAR100, and that running the same strategy on COCO or VOC could be hampered by badly labeled samples (mentioned in Section 3.7.1).

Additionally, the work of Tee & Zhang [Tee and Zhang, 2023] (mentioned in Section 6.6) explored base class selection with inter-sample distance as a difficulty metric. What we need to mention is they also investigated the effect of replaying easy examples during the first part of training, and gradually shift to harder ones for fine-tuning. This seems to have yielded a slight performance boost on CIFAR100, so it might be worth to investigate. The only downside is it would require storing more base samples for retraining the network, but this may not be a concern in an applicative setting.

Finally, while prototype-based sampling methods with balanced batches yielded better results in few-shot settings, we must not forget that a few classes appear in ImageNet as well, as we discussed in Section 3.7.2 - Concept Leakage. This means the network's backbone was already primed for them. It would be good to perform further sampling tests on a backbone trained without those classes, similar to Zhu et al.'s work [Zhu et al., 2021], to fully isolate the effect of the trialled strategies.

### 8.2.3 Resource Requirements

Since training including fine-tuning only takes a long time, reducing power consumption will be an important problem to solve. An important area to investigate is the reduction of training hardware requirements, since batches of 2x8 or 4x4 samples require at least multiple consumer-grade GPUs in parallel: even a single industrial-grade GPU will not be good enough due to large mini-batches decreasing performance as we've shown in Section 7.3. It would be useful to investigate training methods that improve upon the optimiser, as it has been previously done for Task-Incremental Learning [Mirzadeh et al., 2020].

We also hope that our revision of the G-FSOD benchmarks may help improve performance and shorten training times for transformer-hybrid methods such as DE-TReg [Bar et al., 2022], which are notoriously hard to fine-tune [Liu et al., 2021b].

# Appendix A

# Full Results

We include here some results which are required for completeness, but do not add any insights and would have encumbered the main body of the paper.

## A.1 Ablation Experiment: Gradient Averaging

In this experiment, we ask whether it is the averaging of the base and novel gradients which drives performance to be closer to the default DeFRCN setting. To do so, we simply change the A-GEM algorithm to always average the gradients if their angle is greater than 90 degrees, as done by CFA. However, when the angle is not acute we keep the default A-GEM projection step. Denoted by A-GEM + Averaging, this update rule is not the same as Hu et al.'s A-A-GEM [Hu et al., 2020] because the gradient averaging happens in the default case when no gradient correction is needed.

As shown in Table A.1, the performance of A-GEM + Averaging is the same as CFA and CAG, lending weight to the hypothesis that it's not enhanced projection schemes that preserve performance in a class-incremental or fine-tuning setting, but rather the simple averaging of gradients. In any case, using the data directly is more efficient as shown by *Vanilla* performance.

| Model | 5-shot bAP | 5-shot nAP | 10-shot bAP | 10-shot nAP |
|---|---|---|---|---|
| Vanilla | 49.6 ± 0.4 | 38.1 ± 1.1 | 49.8 ± 0.3 | 40.0 ± 0.9 |
| MEGA-I | 40.4 ± 0.7 | 35.0 ± 2.0 | 37.7 ± 1.2 | 37.6 ± 1.7 |
| A-GEM | 42.2 ± 0.8 | 36.6 ± 1.1 | 38.3 ± 1.9 | 38.7 ± 0.8 |
| MEGA-II | 46.9 ± 0.7 | 32.7 ± 0.9 | 46.0 ± 0.5 | 34.1 ± 1.4 |
| A-GEM + Averaging | 49.8 ± 0.4 | 38.6 ± 1.0 | 49.8 ± 0.2 | 40.4 ± 0.8 |
| Averaging-only | 49.8 ± 0.4 | 38.5 ± 0.9 | 49.9 ± 0.2 | 40.6 ± 0.8 |
| CFA | 49.7 ± 0.5 | 38.5 ± 0.9 | 49.9 ± 0.2 | 40.5 ± 0.7 |
| CAG | 49.8 ± 0.4 | 38.5 ± 0.8 | 49.9 ± 0.2 | 40.5 ± 0.8 |

Table A.1: Gradient methods: G-FSOD results on 10 VOC data splits. *Vanilla* indicates the default DeFRCN training process.

## A.2 Additional Results for Gradient Methods (COCO)

In Table A.2, we show that the same effect we observed in the main body of our work on up to 10 shots also holds in a 30-shot setting, which means gradient methods are not useful in when transitioning to larger scale learning. We did not run the 30-shot experiments on all methods, given the length of training required for averaging results over 10 different seeds.

| Model | 30-shot bAP | 30-shot nAP |
|---|---|---|
| Vanilla | 34.7 ± 0.1 | 21.0 ± 0.4 |
| A-GEM | 25.9 ± 0.9 | 16.3 ± 0.3 |
| CFA | 34.5 ± 0.1 | 21.2 ± 0.4 |
| MEGA-II | 27.8 ± 0.3 | 19.9 ± 0.3 |

Table A.2: Gradient methods: 30-shot G-FSOD results on 10 COCO data splits. DeFRCN was used as the base method in every row.

## A.3 Complete Results for Gradient Methods (VOC)

In this section, we include results for all VOC class splits for the sake of completeness. We only showed the first split in the main body since the full results do not provide

additional insights.

| Model | 1-shot bAP | 1-shot nAP | 5-shot bAP | 5-shot nAP | 10-shot bAP | 10-shot nAP |
|---|---|---|---|---|---|---|
| Vanilla DeFRCN | 48.6 ± 0.8 | 24.1 ± 2.4 | 49.6 ± 0.4 | 38.1 ± 1.1 | 49.8 ± 0.3 | 40.0 ± 0.9 |
| Experience Replay | 49.0 ± 0.8 | 25.7 ± 2.1 | 49.8 ± 0.5 | 38.5 ± 0.9 | 50.0 ± 0.2 | 40.4 ± 0.9 |
| A-GEM | 48.9 ± 0.7 | 25.4 ± 1.8 | 42.2 ± 0.8 | 36.6 ± 1.1 | 38.3 ± 1.9 | 38.7 ± 0.8 |
| CFA | 49.0 ± 0.8 | 25.8 ± 2.1 | 49.7 ± 0.5 | 38.5 ± 0.9 | 49.9 ± 0.2 | 40.5 ± 0.7 |
| CFA+Loss | 49.1 ± 0.8 | 25.5 ± 2.2 | 49.9 ± 0.4 | 37.6 ± 1.2 | 49.8 ± 0.2 | 39.1 ± 0.9 |
| MEGA-II | 47.2 ± 1.1 | 24.0 ± 1.8 | 46.9 ± 0.7 | 32.7 ± 0.9 | 46.0 ± 0.5 | 34.1 ± 1.4 |
| CAG | 48.8 ± 0.8 | 25.5 ± 2.2 | 49.8 ± 0.4 | 38.5 ± 0.8 | 49.9 ± 0.2 | 40.5 ± 0.8 |
| Averaging | 49.1 ± 0.7 | 25.7 ± 2.1 | 49.8 ± 0.4 | 38.5 ± 0.9 | 49.9 ± 0.2 | 40.6 ± 0.8 |

Table A.3: Gradient methods: G-FSOD results on 10 VOC data splits for class split VOC-1.

| Model | 1-shot bAP | 1-shot nAP | 5-shot bAP | 5-shot nAP | 10-shot bAP | 10-shot nAP |
|---|---|---|---|---|---|---|
| Vanilla | 49.5 ± 1.1 | 17.5 ± 2.1 | 50.5 ± 0.4 | 26.6 ± 0.9 | 50.9 ± 0.3 | 29.2 ± 0.8 |
| A-GEM | 49.5 ± 0.7 | 16.0 ± 2.2 | 42.9 ± 0.9 | 26.4 ± 0.9 | 39.2 ± 1.2 | 28.7 ± 0.6 |
| CFA | 49.8 ± 1.0 | 16.1 ± 2.2 | 50.7 ± 0.4 | 27.1 ± 1.0 | 51.0 ± 0.2 | 29.7 ± 0.7 |
| MEGA-II | 48.1 ± 1.1 | 14.7 ± 2.0 | 47.7 ± 0.6 | 24.0 ± 0.8 | 47.1 ± 0.6 | 25.8 ± 0.7 |
| CAG | 49.5 ± 1.1 | 15.9 ± 2.1 | 50.7 ± 0.4 | 27.1 ± 1.0 | 51.0 ± 0.3 | 29.5 ± 0.6 |
| Averaging | 49.8 ± 1.0 | 15.6 ± 2.2 | 50.9 ± 0.4 | 27.4 ± 0.9 | 51.0 ± 0.2 | 29.6 ± 0.3 |

Table A.4: Gradient methods: G-FSOD results on 10 VOC data splits, for class split VOC-2

| Model | 1-shot bAP | 1-shot nAP | 5-shot bAP | 5-shot nAP | 10-shot bAP | 10-shot nAP |
|---|---|---|---|---|---|---|
| Vanilla | 49.4 ± 0.7 | 19.9 ± 3.9 | 50.8 ± 0.4 | 32.9 ± 1.0 | 51.0 ± 0.3 | 35.8 ± 1.0 |
| A-GEM | 49.9 ± 0.5 | 20.3 ± 3.5 | 43.7 ± 1.2 | 31.5 ± 0.7 | 40.8 ± 1.4 | 34.5 ± 0.7 |
| CFA | 49.8 ± 0.5 | 20.8 ± 3.8 | 50.9 ± 0.3 | 33.1 ± 0.9 | 51.1 ± 0.3 | 36.0 ± 1.1 |
| MEGA-II | 47.9 ± 0.6 | 19.7 ± 3.5 | 48.0 ± 0.5 | 30.0 ± 1.3 | 47.0 ± 0.7 | 31.1 ± 1.0 |
| CAG | 49.6 ± 0.6 | 20.6 ± 3.8 | 50.9 ± 0.3 | 33.2 ± 1.0 | 51.2 ± 0.3 | 35.9 ± 1.0 |
| Averaging | 49.8 ± 0.6 | 21.0 ± 3.7 | 50.9 ± 0.3 | 33.0 ± 1.0 | 51.2 ± 0.3 | 36.0 ± 1.0 |

Table A.5: Gradient methods: G-FSOD results on 10 VOC data splits, for class split VOC-3

## A.4 Complete Results for Sampling Methods (VOC)

We included only VOC Split 1 for brevity in Section 6.6, so here we publish all three splits.

| **Sampling Algorithm** | **1-shot** | | **5-shot** | | **10-shot** | |
|---|---|---|---|---|---|---|
| | bAP | nAP | bAP | nAP | bAP | nAP |
| Original (DeFRCN) | 48.6 ± 1.0 | 24.1 ± 2.9 | 49.6 ± 0.5 | 38.1 ± 1.3 | 49.8 ± 0.3 | 40.0 ± 1.1 |
| Original (DeFRCN) * | 48.4 ± 0.4 | 22.5 ± 1.7 | 49.6 ± 0.3 | 37.3 ± 0.8 | 49.9 ± 0.2 | 39.8 ± 0.7 |
| Original (DeFRCN) (ER) | 49.0 ± 0.8 | 25.7 ± 2.1 | 49.8 ± 0.5 | 38.5 ± 0.9 | 50.0 ± 0.2 | 40.4 ± 0.9 |
| Random (No Ranking) | 50.3 ± 0.6 | 21.0 ± 3.1 | 50.5 ± 0.4 | 37.9 ± 1.2 | 50.4 ± 0.4 | 40.2 ± 1.2 |
| ProtoDist (Bottom-K) | 51.4 ± 0.4 | 22.7 ± 3.8 | 51.5 ± 0.3 | 37.7 ± 1.5 | 51.2 ± 0.1 | 39.4 ± 1.4 |
| ProtoDist (Histogram) | 51.4 ± 0.4 | 22.7 ± 3.8 | 51.1 ± 0.2 | 37.5 ± 1.2 | 51.0 ± 0.3 | 40.1 ± 0.8 |
| ProtoDist Ratio (Bottom-K) | 50.8 ± 0.3 | 21.7 ± 3.3 | 51.3 ± 0.3 | 37.9 ± 1.0 | 51.3 ± 0.2 | 39.9 ± 0.9 |
| ProtoDist Ratio (Histogram) | 50.8 ± 0.3 | 21.7 ± 3.3 | 51.1 ± 0.3 | 37.8 ± 1.2 | 51.1 ± 0.2 | 39.6 ± 0.9 |
| ProtoDist Ratio (Bot-K) (ER) | 50.7 ± 0.4 | 24.8 ± 2.9 | 51.2 ± 0.3 | 38.4 ± 1.0 | 51.2 ± 0.2 | 40.4 ± 0.9 |

Table A.6: Study of Replay Strategies: G-FSOD **base/novel AP** on VOC Split 1, averaged over 10 data splits. The DeFRCN baseline labelled as "Original" uses the same random sampling and instance limit as TFA. (∗=paper's results)

| Sampling Algorithm | 1-shot | | 5-shot | | 10-shot | |
|---|---|---|---|---|---|---|
| | bAP | nAP | bAP | nAP | bAP | nAP |
| Original (DeFRCN) | 49.5 ± 1.4 | 16.2 ± 2.5 | 50.5 ± 0.5 | 26.6 ± 1.0 | 50.9 ± 0.4 | 29.2 ± 1.0 |
| Original (DeFRCN) * | 49.6 ± 0.4 | 14.6 ± 1.5 | 51.0 ± 0.2 | 25.8 ± 0.9 | 51.3 ± 0.2 | 29.3 ± 0.7 |
| Original (DeFRCN) (ER) | 49.8 ± 1.1 | 16.1 ± 2.8 | 50.9 ± 0.6 | 26.8 ± 1.2 | 51.1 ± 0.4 | 29.4 ± 0.7 |
| Random (No Ranking) | 50.4 ± 0.4 | 13.1 ± 2.5 | 50.9 ± 0.6 | 25.8 ± 1.4 | 51.2 ± 0.4 | 28.9 ± 0.8 |
| ProtoDist (Bottom-K) | 51.8 ± 0.5 | 13.8 ± 2.6 | 51.8 ± 0.2 | 26.1 ± 1.2 | 52.3 ± 0.3 | 29.4 ± 0.8 |
| ProtoDist (Histogram) | 51.8 ± 0.5 | 13.8 ± 2.6 | 51.7 ± 0.3 | 26.1 ± 1.4 | 51.6 ± 0.3 | 29.6 ± 0.6 |
| ProtoDist Ratio (Bottom-K) | 51.9 ± 0.3 | 13.7 ± 2.8 | 51.6 ± 0.3 | 26.0 ± 1.4 | 52.3 ± 0.1 | 29.4 ± 0.8 |
| ProtoDist Ratio (Histogram) | 51.9 ± 0.3 | 13.7 ± 2.8 | 51.6 ± 0.3 | 26.1 ± 1.4 | 51.5 ± 0.3 | 29.6 ± 0.8 |
| ProtoDist Ratio (Bot-K) (ER) | 51.5 ± 0.5 | 15.6 ± 2.8 | 51.9 ± 0.3 | 27.0 ± 1.4 | 52.1 ± 0.2 | 30.0 ± 0.7 |

Table A.7: Study of Replay Strategies: G-FSOD **base/novel AP** on VOC Split 2, averaged over 10 data splits.



| Sampling Algorithm | 1-shot | | 5-shot | | 10-shot | |
|---|---|---|---|---|---|---|
| | bAP | nAP | bAP | nAP | bAP | nAP |
| Original (DeFRCN) | 49.4 ± 0.9 | 19.9 ± 4.7 | 50.8 ± 0.5 | 32.9 ± 1.2 | 51.0 ± 0.3 | 35.8 ± 1.3 |
| Original (DeFRCN) * | 49.4 ± 0.4 | 17.9 ± 1.6 | 51.0 ± 0.2 | 32.3 ± 0.9 | 51.3 ± 0.2 | 34.7 ± 0.7 |
| Original (DeFRCN) (ER) | 49.8 ± 0.7 | 21.0 ± 4.1 | 51.0 ± 0.5 | 33.4 ± 1.2 | 51.3 ± 0.4 | 35.9 ± 1.4 |
| Random (No Ranking) | 51.3 ± 0.4 | 16.4 ± 3.9 | 52.0 ± 0.2 | 33.2 ± 1.1 | 52.0 ± 0.2 | 35.4 ± 1.3 |
| ProtoDist (Bottom-K) | 51.6 ± 0.4 | 18.4 ± 4.7 | 52.0 ± 0.2 | 33.3 ± 1.1 | 52.0 ± 0.2 | 35.2 ± 1.3 |
| ProtoDist (Histogram) | 51.6 ± 0.4 | 18.4 ± 4.7 | 51.9 ± 0.4 | 33.5 ± 1.1 | 51.9 ± 0.3 | 35.5 ± 1.3 |
| ProtoDist Ratio (Bottom-K) | 51.8 ± 0.3 | 17.3 ± 4.2 | 52.0 ± 0.4 | 33.2 ± 1.1 | 52.1 ± 0.2 | 35.3 ± 1.3 |
| ProtoDist Ratio (Histogram) | 51.8 ± 0.3 | 17.3 ± 4.2 | 51.9 ± 0.3 | 33.0 ± 1.1 | 51.9 ± 0.3 | 35.5 ± 1.3 |
| ProtoDist Ratio (Bot-K) (ER) | 51.7 ± 0.4 | 20.9 ± 4.3 | 51.9 ± 0.4 | 33.7 ± 1.2 | 52.1 ± 0.3 | 35.8 ± 1.3 |

Table A.8: Study of Replay Strategies: G-FSOD **base/novel AP** on VOC Split 3, averaged over 10 data splits.

# Bibliography

[Ali-Gombe et al., 2018] Ali-Gombe, A., Elyan, E., Savoye, Y., and Jayne, C. (2018). Few-shot Classifier GAN. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

[Aljundi et al., 2019] Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. (2019). Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*, volume 32.

[Allen-Zhu and Li, 2020] Allen-Zhu, Z. and Li, Y. (2020). Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*.

[Andrychowicz et al., 2016] Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and De Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29.

[Antonelli et al., 2022] Antonelli, S., Avola, D., Cinque, L., Crisostomi, D., Foresti, G. L., Galasso, F., Marini, M. R., Mecca, A., and Pannone, D. (2022). Few-Shot Object Detection: A Survey. *ACM Computing Surveys*, 54(11s):1–37.

[Antoniou et al., 2017] Antoniou, A., Storkey, A., and Edwards, H. (2017). Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*.

[Ashok and Aekula, 2022] Ashok, A. and Aekula, H. (2022). When Does Self-supervision Improve Few-shot Learning? - A Reproducibility Report. In *ML Reproducibility Challenge 2021 (Fall Edition)*.

[Bar et al., 2022] Bar, A., Wang, X., Kantorov, V., Reed, C. J., Herzig, R., Chechik, G., Rohrbach, A., Darrell, T., and Globerson, A. (2022). DETReg: Unsupervised Pretraining with Region Priors for Object Detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2022-June.

[Bendou et al., 2022] Bendou, Y., Hu, Y., Lafargue, R., Lioi, G., Pasdeloup, B., Pateux, S., and Gripon, V. (2022). Easy—Ensemble Augmented-Shot-Y-Shaped Learning: State-of-The-Art Few-Shot Classification with Simple Components. *Journal of Imaging*, 8(7).

[Berahmand et al., 2024] Berahmand, K., Daneshfar, F., Salehi, E. S., Li, Y., and Xu, Y. (2024). Autoencoders and their applications in machine learning: a survey. *Artificial Intelligence Review*, 57(2):28.

[Bernico et al., 2019] Bernico, M., Li, Y., and Zhang, D. (2019). Investigating the impact of data volume and domain similarity on transfer learning applications. In *Advances in Intelligent Systems and Computing*, volume 881.

[Bertinetto et al., 2019] Bertinetto, L., Torr, P. H., Henriques, J., and Vedaldi, A. (2019). Meta-learning with differentiable closed-form solvers. In *7th International Conference on Learning Representations, ICLR 2019*.

[Blumer et al., 1987] Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1987). Occam's Razor. *Information Processing Letters*, 24(6):377–380.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine learning*, 24:123–140.

[Brereton, 2015] Brereton, R. G. (2015). The t-distribution and its relationship to the normal distribution. *Journal of Chemometrics*, 29(9).

[Bromley et al., 1993] Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., and Shah, R. (1993). Signature Verification using a "Siamese" Time Delay Neural Network. In

Cowan, J., Tesauro, G., and Alspector, J., editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan-Kaufmann.

[Bucilă et al., 2006] Bucilă, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, volume 2006.

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2).

[Buzzega et al., 2020] Buzzega, P., Boschini, M., Porrello, A., Abati, D., and Calderara, S. (2020). Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930.

[Cao et al., 2022] Cao, Y., Wang, J., Lin, Y., and Lin, D. (2022). MINI: Mining Implicit Novel Instances for Few-Shot Object Detection.

[Carion et al., 2020] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12346 LNCS.

[Caron et al., 2020] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. (2020). Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems*, volume 2020-December.

[Caron et al., 2021] Caron, M., Touvron, H., Misra, I., Jegou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging Properties in Self-Supervised Vision Transformers. In *Proceedings of the IEEE International Conference on Computer Vision*.

[Castro et al., 2018] Castro, F. M., Marín-Jiménez, M. J., Guil, N., Schmid, C., and Alahari, K. (2018). End-to-end incremental learning. In *Lecture Notes in Computer*

*Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 11216 LNCS.

[Chaudhry et al., 2019a] Chaudhry, A., Marc'Aurelio, R., Rohrbach, M., and Elhoseiny, M. (2019a). Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations, ICLR 2019*.

[Chaudhry et al., 2019b] Chaudhry, A., Rohrbach, M., Elhoseiny, M., Ajanthan, T., Dokania, P. K., Torr, P. H. S., and Ranzato, M. . A. (2019b). Continual Learning with Tiny Episodic Memories. In *ICML*, number Cl.

[Chen et al., 2018] Chen, H., Wang, Y., Wang, G., and Qiao, Y. (2018). LSTD: A low-shot transfer detector for object detection. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*.

[Chen et al., 2019a] Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., and others (2019a). MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

[Chen et al., 2020] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. (2020). A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

[Chen et al., 2019b] Chen, W. Y., Wang, Y. C. F., Liu, Y. C., Kira, Z., and Huang, J. B. (2019b). A closer look at few-shot classification. In *7th International Conference on Learning Representations, ICLR 2019*.

[Chen and He, 2020] Chen, X. and He, K. (2020). Exploring Simple Siamese Representation Learning. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15745–15753.

[Chopra et al., 2005] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546.

[Clouâtre and Demers, 2019] Clouâtre, L. and Demers, M. (2019). Figr: Few-shot image generation with reptile. *arXiv preprint arXiv:1901.02199*.

[Dai et al., 2021] Dai, Z., Cai, B., Lin, Y., and Chen, J. (2021). UP-DETR: Unsupervised Pre-training for Object Detection with Transformers. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893.

[Dawson et al., 2023] Dawson, H. L., Dubrule, O., and John, C. M. (2023). Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification. *Computers & Geosciences*, 171:105284.

[Dhillon et al., 2019] Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. (2019). A baseline for few-shot image classification. *arXiv preprint arXiv:1909.02729*.

[Domingos, 2012] Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78–87.

[Dong et al., 2022] Dong, X., Liao, S., Du, B., and Shao, L. (2022). Pseudo-Labeling Based Practical Semi-Supervised Meta-Training for Few-Shot Learning. *arXiv preprint arXiv:2207.06817*.

[Dvornik et al., 2019] Dvornik, N., Mairal, J., and Schmid, C. (2019). Diversity with cooperation: Ensemble methods for few-shot classification. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-October.

[Elsken et al., 2019] Elsken, T., Staffler, B. S., Metzen, J. H., and Hutter, F. (2019). Meta-Learning of Neural Architectures for Few-Shot Learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12362–12372.

[Everingham et al., 2010] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–338.

[Fan et al., 2021] Fan, Z., Ma, Y., Li, Z., and Sun, J. (2021). Generalized Few-Shot Object Detection Without Forgetting. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Fang et al., 2024] Fang, A., Kornblith, S., and Schmidt, L. (2024). Does progress on ImageNet transfer to real-world datasets? *Advances in Neural Information Processing Systems*, 36.

[Farajtabar et al., 2020] Farajtabar, M., Azizan, N., Mott, A., and Li, A. (2020). Orthogonal Gradient Descent for Continual Learning. In *Proceedings of Machine Learning Research*, volume 108.

[Fei-Fei et al., 2003] Fei-Fei, L., Fergus, R., and Perona, P. (2003). A Bayesian approach to unsupervised one-shot learning of object categories. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2.

[Feng et al., 2023] Feng, K., Xu, L., Zhao, D., Liu, S., and Huang, X. (2023). Toward Model Compression for a Deep Learning–Based Solar Flare Forecast on Satellites. *The Astrophysical Journal Supplement Series*, 268(2).

[Finn et al., 2017] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 3.

[Freund et al., 1999] Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612.

[Gao et al., 2022a] Gao, B.-B., Chen, X., Huang, Z., Nie, C., Liu, J., Lai, J., JIANG, G., Wang, X., and Wang, C. (2022a). Decoupling Classifier for Boosting Few-shot Object Detection and Instance Segmentation. In Koyejo, S., Mohamed, S., Agarwal,

A., Belgrave, D., Cho, K., and Oh, A., editors, *Advances in Neural Information Processing Systems*, volume 35, pages 18640–18652. Curran Associates, Inc.

[Gao et al., 2022b] Gao, B.-B., Chen, X., Huang, Z., Nie, C., Liu, J., Lai, J., JIANG, G., Wang, X., and Wang, C. (2022b). Decoupling Classifier for Boosting Few-shot Object Detection and Instance Segmentation (Review). In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K., editors, *Advances in Neural Information Processing Systems*.

[Gidaris et al., 2019] Gidaris, S., Bursuc, A., Komodakis, N., Perez, P. P., and Cord, M. (2019). Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-October.

[Gidaris et al., 2018] Gidaris, S., Singh, P., and Komodakis, N. (2018). Unsupervised representation learning by predicting image rotations. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

[Girshick et al., 2014] 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 Computer Society Conference on Computer Vision and Pattern Recognition*.

[Goodfellow et al., 2014a] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. *Advances in neural information processing systems*, 27.

[Goodfellow et al., 2014b] Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2014b). An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*.

[Grill et al., 2020] Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. A., Guo, Z. D., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. (2020). Bootstrap your own latent a

new approach to self-supervised learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

[Guirguis et al., 2022] Guirguis, K., Hendawy, A., Eskandar, G., Abdelsamad, M., Kayser, M., and Beyerer, J. (2022). CFA: Constraint-based Finetuning Approach for Generalized Few-Shot Object Detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2022-June.

[Guirguis et al., 2023] Guirguis, K., Meier, J., Eskandar, G., Kayser, M., Yang, B., and Beyerer, J. (2023). NIFF: Alleviating Forgetting in Generalized Few-Shot Object Detection via Neural Instance Feature Forging. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2023-June.

[Guo et al., 2020a] Guo, Y., Codella, N. C., Karlinsky, L., Codella, J. V., Smith, J. R., Saenko, K., Rosing, T., and Feris, R. (2020a). A Broader Study of Cross-Domain Few-Shot Learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12372 LNCS.

[Guo et al., 2020b] Guo, Y., Liu, M., Yang, T., and Rosing, T. (2020b). Improved schemes for episodic memory-based lifelong learning. In *Advances in Neural Information Processing Systems*, volume 2020-December.

[Gutmann and Hyvärinen, 2010] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Teh, Y. W. and Titterington, M., editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 297–304, Chia Laguna Resort, Sardinia, Italy. PMLR.

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society*

*conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742.

[Han et al., 2022a] Han, G., Huang, S., Ma, J., He, Y., and Chang, S. F. (2022a). Meta Faster R-CNN: Towards Accurate Few-Shot Object Detection with Attentive Feature Alignment. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence, AAAI 2022*, volume 36.

[Han et al., 2022b] Han, G., Ma, J., Huang, S., Chen, L., and Chang, S. F. (2022b). Few-Shot Object Detection with Fully Cross-Transformer. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2022-June.

[Han et al., 2023] Han, J., Ren, Y., Ding, J., Yan, K., and Xia, G. S. (2023). Few-Shot Object Detection via Variational Feature Aggregation. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, volume 37.

[Hayes et al., 2021] Hayes, T. L., Krishnan, G. P., Bazhenov, M., Siegelmann, H. T., Sejnowski, T. J., and Kanan, C. (2021). Replay in deep learning: Current approaches and missing biological elements. *Neural Computation*, 33(11).

[He et al., 2020] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. (2020). Momentum Contrast for Unsupervised Visual Representation Learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[He et al., 2019] He, Z., Zhang, T., and Lee, R. B. (2019). Model inversion attacks against collaborative inference. In *ACM International Conference Proceeding Series*.

[Helber et al., 2019] Helber, P., Bischke, B., Dengel, A., and Borth, D. (2019). Eurosat: A novel dataset and deep learning benchmark for land use and land cover

classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7).

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Hosang et al., 2017] Hosang, J., Benenson, R., and Schiele, B. (2017). Learning non-maximum suppression. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January.

[Hsu et al., 2019] Hsu, K., Levine, S., and Finn, C. (2019). Unsupervised learning via meta-learning. In *7th International Conference on Learning Representations, ICLR 2019*.

[Hu et al., 2020] Hu, G., Zhang, W., Ding, H., and Zhu, W. (2020). Gradient episodic memory with a soft constraint for continual learning. *arXiv preprint arXiv:2011.07801*.

[Huang et al., 2016] Huang, G., Liu, Z., and Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

[Huang et al., 2023] Huang, Q., Zhang, H., Xue, M., Song, J., and Song, M. (2023). A survey of deep learning for low-shot object detection. *ACM Computing Surveys*, 56(5):1–37.

[Hung et al., 2019] Hung, S. C., Tu, C. H., Wu, C. E., Chen, C. H., Chan, Y. M., and Chen, C. S. (2019). Compacting, picking and growing for unforgetting continual learning. In *Advances in Neural Information Processing Systems*, volume 32.

[Jia Deng et al., 2009] Jia Deng, Wei Dong, Socher, R., Li-Jia Li, Kai Li, and Li Fei-Fei (2009). ImageNet: A large-scale hierarchical image database.

[Kalantidis et al., 2020] Kalantidis, Y., Sariyildiz, M. B., Pion, N., Weinzaepfel, P., and Larlus, D. (2020). Hard negative mixing for contrastive learning. In *Proceedings of the*

*34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

[Kang et al., 2019] Kang, B., Liu, Z., Wang, X., Yu, F., Feng, J., and Darrell, T. (2019). Few-shot object detection via feature reweighting. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-October.

[Karlinsky et al., 2019] Karlinsky, L., Shtok, J., Harary, S., Schwartz, E., Aides, A., Feris, R., Giryes, R., and Bronstein, A. M. (2019). Repmet: Representative-based metric learning for classification and few-shot object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June.

[Kim et al., 2021] Kim, T., Oh, J., Kim, N. Y., Cho, S., and Yun, S. Y. (2021). Comparing Kullback-Leibler Divergence and Mean Squared Error Loss in Knowledge Distillation. In *IJCAI International Joint Conference on Artificial Intelligence*.

[Kirkpatrick et al., 2017] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences of the United States of America*, 114(13).

[Koch, 2015] Koch, G. R. (2015). Siamese Neural Networks for One-Shot Image Recognition. In *ICML deep learning workshop, vol. 2. 2015. 2015*.

[Kothapalli, 2022] Kothapalli, V. (2022). Neural collapse: A review on modelling principles and generalization. *arXiv preprint arXiv:2206.04041*.

[Krizhevsky et al., ] Krizhevsky, A., Nair, V., and Hinton, G. CIFAR-10 (Canadian Institute for Advanced Research).

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

[Kukleva et al., 2022] Kukleva, A., Kuehne, H., and Schiele, B. (2022). Generalized and Incremental Few-Shot Learning by Explicit Learning and Calibration without Forgetting.

[Lake et al., 2015] Lake, B. M., Salakhutdinov, R., and Tenenbaum, J. B. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266).

[Lampert et al., 2009] Lampert, C. H., Nickisch, H., and Harmeling, S. (2009). Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009*.

[LeCun et al., 1998] LeCun, Y., Cortes, C., and Burges, J. (1998). The MNIST database of handwritten digits.

[Li et al., 2021a] Li, B., Yang, B., Liu, C., Liu, F., Ji, R., and Ye, Q. (2021a). Beyond Max-Margin: Class Margin Equilibrium for Few-shot Object Detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Li et al., 2020a] Li, H., Jiang, H., Gu, X., Peng, J., Li, W., Hong, L., and Tao, C. (2020a). CLRS: Continual Learning Benchmark for Remote Sensing Image Scene Classification. *Sensors*, 20(4).

[Li et al., 2020b] Li, J., Zhou, P., Xiong, C., and Hoi, S. C. H. (2020b). Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*.

[Li and Talwalkar, 2020] Li, L. and Talwalkar, A. (2020). Random search and reproducibility for neural architecture search. In *Uncertainty in artificial intelligence*, pages 367–377.

[Li et al., 2023a] Li, Y., Guo, L., and Ge, Y. (2023a). Pseudo Labels for Unsupervised Domain Adaptation: A Review.

[Li et al., 2021b] Li, Y., Zhu, H., Cheng, Y., Wang, W., Teo, C. S., Xiang, C., Vadakkepat, P., and Lee, T. H. (2021b). Few-shot object detection via classification refinement and distractor retreatment. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Li and Hoiem, 2017] Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

[Li et al., 2023b] Li, Z., Shang, X., He, R., Lin, T., and Wu, C. (2023b). No Fear of Classifier Biases: Neural Collapse Inspired Federated Learning with Synthetic and Fixed Classifier. In *Proceedings of the IEEE International Conference on Computer Vision*.

[Li et al., 2017] Li, Z., Zhou, F., Chen, F., and Li, H. (2017). Meta-SGD: Learning to Learn Quickly for Few Shot Learning. *ArXiv*, abs/1707.09835.

[Liao et al., 2021] Liao, T., Taori, R., Raji, I. D., and Schmidt, L. (2021). Are we learning yet? a meta review of evaluation failures across machine learning. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[Lin et al., 2023] Lin, S., Wang, K., Zeng, X., and Zhao, R. (2023). An Effective Crop-Paste Pipeline for Few-shot Object Detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2023-June.

[Lin et al., 2014] 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*, pages 740–755.

[Lin et al., 2021] Lin, Z., Shi, J., Pathak, D., and Ramanan, D. (2021). The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth conference on neural information processing systems datasets and benchmarks track (round 2)*.

[Liu et al., 2021a] Liu, B., Liu, X., Jin, X., Stone, P., and Liu, Q. (2021a). Conflict-Averse Gradient Descent for Multi-task Learning. In *Advances in Neural Information Processing Systems*, volume 23.

[Liu et al., 2022] Liu, G., Wang, T., Zhang, S., and He, K. (2022). Generating Pseudo-labels Adaptively for Few-shot Model-Agnostic Meta-Learning. *arXiv preprint arXiv:2207.04217*.

[Liu et al., 2023] Liu, T., Zhang, L., Wang, Y., Guan, J., Fu, Y., Zhao, J., and Zhou, S. (2023). Recent Few-shot Object Detection Algorithms: A Survey with Performance Comparison. *ACM Transactions on Intelligent Systems and Technology*, 14(4).

[Liu et al., 2016] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9905 LNCS.

[Liu et al., 2025] Liu, W., Wu, X.-J., Zhu, F., Yu, M.-M., Wang, C., and Liu, C.-L. (2025). Class incremental learning with self-supervised pre-training and prototype learning. *Pattern Recognition*, 157:110943.

[Liu et al., 2020] Liu, X., Wu, C., Menta, M., Herranz, L., Raducanu, B., Bagdanov, A. D., Jui, S., and Van De Weijer, J. (2020). Generative feature replay for class-incremental learning. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2020-June.

[Liu et al., 2019] Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S. J., and Yang, Y. (2019). Learning to propagate labels: Transductive propagation network for few-shot learning. In *7th International Conference on Learning Representations, ICLR 2019*.

[Liu et al., 2021b] Liu, Y., Sangineto, E., Bi, W., Sebe, N., Lepri, B., and De Nadai, M. (2021b). Efficient Training of Visual Transformers with Small Datasets. In *Advances in Neural Information Processing Systems*, volume 29.

[Liu Jinlu, 2020] Liu Jinlu, Song Liang, Q. Y. (2020). Prototype Rectification for Few-Shot Learning. In Vedaldi Andrea
}and Bischof, H., Thomas, B., and Jan-Michael, F., editors, *Computer Vision – ECCV 2020*, pages 741–756, Cham. Springer International Publishing.

[Lopez-Paz and Ranzato, 2017] Lopez-Paz, D. and Ranzato, M. (2017). Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, volume 2017-December.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2).

[Lu et al., 2022] Lu, Y., Wen, L., Liu, J., Liu, Y., and Tian, X. (2022). Self-Supervision Can Be a Good Few-Shot Learner. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 13679 LNCS.

[Luo et al., 2019] Luo, C., Li, X., Wang, L., He, J., Li, D., and Zhou, J. (2019). How Does the Data set Affect CNN-based Image Classification Performance? *2018 5th International Conference on Systems and Informatics, ICSAI 2018*, pages 361–366.

[Luo et al., 2020] Luo, Y., Yin, L., Bai, W., and Mao, K. (2020). An appraisal of incremental learning methods.

[Ma et al., 2023] Ma, J., Niu, Y., Xu, J., Huang, S., Han, G., and Chang, S. F. (2023). DiGeo: Discriminative Geometry-Aware Learning for Generalized Few-Shot Object Detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2023-June.

[Mallya and Lazebnik, 2018] Mallya, A. and Lazebnik, S. (2018). PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Masana et al., 2022] Masana, M., Liu, X., Twardowski, B., Menta, M., Bagdanov, A. D., and Van De Weijer, J. (2022). Class-incremental learning: survey and perfor-

mance evaluation on image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):5513–5533.

[Mirzadeh et al., 2020] Mirzadeh, S. I., Farajtabar, M., Pascanu, R., and Ghasemzadeh, H. (2020). Understanding the role of training regimes in continual learning. In *Advances in Neural Information Processing Systems*, volume 2020-December.

[Misra and Maaten, 2020] Misra, I. and Maaten, L. v. d. (2020). Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6707–6717.

[Netzer et al., 2011] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A. Y., and others (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7.

[Nguyen and Todorovic, 2019] Nguyen, K. D. M. and Todorovic, S. (2019). Feature Weighting and Boosting for Few-Shot Segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 622–631.

[Nichol et al., 2018] Nichol, A., Achiam, J., and Schulman, J. (2018). On First-Order Meta-Learning Algorithms. *ArXiv*, abs/1803.02999.

[Nilsback and Zisserman, 2008] Nilsback, M. E. and Zisserman, A. (2008). Automated flower classification over a large number of classes. In *Proceedings - 6th Indian Conference on Computer Vision, Graphics and Image Processing, ICVGIP 2008*.

[Noroozi and Favaro, 2016] Noroozi, M. and Favaro, P. (2016). Unsupervised learning of visual representations by solving jigsaw puzzles. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9910 LNCS.

[Ojala et al., 2002] Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987.

[Ouyang et al., 2023] Ouyang, Y., Wang, X. q., Hu, R. z., and Xu, H. h. (2023). Few-shot object detection based on positive-sample improvement. *Defence Technology*, 28.

[Pfeiffer, 2020] Pfeiffer, B. E. (2020). The content of hippocampal "replay".

[Portugal et al., 2018] Portugal, I., Alencar, P., and Cowan, D. (2018). The use of machine learning algorithms in recommender systems: A systematic review. *Expert Systems with Applications*, 97:205–227.

[Prabhu et al., 2023] Prabhu, A., Al Kader Hammoud, H. A., Dokania, P. K., Torr, P. H. S., Lim, S.-N., Ghanem, B., and Bibi, A. (2023). Computationally budgeted continual learning: What does matter? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3698–3707.

[Prabhu et al., 2020] Prabhu, A., Torr, P. H., and Dokania, P. K. (2020). GDumb: A Simple Approach that Questions Our Progress in Continual Learning. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12347 LNCS.

[Qiao et al., 2021] Qiao, L., Zhao, Y., Li, Z., Qiu, X., Wu, J., and Zhang, C. (2021). DeFRCN: Decoupled Faster R-CNN for Few-Shot Object Detection. In *Proceedings of the IEEE International Conference on Computer Vision*.

[Radford et al., 2015] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[Raghu et al., 2019] Raghu, A., Raghu, M., Bengio, S., and Vinyals, O. (2019). Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. *ArXiv*, abs/1909.09157.

[Rahman et al., 2021] Rahman, M., Prodhan, R., Shishir, Y., and Ripon, S. (2021). Analyzing and Evaluating Boosting-Based CNN Algorithms for Image Classification. In *2021 International Conference on Intelligent Technologies (CONIT)*, pages 1–6.

[Raina et al., 2007] Raina, R., Battle, A., Lee, H., Packer, B., and Ng, A. Y. (2007). Self-taught learning: Transfer learning from unlabeled data. In *ACM International Conference Proceeding Series*, volume 227.

[Ravi and Larochelle, 2017] Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

[Rebuffi et al., 2017] Rebuffi, S. A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). iCaRL: Incremental classifier and representation learning. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-December.

[Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January.

[Redmon and Farhadi, 2018] Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[Ren et al., 2018] Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. (2018). Meta-learning for semi-supervised few-shot classification. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*.

[Ren et al., 2022] Ren, P., Xiao, Y., Chang, X., Huang, P. Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. (2022). A Survey of Deep Active Learning.

[Ren et al., 2017] Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6).

[Richard Bellman, 1957] Richard Bellman (1957). Dynamic Programming. *Princeton University Press*.

[Riemer et al., 2019] Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. (2019). Learning to learn without forgetting by maximizing transfer and minimizing interference. In *7th International Conference on Learning Representations, ICLR 2019*.

[Robb et al., 2020] Robb, E., Chu, W.-S., Kumar, A., and Huang, J.-B. (2020). Few-shot adaptation of generative adversarial networks. *arXiv preprint arXiv:2010.11943*.

[Rodríguez et al., 2020] Rodríguez, P., Laradji, I., Drouin, A., and Lacoste, A. (2020). Embedding Propagation: Smoother Manifold for Few-Shot Classification. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12371 LNCS.

[Rolnick et al., 2019] Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. P., and Wayne, G. (2019). Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32.

[Rubner et al., 2000] Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). Earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2).

[Rusu et al., 2016] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

[Rusu et al., 2019] Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2019). Meta-learning with latent embedding optimization. In *7th International Conference on Learning Representations, ICLR 2019*.

[Rypeść et al., 2024] Rypeść, G., Cygert, S., Khan, V., Trzciński, T., Zieliński, B., and Twardowski, B. (2024). Divide and not forget: Ensemble of selectively trained experts in Continual Learning. *arXiv preprint arXiv:2401.10191*.

[Saha et al., 2021] Saha, G., Garg, I., and Roy, K. (2021). GRADIENT PROJECTION MEMORY FOR CONTINUAL LEARNING. In *ICLR 2021 - 9th International Conference on Learning Representations*.

[Santoro et al., 2016] Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. (2016). Meta-Learning with Memory-Augmented Neural Networks. In *33rd International Conference on Machine Learning, ICML 2016*, volume 4.

[Santoro et al., 2017] Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30.

[Santra et al., 2021] Santra, S., Hsieh, J. W., and Lin, C. F. (2021). Gradient Descent Effects on Differential Neural Architecture Search: A Survey. *IEEE Access*, 9.

[Saxena et al., 2022] Saxena, R., Shobe, J. L., and McNaughton, B. L. (2022). Learning in deep neural networks and brains with similarity-weighted interleaved learning. *Proceedings of the National Academy of Sciences of the United States of America*, 119(27).

[Schonfeld et al., 2019] Schonfeld, E., Ebrahimi, S., Sinha, S., Darrell, T., and Akata, Z. (2019). Generalized zero-and few-shot learning via aligned variational autoencoders. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8247–8255.

[Schroff et al., 2015] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

[Schwarz et al., 2018] Schwarz, J., Altman, D., Dudzik, A., Vinyals, O., Whye Teh, Y., and Pascanu, R. (2018). Towards a natural benchmark for continual learning. *Continual learning Workshop NeurIPS*, (Cl).

[Serra et al., 2018] Serra, J., Suris, D., Mirón, M., and Karatzoglou, A. (2018). Overcoming Catastrophic forgetting with hard attention to the task. In *35th International Conference on Machine Learning, ICML 2018*, volume 10.

[Shi et al., 2020] Shi, X., Salewski, L., Schiegg, M., and Welling, M. (2020). Relational Generalized Few-Shot Learning. In *31st British Machine Vision Conference, BMVC 2020*.

[Shin et al., 2017] Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, volume 2017-December.

[Shors et al., 2012] Shors, T. J., Anderson, M. L., Curlik, D. M., and Nokia, M. S. (2012). Use it or lose it: How neurogenesis keeps the brain fit for learning.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *CoRR*, abs/1409.1556.

[Snell et al., 2017] Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical Networks for Few-shot Learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

[Song et al., 2022] Song, Y., Wang, T.-Y., Cai, P., Mondal, S. K., and Sahoo, J. P. (2022). A Comprehensive Survey of Few-shot Learning: Evolution, Applications, Challenges, and Opportunities. *ACM Computing Surveys*, 55:1 – 40.

[Springenberg, 2015] Springenberg, J. T. (2015). Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks. *CoRR*, abs/1511.06390.

[Su et al., 2020] Su, J. C., Maji, S., and Hariharan, B. (2020). When Does Self-supervision Improve Few-Shot Learning? In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12352 LNCS.

[Suárez et al., 2021] Suárez, J. L., García, S., and Herrera, F. (2021). A tutorial on distance metric learning: Mathematical foundations, algorithms, experimental analysis, prospects and challenges. *Neurocomputing*, 425:300–322.

[Sun et al., 2021a] Sun, B., Li, B., Cai, S., Yuan, Y., and Zhang, C. (2021a). FSCE: Few-Shot Object Detection via Contrastive Proposal Encoding. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Sun et al., 2019] Sun, Q., Liu, Y., Chua, T. S., and Schiele, B. (2019). Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2019-June.

[Sun et al., 2021b] Sun, X., Wang, B., Wang, Z., Li, H., Li, H., and Fu, K. (2021b). Research Progress on Few-Shot Learning for Remote Sensing Image Interpretation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14:2387–2402.

[Sung et al., 2018] Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P. H., and Hospedales, T. M. (2018). Learning to Compare: Relation Network for Few-Shot Learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Szabo and Horvath, 2022] Szabo, G. and Horvath, A. (2022). Mitigating the Bias of Centered Objects in Common Datasets. In *Proceedings - International Conference on Pattern Recognition*, volume 2022-August.

[Taherkhani et al., 2020] Taherkhani, A., Cosma, G., and McGinnity, T. M. (2020). AdaBoost-CNN: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, 404:351–366.

[Tee and Zhang, 2023] Tee, R. J. and Zhang, M. (2023). Integrating Curricula with Replays: Its Effects on Continual Learning. In *Proceedings of the Inaugural 2023 Summer Symposium Series 2023*.

[Tian et al., 2020] Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. (2020). Rethinking Few-Shot Image Classification: A Good Embedding is All You Need? In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV*, pages 266–282, Berlin, Heidelberg. Springer-Verlag.

[Tian et al., 2022] Tian, Z., Zhao, H., Shu, M., Yang, Z., Li, R., and Jia, J. (2022). Prior Guided Feature Enrichment Network for Few-Shot Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(2):1050–1065.

[Tong and Wu, 2023] Tong, K. and Wu, Y. (2023). Rethinking PASCAL-VOC and MS-COCO dataset for small object detection. *Journal of Visual Communication and Image Representation*, 93.

[Triantafillou et al., 2020] Triantafillou, E., Zhu, T., Dumoulin, V., Lamblin, P., Evci, U., Xu, K., Goroshin, R., Gelada, C., Swersky, K., Manzagol, P. A., and Larochelle, H. (2020). Meta-Dataset: A Dataset Of Datasets For Learning to Learn from Few Examples. In *8th International Conference on Learning Representations, ICLR 2020*.

[Uijlings et al., 2013] 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(2).

[Van De Ven et al., 2021] Van De Ven, G. M., Li, Z., and Tolias, A. S. (2021). Class-incremental learning with generative classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3611–3620.

[Van de Ven et al., 2020] Van de Ven, G. M., Siegelmann, H. T., and Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature communications*, 11(1):4069.

[van de Ven et al., 2022] van de Ven, G. M., Tuytelaars, T., and Tolias, A. S. (2022). Three types of incremental learning. *Nature Machine Intelligence*, 4(12).

[van den Oord et al., 2018] van den Oord, A., Li, Y., and Vinyals, O. (2018). Representation Learning with Contrastive Predictive Coding. *ArXiv*, abs/1807.03748.

[Veilleux et al., 2021] Veilleux, O., Boudiaf, M., Piantanida, P., and Ayed, I. B. (2021). Realistic Evaluation of Transductive Few-Shot Learning. In *Advances in Neural Information Processing Systems*, volume 12.

[Vilalta and Drissi, 2002] Vilalta, R. and Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial intelligence review*, 18:77–95.

[Vinyals et al., 2016] Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. (2016). Matching Networks for One Shot Learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

[Walach and Wolf, 2016] Walach, E. and Wolf, L. (2016). Learning to Count with CNN Boosting. In *European Conference on Computer Vision*.

[Wang et al., 2023] Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2023). YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors.

[Wang et al., 2022] Wang, F. Y., Zhou, D. W., Ye, H. J., and Zhan, D. C. (2022). FOSTER: Feature Boosting and Compression for Class-Incremental Learning. In

*Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 13685 LNCS.

[Wang et al., 2020a] Wang, J., Wu, J., Bai, H., and Cheng, J. (2020a). M-nas: Meta neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6186–6193.

[Wang et al., 2020b] Wang, X., Huang, T. E., Darrell, T., Gonzalez, J. E., and Yu, F. (2020b). Frustratingly simple few-shot object detection. In *37th International Conference on Machine Learning, ICML 2020*, volume PartF168147-13.

[Wang et al., 2017] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. (2017). ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-January.

[Wang et al., 2024] Wang, Z., Yang, B., Yue, H., and Ma, Z. (2024). Fine-Grained Prototypes Distillation for Few-Shot Object Detection. *Proceedings of the 38th AAAI Conference on Artificial Intelligence (AAAI-24)*.

[Wei et al., 2022] Wei, X.-S., Xu, H.-Y., Zhang, F., Peng, Y., and Zhou, W. (2022). An Embarrassingly Simple Approach to Semi-Supervised Few-Shot Learning. *Advances in Neural Information Processing Systems*, 35:14489–14500.

[Welinder et al., 2010] Welinder, P., Branson, S., Mita, T., Wah, C., and Schroff, F. (2010). Caltech-UCSD Birds 200. *Caltech-UCSD Technical Report*, 200.

[Wen et al., 2020] Wen, Y., Tran, D., and Ba, J. (2020). BATCHENSEMBLE: AN ALTERNATIVE APPROACH TO EFFICIENT ENSEMBLE AND LIFELONG LEARNING. In *8th International Conference on Learning Representations, ICLR 2020*.

[Wikenheiser and Redish, 2015] Wikenheiser, A. M. and Redish, A. D. (2015). Hippocampal theta sequences reflect current goals. *Nature Neuroscience*, 18(2).

[Wu et al., 2020] Wu, J., Liu, S., Huang, D., and Wang, Y. (2020). Multi-scale Positive Sample Refinement for Few-Shot Object Detection. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12361 LNCS.

[Wu et al., 2022a] Wu, S., Pei, W., Mei, D., Chen, F., Tian, J., and Lu, G. (2022a). Multi-faceted Distillation of Base-Novel Commonality for Few-Shot Object Detection. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 13669 LNCS.

[Wu et al., 2022b] Wu, T.-Y., Swaminathan, G., Li, Z., Ravichandran, A., Vasconcelos, N., Bhotika, R., and Soatto, S. (2022b). Class-incremental learning with strong pre-trained models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610.

[Xiao et al., 2010] Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492.

[Xu and Le, 2022] Xu, J. and Le, H. (2022). Generating representative samples for few-shot classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9003–9013.

[Xudie Ren et al., 2017] Xudie Ren, Haonan Guo, Shenghong Li, Shilin Wang, and Jianhua Li (2017). A Novel Image Classification Method with CNN-XGBoost Model. In *Digital Forensics and Watermarking*, pages 378–390, Cham. Springer International Publishing.

[Yan et al., 2019] Yan, X., Chen, Z., Xu, A., Wang, X., Liang, X., and Lin, L. (2019). Meta R-CNN: Towards general solver for instance-level low-shot learning. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2019-October.

[Yang et al., 2023a] Yang, Y., Yuan, H., Li, X., Lin, Z., Torr, P., and Tao, D. (2023a). Neural collapse inspired feature-classifier alignment for few-shot class incremental learning. *arXiv preprint arXiv:2302.03004*.

[Yang et al., 2019] Yang, Y., Zheng, K., Wu, C., and Yang, Y. (2019). Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors (Switzerland)*, 19(11).

[Yang et al., 2023b] Yang, Z., Yang, Z., Liu, Y., Li, P., and Liu, Y. (2023b). Restricted orthogonal gradient projection for continual learning. *AI Open*, 4.

[Yoon et al., 2022] Yoon, J., Madaan, D., Yang, E., and Hwang, S. J. (2022). Online Coreset Selection for Rehearsal-based Continual Learning. In *ICLR 2022 - 10th International Conference on Learning Representations*.

[Yoon et al., 2017] Yoon, J., Yang, E., Lee, J., and Hwang, S. J. (2017). Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

[Yu et al., 2020a] Yu, T., Kumar, S., Gupta, A., Levine, S., Hausman, K., and Finn, C. (2020a). Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 2020-December.

[Yu et al., 2020b] Yu, Z., Chen, L., Cheng, Z., and Luo, J. (2020b). Transmatch: A transfer-learning scheme for semi-supervised few-shot learning. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[Zenke et al., 2017] Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *34th International Conference on Machine Learning, ICML 2017*, volume 8.

[Zhang et al., 2020] Zhang, C., Cai, Y., Lin, G., and Shen, C. (2020). DeepEMD: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

[ZHANG et al., 2018] ZHANG, R., Che, T., Ghahramani, Z., Bengio, Y., and Song, Y. (2018). MetaGAN: An Adversarial Approach to Few-Shot Learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Zhang et al., 2018] Zhang, R., Che, T., Ghahramani, Z., Bengio, Y., and Song, Y. (2018). MetaGAN: An Adversarial Approach to Few-Shot Learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

[Zhang, 2017] Zhang, X. (2017). A method of data label checking and the wrong labels in mnist and cifar10. *Available at SSRN 3072167*.

[Zhang et al., 2021] Zhang, Y., Huang, S., and Zhou, F. (2021). Generally Boosting Few-Shot Learning with HandCrafted Features. In *Proceedings of the 29th ACM International Conference on Multimedia*, MM '21, pages 3143–3152, New York, NY, USA. Association for Computing Machinery.

[Zhou et al., 2022] Zhou, D.-W., Wang, Q.-W., Ye, H.-J., and Zhan, D.-C. (2022). A model or 603 exemplars: Towards memory-efficient class-incremental learning. *arXiv preprint arXiv:2205.13218*.

[Zhu et al., 2021] Zhu, C., Chen, F., Ahmed, U., Shen, Z., and Savvides, M. (2021). Semantic Relation Reasoning for Shot-Stable Few-Shot Object Detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.