Few Shot Metric Learning for Remote Sensing Image Scene Classification

By

Muhammad Adil Talay

206573

Master of Science in Systems Engineering

Supervisor

Dr. Shahzad Rasool

Department of Computational Engineering

Research Center for Modelling and Simulation (RCMS)

National University of Sciences and Technology (NUST)

Islamabad, Pakistan

August 2020

Few Shot Metric Learning for Remote Sensing Image Scene Classification

By

Muhammad Adil Talay

206573

Supervisor

Dr. Shahzad Rasool

A thesis submitted in conformity with the requirements for the degree of *Master of Science* in Systems Engineering

Department of Computational Engineering
Research Center for Modelling and Simulation (RCMS)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan

August 2020

To Pakistan Air Force

Declaration

I, *Muhammad Adil Talay* declare that this thesis titled "Few Shot Metric Learning for Remote Sensing Image Scene Classification" and the work presented in it are my own and has been generated by me as a result of my own original research.

I confirm that:

1. This work was done wholly or mainly while in candidature for a Master of Science degree at NUST
2. Where any part of this thesis has previously been submitted for a degree or any other qualification at NUST or any other institution, this has been clearly stated
3. Where I have consulted the published work of others, this is always clearly attributed
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work
5. I have acknowledged all main sources of help
6. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself

Muhammad Adil Talay,
206573

Copyright Notice

Acknowledgments

First of all, I would like to thank Allah Almighty the Greatest, for enabling me to complete this work. Thereafter, I am grateful to my parents, wife, son, siblings and grandparents for their continuous support and encouragement. I am indebted to the constant help from my supervisor Dr. Shahzad Rasool and appreciate the timely guidance from my GEC members Dr. Muhammad Tariq Saeed and Dr. Mian Ilyas Ahmad. Finally, I am also obliged to Dr. Rizwan Riaz, Principal RCMS, for his encouragement and provision of a nurturing environment at RCMS, NUST.

Contents

List of Tables

List of Figures

Abstract

Visual recognition in aerial imagery plays an important role in a wide range of applications such as surveillance, monitoring, detection and management of natural and man-made disasters. Current advances in deep learning show promising results for computer vision tasks of classification, detection, segmentation and tracking. The meta-learning branch of deep learning seeks to train models that learn new concepts with few labeled examples, to save data annotation cost and solve the problem of data scarcity. Recent works show the superior performance of metric-learning approaches for meta-learning among others. This research evaluates two state-of-the-art metric-learning methods, namely Prototypical Networks and Relation Networks, in remote sensing imagery and explores avenues to improve performance by utilizing efficient networks with different depths for feature extraction and jointly training on multi-domain data. The performance of the same efficient networks is also evaluated for object detection in satellite imagery, to aid in the wise selection of feature extraction backbone for a meta-learning object detector. Our results suggest that Prototypical Networks are faster to train and more accurate than Relation Networks when the number of training classes are limited. Furthermore, jointly training on natural and satellite imagery for few shot classification is shown to slightly improve accuracy, given a suitable feature extraction backbone. Finally, we conclude that MobileNet v2 might serve as the potential network design to begin design space exploration of feature extraction backbones targeted for accurate and efficient meta-learning as it outperforms its competitors in both the tasks of object detection and few shot classification.

Chapter 1

Introduction

1.1 Motivation

Visual recognition in aerial imagery has many important applications such as in surveillance, urban planning, transportation planning, traffic supervision, environmental monitoring and man-made or natural disaster management.

Two branches of visual recognition are scene classification and object detection. Scene classification aims at developing a semantic-level understanding of the whole image and assign semantic labels to each scene class. Conversely, object detection aims at recognizing different scene components or entities in an image and assign object level labels to each entity along with recognizing its location or bounding box coordinates.

The effectiveness of any method used for any visual recognition problem depends heavily upon its ability to exploit multiple cues from the image to generate better discriminative features, and also the level of spatial patterns recognized by a method. Research in the last few decades has progressed from pixel-level to object-level to semantic-level understanding of the images. Yet still, traditional hand-engineered features and even the shallow-learning-based features have limited description ability which may even be impoverished in challenging scenarios. Hence, research has taken a shift in the last two decades to deep-learning features for their powerful representation of both the local and global information in an image. Loosely inspired from the human visual system,

convolutional neural networks are the most popular deep learning method for visual recognition.

The success of deep learning lies in both the availability of appropriate data and accurate models. The emergence of challenging benchmarks through the last decade has steered the development of novel network designs with remarkable accuracy. However, in the context of visual recognition in aerial imagery using embedded platforms, these network designs face a few limitations:

1. The design of these networks is not optimized for resource-constrained environments.

2. The object size in aerial imagery is very small compared to the image size which poses a challenge to state-of-the-art detectors that have degraded performance in detecting tiny targets.

3. The availability of data in some domains like satellite imagery is insufficient to obtain the desired accuracy through the standard training procedure.

4. The images collected through different technologies have great variation and disparity to effectively create a data distribution shift which leads to degraded accuracy.

Therefore, an integrated solution is needed that has the ability to learn from little amount of data, detect small objects in satellite imagery and is optimized for embedded platforms. On a note, we find that the terms *aerial imagery* and *satellite imagery* are used interchangeably in the literature although aerial imagery is inclusive of low-altitude top-view images taken through unmanned aerial vehicles (UAVs). As such, we also sometimes refer to satellite images as aerial images in this work.

1.2 Problem Statement

This thesis concentrates on evaluating state-of-the-art technologies for visual recognition in remote sensing imagery. To be specific, the research addresses the two problems

of scene classification and object detection separately. For scene classification, the performance of state-of-the-art solutions for scarce and diverse data known as few shot learning is compared with efficient backbones. Whereas for object detection, we employ a state-of-the-art detector by applying modest alterations to improve computational efficiency. We perceive that this work may serve as the stepping-stone to tackle the challenges presented by aerial imagery.

1.3 Objectives

- To compare the performance of state-of-the-art few shot learning methods on aerial imagery.
- To assess the performance of efficient networks for feature extraction in the few shot learning scenario.
- To analyze the generalizability of few shot learning methods to distinct domains.
- To examine the accuracy of efficient backbones for object detection in satellite imagery.

1.4 Thesis Organization

The thesis is organized into six chapters, the details of each are as follows:

1. Introduction: Describing the evolution and limitations of visual recognition systems and defining the objectives of this research.
2. Literature Review: Reviewing the different available datasets and discussing the development of deep learning methods to identify research gaps.
3. Proposed Methodology: Overviewing the problem and the proposed approach.
4. Implementation: Presenting the implementation details.

5. Results and Discussion: Presenting the results of experiments and analysing the patterns.

6. Conclusion: Arriving at conclusions while setting the stage for future work.

Chapter 2

Literature Review

Deep learning is a data-driven technology that owes greatly to the organization or invention of publicly available benchmarks for recent breakthroughs in the field, as it pushes the development of new algorithms, while allowing comparison to the baselines using standard evaluation protocols. In this section, we first review the available datasets related to the problem. Then, we discuss the available deep learning methods and identify the gap in knowledge.

2.1 Benchmarks

In this section, we review the classification and detection benchmarks designed for both the standard deep learning approach and the few shot learning approach in the domains of natural and satellite imagery. We divide this section into the following subsections: (a) natural scenes, (b) few shot learning and (c) satellite imagery.

2.1.1 Natural Scenes Datasets

Natural scenes or imagery taken on ground has received the most attention in the deep learning world. In 2009, CIFAR dataset was proposed by Krizhevsky [28]. It consists of two subsets CIFAR10 and CIFAR100, each with 60,000 color images of size 32x32. CIFAR10 consists of 10 classes with 6000 images in each class, while CIFAR100 consists

of 100 classes with 600 images per class. Everingham et al. [29] proposed PASCAL Visual Object Classes (VOC) Challenge the same year. The dataset consists of 19.7K images and 20 object classes. The challenge offers five tasks of classification, detection, segmentation, action classification and person layout.

In 2010, Russakovsky et al. [26] proposed the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) that has 1000 object classes with about 1.46 million annotated images. The challenge offers three tasks of image classification, single-object localization and object detection. Lin et al. [27] proposed Microsoft Common Object in Context (COCO) dataset in 2014, which contains about 328K images of 91 object types, for detection and segmentation tasks. The Open Images Dataset V4 was proposed by Kuznetsova et al. [30] in 2018, which consists of about 9.2M images, with 19.8K concepts for classification, 600 object classes for detection and 57 classes for visual relationship recognition.

The increase in the scale of benchmarks from 120K images and 110 classes in CIFAR to 9.2M images and 19.8K concepts in Open Images is due to the fact that machines are not efficient learners of new concepts, like humans. A child may learn about a new object class by looking at a single instance of that object, while a computer would require many object instances to learn about the object category. However, it should be noted here that these large scale datasets are costly to acquire and annotate. Research in the direction to enable the computers to learn new concepts from a few examples is known as few shot learning, for which we briefly discuss three benchmark datasets.

2.1.2 Few Shot Learning Datasets

To the best of our knowledge, the first dataset for few shot learning was proposed by Lake et al. [25] as The Omniglot Challenge. The dataset is about handwritten character recognition with 1623 characters from 50 different alphabets. However, it lacks the complexity offered by images of natural scenes. Vinyals et al. [15] proposed to create MiniImageNet from the ImageNet dataset [26] as the latter is notoriously large, but they did not release it. Ravi and Larochelle [31] proposed their own version

of MiniImageNet, consisting of 60,000 images with 600 images per class for 100 classes.

Triantafillou et al. [24] note that the typical approach adopted in most few shot learning methods is to train and test the models on two non-overlapping subsets from the *same* dataset, with all the classes having the same number of samples, a concept known as *class-balance*. They argue that the real-world data suffers from class imbalance and training on class-balanced dataset would train the model for an unrealistic problem. They further state that the purpose of few shot learning is to generalize to data from different domains. Thus, they propose Meta-Dataset which is significantly large-scale and comprised of data from different domains, and further introduce class-imbalance to make the problem more realistic.

2.1.3 Satellite Imagery Datasets

Satellite imagery datasets may be for scene classification or object detection. Here we discuss briefly about five scene classification and four object detection benchmarks.

For scene classification, the UC-Merced Land-Use dataset [32] is the earliest among the five. It consists of 2,100 images divided into 21 land-use scene classes with 100 aerial images in each class. All images are in the red, blue, green (RGB) color space, each measuring 256x256 pixels, with a spatial resolution of 0.3m. However, it lacks variations and diversity and hence suffers from saturation of accuracy. Another benchmark is the WHU-RS19 dataset [33] which is composed of 1,005 aerial scene images with about 50 images per class for 19 scene classes. The images are in the spectral bands of RGB, with a spatial resolution of up to 0.5m, measuring 600x600 pixels. This dataset has high variations in illumination, scale, orientation and resolution, and is hence challenging but suffers from small number of images per class. The third benchmark is the RSC11 dataset [34] which has a total of 1,232 images of size 512x512 in the RGB spectrum with a spatial resolution of 0.2m. The 11 scene classes represent complicated scenes and the between-class similarity increases the difficulty of discrimination between them. However, the number of scene classes is relatively small.

Two state-of-the-art benchmarks are the AID [35] and NWPU-RESISC45 [36] datasets.

The AID dataset has a total of 10,000 images for 30 scene classes with different number of images for each scene class, ranging from 220 to 420. All the images are in the RGB color space and of size 600x600 with spatial resolution varying from 8m to 0.5m. On the other hand, the NWPU-RESISC45 dataset is composed of 31,500 images divided into 45 scene classes with 700 images per class. The spatial resolution varies from 30m to 0.2m, and the images are of size 256x256 in the RGB color space. Both the datasets are large scale with rich variations, high intra-class diversity and smaller inter-class dissimilarity.

For object detection, the earliest dataset mentioned in the literature for multi-class object detection was proposed by Cheng and Han [37] in 2014 as NWPU-VHR10 which has 800 images from which 715 images are RGB and 85 are pan-sharpened color infrared images. The spatial resolution of the RGB images ranges from 0.5m to 2m while that of infrared images is 0.08m. The dataset consists of 10 geospatial object classes and is widely used in the earth observation community. Another dataset is VEDAI, proposed by Razakarivony and Jurie [38] in 2015, to aid the development of new algorithms for multi-class small object detection in aerial imagery. It has a total of 1210 images of size 1024x1024 pixels, with a spatial resolution of 12.5 cm, in the RGB and near infrared color space. The dataset has nine different vehicle classes and is challenging due to complex background. Nevertheless, both these datasets are not large scale as the object instances in either of the two is not more than 4000.

To cater for the need of a large scale dataset for object detection in satellite images, Xia et al. proposed DOTA [39] in 2017 while Li et al. proposed DIOR [40] in 2018. Both these datasets are large scale with object instances numbering to about 190K in each, with large variations in orientations, spatial resolution, between-class sizes and within-class sizes, as well as high background complexity. Yet there are a number of differences between the two. The DOTA dataset provides additional information about the spatial resolution, has greater variety of object density, the image sizes vary between 800x800 to 4000x4000, the total number of images is 2806 and the number of object classes is 15. Conversely, the DIOR dataset has fixed image size of 800x800, with 23,463 images and is composed of 20 object classes that have high inter-class similarity

and intra-class diversity.

2.2 Data Driven Deep Learning Methods

Reviewing the evolution of benchmark datasets, we next discuss the development of deep learning methods owing to these benchmarks. As convolutional neural networks (CNNs) are the most prevalent deep learning method among the computer vision community for outstanding performance in learning visual recognition tasks, in this subsection, we only review the development of CNNs through the last decade. Firstly, we discuss the CNN architectures for classification and then briefly overview the network designs for detection.

2.2.1 CNNs for Classification

Due to the limitations of hand-crafted features and the fruitful performance of deep learning networks, the last decade has seen the shift of research focus from 'feature engineering' to 'network engineering'. Hereunder, we only discuss two directions of research in network design related to our work: (1) design of efficient networks, (2) design of few shot learning networks.

Design of Efficient Networks

Scientists have adopted different directions into design space exploration of neural networks to increase efficient use of model parameters for performance gains in accuracy; to optimize the networks for real-time applications; to meet the memory and power constraints; and to build simplified modular units in order to reduce complexity while increasing scalability, all with the prime goal of designing neural networks with better generalizability besides optimal accuracy under limited resource budget.

The seminal work for convolutional neural networks was proposed at the end of the twentieth century by LeCun et al. , known as LeNet [41]. The network has two convo-

lutional layers each followed by a pooling layer and two fully connected (FC) layers at the end. However, the development of novel algorithms only took a quantum leap with the emergence of AlexNet [1] proposed by Krizhevsky et al., which outperformed all hand-engineered approaches on the ILSVRC2012. A distinguishing feature of AlexNet from LeNet was its increased capacity, both in terms of the total number of layers (depth) and the number of channels in each layer (width). AlexNet is composed of five convolutional layers with different number of channels for each layer, the least of which is 96, while the maximum is 384. The convolutional layers are followed by three FC layers, from which the first two have 4096 channels while the last has 1000 channels (see figure 2.1).

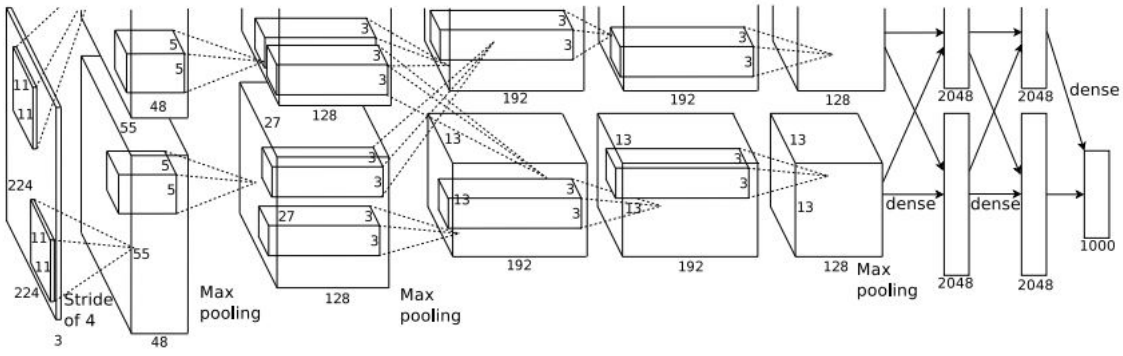


Figure 2.1: AlexNet architecture [1].

Following the success of AlexNet, Lin et al. proposed Network In Network (NIN) [2] in which they use 1×1 kernels to add further non-linearity to the network to enhance discriminability (see fig 2.2). They also discuss that large models of the likes of AlexNet are computationally expensive and prone to over-fitting. The authors note that most of the parameters of such large models belong to the FC layers and as an alternative they propose global average pooling (GAP) in which a 3-D feature map of size '$H \times W \times C$' (where H, W and C refer to the height, width and depth of the feature map) is converted into a feature map of size '$H \times W \times N$' (where N refers to the number of classes labels from which the target image is to be assigned a class label). Thereafter, an average of this feature map is computed in the H and W dimensions to obtain a feature map of size N which is then fed to the SoftMax layer for computation of class probabilities

for classification. This alternative to FC layers reduces over-fitting as it requires no parameters to train.

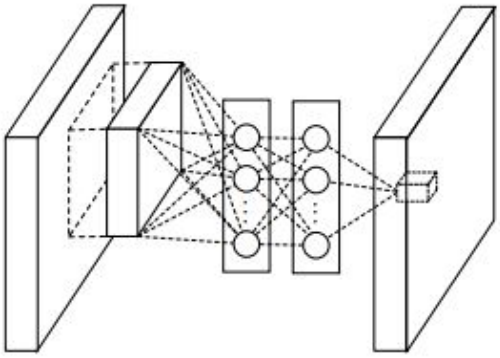


Figure 2.2: Network In Network module [2].

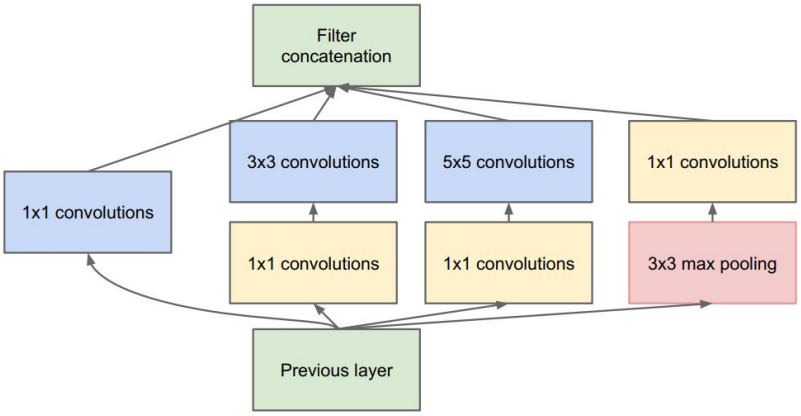


Figure 2.3: Inception v1 module with the split-transform-merge strategy [3].

Simonyan and Zisserman proposed VGG networks [42] in which they show that increasing network depth improves accuracy. They also introduce factorized convolutions and argue that a stack of kernels with small receptive fields is equivalent to kernels with large receptive fields and advantageous due to the reduction in model parameters and increased ability to add more non-linearity. They further adopt a modular approach in designing the network and propose to double the number of channels whenever the image is down-sampled to half, to keep the complexity of all modules similar. Concurrently, Szegedy et al. proposed GoogleNet based on the Inception v1 module [3], in which they use different kernel sizes in a split-transform-merge fashion to capture visual information at different scales (see figure 2.3). They further exploit the 1×1

filters for dimension reduction, resulting in a computationally efficient design which allows the network depth to reach 22 layers.

Nevertheless, deep networks suffer from vanishing/exploding gradients problem. Ioffe and Szegedy proposed batch normalization [43] while He et al. proposed advanced initialization along with learnable activation [44] to combat the problem, hence enabling faster convergence. Yet, He et al. [4] note that increasing network depth leads to accuracy saturation and adding more layers would cause performance degradation. As a solution, they propose ResNet which is based on residual connections that are used to add the output of the preceding module to the output of the succeeding module (see figure 2.4), in order to ensure that stacking more layers would only lead to improved performance.

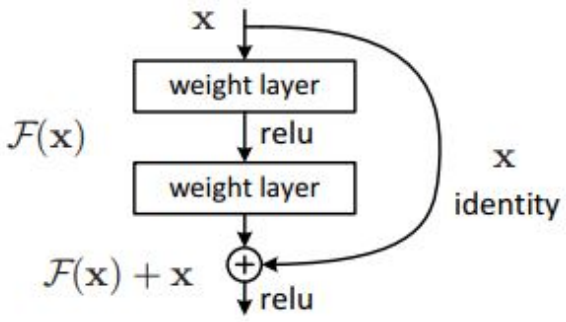


Figure 2.4: Residual connection [4].

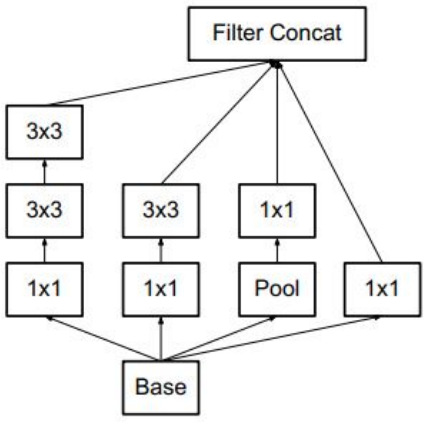


Figure 2.5: Inception v3 module. Notice that the 5×5 convolution from the Inception v1 is factorized into two stacked 3×3 convolutions [5].

Meanwhile, Szegedy et al. [5] note that the complexity of Inception v1 architecture

makes it inefficient in terms of scalability and hence discuss some principles of scalability which they follow to propose Inception v3. The authors adopt the factorized convolutions to replace kernels with large receptive fields (see figure 2.5) in the proposed network and further discuss the usability of separable convolutions that use $n \times 1$ kernel followed by $1 \times n$ instead of an $n \times n$ filter (see figure 2.6).

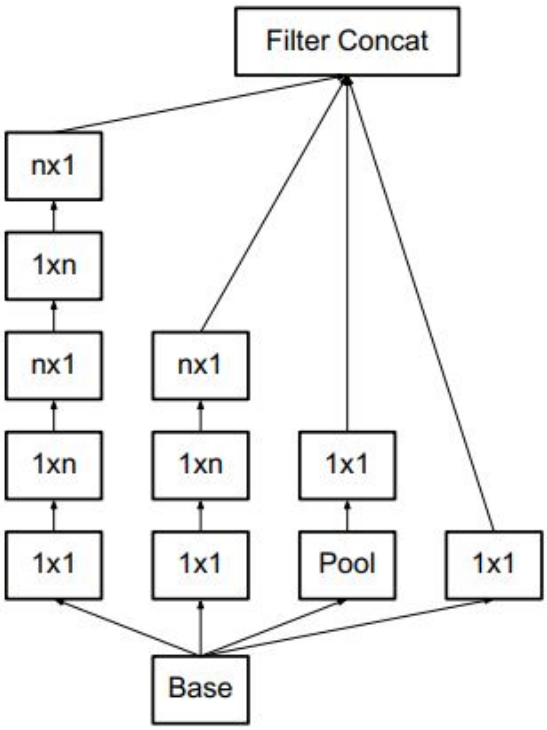


Figure 2.6: Inception v3 module with separable convolutions [5].

In the follow-up work, Szegedy et al. [45] propose a more uniform and simplified architecture built on the inception module to increase scalability which they name Inception v4, in which they also utilize separable convolutions. The authors also evaluate the effects of introducing residual connections on the Inception network and conclude that the improvement in accuracy was not significant, but training speed was dramatically improved. Conversely, Xie et al. [6] adopt the split-transform-merge strategy from the Inception module into the ResNet architecture by exploiting group convolutions for splitting while using summation for merging (see figure 2.7) and conclude that the size of transformation or cardinality is an essential dimension besides depth and width, which allows to increase model accuracy while maintaining complexity.

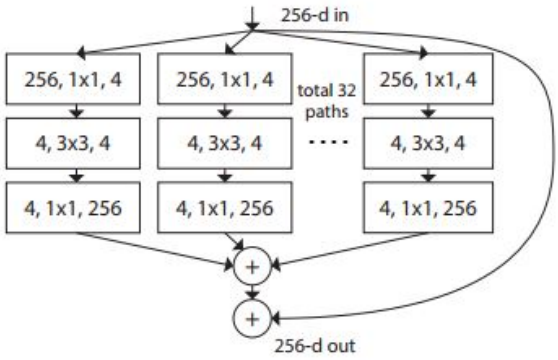


Figure 2.7: ResNext module illustrating cardinality [6].

Another extension of ResNet was proposed by Huang et al. as DenseNet [7] in which the authors introduce dense connectivity by feeding the output of each layer to all the subsequent layers to introduce feature reuse while reducing redundancies (see figure 2.8). In the follow-up work [8], the authors further condense the network by learning important dense connections that are arranged as groups using group convolutions while removing superfluous connections through pruning (see figure 2.9).

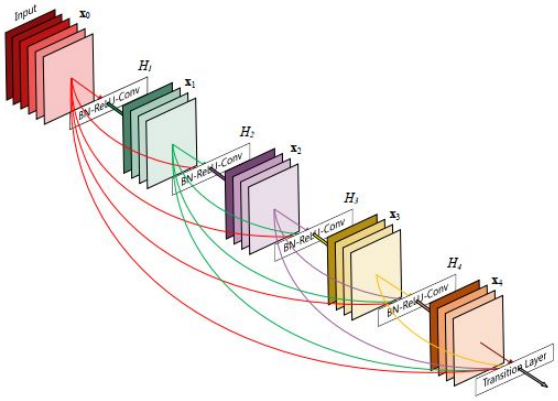


Figure 2.8: A five layer DenseNet [7].

A more dedicated work on small neural networks was proposed by Iandola et al. [9] in which the authors discuss the importance of small networks for efficient distributed training, feasible embedded deployment and faster training and develop a disciplined approach to build networks with few parameters while preserving accuracy. For the given purpose, they propose to replace 3×3 filters with 1×1 filters and reduce the number of input channels to 3×3 kernels. The authors propose the fire module in

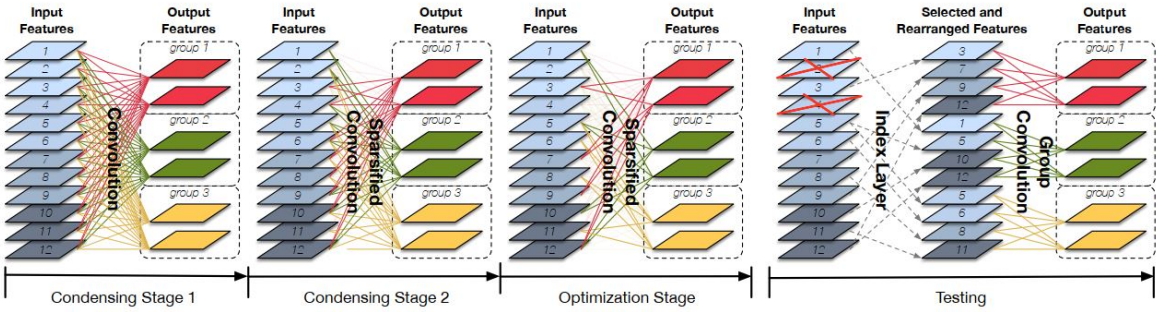


Figure 2.9: CondenseNet procedure [8].

which they first squeeze the input channels, then expand using two sets of filters of sizes 3×3 and 1×1 and finally concatenate the outputs from both sets of kernels (see figure 2.10). With the given strategy, they achieve the accuracy of AlexNet with 50 times fewer parameters. Furthermore, although the authors at first suggested to downsample late in the network to improve accuracy but thereafter found that it leads to increased computational complexity [46] and hence suggest evenly-spaced downsampling which is in accordance to the principle suggested by Szegedy et al. [5] to decrease the representation size gently from the input to the output. Gholami et al. [47] extend the work by using aggressive channel reduction, separable convolutions and residual type connections to further reduce the model parameters.

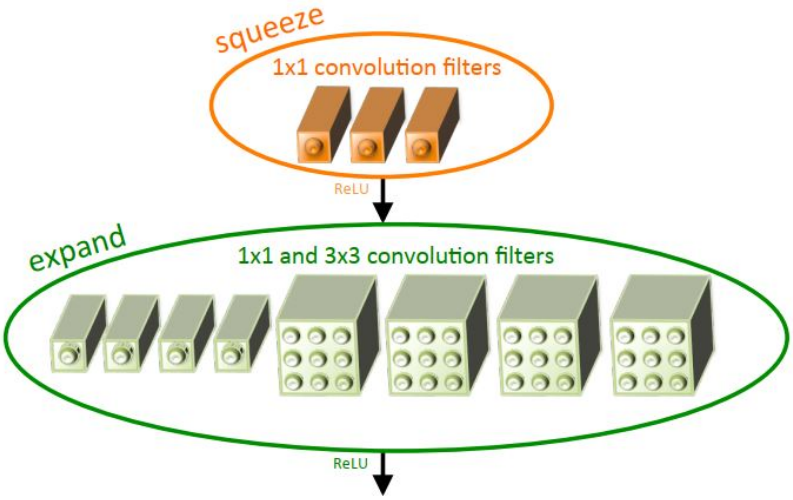


Figure 2.10: Fire module [9].

Another line of work to reduce network parameters adopts depthwise separable convolutions in which channel-wise spatial convolution is performed first followed by pointwise

(1×1) convolution. Chollet [48] proposed to replace Inception modules with depthwise separable convolutions to propose a novel architecture Xception and achieved accuracy improvements on ImageNet and JFT datasets. Howard et al. also adopt depthwise separable convolutions to propose a novel architecture MobileNet v1 [49] in which they further incorporate two hyperparameters of width and resolution multipliers for controlling the model capacity. The width multiplier is used to decrease/increase the number of channels at each stage while the resolution multiplier is implicitly set by setting the input resolution. Sandler et al. [10] extend the work by proposing linear bottlenecks in which they first expand the number of channels according to the expansion ratio using pointwise (1×1) convolutions, then perform channel-wise spatial convolutions and finally squeeze the channels using pointwise convolutions but do not apply any non-linear activation to the last layer to prevent information loss. The authors further introduce an inverted residual connection into the block by which the input to the linear bottleneck is added to its output (see figure 2.11). The full architecture built from these modules is referred as MobileNet v2.

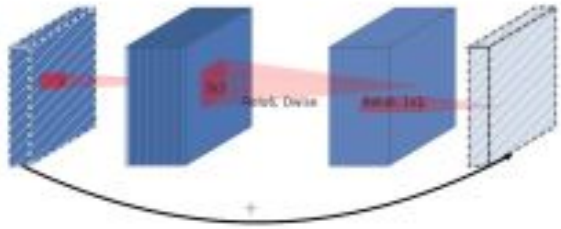


Figure 2.11: Inverted Residual block [10].

Another competing architecture contemporary to MobileNet was proposed by Zhang et al. [11] in which the authors pay attention to efficiently increase the number of channels for more powerful representation, for which they propose to use pointwise group convolutions to reduce computation. However, using pointwise convolutions in groups blocks cross-channel information flow leading to weakened representations, hence defeating the purpose. Therefore, they propose to use a channel shuffle operation to shuffle the information across channels to mitigate the adverse side effect of group convolutions (see figure 2.12). Ma et al. [12] improve the work by first developing the

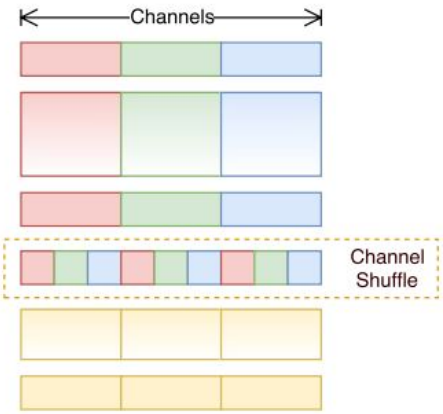


Figure 2.12: Channel shuffle operation of ShuffleNet v1 [11].

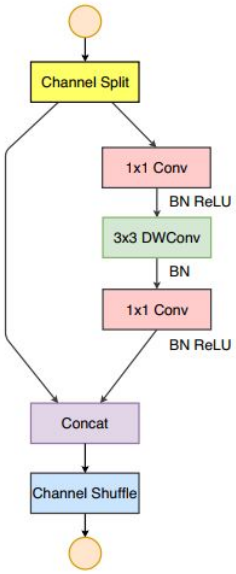


Figure 2.13: Channel split operation of ShuffleNet v2 [12].

principles to design efficient networks and then proposing a modified architecture in which they remove group convolutions as it increases the memory access cost. They further introduce a channel split operation which splits the input channels into two branches, from which no operation is performed on one branch and it acts as a residual connection, while pointwise convolution followed by depthwise separable convolution is performed on the other branch. Finally, both the branches are concatenated and fed into the channel shuffle operator (see figure 2.13). The complete architecture is named ShuffleNet v2.

A more recent work by Mehta et al. [13] proposed an efficient spatial pyramid (ESP)

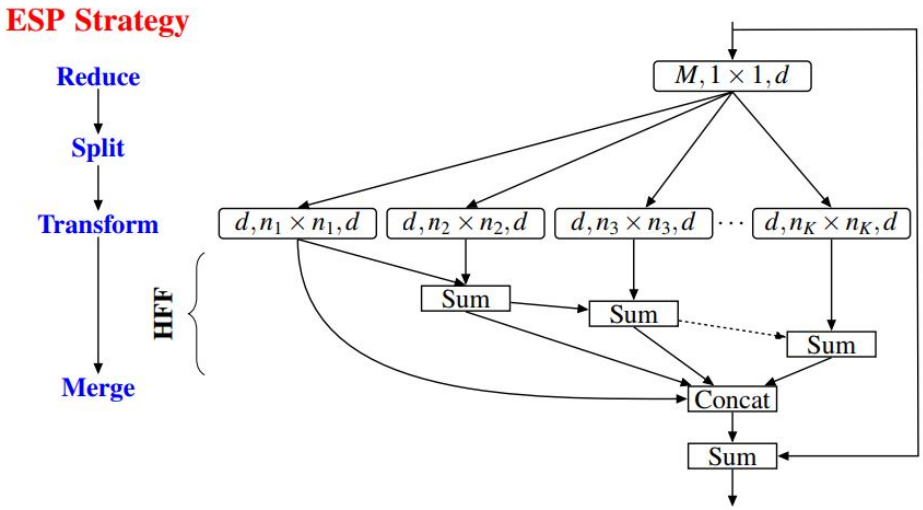


Figure 2.14: ESP module of ESPNet v1 [13].

module which is built using the reduce-split-transform-merge strategy (see figure 2.14). The reduction is done using the pointwise convolutions while the splitting and transforming is performed using a spatial pyramid of dilated convolutions and the outputs from all pyramidal levels are concatenated. As a note, dilated convolutions inject zeros between the convolution kernels to increase the effective receptive field. The use of spatial pyramids leads to gridding artifacts that are solved by hierarchical feature fusion (HFF). The authors also introduce a skip connection in the ESP module to improve information flow. To further increase network efficiency, the authors propose extremely efficient spatial pyramid (EESP) [14] in the subsequent work, in which they replace the pointwise convolutions with group pointwise convolutions and decompose the dilated convolutions into depthwise dilated separable convolutions from the original ESP module (see figure 2.15). The resulting network is called ESPNet v2 and is more power-efficient than MobileNet v2 or ShuffleNet v2.

We note here that the journey of network design began with the objective of powerful discriminative feature learning. With the ever-growing popularity of deep learning applications, resource efficiency was later incorporated in design to meet the commercial demands. We observe that as the last few years have seen the shift from feature design to network design, research is now progressing from feature learning to network learning to tackle the forthcoming challenges [50–54].

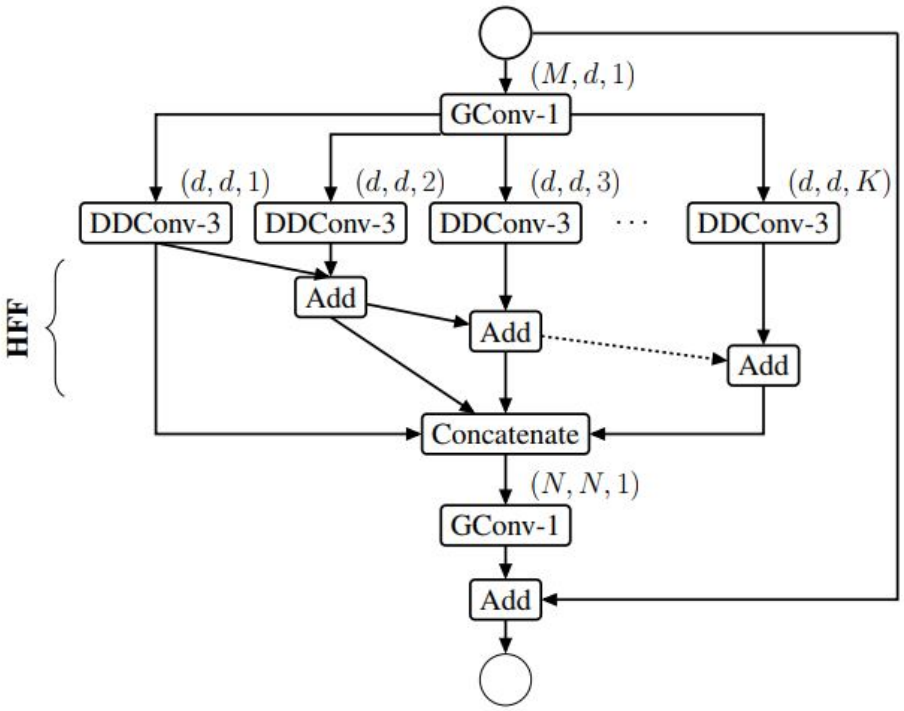


Figure 2.15: EESP module of ESPNet v2 [14].

Design of Few Shot Learning Networks

Few shot learning [55] aims at learning new concepts from limited examples, more like humans, to reduce the efforts for data collection, save the computational cost of model re-training and improve generalization to unseen classes from completely different domains. The standard approach for training deep learning models is to randomly initialize the weights and train on large-scale labeled datasets with gradient-based optimizer for thousands of iterations until the models converge to a good solution. However, training the models using this standard approach on little data leads to over-fitting, hence poor generalizability.

A straightforward solution to prevent over-fitting is to augment the data by applying various transformations on the train set. Recent approaches learn appearance variations in the base classes and hallucinate data by generating the learned variations for the novel classes [56]. However, this approach is data-focused and usually works together with model-focused few shot learning approaches. Another simple yet intuitive solution is to first train the model on a large-scale labeled dataset from a related

distribution and thereafter fine-tune the last few layers on the target distribution. This concept of knowledge transfer from source domain to target domain is referred in the literature as transfer learning [57]. Nonetheless, a major demerit of this approach is that feature transferability deteriorates as the difference between the source and target domains increases [58].

To address this challenge, some works propose a meta-learning approach in which the data classes are first split into train, validation and test sets. Thereafter, few shot tasks are generated from the train set and the model is optimized to minimize the loss between the support and query samples. The hyper-parameters are tuned on the validation set and the model's ability to generalize to unseen tasks is evaluated on the test classes. A popular approach that adopts such a strategy may be referred to as *optimized adaptation*, as it learns the weights that *capture the common features* in the train classes. The learnt weights may then serve as a *good initialization* to further fine-tune the model on the test classes [59]. Extensions to such methods include simplifying the approach [60] and learning the update direction and learning rate besides initialization [61].

Another line of work adopts the meta-learning approach to *learn a discriminative metric space* in which the query images are classified by determining the similarity with the support images, which obviates the need for fine-tuning. In effect, metric learning methods exploit only the feature extraction layers of CNNs, and the classification layers are replaced by similarity measuring module. Siamese Networks [62] are the earliest CNN-based metric learning method proposed by Koch which consist of two identical networks, that is, both the architecture and parameters are same. The feature embedding of the support images are computed by one network while the other computes the embedding for the query images and the L1 component-wise distance between the two embedding is computed for classification.

Subsequently, Vinyals et al. proposed Matching Networks [15] that use CNN or long short-term memory (LSTM) with attention mechanism for computing the feature embedding of the support set, while utilize CNN or bidirectional LSTM to compute query embedding and classify by measuring the cosine similarity (see figure 2.16). Snell et al.

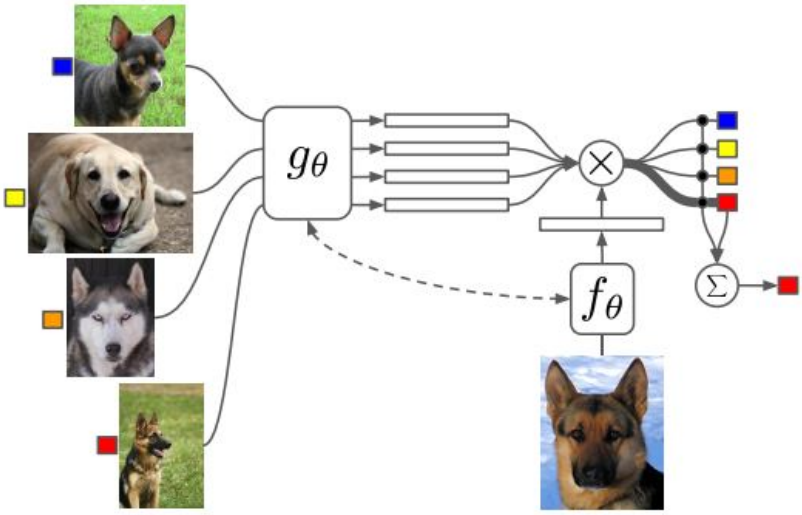


Figure 2.16: Matching Networks [15].

adopt an even simpler approach in their proposed Prototypical Networks [16], in which class prototypes are computed by taking the mean of feature embeddings of the samples from the same class and classification is done on the basis of Euclidean distance between the prototype embedding and query embedding (see figure 2.17). A bit differently, Sung et al. proposed Relation Networks [17] that consist of two modules. The first module is the *feature embedding module* that is used to compute the feature embeddings from the support or query samples. The novelty of the approach lies in the other module termed as the *relation module* that instead of adopting fixed metrics, learns a non-linear deep distance metric to compare the support and query items (see figure 2.18).

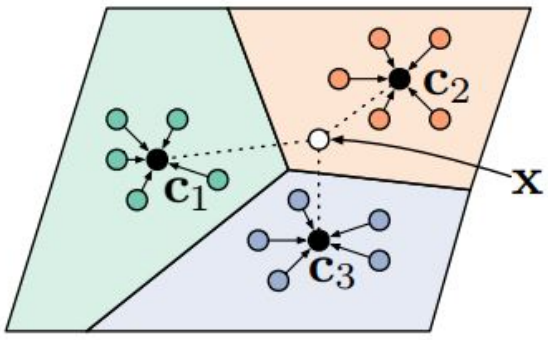


Figure 2.17: Feature embeddings in the Prototypical Network [16]. Here c1, c2 and c3 refer to the class prototypes and x is the query sample whose Euclidean distance from the class prototypes is being computed.

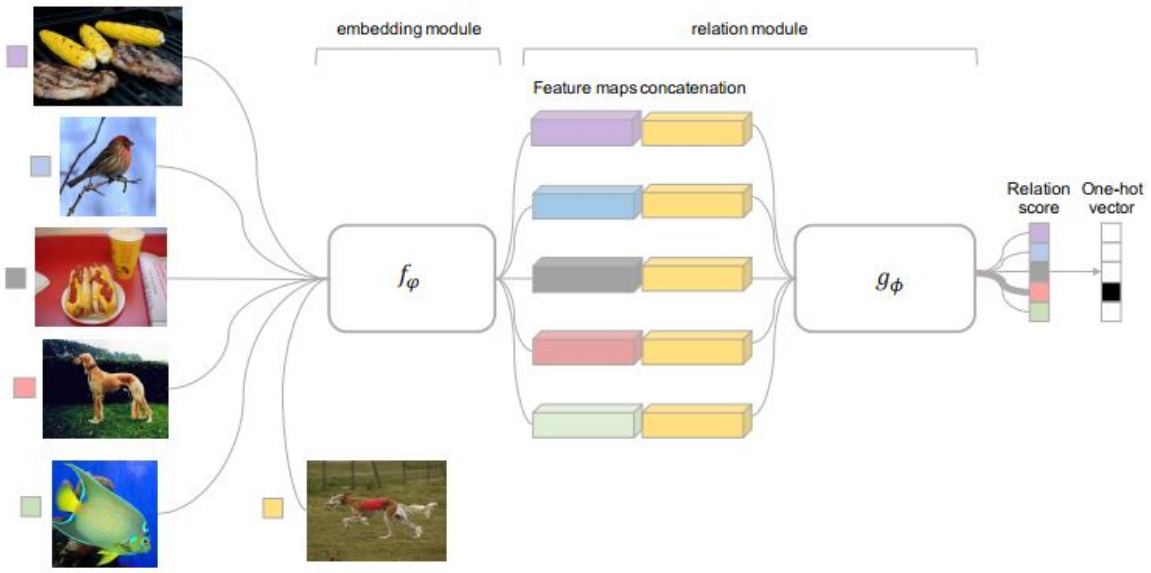


Figure 2.18: Relation Networks [17]. Notice that in the relation module, the feature embedding coloured orange belong to the query sample. The query embedding is concatenated with each embedding from the support samples to learn a non-linear relation between objects belonging to the same class.

Recent studies by Chen et al. [63] and Triantafillou et al. [24] show that transfer learning and metric learning are competitive approaches to address the problem of few shot learning. Chen et al. [63] argue that the transfer learning approach has been severely underestimated and especially for cross-domain scenarios where the novel classes are very different from the base classes, it outperforms the metric learning approaches. The results reported by Triantafillou et al. [24] also partially support the argument.

Finally, Chen et al. [63] also study the effect of reducing intra-class variation by increasing the backbone depth and conclude that it leads to higher accuracy and fine-tune methods show comparable performance to metric learning methods. However, it is worth noting that the performance of transfer learning approach in such scenarios owes greatly to data augmentation which is not a necessity for meta-learning approaches.

2.2.2 Architectures for Detection

Traditional detection approaches adopted sliding windows for object localization along with hand-engineered features for object classification. In 2013, Uijlings et al. pro-

posed Selective Search [64], a hand-engineered object proposal method to replace the computationally expensive sliding windows. Consequently, Girshick et al. proposed Regions with CNN (R-CNN) classifier [65] to replace the hand-engineered classification techniques while utilizing selective search for object proposals. In the follow-up work, although Girshick [66] further improved the architecture to propose Fast R-CNN but the selective search algorithm created a computation bottleneck.

To remedy the problem, Ren et al. proposed Faster R-CNN [18] which is composed of a backbone for feature extraction, followed by a region proposal network (RPN) as a replacement for selective search and Fast R-CNN for object classification along with fine localization. The backbone generates feature maps from the input image, that are fed to the RPN which computes the intersection over union (IOU) between default bounding box positions (anchor boxes) and the ground truths and keeps only those anchor boxes whose IOU is above the given threshold. These anchor boxes are then proposed as regions of interest (ROIs) to the Fast R-CNN head, that first warps the proposed ROIs to fixed size by ROI Pooling and thereafter computes the class scores and regresses the offsets between the anchor boxes and ground truths to obtain accurate bounding box positions (see figure 2.19 for the illustration of the architecture).

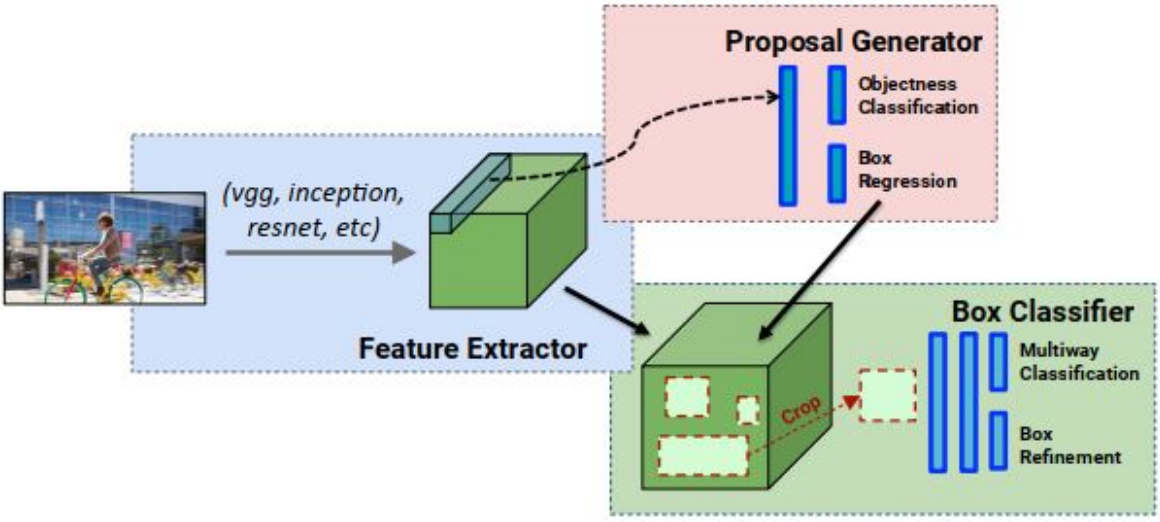


Figure 2.19: Faster R-CNN illustration [18].

Dai et al. [19] note that the FC layers in the Faster R-CNN architecture are computationally expensive as they do not share computation and propose to replace them with

convolutional layers to generate position-sensitive score maps. They further propose position-sensitive ROI pooling that takes votes from the score maps to generate the final prediction. The full architecture is referred as Region-based Fully Convolutional Network (RFCN). We consider this work to be in striking resemblance with the work of Lin et al. [2] who proposed global average pooling as a replacement for the FC layers (refer to figure 2.20 for visualization of the architecture).

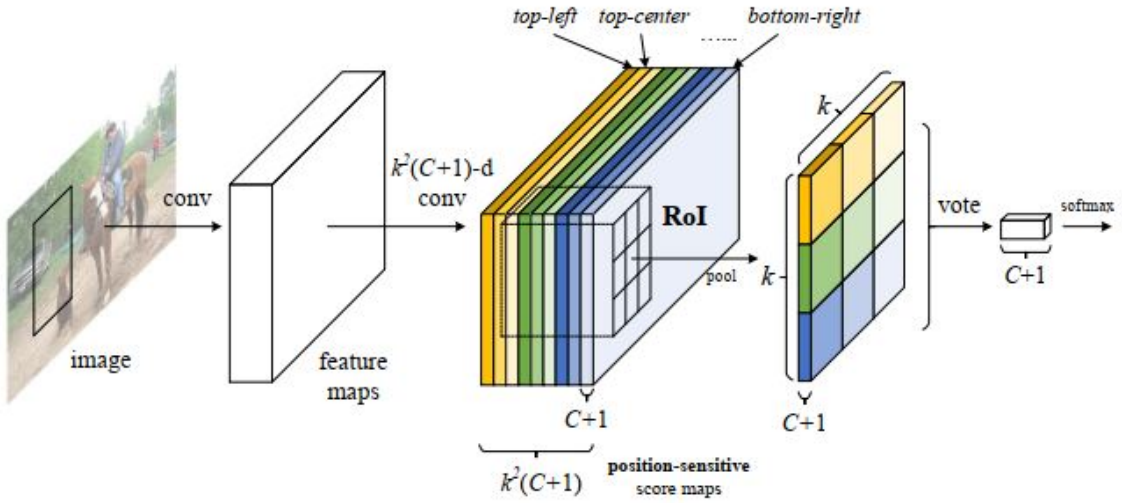


Figure 2.20: R-FCN architecture [19].

It is worth noting that the region-based methods perform detection in two stages, where the first stage performs binary classification to differentiate object regions from the background along with coarse localization of the objects and the second stage further processes those regions for multi-class classification and fine localization. Although such approaches are accurate, yet they are relatively slow for real-time applications and unsuitable for embedded systems for high computational requirements.

Redmon et al. steered research in another direction to develop a one-stage detector and proposed You Only Look Once (YOLO) [20]. The intuition behind the working of YOLO is that the convolutional layers extract features followed by two FC layers where the last FC layer generates the prediction of the shape $S \times S + (B \times 5 + C)$ (see figure 2.21). Here 'S' denotes the height and width of the score map and $(B \times 5 + C)$ refers to the number of channels. In effect, the system partitions the input image into an '$S \times S$' grid, where each grid cell generates 'B' bounding boxes with five predictions

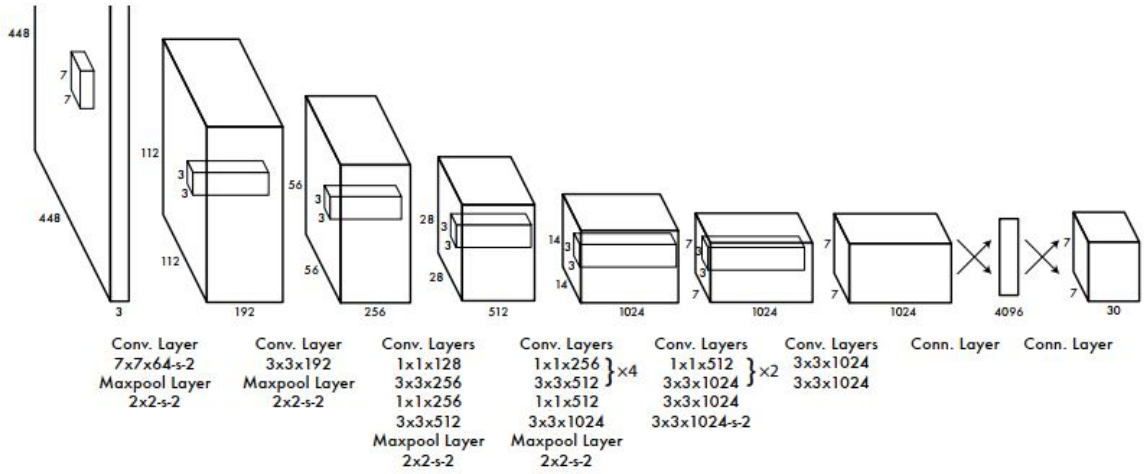


Figure 2.21: YOLO architecture [20].

from which two correspond to the center coordinates (x, y) of the boxes relative to the grid cell, while another two correspond to the height and width of the boxes relative to the whole image and the fifth prediction corresponds to the box objectness score computed through the IOU between the ground truth boxes and the predicted boxes, and finally, 'C' refers to the conditional class probabilities, corresponding to the classes in the dataset, which is only computed if the center of an object is present in the corresponding grid cell. All in all, the proposed system outperforms other methods in speed and generalizability from natural images to other domains like artwork but also suffers from high localization errors as well as struggles to generalize to objects in unusual configurations and also in detecting small objects appearing in groups.

Meanwhile, Liu et al. tread a similar path to propose Single Shot MultiBox Detector (SSD) [21] with a few distinct features from YOLO. Firstly, instead of developing a custom feature extraction backbone, they use the VGG-16 [42] architecture in which they replace the two FC layers consisting of 4096 channels with two convolutional layers, each with 1024 channels but the former with kernel sizes 3×3 and the latter with 1×1. The final 1000-channel FC layer is removed to make the network fully convolutional. Secondly, the authors add some auxiliary layers to the VGG base to generate additional feature maps for detection. Thirdly, to enable detections at multiple scales, feature maps are generated from all the auxiliary layers in addition to some layers

from the base. Fourthly, each cell from each feature map is associated with multiple anchor boxes of various sizes and aspect ratios to efficiently discretize the space of possible output box shapes. Lastly, predictions are made by feeding the multi-scale feature maps to final convolutional layers that predict the bounding box offsets and class probabilities. For illustration, refer to figure 2.22.

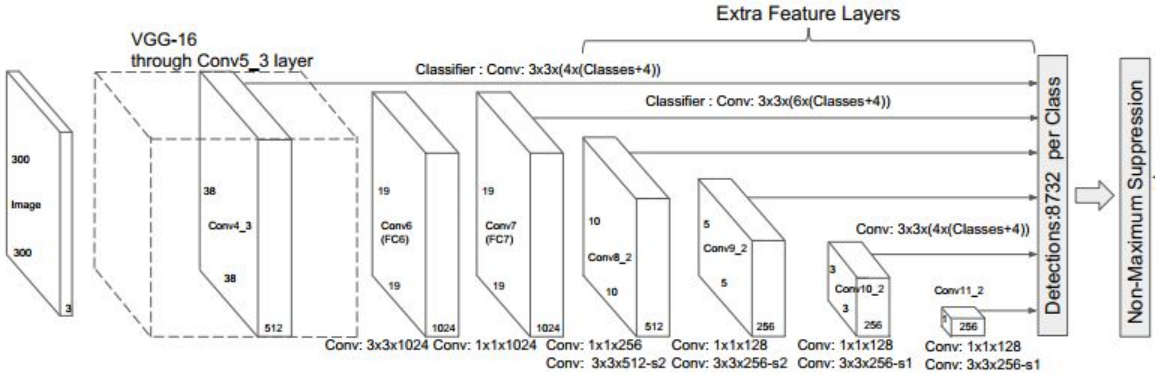


Figure 2.22: SSD architecture [21].

Lin et al. introduced an important improvement apart from the basic architecture as *feature pyramid networks (FPN)* [22], to enhance model accuracy. Although the authors use the FPN on top of Faster RCNN, it is a generic module which may be added on top of any detection architecture. A feature pyramid network consists of three components namely, bottom-up pathway, lateral connections and top-down pathway. The bottom-up pathway is the feed-forward path of the model which consists of multiple stages where an input feature map is down-sampled at every stage. The output feature maps from the bottom-up pathway are generated from multiple stages in the bottom-up pathway and are fed to the lateral connections that up-sample the feature maps to the same size as the feature maps in the previous stage. Finally, in the top-down pathway, the up-sampled feature maps from the latter layers are added to the feature maps in the earlier layers to create semantically strong maps at each scale (see figure 2.23 for illustration). The introduced module significantly improves the average precision (AP) of Faster RCNN from 47.3 to 56.9 on the COCO minival set.

Consequently, Lin et al. [23] note that although the single stage detectors have a speed advantage, yet they lag behind the two stage detectors in accuracy. For the purpose, the authors make three major changes in the SSD architecture. Firstly, they adopt the

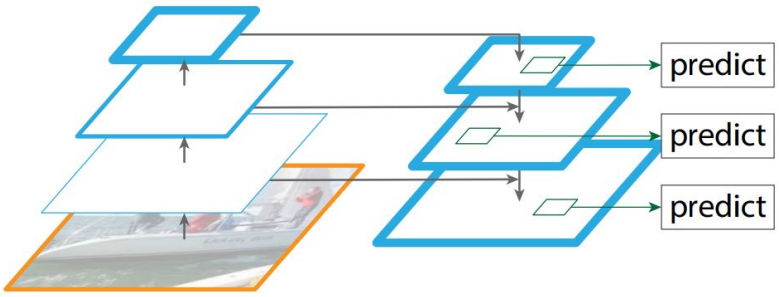


Figure 2.23: Illustration of feature pyramid networks [22].

FPN into the feature extraction backbone of SSD, to obtain stronger features are multiple scales. Secondly, instead of using shared auxiliary layers for object classification and box regression, they use two separate sub-networks for each task. Finally, they introduce a loss function termed as *focal loss* by tweaking the standard cross-entropy loss, to down-weight the loss assigned to well-classified examples and focus on the sparse set of hard examples. The resultant architecture is named RetinaNet which improves the mean average precision of one stage detectors while maintaining competitive speed (see figure 2.24 for illustration).

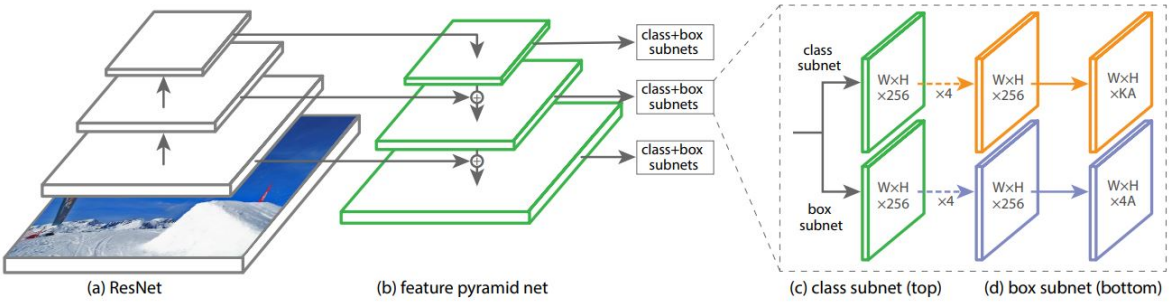


Figure 2.24: Illustration of RetinaNet [23].

Finally, it is worth mentioning that to avoid multiple detections of same object instances, overlapping detections above a certain threshold are suppressed, a technique known as non-maximum suppression (NMS) [67].

2.3 Research Gap

Following the bird's eye view of the available benchmarks, efficient network designs, few shot learning methods and object detection architectures, we now point out some problems not yet addressed.

2.3.1 Meta-Learning for Aerial Scene Classification

To the best of our knowledge, the only work adopting meta-learning for aerial scene classification was carried out by [68] in which the authors exploited an optimized adaptation method to address the challenges of data scarcity and disparity in the domain of remote sensing images. Other studies by [69] and [70] do utilize convolutional networks for metric-learning to achieve state-of-the-art performance, but not in the context of meta-learning. Hence, we find it interesting to investigate the performance of few shot metric learning approaches on aerial scene classification.

2.3.2 Backbones for Few Shot Learning Methods

Reviewing on one side the progress made in the design of efficient networks for feature extraction and on the other side the developments made in the area of few shot learning for image classification, we notice that both efforts are on different parts of the network and combining them might prove fruitful. We observe that the feature extraction backbone commonly used in state-of-the-art few shot learning methods for feature extraction has only four convolutional layers. The apparent reason for using shallow backbones for meta-learning is that the earlier layers learn more generic features while deeper layers learn specific features [58]. Increasing the depth of a meta-learning network too much may lead it to lose its genericity which would deem the network unfit for meta-learning. However, increasing the depth to some extent has been shown to improve accuracy as the representation power increases. Thus, it is an interesting direction to search for an optimal depth that provides the best trade-off between genericity and representation power, as it would help develop better lightweight few shot learning

networks.

2.3.3 Domain Transfer Between Different Meta-Domains

Although researchers have studied the generalizability of meta-learning models in cross-domain scenarios, however, we note that the base and target domains in such scenarios both belong to the same meta-domain. For instance, Chen et al. [63] compared the performance of different meta-learning methods with transfer learning, on domain shift from miniImageNet to CUB datasets. Similarly, Triantafillou et al. [24] compared the cross-domain generalizability of models trained jointly on all the datasets in the Meta-Dataset versus the models trained only on the ILSVRC2012. Likewise, Zhai et al. [68] evaluated the performance of various methods for the task of domain transfer from source to target datasets using different combinations of three aerial scene classification datasets. We observe that the datasets considered in the first two studies belong to the meta-domain of natural images, while those in the last study belong to aerial imagery. We argue that to achieve human-level intelligence, meta-learning methods should generalize well to various meta-domains. Therefore, it would be fascinating to explore methods to achieve the intended performance.

2.3.4 Modern Detectors with Lightweight Backbones for Object Detection in Aerial Imagery

Li et al. [40] evaluated 12 different representative detection methods on their proposed benchmark. Nevertheless, we note that the backbones of the detectors being evaluated are not computationally efficient. Performance evaluation of these modern detectors with state-of-the-art backbones is an exciting path to pursue.

2.4 Conclusion

In this thesis, we primarily focus on aerial scene classification and perform experiments to evaluate the performance of metric learning approaches with varying backbone depths while also considering the scenario of cross-meta-domain generalizability. Thereafter, we pursue the experiments for object detection in remote sensing imagery.

Chapter 3

Proposed Methodology

3.1 Problem Overview

Metric learning has been proven fruitful for aerial scene classification [69, 70], yet few shot metric learning approaches have not been evaluated on satellite imagery. Recent studies that have compared state-of-the-art few shot learning approaches show that *metric learning* methods display superior performance to *optimized adaptation* methods [24, 63]. Therefore, inspired from the work of [68] that adopt an optimized adaptation method for aerial scene classification, we instead choose few shot metric learning approaches for the task.

We find it necessary to mention here that some studies show transfer learning as a competitive approach to meta-learning [24, 63]. Nonetheless, *an analysis of the results reported by Triantafillou et al.* [24] reveals two interesting insights:

1. Finetune approach (transfer learning) has higher accuracy than Prototypical Networks (a representative metric learning method) when the base data distribution has higher complexity and the target data distribution has lower complexity. Conversely, when both the base and target distributions are of high complexity, then Prototypical Networks generally outperform the finetune approach (see figures 3.1 and 3.2).

2. Jointly training on the base and target distributions almost always leads to higher

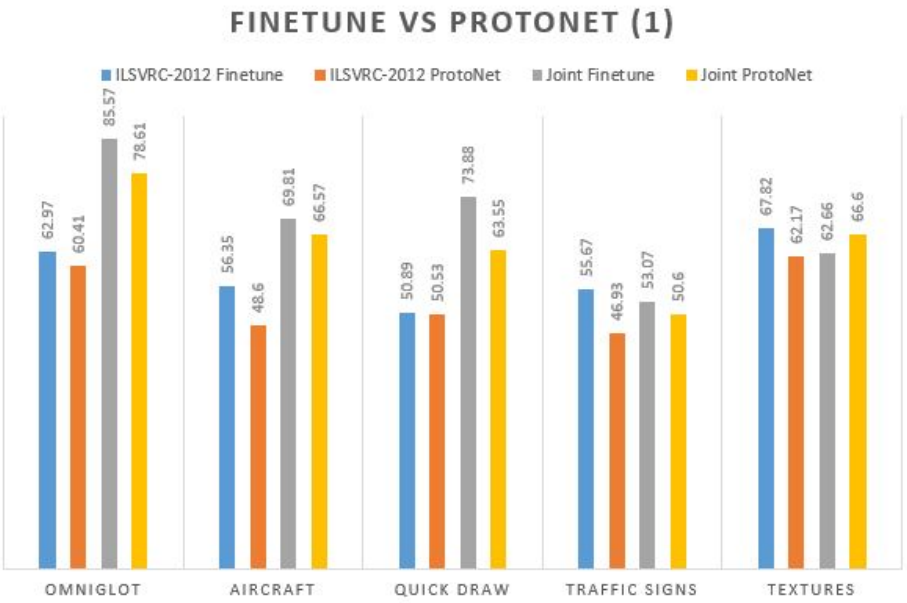


Figure 3.1: Analysis of the results reported in the Meta-Dataset paper [24]. We consider the five datasets: Omniglot [25], Aircraft, Quick Draw, Traffic Signs and Textures, to be of low complexity. *ILSVRC* denotes training only on the ILSVRC dataset (the base distribution) while *Joint* refers to training jointly on all the ten datasets in the Meta-Dataset.

accuracy for Prototypical Networks compared to training solely on the base distribution. In contrast, joint training mostly leads to an accuracy drop for the finetune method and only helps in some cases when the target distribution has lower complexity (see figures 3.1 and 3.2).

Considering these two insights from the analysis, we prefer metric learning over transfer learning in this work.

Another important point to note is that the datasets used for joint training in [24] all belong to the meta-domain of *natural imagery*. We have not come across any work studying the performance of jointly training on datasets from *distinct meta-domains*. We perceive that as the available datasets for aerial imagery have limited number of classes, training meta-learning models on small number of classes might lead to overfitting. A possible solution is to train the models on a dataset of natural images and use them directly on aerial images. However, the great disparity between the two meta-domains may not lead to better generalizability, hence joint training might lead to improved results.

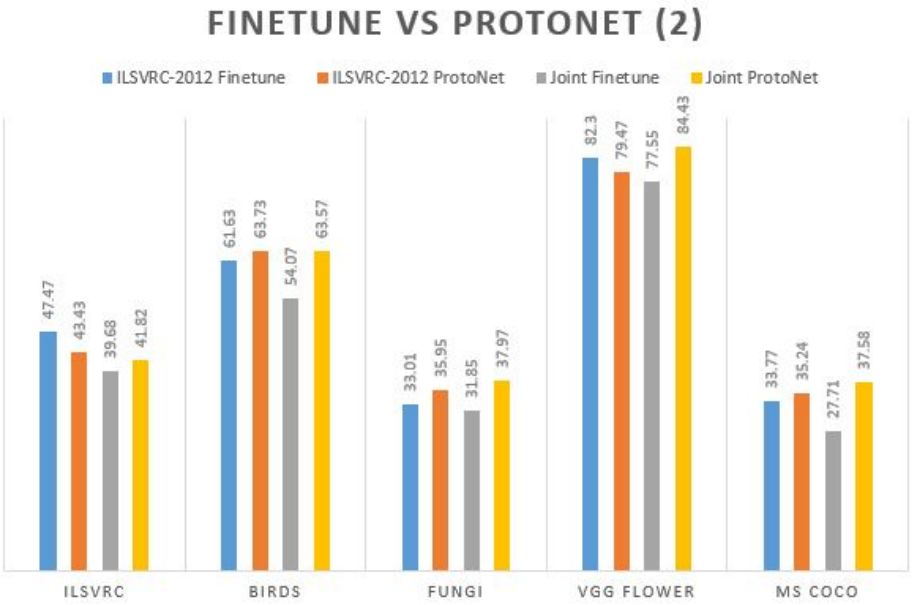


Figure 3.2: Analysis of the results reported in the Meta-Dataset paper [24]. We consider the five datasets: ILSVRC [26], Birds, Fungi, VGG Flower and MS COCO [27], to be of high complexity. *ILSVRC* denotes training only on the ILSVRC dataset (the base distribution) while *Joint* refers to training jointly on all the ten datasets in the Meta-Dataset.

Further improvements in accuracy may be achieved by increasing network depth as suggested by Chen et al. [63] but very deep networks may lose genericity which is an essential feature of meta-learning models. Thus, a study examining the network's performance by varying its depth is required similar to [42]. Finally, considering the unavailability of a dataset of aerial imagery for few shot detection, we instead adopt the standard approach for object detection by training on a large scale dataset with efficient networks, with hope that it would aid in wise selection of network designs for few shot object detection in satellite imagery.

3.2 Our Approach

To address the problems described above, we follow the approach below:

1. Selection of State-of-the-Art Few Shot Metric Learning Methods:
   Based on recent reports, we select Prototypical Networks [16] and Relation Net-

works [17] for our experiments.

2. Validation of Implementation:
   We validate our implementations of the two few shot learning methods by comparing our results on the MiniImageNet [15] few shot classification benchmark with the reported results.

3. Experiments on Aerial Imagery:
   We select the NWPU-RESISC45 [36] dataset for our experiments on satellite imagery as it is state-of-the-art with high intra-class diversity and inter-class similarity, and has relatively larger number of classes besides large number of images per class.

4. Joint Training:
   We evaluate if joint training leads to performance improvement by testing on the NWPU-RESISC45 test classes. For the given purpose, we first evaluate the cross-domain generalizability from MiniImageNet [15] to NWPU-RESISC45 [36] and then compare the performance with models trained jointly on both the datasets.

5. Analyse Depth Effects:
   We choose three efficient networks (MobileNet v2 [10], ShuffleNet v2 [12] and ESPNet v2 [14]) for our experiments and study the effects of varying network depth by using seven depth levels.

6. Detection in Satellite Imagery:
   We then use the same three efficient networks (MobileNet v2 [10], ShuffleNet v2 [12] and ESPNet v2 [14]) for feature extraction in the detection experiments. The detection architecture chosen is RetinaNet as it shows the best performance on the DIOR dataset and we replace its ResNet-50 backbone with the efficient backbones to compare the performance.

Chapter 4

Implementation

In this chapter, we present the details about the experimental setup. In particular, we mention the resources utilized, explain the process of data preparation, relate the implementations used, outline the procedure adopted to modify the network architectures to suit the relevant scenario, and describe the choice of hyper-parameters and training procedure in different experimental settings.

4.1 Resources

We implement the neural networks using the PyTorch framework [71] for its Pythonic and object-oriented programming style, and mid-level complexity that allows more intuition, flexibility and control than Keras [72] while handling inherent intricacies internally thus providing a simpler interface than TensorFlow [73]. For the experiments, we use free instances of NVIDIA Tesla K80 GPU provided by Google Colaboratory [74]. The GPU has 68 GB memory, 12 GB RAM and compute capability of 3.7, thus supported by CUDA library [75] from NVIDIA which requires the compute capability to be at least 3.0.

4.2 Data Preparation

Hereunder, we relate the datasets used in our experiments, how they were split into the train, validation and test sets and the sizes we used for the images. Furthermore, we also describe the data sampling techniques and the format we used to store data.

4.2.1 Datasets

For scene classification experiments, we used the MiniImageNet [15] and NWPU-RESISC45 [36] datasets whereas for object detection, we used the DIOR dataset [40].

4.2.2 Data Splits

The details of how different datasets were split for relevant experiments are presented below:

1. Validation Experiments:
   To validate our experiments, we used the MiniImageNet benchmark and the splits proposed by Vinyals et al. [15] that use 64 classes for train set, 16 for validation set and 20 for test set.

2. Aerial Scene Classification Experiments:
   For experiments on the NWPU-RESISC45 dataset [36], from the 45 classes we use 20 for training, five for validation and 20 for testing. The names of the scene classes used in each split are given in Table 4.1.

3. Domain Transfer Experiments:
   For experiments on how well the meta-learning models generalize from natural images to satellite images, we use MiniImageNet for training and NWPU-RESISC45 for testing. For the train set, we merge the train and test sets of MiniImageNet proposed by Vinyals et al. [15] and for the test set, we merge the train and test sets of NWPU-RESISC45 dataset. Finally, we merge the validation sets of both the datasets to create the validation set.

4. Joint Training Experiments:

   In order to evaluate the effectiveness of joint training, we train jointly on Mini-ImageNet [15] and NWPU-RESISC45 [36] datasets. For the stated purpose, we merge the train sets of both datasets to create the new train set and create the other splits in a similar fashion.

5. Detection Experiments:

   For detection experiments, we use the splits of DIOR dataset proposed by Li et al. [40], which has 5862 images in the train set, 5863 images in the validation set and 11,738 images in the test set.

Splits	Categories
Train	baseball-diamond, beach, chaparral, circular-farmland, forest, freeway, golf-course, island, meadow, medium-residential, mobile-home-park, overpass, railway-station, rectangular-farmland, roundabout, sea-ice, snow-berg, sparse-residential, stadium, wetland
Validation	cloud, harbor, mountain, storage-tank, thermal-power-station
Test	airplane, airport, basketball-court, bridge, church, commercial-area, dense-residential, desert, ground-track-field, industrial-area, intersection, lake, palace, parking-lot, railway, river, runway, ship, tennis-court, terrace

Table 4.1: NWPU-RESISC45 Splits

4.2.3 Image Sizes

For the experiments involving MiniImageNet [15] and NWPU-RESISC45 [36], we resize the images to 126×126 pixels taking inspiration from [24] whereas for DIOR experiments, we keep the original image size of 800×800 pixels.

4.2.4 Data Storage Format

We note that the usual format of storing data is to store images in a folder and the annotations in a file which consists of paths to the images. While loading data, the images are read from the paths using image processing libraries. However, this approach is time-consuming and slows down the training process. Considering this and the small

size of our *classification* datasets, we instead store the data splits for classification experiments as a list of tuples, consisting of image tensors and the corresponding labels, in pickle format and load them to GPU RAM for faster training.

4.2.5 Data Samplers

Unlike traditional deep learning samplers that randomly sample data from the whole dataset according to the batch size, few shot learning requires customized samplers appropriate to the method. For a class-balanced scenario, which also is our case, the few shot sampler takes in three inputs:

1. the number of classes from which the data is to be selected (n-way),
2. the number of examples that serve as the meta-training/support set (k-shot) and
3. the number of examples that would be used for meta-testing of the model (k-query).

The sampler first randomly selects 'n-way' classes from the given data split, then picks a sum of 'k-shot' and 'k-query' images from each class and returns the selection as a batch. It is also important to note that later data processing techniques may require the sampled batch to be in a specific order. For our case, we used different samplers for Prototypical Networks and Relation Networks. Table 4.2 shows an example of how data is arranged by the two samplers.

Method	Sampled Batch
Prototypical Networks	A,B,C,A,B,C,A,B,C
Relation Networks	A,A,A,B,B,B,C,C,C

Table 4.2: Data samplers for few shot learning methods. Please note that A, B and C represent three classes making it a *3-way* scenario while the repetition of an item from each class for three times makes it a *3-shot* scenario.

4.3 Network Implementations

For few shot learning, we adopt the re-implementation of Prototypical Networks by Chen [76] and modify the original implementation of Relation Networks by Sung [77] to harmonize with Chen's implementation [76]. As for efficient networks, we use the official implementations available in PyTorch's [71] *torchvision* library for MobileNet v2 [10] and ShuffleNet v2 [12] while use the original implementation of ESPNet v2 by Mehta et al. [14]. Lastly, we use the RetinaNet implementation by MMDetection Open Toolbox [78].

4.4 Modifications to Networks

We conduct our experiments for classification and detection separately. For classification, we perform the experiments in two phases which we refer to as the *initial* phase and the *follow-up* phase. The details of networks used for the different experiments are given below.

4.4.1 Initial Experiments

For the initial experiments, we use a convolutional network with four layers (Conv-4) in order to validate our implementations to the reported works. For Prototypical Networks [16], we use the Conv-4 implementation by Chen [76] while for Relation Networks [17], we use the implementation by Sung [77]. We set the results from Conv-4 as the baseline to compare the performance of three efficient networks namely, ESPNet v2 [14], MobileNet v2 [10] and ShuffleNet v2 [12]. For each network, we discard the classification layers and only keep the feature extraction layers. We further reduce the number of channels in the final layer of each network to 512 channels to get the final feature map of size $512 \times 4 \times 4 = 8192$ for an input of size 126×126 pixels. Furthermore, we notice that these efficient networks are built from basic modules such that the chain of modules is between two convolutional layers. With this observation,

we use two depth levels for each network, *full* and *half* depth. In full depth, we keep all the modules whereas in half depth, we reduce the number of modules almost to half. We evaluate these networks on three tasks:

1. Generalization to unseen classes in aerial imagery by using the NWPU-RESISC45 dataset [36].

2. Domain transfer from natural to aerial imagery by training on MiniImageNet [15] and testing on NWPU-RESISC45 [36].

3. Joint training on MiniImageNet [15] and NWPU-RESISC45 [36] to assess performance gains over domain transfer.

Thereafter, we perform a set of follow-up experiments to further study the effects of varying depth, the details of which are presented underneath.

4.4.2 Follow-up Experiments

In the follow-up experiments, we make two major changes to the networks. Firstly, we change the number of channels in the final layer of backbones from 512 to 128 channels in order to reduce network capacity, hence lowering the chances of over-fitting. Secondly, we adopt a different scheme of varying depths than the one in the previous experiments. We note that the efficient networks we had selected have different stages, where each stage has a particular configuration of basic modules. Observing that ESPNet v2 [14] and ShuffleNet v2 [12] have three stages of the basic modules, we choose to use three depth levels. For the first level, we only keep the first stage of the basic modules and similarly keep two and three stages for second and third levels respectively. As for MobileNet v2 [10], it has seven stages of the basic modules, so we decide to use three, five and seven stages for first, second and third levels respectively. Not to mention, these networks of different depths would downsample the image to different sizes but it would be desirable for the output to be of the same size. Hence, we increase the stride of convolutional layers and additionally insert a max-pool layer where necessary to make the network strides equal.

4.4.3 Detection Experiments

We use the RetinaNet architecture for object detection in satellite imagery and make only two modifications to the original implementation. Firstly, we replace the ResNet [4] backbone for feature extraction with the selected efficient networks and secondly, we modify the number of input channels to feature pyramid network accordingly.

4.5 Training Procedure

For classification experiments, we perform the experiments in two settings of 5-way 1-shot and 5-way 5-shot. Additionally, for initial experiments, we also train and test in 20-way settings for *domain transfer* and *joint training* experiments. We train our models for 200 epochs with learning rate = 0.001 using the Adam optimization scheme [79]. Two other important hyper-parameters are the number of *sampled batches* which are in fact *mini-datasets generated from the meta-data* and the *number of query samples* (k-query) in a sampled batch. We keep the sampled batches = 500 for both phases of the experiments and use k-query = 5 for the initial experiments while k-query = 15 and k-query = 10 for 5-way 1-shot and 5-way 5-shot settings of the follow-up experiments respectively.

In the detection experiments on DIOR dataset [40], we use ImageNet pretrained backbones for training while the other layers are randomly initialized. We utilize the SGD optimizer with learning rate = 0.001 and train for 12 epochs. Finally, we keep the original image size of 800×800 pixels and the mini-batch size = 2.

Chapter 5

Results and Discussion

We evaluate three state-of-the-art efficient networks with five configurations on five datasets for few shot classification in 5-way 1-shot and 5-way 5-shot settings. We detail below the results obtained from our experiments.

5.1 Validation of Implementations

We first validate our implementations to the reported results on the MiniImageNet dataset [15]. Table 5.1 shows that our results lag behind the reported ones by 2% for Prototypical Networks and 4% for Relation Networks. We attribute this difference to the difference in hyper-parameters settings. Furthermore, the difference of 4% for Relation Networks is because we use the same hyper-parameters used for Prototypical Networks and do not optimize them for Relation Networks.

	ProtoNet		RelationNet	
	5-way 1-shot	5-way 5-shot	5-way 1-shot	5-way 5-shot
Reported	49.42	68.2	50.44	65.32
Ours	47.31	68.05	47.84	61.74

Table 5.1: Accuracy comparison of our implementations with the reported results on the MiniImageNet benchmark

		5-way 1-shot		5-way 5-shot	
		ProtoNet	RelationNet	ProtoNet	RelationNet
Conv-4 (baseline)		45.68	46.02	65.38	50.59
ESPNet	First	45.69	41.63	64.25	52.16
	Second	47.96	43.39	65.15	55.2
	Third	44.59	42.41	60.58	54.95
	Half	49.78	42.84	58.85	51.38
	Full	48.19	40.74	63.23	42.20
MobileNet	First	47.46	45.16	63.9	54.43
	Second	49.26	44.42	66.32	55.97
	Third	48.14	45.82	66.14	58.95
	Half	**51.36**	42.94	**67.63**	53.64
	Full	**51.02**	39.88	**67.69**	52.25
ShuffleNet	First	48.51	40.36	65.55	53.26
	Second	48.11	44.78	66.25	54.99
	Third	48.15	44.87	67.24	52.27
	Half	46.32	40.03	65.22	53.40
	Full	48.11	41.78	64.84	50.91

Table 5.2: Comparison of average accuracy between Prototypical Networks and Relation Networks on the NWPU-RESISC45 dataset, where first, second, third, half and full refer to different depth levels of the networks.

5.2 Generalization to Unseen Classes

To evaluate the ability to recognize unseen classes given only a few examples, we train and test Prototypical Networks and Relation Networks with different backbones on the NWPU-RESISC45 dataset [36]. The results are shown in Table 5.2 and depicted in figure 5.1

Our results suggest that Prototypical Networks clearly outperform Relation Networks in all scenarios. The apparent reason to this is that although Relation Networks learn a non-linear distance metric which may be more discriminative than Euclidean distance, the learning of such a metric itself requires large amounts of training. Another probable reason might be that the *relation module* in Relation Networks which is responsible for learning the distance metric is a neural network and the current design of the network might not be suitable for aerial imagery with high within-class diversity and between-class similarity. Finally, the poor performance of Relation Networks in part is also attributed to the unoptimized hyper-parameters. Another observation we make is

(a) 5-way 1-shot (ProtoNet)

(b) 5-way 1-shot (RelationNet)

(c) 5-way 5-shot (ProtoNet)

(d) 5-way 5-shot (RelationNet)

Figure 5.1: Average accuracy on NWPU-RESISC45 test classes. ProtoNet and RelationNet refer to Prototypical Network and Relation Network respectively, while first, second, third, half and full refer to different depth levels of the feature extraction backbone.

that the MobileNet v2 with half and full depth configurations provide the maximum accuracy advantage compared to other networks and configurations.

5.3 Effectiveness of Joint Training

We evaluate the effectiveness of joint training in two scenarios. In the first scenario, we use the model weights from training on MiniImageNet and test on the NWPU-RESISC45 test-set. In the second scenario, we use the results obtained from NWPU-RESISC45 experiments. We compare the results of joint training to both sets of results to examine if it improves the performance.

5.3.1 Domain Transfer vs. Joint Training

Triantafillou et al. evaluated different few shot learning methods on the Meta-Dataset. For evaluation, the authors compare the performance of models trained only on ImageNet [26] with those trained on all the datasets in the Meta-Dataset. They conclude that the performance gain achieved by training on the Meta-Dataset is not significant compared to training only on ImageNet and in some cases joint training leads to performance drop. We however observe that this might be due to the fact that all the datasets in the Meta-Dataset belong to the domain of natural images and despite that the datasets belong to different data distributions, they still have high degree of similarity. Thus, we evaluate if joint training on two highly dissimilar data distributions leads to performance improvement over domain transfer from one to another distribution and our results clearly suggest that models jointly trained have superior performance (see tables 5.3 and 5.4, and Figure 5.2).

	5-way 1-shot			
	ProtoNet		RelNet	
	Transfer	Joint	Transfer	Joint
Conv-4	36.22	39.78	37.21	42.72
ESPNet_Half_Depth	40.02	45.49	32.56	20.61
ESPNet_Full_Depth	37.41	45.05	33.74	41.94
MobileNet_Half_Depth	35.99	45.26	32.97	37.57
MobileNet_Full_Depth	35.19	43.41	32.66	38.23
ShuffleNet_Half_Depth	38.04	43.21	31.61	35.46
ShuffleNet_Full_Depth	35.37	45.87	32.03	40.93

Table 5.3: Average Accuracy comparison between domain transfer and joint training on the NWPU-RESISC45 test classes (5-way 1-shot). Note that Transfer refers to training on MiniImageNet train classes and testing on NWPU-RESISC45 test classes, which is a transfer from natural to satellite imagery, while Joint refers to training jointly on MiniImageNet and NWPU-RESISC45 train classes and testing on NWPU-RESISC45 test classes.

On a note, we would like to mention that we did not use the full ImageNet dataset for our experiments due to memory and time constraints and opted to instead use MiniImageNet in this work.

	5-way 5-shot			
	ProtoNet		RelNet	
	Transfer	Joint	Transfer	Joint
Conv-4	60.81	66.03	47.96	52.82
ESPNet_Half_Depth	55.31	63.89	40.14	49.66
ESPNet_Full_Depth	52.14	63.41	45.56	50.50
MobileNet_Half_Depth	53.39	66.00	45.64	47.12
MobileNet_Full_Depth	52.01	64.91	41.35	51.83
ShuffleNet_Half_Depth	53.14	62.41	39.21	55.04
ShuffleNet_Full_Depth	50.34	62.90	39.62	50.24

Table 5.4: Average Accuracy comparison between domain transfer and joint training on the NWPU-RESISC45 test classes (5-way 5-shot). Note that Transfer refers to training on MiniImageNet train classes and testing on NWPU-RESISC45 test classes, which is a transfer from natural to satellite imagery, while Joint refers to training jointly on MiniImageNet and NWPU-RESISC45 train classes and testing on NWPU-RESISC45 test classes.

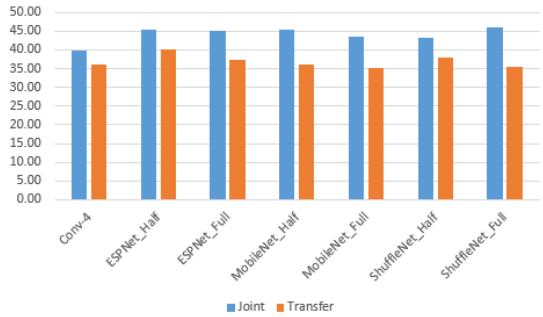
5.3.2 Sole Training vs. Joint Training

After we confirm the effectiveness of joint training, we analyse whether it also gives a performance advantage over training solely on aerial imagery. For the purpose, we compare the results from the NWPU-RESISC45 and joint training experiments. Analysing the results from figure 5.3 and 5.4, we conclude that joint training usually gives poorer performance than sole training while in some cases it shows comparable or slightly superior results. Therefore, we infer that provided a well-designed backbone, joint training would provide superior performance than sole training.

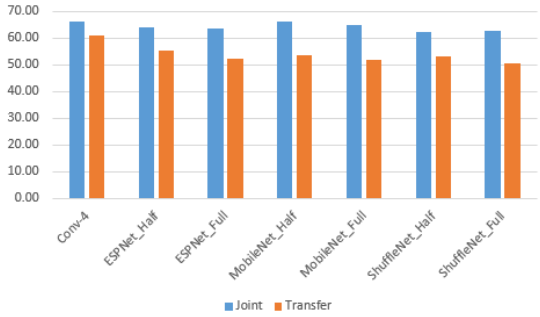
5.4 Effects of Varying Depth

Searching for optimal depth to increase representation power without compromising genericity would push forward the performance and development of meta-learning models. We started with the assumption that the accuracy of models increase with depth due to the ability to learn more powerful representations but after a threshold, the accuracy drops due to loss in genericity.

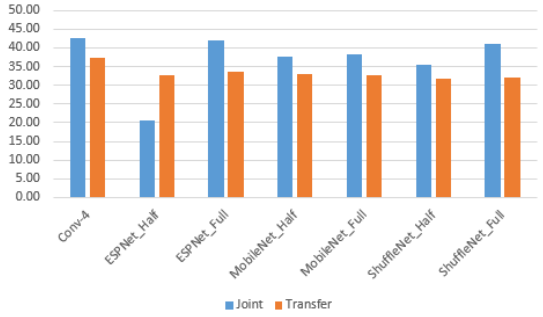
We analyse the results in 5.1 and conclude that the effects of depth are unclear from the graphs. Therefore, we perform another set of experiments with Prototypical Networks



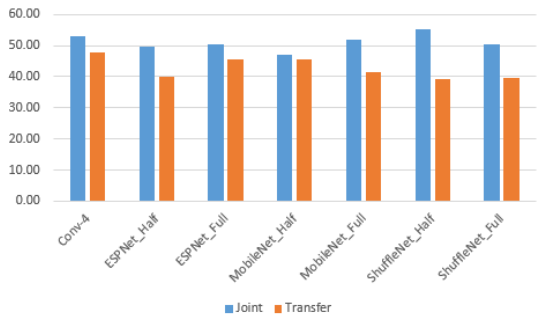
(a) 5-way 1-shot (ProtoNet)



(b) 5-way 5-shot (ProtoNet)

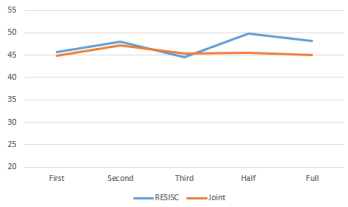


(c) 5-way 1-shot (RelationNet)

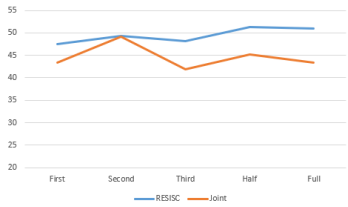


(d) 5-way 5-shot (RelationNet)

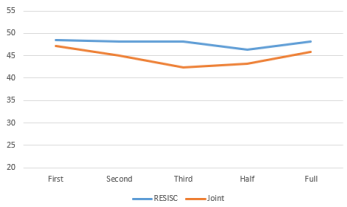
Figure 5.2: Average Accuracy comparison between domain transfer and joint training on the NWPU-RESISC45 test classes. Note that Transfer refers to training on MiniImageNet train classes and testing on NWPU-RESISC45 test classes, which is a transfer from natural to satellite imagery, while Joint refers to training jointly on MiniImageNet and NWPU-RESISC45 train classes and testing on NWPU-RESISC45 test classes.



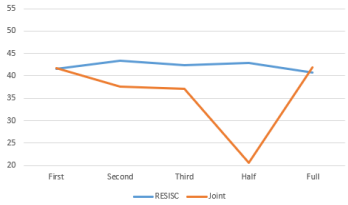
(a) ESPNet (ProtoNet)



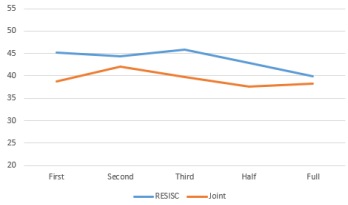
(b) MobileNet (ProtoNet)



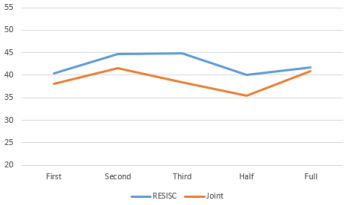
(c) ShuffleNet (ProtoNet)



(d) ESPNet (RelationNet)

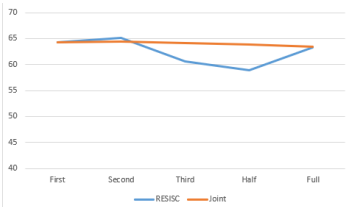


(e) MobileNet (RelationNet)

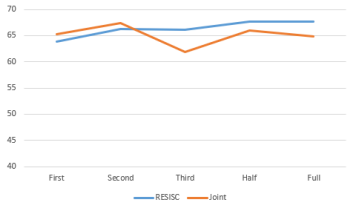


(f) ShuffleNet (RelationNet)

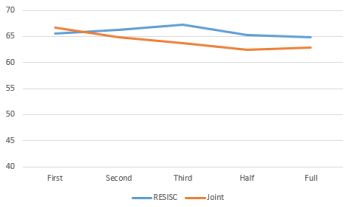
Figure 5.3: Average Accuracy comparison between sole training and joint training on the NWPU-RESISC45 test classes (5-way 1-shot). Note that RESISC refers to training on NWPU-RESISC45 train classes and testing on NWPU-RESISC45 test classes, while Joint refers to training jointly on MiniImageNet and NWPU-RESISC45 train classes and testing on NWPU-RESISC45 test classes.



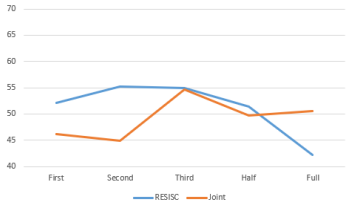
(a) ESPNet (ProtoNet)



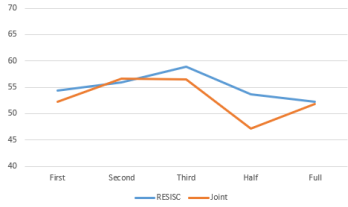
(b) MobileNet (ProtoNet)



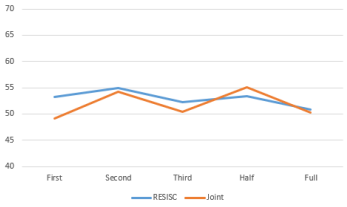
(c) ShuffleNet (ProtoNet)



(d) ESPNet (RelationNet)

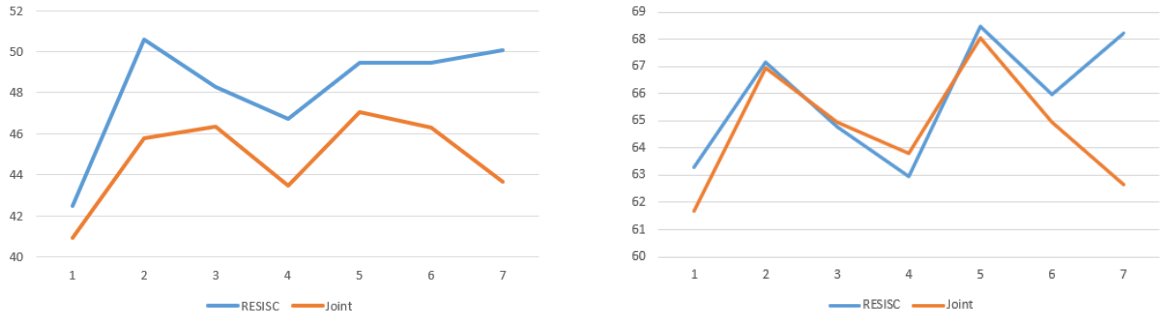


(e) MobileNet (RelationNet)



(f) ShuffleNet (RelationNet)

Figure 5.4: Average Accuracy comparison between sole training and joint training on the NWPU-RESISC45 test classes (5-way 5-shot). Note that RESISC refers to training on NWPU-RESISC45 train classes and testing on NWPU-RESISC45 test classes, while Joint refers to training jointly on MiniImageNet and NWPU-RESISC45 train classes and testing on NWPU-RESISC45 test classes.



(a) 5-way 1-shot **(b)** 5-way 5-shot

Figure 5.5: Analysis of increasing network depth with Prototypical Networks. The backbone network used is MobileNet v2 with seven different depth levels.

and MobileNet v2 backbone using seven depth levels to gain a deeper understanding of the effects of increasing network depth. The results shown in figure 5.5 form a zig-zag pattern and we observe that although in the 5-way 5-shot scenario, depth level five of MobileNet v2 gives greater accuracy than depth level two, but we observe an accuracy drop between the two levels. Therefore, we conclude that the current design of MobileNet v2 is not specifically designed for the problem of prototype learning. We consider it an interesting future direction to develop feature extraction networks specifically to enhance the accuracy of few shot metric learning, or in general, the meta-learning approaches.

5.5 Detection in Satellite Imagery

For the detection task, we first train the RetinaNet detector with ResNet-50 backbone and compare our results to the reported results in [40]. However, the authors did not report the training settings used and so we used the standard training setting of the MMDetection Toolbox [78] that uses the SGD optimizer with learning rate = 0.001, momentum = 0.9, weight decay = 0.0001 and trained for 12 epochs. The comparison of our results (see figure 5.6) to the reported results show that our results lag behind the reported ones by a significant margin, which we consider to be due to a difference in training settings.

Furthermore, we train the RetinaNet detector using the same settings with the ResNet-

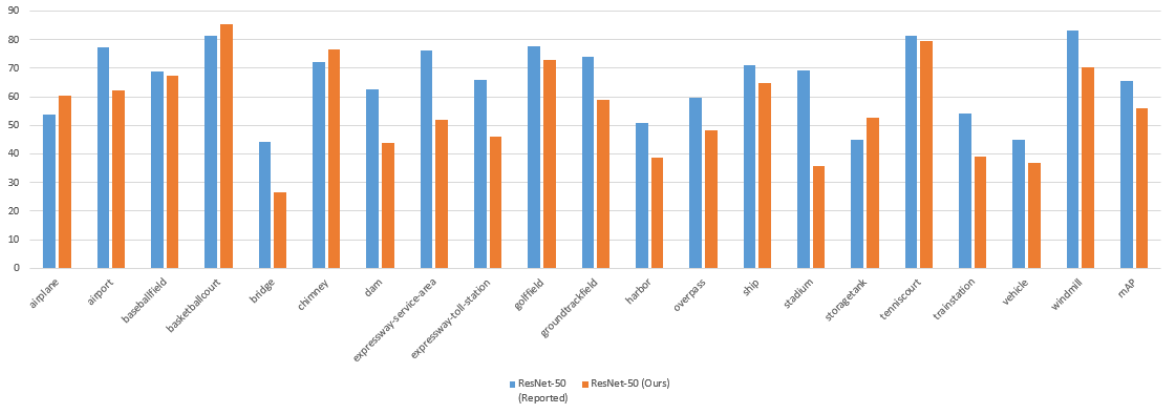
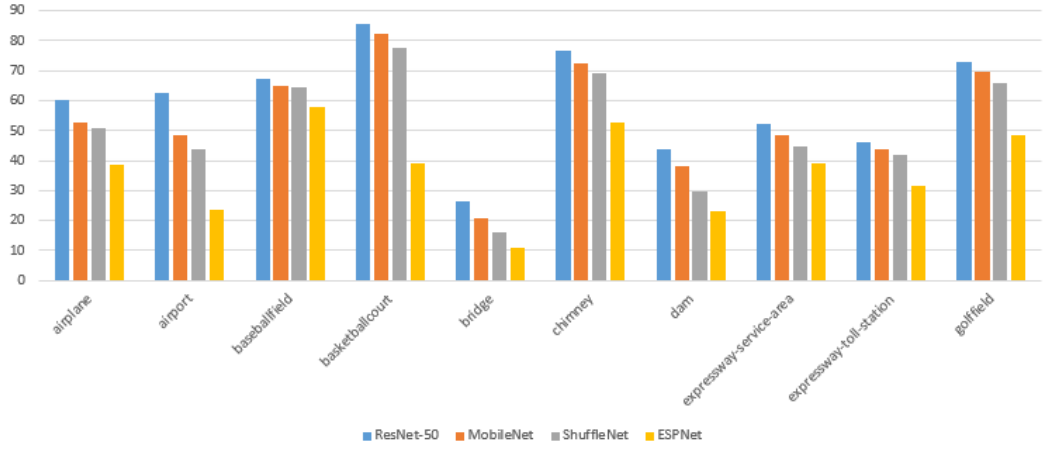
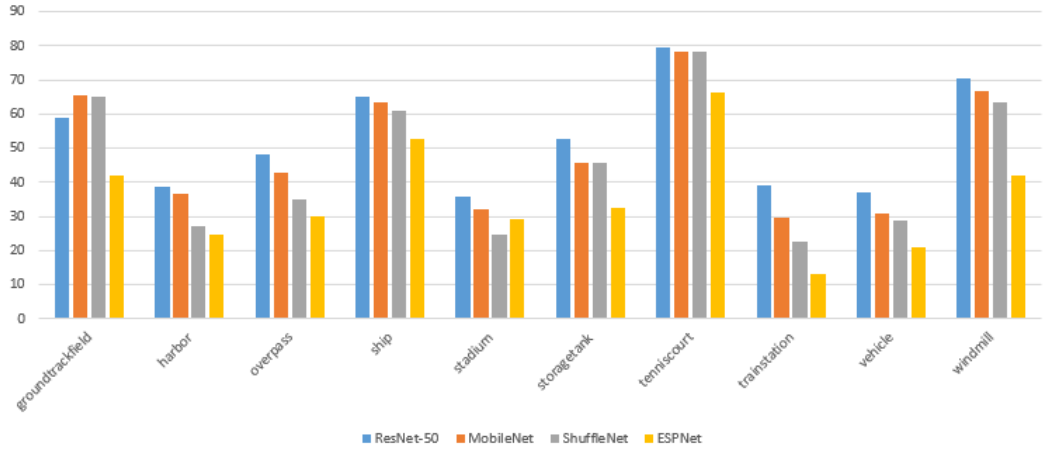


Figure 5.6: Comparison of our detection results on the DIOR test set with the reported results using the RetinaNet architecture with ResNet-50 backbone.



(a) Detection results on first 10 classes in the DIOR dataset.



(b) Detection results on last 10 classes in the DIOR dataset.

Figure 5.7: Average precision of RetinaNet on DIOR test set with different backbones.

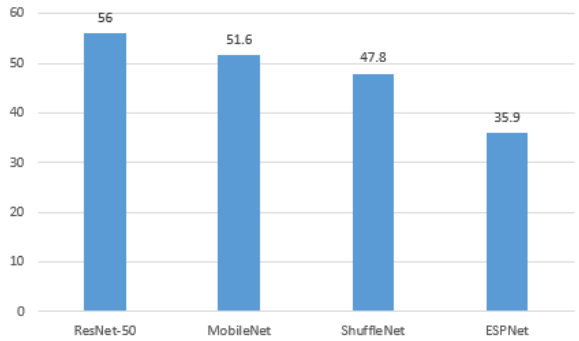


Figure 5.8: Mean average precision of RetinaNet with four different backbones on the DIOR test set.

50 backbone replaced by ESPNet v2, MobileNet v2 and ShuffleNet v2. The average precision (AP) results are shown in figure 5.7, which show that MobileNet gives the closest results to ResNet-50, while ShuffleNet shows to be the second closest and ESP-Net gives the poorest performance on the detection task. The mean average precision results are shown in figure 5.8.

Chapter 6

Conclusion

In this work we evaluated the performance of state-of-the-art metric learning approaches on aerial imagery. Our results suggest that Prototypical Networks show better performance than Relation Networks on a satellite dataset with less number of classes. We also found that jointly training on base and target tasks leads to slightly improved performance provided appropriate feature extraction backbones. Moreover, our results suggest a need to develop deep networks specially designed for the problem of meta-learning as well as to prepare a dataset for meta-learning consisting of classes from highly dissimilar domains like natural, satellite and biomedical imagery. Finally, we find that from the three competitive efficient networks we selected, the MobileNet architecture shows better performance in both the few shot scene classification and object detection tasks. In future, we plan to further research the effects of different architectural choices in the network design space to improve the few shot learning accuracy. Using the network learning approaches [50–54] for feature extraction, and a combination of metric learning approaches [15–17] with optimized adaptation approaches [59–61] for classification, might prove fruitful. Yet another fascinating direction would be to develop a semi-supervised learning approach to cater for unlabeled examples in the train set, similar to the work by [80]. In conclusion, for accurate few shot object detection in satellite imagery, development of relevant dataset and stronger detection architectures would be our next course of choice.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[2] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv preprint arXiv:1312.4400, Neural and Evolutionary Computing, Computing Research Repository (CoRR)*, 2013.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2014.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015.

[5] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, June 2016.

[6] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2016.

[7] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2016.

[8] Gao Huang, Shichen Liu, Laurens van der Maaten, and Kilian Q. Weinberger. CondenseNet: An Efficient DenseNet Using Learned Group Convolutions. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2752–2761, 2017.

[9] Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size. *arXiv preprint arXiv:1602.07360, Computer Vision and Pattern Recognition, Computing Research Repository (CoRR)*, 2017.

[10] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[11] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2017.

[12] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. In *The European Conference on Computer Vision (ECCV)*, pages 116–131, September 2018.

[13] Sachin Mehta, Mohammad Rastegari, Anat Caspi, Linda Shapiro, and Hannaneh Hajishirzi. ESPNet: Efficient Spatial Pyramid of Dilated Convolutions for Se-

mantic Segmentation. In *The European Conference on Computer Vision (ECCV)*, pages 552–568, September 2018.

[14] Sachin Mehta, Mohammad Rastegari, Linda Shapiro, and Hannaneh Hajishirzi. ESPNetv2: A Light-Weight, Power Efficient, and General Purpose Convolutional Neural Network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9190–9200, June 2019.

[15] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems 29*, pages 3630–3638. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6385-matching-networks-for-one-shot-learning.pdf.

[16] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical Networks for Few-shot Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4077–4087. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/6996-prototypical-networks-for-few-shot-learning.pdf.

[17] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H.S. Torr, and Timothy M. Hospedales. Learning to Compare: Relation Network for Few-Shot Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1199–1208, June 2018.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[19] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: Object Detection via Region-based Fully Convolutional Networks. In D. D. Lee,

M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 379–387. Curran Associates, Inc., 2016. URL http://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks.pdf.

[20] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2015.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single Shot Multibox Detector. In *European conference on computer vision (ECCV)*, pages 21–37. Springer, 2016.

[22] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature Pyramid Networks for Object Detection. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2016.

[23] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[24] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgAGAVKPr.

[25] Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266): 1332–1338, 2015. ISSN 0036-8075. doi: 10.1126/science.aab3050. URL https://science.sciencemag.org/content/350/6266/1332.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115:211–252, 2014.

[27] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[28] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. Technical report, University of Toronto, Toronto, Canada, 2009. URL https://www.cs.toronto.edu/~kriz/cifar.html.

[29] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88:303–338, 2009.

[30] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The Open Images Dataset V4. *International Journal of Computer Vision*, pages 1 – 26, 2020.

[31] Sachin Ravi and Hugo Larochelle. Optimization as a Model for Few-Shot Learning. In *International Conference on Learning Representations (ICLR)*, Toulon, France, April 2017. URL https://openreview.net/pdf?id=rJY0-Kcll.

[32] Yi Yang and Shawn Newsam. Bag-of-Visual-Words and Spatial Extensions for Land-Use Classification. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, page 270–279, New York, NY, USA, 2010. Association for Computing Machinery. ISBN 9781450304283. doi: 10.1145/1869790.1869829. URL https://doi.org/10.1145/1869790.1869829.

[33] Guofeng Sheng, Wen Yang, Tao Xu, and Hong Sun. High-resolution satellite scene classification using a sparse coding based multiple feature combination. *International Journal of Remote Sensing*, 33(8):2395–2412, 2012. doi: 10.1080/01431161.2011.608740. URL https://doi.org/10.1080/01431161.2011.608740.

[34] Lijun Zhao, Ping Tang, and Lianzhi Huo. Feature significance-based multibag-of-visual-words model for remote sensing image scene classification. *Journal of Applied Remote Sensing*, 10(3):1 – 21, 2016. doi: 10.1117/1.JRS.10.035004. URL https://doi.org/10.1117/1.JRS.10.035004.

[35] G. Xia, J. Hu, F. Hu, B. Shi, X. Bai, Y. Zhong, L. Zhang, and X. Lu. AID: A Benchmark Data Set for Performance Evaluation of Aerial Scene Classification. *IEEE Transactions on Geoscience and Remote Sensing*, 55(7):3965–3981, July 2017. ISSN 1558-0644. doi: 10.1109/TGRS.2017.2685945.

[36] G. Cheng, J. Han, and X. Lu. Remote Sensing Image Scene Classification: Benchmark and State of the Art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

[37] Gong Cheng and Junwei Han. A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:11 – 28, 2016. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2016.03.014. URL http://www.sciencedirect.com/science/article/pii/S0924271616300144.

[38] Sebastien Razakarivony and Frederic Jurie. Vehicle detection in aerial imagery : A small target detection benchmark. *Journal of Visual Communication and Image Representation*, 34:187 – 203, 2016. ISSN 1047-3203. doi: https://doi.org/10.1016/j.jvcir.2015.11.002. URL http://www.sciencedirect.com/science/article/pii/S1047320315002187.

[39] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. DOTA: A Large-Scale Dataset for Object Detection in Aerial Images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3974–3983, June 2018.

[40] Ke Li, Gang Wan, Gong Cheng, Liqiu Meng, and Junwei Han. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159:296 – 307, 2020. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2019.11.023. URL http://www.sciencedirect.com/science/article/pii/S0924271619302825.

[41] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pages 2278–2324, 1998.

[42] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556, Computer Vision and Pattern Recognition, Computing Research Repository (CoRR)*, 2014.

[43] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 July 2015. PMLR. URL http://proceedings.mlr.press/v37/ioffe15.html.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[45] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *AAAI Conference on Artificial Intelligence*, 2017. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14806/14311.

[46] Forrest N. Iandola and Kurt Keutzer. Keynote: small neural nets are beautiful: enabling embedded systems with small deep-neural- network architectures. *2017*

International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS), pages 1–10, 2017.

[47] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter H. Jin, Sicheng Zhao, and Kurt Keutzer. SqueezeNext: Hardware-Aware Neural Network Design. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1719–171909, 2018.

[48] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1800–1807, 2016.

[49] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861, Computer Vision and Pattern Recognition, Computing Research Repository (CoRR)*, 2017.

[50] Barret Zoph, V. Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning Transferable Architectures for Scalable Image Recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2017.

[51] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Tien-Ju Yang, and Edward Choi. MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1586–1595, 2017.

[52] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V. Le. MnasNet: Platform-Aware Neural Architecture Search for Mobile. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2815–2823, 2018.

[53] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet:

Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10726–10734, 2018.

[54] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, Peter Vajda, Matthew Uyttendaele, and Niraj K. Jha. ChamNet: Towards Efficient Network Design Through Platform-Aware Model Adaptation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11390–11399, 2018.

[55] Yaqing Wang, Quanming Yao, James T. Kwok, and Lionel M. Ni. Generalizing from a Few Examples: A Survey on Few-Shot Learning. *ACM Computing Surveys*, 53(3), June 2020. ISSN 0360-0300. doi: 10.1145/3386252. URL https://doi.org/10.1145/3386252.

[56] Bharath Hariharan and Ross B. Girshick. Low-Shot Visual Recognition by Shrinking and Hallucinating Features. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 3037–3046, 2016.

[57] Sinno Jialin Pan and Qiang Yang. A Survey on Transfer Learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345–1359, 2010.

[58] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3320–3328. Curran Associates, Inc., 2014. URL http://papers.nips.cc/paper/5347-how-transferable-are-features-in-deep-neural-networks.pdf.

[59] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *ICML*, pages 1126–1135, 2017. URL http://proceedings.mlr.press/v70/finn17a.html.

[60] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning

algorithms. *arXiv preprint arXiv:1803.02999, Machine Learning, Computing Research Repository (CoRR)*, 2018.

[61] Zhenguo Li, Fengwei Zhou, Fei Chen, and Huang Li. Meta-SGD: Learning to Learn Quickly for Few Shot Learning. *arXiv preprint arXiv:1707.09835, Machine Learning, Computing Research Repository (CoRR)*, 2017.

[62] Gregory R. Koch. Siamese Neural Networks for One-Shot Image Recognition. *Deep Learning Workshop, International Conference on Machine Learning*, 2, 2015.

[63] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A Closer Look at Few-shot Classification. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HkxLXnAcFQ.

[64] Jasper R. R. Uijlings, Kea van de Sande, Theo Gevers, and Arnold W. M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104:154–171, 2013.

[65] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2013.

[66] Ross B. Girshick. Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015.

[67] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-NMS – Improving Object Detection With One Line of Code. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 5561–5569, Oct 2017.

[68] Min Zhai, Huaping Liu, and Fuchun Sun. Lifelong Learning for Scene Recognition in Remote Sensing Images. *IEEE Geoscience and Remote Sensing Letters*, 16: 1472–1476, 2019.

[69] Gong Cheng, Ceyuan Yang, Xiwen Yao, Lei Guo, and Junwei Han. When Deep Learning Meets Metric Learning: Remote Sensing Image Scene Classification via

Learning Discriminative CNNs. *IEEE Transactions on Geoscience and Remote Sensing*, 56:2811–2821, 2018.

[70] Li Yan, Ruixi Zhu, Nan Mo, and Yi Liu. Cross-Domain Distance Metric Learning Framework With Limited Target Samples for Scene Classification of Aerial Images. *IEEE Transactions on Geoscience and Remote Sensing*, 57:3840–3857, 2019.

[71] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[72] François Chollet et al. Keras. https://keras.io, 2015.

[73] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. URL https://www.tensorflow.org/.

[74] Ekaba Bisong. Google Colaboratory. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*, pages 59–64. Springer, 2019.

[75] David Kirk. NVIDIA CUDA Software and GPU Parallel Computing Architecture. In *Proceedings of the 6th International Symposium on Memory Management*, ISMM '07, page 103–104. Association for Computing Machinery, 2007. ISBN 9781595938930. URL https://doi.org/10.1145/1296907.1296909.

[76] Yinbo Chen. A re-implementation of "Prototypical Networks for Few-shot Learning", 2018. URL https://github.com/cyvius96/prototypical-network-pytorch.

[77] Flood Sung. PyTorch code for CVPR 2018 paper: Learning to Compare: Relation Network for Few-Shot Learning (Few-Shot Learning part), 2018. URL https://github.com/floodsung/LearningToCompare_FSL.

[78] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155, Computer Vision and Pattern Recognition, Computing Research Repository (CoRR)*, 2019.

[79] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980, Machine Learning, Computing Research Repository (CoRR)*, 2014.

[80] Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-Learning for Semi-Supervised Few-Shot Classification. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJcSzz-CZ.