An Exploration of Domain Generalisation Through Vision Benchmarking, Masking, and Pruning

Hamza Riaz, B.Sc., M.Sc.

Supervised by Prof. Alan F. Smeaton

Dublin City University

DCU

Ollscoil Chathair
Bhaile Átha Cliath
Dublin City University

A thesis presented for the degree of Doctor of Philosophy

School of Computing

Dublin City University

January 2025

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Hamza Riaz, ID No.: 20213348, Date: 06-01-2025

Acknowledgements

I would like to express my deepest gratitude to everyone who has supported and guided me throughout my PhD journey. First and foremost, I am profoundly grateful to my supervisor, Prof. Alan Smeaton, for his invaluable mentorship, encouragement, and constant support. His expertise and insightful feedback have been critical to shaping this thesis and my overall Ph.D. career, especially Alan's challenging questions and motivations for my research was phenomenal. I also extend my heartfelt thanks to Dr. Hyowon Lee, for acting as an independent supervisor with guidance and constructive discussions for the maintaining mental health and managing the PhD workload.

I deeply appreciate the Dublin City University, School of Computing for providing the facilities and an environment conducive to research. I also wish to acknowledge Science Foundation Ireland Centre for Research Training in Machine Learning (ML-Labs) for the support Grant number 18/CRT/6183, which made this research possible. In general, by organising research activities, like summer schools, seminars, and panel discussions by ML-labs & the team, was really helpful to boost our other important skills and knowledge base. Moreover, my thanks also go to Insight SFI Research Centre for Data Analytics for their generous collaboration.

Angela Lally's remarkable student management skills and guidance, especially in planning my travels for conferences, visas, and many more, was unwavering. I am deeply grateful for her thoughtful advice and for helping me even during non-office hours for urgent matters.

I am deeply thankful to my labmates—Bulat, Eric, Phuc, Anwesha, Sidra, Mahreen, Sanjay, and others—for creating an environment of sociability and intellectual exchange. The friendly discussions, shared ideas, lunch breaks, and moments of relaxation together significantly enhanced my enthusiasm for research. These interactions not only boosted my productivity but also provided me with the motivation and excitement to visit the lab more often, making my PhD journey both rewarding and enjoyable.

To my family, especially my late father, whose dream was to see me as a scientist, I owe my deepest gratitude. His love and hard work for our family inspired me to become the first Ph.D. holder in our family. I am also incredibly thankful to my

mother for her constant care, for managing my siblings back home, and for ensuring I could focus on my Ph.D. and adapt to life in Ireland without worrying about daily matters.

To my friends—Rahul, Raonak, Danilo, Andrius, Sarmad, and Ciara—you became my home and comfort zone in Ireland. Your support in planning trips and tours was invaluable for my mental health, providing me with much-needed relaxation during intense and challenging times. I will always be grateful for your companionship and for making my journey in Ireland more enjoyable and balanced.

In the end, I would like to express my profound gratitude to my wife, Fatima, who has been one of the most precious and supportive people in my life. You were my go-to person throughout this Ph.D. journey, and I owe you more than words can express. This journey would have been incomplete without your patience, consistent support, and encouragement. I know that balancing your own Ph.D. while taking care of me, advising me, and listening to my complaints was not easy. Yet, you did it all with grace and love. For all your countless efforts and support in every aspect of my life, I am deeply and eternally thankful to you.

Finally, it is difficult to name every individual who guided me or shared moments with me during this journey. However, I would like to express my heartfelt gratitude to all my friends, especially my Pakistani friends in Ireland and my international friends, for their support, companionship. throughout this time. Your presence made this journey more meaningful and memorable.

Contents

List of Tables

List of Figures

List of Abbreviations

AI Artificial Intelligence.

ANN Artificial Neural Network.

BEIT BERT Pre-Training of Image Transformers.

CLIP Contrastive Language-Image Pre-Training.

CLIPSeg Image Segmentation Using Text and Image Prompts.

CNNs Convolutional Neural Network.

CV Computer Vision.

DA Domain Adaptation.

DANN Domain Adversarial Neural Networks.

DeiT Data-Efficient Image Transformer.

DETR End-to-End Object Detection with Transformers.

DG Domain Generalisation.

DL Deep Learning.

DNN Deep Neural Network.

DRL Deep Reinforcement Learning.

ERM Empirical Risk Minimisation.

FPN Feature Pyramid Networks.

GANs Generative Adversarial Networks.

GPUs graphics processing units.

Grounding DINO Marrying DINO with Grounded Pre-Training for Open-Set Object Detection.

IID Independent and Identically Distribution.

Kosmos-2 Grounding Multimodal Large Language Models to the World.

LinkNet Exploiting Encoder Representations for Efficient Semantic Segmentation.

LLM Large Language Models.

LSTM Long Shot-Term Memory.

MACs Multiply-Accumulate Operations.

MHSA Multi-Head Self Attention Layer.

MIM Mask Image Modelling.

ML Machine Learning.

MLP Multi-Layer perceptron.

MSE Mean Squared Error.

NLP Natural Language Processing.

OOD Out-of-Distribution.

OWL-ViT Simple Open-Vocabulary Object Detection with Vision Transformers.

PSPNet Pyramid Scene Parsing Network.

PVT Pyramid Vision Transformer.

RCNN Region-based Convolutional Neural Network.

ReLU Rectified Linear Activation.

RL Reinforcement Learning.

SAM Segment Anything Model.

SegGPT Segmenting Everything In Context.

SegNet A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation.

SigLIP Sigmoid Loss for Language Image Pre-Training.

SL Supervised Learning.

SSD Single Shot MultiBox Detector.

SSL Semi-Supervised Learning.

UNet Convolutional Networks for Biomedical Image Segmentation.

USL Unsupervised Learning.

ViT Vision Transformer.

ViTMAE Masked Autoencoders Are Scalable Vision Learners.

VLM Vision-Language Model.

VNet Convolutional Neural Networks for Volumetric Data.

YOLO You Only Looks Once.

YOLOS You Only Look at One Sequence.

List of Publications

- **Riaz, Hamza**, and Alan F. Smeaton. "Domain Generalisation with Bidirectional Encoder Representations from Vision Transformers." Published in Irish Machine Vision and Image Processing Conference (IMVIP), Galway, August 2023.
- **Riaz, Hamza**, and Alan F. Smeaton. "Vision-Based Machine Learning Algorithms for Out-of-Distribution Generalisation." Springer Computing Conference, 22-23 June 2023, London, United Kingdom.
- Aparajita Dey-Plissonneau, Hyowon Lee, Michael Scriney, Alan F. Smeaton, Vincent Pradier, **Hamza Riaz**. "Facilitating reflection in teletandem through automatically generated conversation metrics and playback video". Published in CALL and Professionalisation: Short Papers from EUROCALL 2021, Elsevier, November 2021.
- Hyowon Lee, Mingming Liu, **Hamza Riaz**, Navaneethan Rajasekaren, Michael Scriney and Alan F. Smeaton. "Attention Based Video Summaries of Live Online Zoom Classes". Published in the 35th AAAI Conference on Artificial Intelligence, Workshop on Imagining Post-COVID Education with Artificial Intelligence, 9 February 2021.
- **Riaz, Hamza.**, Muhammad Uzair, Habib Ullah, Mohib Ullah. "Anomalous human action detection using a cascade of deep learning models". Published in 9th European Workshop on Visual Information Processing (EUVIP) 2021, IEEE, 2021/6/23).

An Exploration Study for Domain Generalisation Through Vision Benchmarking, Masking, and Pruning

Hamza Riaz

Abstract

There are many computer vision applications including object segmentation, classification, object detection, and reconstruction for which Machine Learning (ML) shows state-of-the-art performance. Nowadays, we can build ML tools for such applications with real-world accuracy. However, each tool works well within the domain in which it has been trained and developed. Often, when we train a model on a dataset in one specific domain and test on another unseen domain known as an Out-of-Distribution (OOD) dataset, models or ML tools show a decrease in performance.

Previously, in the lterature different solutions have been proposed to tackle with Domain Shifting problem which occurs during the inference of models, like adversarial training, feature alignment, learning distribution invariant features, meta learning and many more. Similarly, to understand the behaviour of ML models for serious challenges of Domain Generalisation (DG), Domain Adaptation (DA), and Domain Shifting, in summary, this thesis presents novel work at the intersection of vision-based technologies for domain-specific and domain-generalised methods, vision transformers for DG, synthetic data generation for OOD data with detailed analysis, and the effects of pruning on DG.

The underlying hypothesis is that to solve complex challenges like DG and DA, "it is possible to say that domain-generalised learning which can refer to dynamic learning could be better than domain-specific learning which can refer to static learning". It means that under domain shifting, dynamic learning can also have better, reliable, and faster adaptation than static learning.

Some initial experiments are conducted on two popular vision-based benchmarks, PACS and Office Home and we introduce an implementation pipeline for domain generalisation methods and conventional deep learning models. The results illustrates that domain generalised models have better accuracy than domain specific methods for these chosen benchmarks.

Since domain generalisation involves pooling knowledge from source domain(s) into a single model that can generalise to unseen target domain(s), recent trends motivate us to conduct an investigation into the factors which could affect the DG ability of a model and this inspired us to explore vision transformers.

Initially, we examined four vision transformer architectures namely ViT, LeViT, DeiT, and BEIT on out-of-distribution data. Due to advantages like self-attention, self-supervised fine-tuning, and mask image modeling, we use the BEIT architecture for further experiments on three benchmarks PACS, Home Office, and DomainNet.

In summary, under few conditions and selected measurement metrics, our experiments demonstrate that it is true to say domain generalised learning provides better solutions than domain specific learning.

Chapter 1

Introduction

The field of ML has created tremendous success stories by solving many complex problems like object identification and detection in images, segmentation and reconstruction in videos, Natural Language Processing (NLP), medical image analysis, robotics, and many more. The developments in ML algorithms and databases, and fusion of various fields of ML help researchers to achieve high-level goals. The majority of the current applications are built on traditional ML where we usually have millions of example datapoints with labels to train a model and this is called supervised learning Supervised Learning (SL). On the other hand, Unsupervised Learning (USL) trains algorithms using datasets without labels. In this method, the algorithms receive data without any labels and ML models discover patterns independently without external direction. Moreover, Semi-Supervised Learning (SSL) combines both labeled and unlabeled datasets to train algorithms. Typically, in semi-supervised learning, algorithms start with a small set of labeled data to guide their learning process, followed by larger amounts of unlabeled data to further refine the model. Since 2011, with the help of Deep Learning (DL) which is a sub-domain of ML which deals with various sensory data to automatically extract features, researchers are now using DL to handle various supervised and unsupervised learning problems [1]. Pure conventional DL based models can be considered as static systems, and still have many problems including overfitting, commonly needed huge datasets, and they do not have significant potential for generalisation and domain adaptation.

In computer science, researchers can use a technique called Reinforcement Learning (RL) which is a reward-based learning technique to learn from experiences and interactions with environments. The combination of DL and RL is known as Deep Reinforcement Learning (DRL) which utilises the power of large datasets and experiences. It performs feature extraction, classification, planning, reasoning and taking action with one algorithm. Like DL, it also has common challenges including insufficient sample sizes, overfitting and instability, requiring plenty of compute time

for the training of models or agents (which represent the learning algorithm), and lacking in generalisation and domain adaptation [2]. Term "agents" is widely used in RL to describe the learning algorithms. The motivation behind this fusion of RL and DL into DRL was to develop methods which have potential to learn quickly, just like humans do from few examples. In DRL, even transfer learning requires many hours of training as shown by researchers where they tried to use knowledge of an agent from one game into another. On the contrary, humans do the same job in few minutes [3].

Since all these learnings are inspired by humans, therefore it is insightful to briefly understand how the human brain does learning. Throughout our whole lives, we have various stages of learning, and as we know neurons and their connections are the main reasons behind all possible learnings. Simply put, biological neurons collect signals from one branch, perform transformations on those signals and convert them to electrical spikes that are passed to other branches. This is how we auto-extract features and store information in our brains. Nowadays researchers often use artificial neural networks to mimic the brain's functions. Moreover, the fields of psychology and neuroscience have been very helpful in explaining the process of human learning [4]. For example, reinforcement learning is inspired by Skinner's work on operational conditioning where humans learn by interacting with environments through trial and error, rewards and punishments [5]. Similarly, supervised and unsupervised learning are also derived from human cognitive level learnings where humans perform learning under someone's supervision or in a few situations, humans can learn without any supervision [4].

Moving towards the technical side again, current ML and DL based models have become state-of-the-art in domains like healthcare, finance, policy-making, new age search engines, AI chatbots like ChatGPT, Claude, Copilot etc, [6, 7, 8]. In the history of AI, this is the first time that researchers have come this close to general-purpose AI. For instance, now the new generation of AI models have the potential to pass the famous "Turing-test" or is closer to passing the test. Meanwhile, in healthcare, companies like Google, Meta, Apple, IBM and others are utilising their resources to help mankind by working on disease predictions, heart attack detection, diabetes, Alzheimer's, cancers, etc. For example, a model named Med-PaLM2 [9, 10], was developed by Google to handle questions related to the medical domain. One of the main reasons behind the exponential growth in the field of AI is the development of Large Language Models (LLM) that are based on transformers, as will be explained in Section 2.7.8.

1.1 Definition of AI Related Terms

Before delving into hypotheses and research questions, and to be able to grasp the problem that this thesis addresses, it is important for readers to familiarise ourselves with the fundamental definitions that will be used consistently throughout this thesis.

Artificial Intelligence. We can find many definitions of Artificial Intelligence (AI) in the literature and many famous researchers have defined AI in their own ways which depends on the perspective of the problem statement [11, 1]. Therefore, to the best of our understanding, AI can be defined as an automatic mechanism for problem-solving and decision-making to the simulation of human intelligence and beyond. AI is broad and continually evolving, encompassing various subfields including machine learning, natural language processing, computer vision, and more.

Machine Learning (ML). ML is a subfield of AI which focuses on data and algorithms for model development that enable machines to perform tasks without requiring human intelligence.

Deep Learning DL is a branch of ML that employs multi-layered neural networks, known as Deep Neural Network (DNN), to mimic the intricate decision-making abilities of the human brain. It is a method in machine learning that allows computers to learn from past data and comprehend the world by organising information into hierarchical concepts

Computer Vision. Computer Vision (CV) is a branch of AI focused on teaching computers and systems to derive useful insights from digital images, videos, and similar visual data. It utilises machine learning techniques and neural networks to educate these systems on understanding and analysing visual information.

Artificial Neural Networks. Artificial Neural Network (ANN), commonly known as neural networks, are computational systems designed to learn and process data inputs to produce desired outputs. Inspired by the structure of the human brain, they consist of interconnected nodes called neurons arranged into various layers including input, hidden, and output layers.

Domain Generalisation. DG is a concept in machine learning that deals with the challenge of learning a model from one or several training domains that can generalise to an unseen test domain. These training domains have different probability distributions. The goal of DG is to achieve OOD generalisation. This means that

the model is trained using only source data, and it is expected to perform well on data from a distribution that it has not seen during training [12].

Domain Adaptation. Domain adaptation is a sub-discipline of transfer learning that deals with scenarios in which a statistical or neural model trained on a source distribution is used in the context of a different (but related) target distribution. When this happens, we usually speak of a domain shifting.

Domain Shifting. Domain Shift, also known as distributional shift, is a change in the statistical distribution of data between different domains (like the training, validation, and test sets) for a model. In the context of machine learning, a domain shift occurs when the data distribution between an algorithm's training set and a dataset it encounters when deployed changes

Transfer Learning. During the training of a ML model, the process of using preserved knowledge from one task (source domain) to another different (but closely related) task (target domain) is called transfer learning. In practice, transfer learning starts with a pre-trained model and instead of training a model from scratch on the target domain, it is recommended to reuse the parameters of a pre-trained model.

Masking. Masking is the technique of deliberately concealing, changing or emphasising particular bits of data during analysis. Masking can be used with vision, sequences, or other types of input data, depending on the application or model.

Pruning. Pruning of a DNN is a process for optimising a trained neural network by reducing the number of parameters or computational resources it uses, without significantly impacting its predictive performance [13].

1.2 General Problems in AI

Regardless of the major developments in AI, there are many well-known general challenges which need to be mentioned here. For example, whenever we train any model X, to solve any problem Y by learning the mappings/representations across problem Y, the explainability of the model is always a big question. This arises when researchers ask themselves how the model reaches the decision Y, or how the AI model learns decision boundaries. Furthermore, according to one of the Godfathers of AI Geoffrey Hinton, most of the available large AI models are like a "black-box". Similarly, safety and privacy are also problems. In large language models for example, users can manipulate prompts in such a way that they can create

sensitive and dangerous content using modern AI-based ChatBots. In addition, as we know, all the algorithms are trained by using data which means that biases like inductive, sample, and human should be reduced or a model needs to be aware of them.

In general, traditional ML/DL networks are data-hungry and slow learner systems. For instance, to train a simple supervised learning image classifier with absolute success, ideally it needs millions of examples in its data space. In addition, ML/DL does not handle critical parts of an intelligent system like planning, reasoning, and making decisions, as it is useful only to solve domain-specific and comparatively simple downstream problems within that domain. For instance, if we train an model M on a data distribution X and we inference that model on a new unseen but related distribution Y, due to this phenomenon the accuracy of model M could be affected. Therefore, to fill this gap, we will explore various ML models for processing data outside of their target domain where we want to fit a new data distribution. For this to occur, trained algorithms need to perform knowledge transfer which also takes compute time.

1.3 Hypotheses and Research Questions

The overall research hypothesis in this thesis explores various areas of DL including conventional DL models, domain generalised methods, vision transformers, masking, pruning, and more.

Hypothesis ($H\phi$):
In understanding the role of neural networks in relatively emerging areas of machine learning to solve complex challenges like domain generalisation and adaptation, it is possible that domain-generalised learning which can refer to dynamic learning could be better than domain-specific learning which can refer to static learning.

To expand on this research hypothesis we have extended it into a number of sub-hypotheses, so $H\phi \rightarrow H_1, H_2$ and H_3 and also extended it to three specific research questions, namely RQ1, RQ2 and RQ3. The relationship between these is best described in the diagram in Figure 1.1.

We now introduce each of the sub-hypotheses and research questions in turn:

Sub-Hypothesis 1 (H1): Overall, researchers considered machine learning as an unpredictable field when it comes to real time applications because of the many sensitive factors like biases, explainability and generalisation, but domain shift can also be assumed to be one of major factors. During the domain shifting phenomenon,

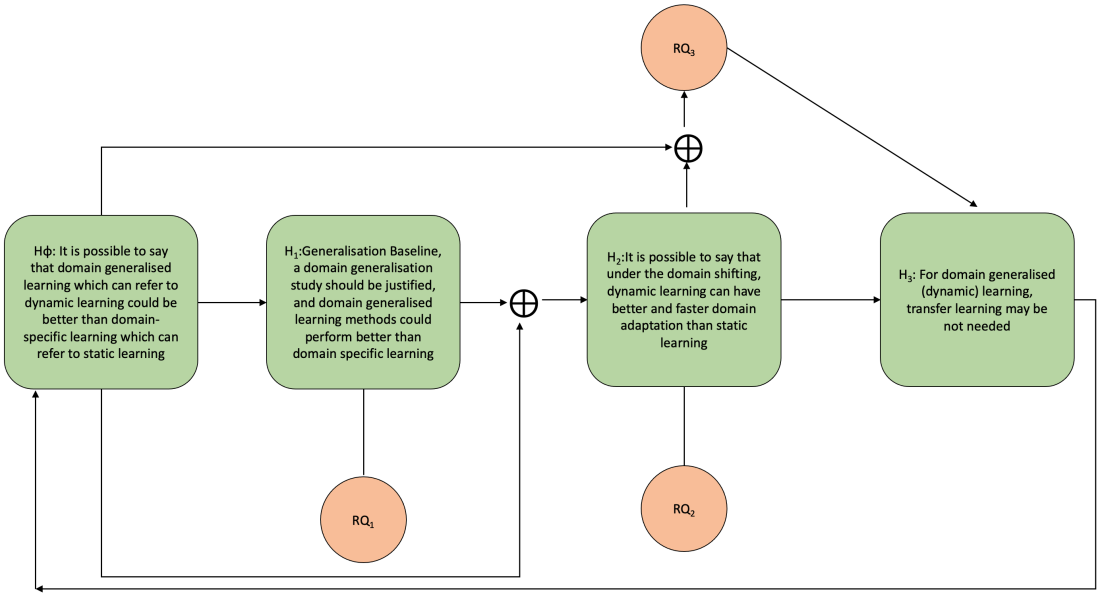


Figure 1.1: Relationship between research hypothesis and corresponding research questions.

applications based on areas like supervised, semi-supervised, unsupervised, and reinforcement learning show poor performance because of poor domain generalisation in the underlying machine learning. Therefore, a domain generalisation study should be justified where we will discover that domain generalised learning methods could perform better than domain-specific learning.

Research Question 1 (RQ1): In the literature, there is no precise answer to the question of how to measure domain generalisation therefore the first research question should be, what are the various ways of measurement for domain generalisation and what are the experiments that could be performed to show that dynamic learning can have better domain generalisation than static learning?

Sub-Hypothesis 2 (H2): Based on H1, it is possible to say that under domain shifting, dynamic learning can have better and faster domain adaptation than static learning.

Research Question 2 RQ2: For real world data, how much elasticity or variation in the datasets is needed in order for domain-specific and domain generalised methods to break? Moreover, as part of this it will be interesting to see the results of experiments to show that domain generalised learning has better domain adaptation than domain-specific.

Research Question 3 RQ3: This RQ relates to the accuracy of models and if H1 and H2 are true as explored by research questions RQ1 and RQ2 then it is important to see whether domain generalised learning can actually have better accuracy than domain-specific methods?

Sub-Hypothesis 3 (H3): Normally, when we perform domain shifting for domain-specific methods and to get reliable results, we have to perform transfer learning so that the model can adapt to a new domain. However for domain generalised (dynamic) learning, transfer learning may be not needed. Moreover, experiments could show how can we replace the very expensive transfer learning by some alternative.

1.3.1 Contributions (Cs): Research Questions and Hypotheses Workings in the Chapters

This section provides clear ideas about the research questions and hypotheses that we have answered in the related chapters.

C1: In this thesis, chapter 3 tackles the first RQ1, which is mostly related to exploring: how to measure DG, the different metrics, & experimental setups required. In addition, chapter 3 also provides information about the first hypothesis H1. H1 asos if a study is required which will initially prove that domain-generalised methods could have better DG than domain specific methods.

C2: In the second contribution, chapter 5 focuses on RQ3 which asks: will DG models have better accuracy, it also investigates the relationship between DG and the attention distances of models.

C3: Chapter 6 discusses RQ2 which is related to how much elasticity or variation can we add in order to break a DG model? Moreover, it also covers H3 according to which for domain shifting, whether transfer learning or fine tuning is required and under the right conditions transfer learning may not needed.

C4: At the end of the thesis, the research work in chapter 7 presents an emphasis on H2: Dynamic learning can have better and faster adaptation and talk about structural pruning of vision transformers and its effects on DG.

1.4 Thesis Structure

The remainder of the thesis is structured as follows:

Chapter 2: Background This chapter presents a brief overview of ML and its important sub-fields, such as DL, RL, and meta-learning, while also addressing common problems encountered by these sub fields of ML. It describes the settings in which classic ML models may be used effectively and contrasts them with circumstances in which DL models are more appropriate, emphasising the difference between both fields. The chapter also provides an overview of several different ML system designs, demonstrating how they have contributed to recent advances in the area.

In addition, a historical framework is presented which summarises the evolution of ML and its sub-areas according to applications. This chapter introduces the important literature for domain generalisation with respect to different fields of machine learning. Towards the end, the chapter introduces the idea of OOD data which is term referring to such data distributions that differ significantly from the training data, and discusses existing benchmarks used to evaluate model performance with OOD data.

Chapter 3: Vision-Based Machine Learning Algorithms for Out-of Distribution Generalisation This chapter compares vision-based systems geared for domain-specific tasks to those focused towards domain generalisation. We emphasise that typical deep learning approaches based on Convolutional Neural Network (CNNs) suffer when faced with domain shifts—where the data distribution varies between training and testing.

Chapter 4: Overview of Vision Transformers This chapter provides the theoretical and technical background which is required for investigations of such factors that could effect the performance of DG. This chapter also helps us to find such models which could have better generalisation. Moreover, it focuses on vision transformers and relatively modern models of ML.

Chapter 5: Domain Generalisation with Bidirectional Encoder Representations from Vision Transformer This chapter presents experiments and a feasibility study for vision transformers to solve DG. This work also provides reasons behind the good performance of the BEIT model and builds relationships between DG and global attention distances.

Chapter 6: Measuring the Resilience of Domain Generalisation in the Presence of Newly Generated Out-of-Distribution Noisy Images In this chapter, we enhance Segment Anything Model (SAM) and Marrying DINO with Grounded Pre-Training for Open-Set Object Detection (Grounding DINO) models in a network cascade to generate new OOD data distributions so that we can measure the resilience of the trained models.

Chapter 7: Pruning of Vision Transformers and its Effects on Domain Generalisation This chapter describes a framework and presents experiments for pruning of vision transformers in a grouped structure and implemented using different pruning criteria metrics. This research explores the effects of grouped structure pruning on DG when implemented for benchmarks like PACS, Office-Home, and DomainNet.

Chapter 8: Conclusions In the concluding chapter we summarise the research in all previous chapters by revisiting the research questions and hypotheses and how each chapter contributed in them. Moreover, it also mentions some broad limitations and future research direction for this thesis.

Chapter 2

Background

This chapter presents a brief background of machine learning (ML) and its sub-areas including deep learning (DL), reinforcement learning(RL), and meta-learning, and their common issues in general. The chapter also discusses where could we use traditional ML-based models for applications, and in which situations it is recommended to use DL-based models, and how all these types of models are different from each other. Then follows a brief explanation of various system architectures from the ML field and how they could help us to understand the overall progress that has been in the field in recent years. A history tree is also presented as part of the background for some of these mentioned areas of ML. This chapter also introduces the concept of domain generalisation in simple terms and the role of domain generalisation in various ML methods. This helps us to understand the importance of domain generalisation so that newly trained models are equipped to handle challenges like domain shifting more easily. Towards the end of this chapter, we also introduce and describe the term "out-of-distribution" referring to data which is outside the bounds of other data used to train a model and the available related benchmarks used to assess performance with out-of-distribution data. We also include a brief introduction to pruning in relation to domain generalisation.

2.1 Conventional Machine Learning

The journey of Machine Learning starts from about 1943 when Walter Pitts and Warren McCulloch presented the first mathematical model of a neural network [14] and 81 years later, after many ups and downs, ML is a large and active area of research. The development in this area is reflected by its success in recent advanced level applications like language translators, image captioning, video annotations, and data reconstruction. All these applications can now be done using machine learning and to a level of performance which equals or exceeds that of humans. With the development of stable databases and graphics processing units (GPUs), discoveries

like new mathematical functions, structures of neural network (NN) layers, and especially fusion of one type of ML with others are major factors behind these tremendous achievements. This section introduces the fundamentals of machine learning.

By definition, machine learning is the branch of artificial intelligence (AI) that deals with data and algorithms and gives the ability to machines to learn on their own from the data on which they have been trained. Conventionally, ML was considered as a complex and difficult task because, after capturing and pre-processing data which is used for training, many signal-processing filters like means, smoothing, Sobel [15] or other feature extractors like histogram of oriented gradients, wavelet transform, principal component analysis, etc. [16, 17, 18]

For example, if we consider images as a form of input data then simple filters can help to identify the edges around objects in the image or to cancel any visual noise caused by compression of the image into a format like JPEG. Similarly, feature extractors including for example Harris Corner Detection, Shi-Tomasi Corner Detector, Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Features from Accelerated Segment Test (FAST), Binary Robust Independent Elementary Features (BRIEF), Oriented FAST and Rotated BRIEF (ORB) can each provide high-level representations from raw data [19]. With changing data types, the types of filter that can be used and the selection criteria for extracting those features also change which makes the feature engineering job very challenging to handle. Even for slightly bigger datasets, the number of extracted features can become insanely high and cannot be managed manually [19].

Currently, within the area of data science, to solve comparatively smaller and basic problems, we still utilise similar methods but with a modern touch. Nowadays, the development of popular libraries including pandas, NumPy, sci-kit-learn, matplotlib, and many more have given a boost to the previous established machine learning systems [20, 21, 22, 23]. Moreover, some of the well-used ML algorithms including basic forms of Linear Regression, Logistic Regression, Decision Trees, SVM, Naive Bayes, KNN, K-Means, Random Forest, Dimensionality Reduction Algorithms, and Gradient Boosting algorithms overcome feature handling limitations. Yet still these machine learning algorithms can be used for simple problems and there is lot that one can learn about these methods in the book "Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies" [24].

2.2 Deep Learning

Neural networks and deep learning (DL) have been around for several decades and are a complex concent to understand. It is interesting to learn that deep learning started back in the 1940s and has had different names over the years, including cybernetics, connectionism, and what we now call Artificial Neural Networks (ANNs) [14]. DL is a sub-domain of ML and is basically inspired by the deep layer connections of Neural Networks in the human brain. It was first introduced in an attempt to solve difficult problems e.g classification, object detection and segmentation in images, scene understanding, and natural language processing (NLP). For vision-based applications, CNNs are the backbone of DL which were introduced in 1998 by Yann LeCun to do document recognition [25]. Even though CNNs started in the late 1990s where there was a lack of big databases and GPUs, researchers were unable to excel in the task of showing how powerful they could be. Then, after almost 14 years, a leap in state-of-the-art development was made by Krizhevsky, Sutskever and Hinton in 2012, in the image recognition field on the ImageNet data challenge [26]. Since then, DL has dominated the domain of machine learning and this is because it can give better performance than human level on tasks like image recognition.

Fundamentally, DL consists of stacked CNN layers which are designed to handle visual imagery information (e.g images, videos). CNNs perform basic mathematical convolution operations on images at each of the layers and can be configured to automatically extract sophisticated features. Recognition of handwritten digits was one of the first successful uses of CNNs and only needs three or four convolutional layers. With as much as 25 layers it is possible to design and build an accurate classifier for a face recognition system.

DL also has other functions or layers and we describe them here. Usually, the output of every convolutional layer passes through what are called activation layers. This is sometimes known as a transfer function and many **activation functions** are non-linear functions and the selection of which activation function to use has great impact on the performance of the final model. Moreover, if the output of the function has a limited range of values then it can be called a squashing function. The purpose of activation layers is to enhance the potential of the model to learn more complex non-linear representations. This is summarised in the quote "in order to get access to a much richer hypothesis space that would benefit from deep representations, you need a non-linearity, or activation function" [27]. Rectified Linear Activation (ReLU), Logistic (Sigmoid) and Hyperbolic Tangent (Tanh) are popular activation functions [28, 29].

Previously, in ML, the processing of images needed high computational power but CNNs reduce this cost. To further reduce computational power, mostly after

every convolutional layer and activation layer, the design of a CNN will indicate that it should be followed by a **pooling layer**. The purpose of pooling is to do downsampling of input feature maps (reducing the dimensions) from the previous convolutional layer without losing important information about feature maps. Max pooling and average pooling are popular examples of this. There are also **dense layers** known as fully connected layers which convert high dimensionality feature maps into arrays or vectors of the desired number of outputs. These fully connected layers become the input to the classification layer which converts these numbers into a probability distribution. Softmax and sigmoid are examples of commonly used functions.

The objective function is a function that we want to minimise or maximise during the training of a model. When we minimise it then we can call it a cost or **loss function**. Cross-entropy, and Mean Squared Error (MSE)are some examples of loss functions. Similarly, to start training, we also have to define the optimiser function. This boosts the training process by controlling the direction of the gradient steps during backpropagation and helps the model to converge. Gradient Descent and Stochastic Gradient Descent [30], Stochastic Gradient descent with momentum, Mini-Batch Gradient Descent, Adaptive Gradient Algorithm (Adagrad), Root Mean Square Propagation (RMSProp), Adaptive Delta (AdaDelta) and Adaptive Moment Estimation (Adam) are common functions used in the research community [24].

A conventional CNN is the combination of above-mentioned layers where each stack of layers extracts different levels of feature maps. Early layers extract low-level features like edges and boundaries in the case of images, higher layers contain high-level features like the shape and structure of objects. Regardless of the many successful applications of CNNs, they also have limitations of understanding the relationship between objects present in the image with respect to location [31]. It means that CNNs focus on local regions of interest in images and have less global understanding of input data. Furthermore, CNNs usually needed a large number of annotated images as datasets which is often very difficult to get in medical and in many other fields.

Depending upon the applications, the best configurations of CNNs will have structural differences accordingly. To solve classification problems, researchers explore AlexNet, VGGNets, InceptionNets, ResNets, DenseNet, MobileNet, and EfficientNet [26, 32, 33, 34, 35, 36, 37]. Additionally, in the literature vision transformer based methods proved to be very resourceful such as the following, ViT [38], BEIT [39], BiT [40], DeiT [41], SwinTransformer [42], ConvNeXT V2 [43], DINO V2 [44], Masked Autoencoders Are Scalable Vision Learners (ViTMAE) [45], Sigmoid Loss for Language Image Pre-Training (SigLIP) [46], GIP [47], Contrastive Language-Image Pre-Training (CLIP) [48], ViTMatte [49], and Pyramid Vision Transformer

(PVT) [50]. The workings of these methods is quite complicated but vision transformers are the backbone of most of these methods and these are explained later in this Chapter on the section an architectures, Section 2.7. Moreover, some of the methods like SigLIP [46], CLIP [48], and GIT [47] are combinations of both vision and language based models. Such learning algorithms utilise contrastive learning methods which is another well-known ML technique and it is also known as representation learning. Contrastive learning works by training a model to differentiate between similar and dissimilar pairs of data points by maximising the similarity within the same class and minimising it between different classes.

In the same way, to perform an object detection task, Region-based Convolutional Neural Network (RCNN), Single Shot MultiBox Detector (SSD),Feature Pyramid Networks (FPN) and You Only Looks Once (YOLO) families of models are popular choices [51, 52, 53, 54, 55, 56, 57]. These algorithms usually release various updated versions over time in order to overcome their own limitations. For instance, the first model of YOLO [55] came out in 2015 but after almost 9 years research on YOLO is still ongoing and 9 different versions have been developed by the ML community [58, 59]. The majority of these object detectors are based on CNNs but recently due to the success of vision transformers researchers have also explored them for object detection applications. Therefore, You Only Look at One Sequence (YOLOS) [60], ViTDet [61], End-to-End Object Detection with Transformers (DETR) [62], Deformable DETR [63], Grounding DINO [64], Grounding Multimodal Large Language Models to the World (Kosmos-2) [65], Simple Open-Vocabulary Object Detection with Vision Transformers (OWL-ViT) [66], and OWLV2 [67] are well-known transformer based models. Similar to classification tasks models like Kosmos-2, OWL-ViT, and OWLV2, they also use visual information with textual concepts to localise objects within an image with higher confidence.

For vision applications like segmentation, we have models like A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation (SegNet), Convolutional Networks for Biomedical Image Segmentation (UNet), Exploiting Encoder Representations for Efficient Semantic Segmentation (LinkNet), Pyramid Scene Parsing Network (PSPNet), Convolutional Neural Networks for Volumetric Data (VNet), deeplabs and Mask RCNN [68, 69, 70, 71, 72, 73, 74, 75]. Recently, ML models started evolving towards general purpose AI and especially for segmentation tasks we see that relatively generalised models have been developed. Therefore, models like SAM [76], Segmenting Everything In Context (SegGPT) [77], SegFormer [78], MaskFormer [79], Image Segmentation Using Text and Image Prompts (CLIPSeg) [80], GroupViT [81], and Oneformer [82] are considered most relevant models for the segmentation task in vision.

2.3 Reinforcement Learning

Reinforcement Learning (RL) is a development from supervised and unsupervised learning where instead of learning from examples, RL learns from experiences, which means we do not need labelled data for it to operate. RL is about mapping situations to action spaces and maximising a reward signal. The learners, previously introduced using the term agents, are not guided about what actions to take and during training, by exploring the world, they learn what type of actions yield maximum reward in an action space. In short, features like trial-and-error search and delayed reward make RL a distinguishing learning method. The agent is the model that tries to maximise feedback via the reward function.

Before going into other details about RL, it is important to know the basic elements of how it operates. Other than the agents and the environment, there are four major components of RL namely a policy, a reward signal, a value function, and a model of the environment. **Policy** describes the behaviour of agents and it is core of RL. Roughly speaking, policy alone is sufficient to find behaviour. A **reward** signal or function describes the goal of the RL problem. The agent's objective is to maximise this reward function in the long run. However a reward signal identifies good options for the agent in an immediate short-term manner, but a **value function** highlights good options in the long run. **A model of the environment** is a system that mimics the behaviour of the environment.

Standalone RL is a common field used to tackle problems related to control theory. However, when we combine it with DL then many applications which perform at a level greater than a human, become possible. DRL, which is deep reenforcement learning, has great applications in robotics, NLP, and computer systems [2]. In the Atari game challenge, Watson's wagering strategy, and the Go and chess-playing programs especially AlphaGo and AlphaGo Zero, defeated human masters after only few days of doing learning from experience [83, 84, 85].

DRL has three types: model-free RL, model-based RL and advanced RL. With the help of a large number of samples, **model-free RL** estimates three components of an agent, the states, a value function, and a reward function and it optimises the action policy. Furthermore, algorithms for model-free RL include DDQN, DDPG, SAC, NAF, Rainbow, Ape-X, and TD3 are off-policy. This means that these methods are independent of the agent's actions and learn the optimal policy regardless of the agent motivation [83, 86, 87, 88, 89, 90, 91, 92]. Similarly, TRPO, PPO, ACER, and A3C [93, 94, 95, 96] are on-policy methods, which try to improve the policy by interaction with environments. On-policy reinforcement learning is a good choice when we intend to optimise the parameters of an agent by doing much exploration. Conversely, in off-policy, the agent does not explore much. Each of these

methods produced state-of-the-art results in application domains like video games, maze runners, locomotion and hierarchical tasks.

The second type of DRL called **model-based RL** learns the transition dynamics and predicts the next state s(t+1) in the reaction of an action a(t) based on its current state s(t). In a comparison between model-free RL with model-based RL, model-free RL learns policy in a data-efficient way and continuous interaction with the environment is also not needed. GPS, PETS, I2As, PILCO, E2C [97, 98, 99, 100, 101, 102, 103] are common methods for model-based RL. Moreover, large language models also have affected the RL field and recently tremendous progress has been seen in this regard [104].

Insufficient samples as well as inefficiency in using those samples, overfitting and instability, exploitation and exploration trade-offs, training or transfer learning timing, and a demanding reward function are all common challenges for RL. Therefore, to work on these drawbacks, the third type of DRL known as **advanced RL** is extremely helpful and is actually a combination of different machine learning techniques like transfer RL, inverse RL, meta RL. Moreover, Bootstrapped DQN, VIME, MMP, NIRL, MaxEnt IRL, GAIL, AIRL, SNAIL, MAML, MAESN, and CAVIA are also important methods in this area of RL [2].

2.4 Meta-Learning

Meta-learning is a comparatively recent branch of ML which started in about the year 2016. In comparison to traditional ML that solved tasks like classification and prediction from scratch using training data and fixed learning algorithms, meta-learning means **learning to learn**. Its purpose is to improve the learning algorithm itself by using the experience of multiple learning episodes. Meta-learning deals more efficiently and effectively with some of the problems of conventional deep learning like data and computation, and generalisation hence the interest in the topic. Even though deep learning has had great successes in many applications, it has clear limitations in applications where data is intrinsically rare or expensive [105] to collect, and the large amount of compute resources usually required by deep learning are unavailable [106, 107].

Another factor behind the interest in research on meta-learning is because it is relatively closer in concept to human and animal learning. For example, it has been used in strategies where algorithms perform lifetime and evolutionary learning [108], just like humans and animals do. In summary, meta-learning is an alternative paradigm of ML modelling in which algorithms learn from experience over multiple learning episodes rather than just learning to solve one task, and they do this by encompassing and learning over the distribution of related tasks.

Before delving further into meta-learning, it is important to know about the process of how meta-learning based methods work. Meta-learning has two main loops of training known as the outer (or meta loop) learning algorithm and the inner (or base loop) algorithm. The base learning algorithm behaves like a conventional feed-forward ML prediction model and solves classification-like tasks. Meta-learning algorithms update their base learning algorithms in such sophisticated ways so as to learn the model with improvements in the outer objectives or goals. For example, these goals can be the generalisation of an overall model or the learning speed of the inner algorithm.

Generally, meta-learning can be used to solve supervised, unsupervised learning and reinforcement learning (RL) related problems. Single-shot and few shot learnings are examples of meta-learning in supervised learning. MAML, MAESN, RL2 and CAVIA [109, 110, 111] are examples of meta-learning implementations in RL domains, known as meta-RL [2]. Meta-RL, also contains two optimisation loops. The outer loop samples a new environment with respect to each iteration and updates parameters in such ways which determine the behaviour of agents. Meanwhile, in the inner loop, the agent interacts with each respective sampled environment and optimises its parameters for the maximal reward. We will return to cover more details on meta learning later in this thesis.

Even though, RL and Meta learning-based methods are not within the scope of this thesis we feel it is important to mention about the evolution and history of the ML field in general. This is a indication of continuous progression in various branches of the ML field. Therefore, Figure 2.1 tries to give a basic visual understanding of the evolution tree across various fields of ML.

2.5 History/Journey of Deep Learning

The journey of the development of Deep Learning started in the 1943 with the modeling of neural networks and after that psychologists and behaviourists started to study and define the cognitive processes of human learning and tried to convert those ideas into reality by building the hardware of artificial neurons [4, 14]. For the past 70 years, ML has had various phases of development like solving the XOR problem, gradient descent and backpropagation, long short term memory (LSTM), CNNs, GPUs, and vision transformers. However, a major boost in progress occurred after the commercial availability of GPUs and large databases of data. Therefore, to try to summarise these developments in this section we present a summary diagram of the timeline of the development of the overall ML field, especially since the development of CNNs, GPUs and databases.

Figure 2.1 briefly summarises the history and development of machine learning

specifically in the domains of DL, DRL, and Meta-learning, including Deep Meta RL. Depending on applications like classification, object detection, segmentation, reconstruction, and NLP, all of these application areas of ML have many algorithms which support the purpose of the application. This includes references to the most popular implementations and algorithms, selected for inclusion based on the number of citations from Google Scholar as of May 2024 which are indicated in parentheses in the diagram.

Meanwhile, as the main area of interest in this thesis is DL, therefore, we highlight its most recent references. But for the topics of RL and meta-learning, more references could be added in recent years but they are not the main scope of the thesis so we neglect them here.

2.6 Common Challenges in Machine Learning

Having introduced the main types of machine learning covering conventional machine learning, deep learning, reinforcement learning and meta-learning, we now briefly review the challenges that are common across these types of ML. All the given challenges are important to solve for the research community. Recently, explainable AI has become one of the most crucial challenge in the ML field because of the development of large language models [6, 7, 8]. However, we believe that domain generalisation is still under explored in DL for large models like vision transformers.

Overfitting and Instability in ML: Overfitting is a very common problem in ML and is a situation in which a statistical model only performs well for its training data but it cannot perform accurately for unseen or testing data. When overfitting happens it also means that the model is learning random details and noise from the training data which impacts the performance of the model negatively on new data. Moreover, it also means that after overfitting a model is reacting according to noise and random fluctuations in training data instead of picking useful features from the data and this issue in modelling leads towards poor generalisation of the resulting model.

Overfitting is also related to the bias and variance of the model. Bias is the difference between the actual value and the predicted value by the model. High bias means the model is focusing on training data which can oversimplify the model, known as underfitting. Conversely, variance measures the variability of the model and high variance relate to overfitting or poor generalisation.

To overcome this issue, several methods have been proposed including k-fold cross-validation, data augmentation, use of larger datasets, regularisation methods like activity regularisation, weight decay, early stopping, dropout, adding extra

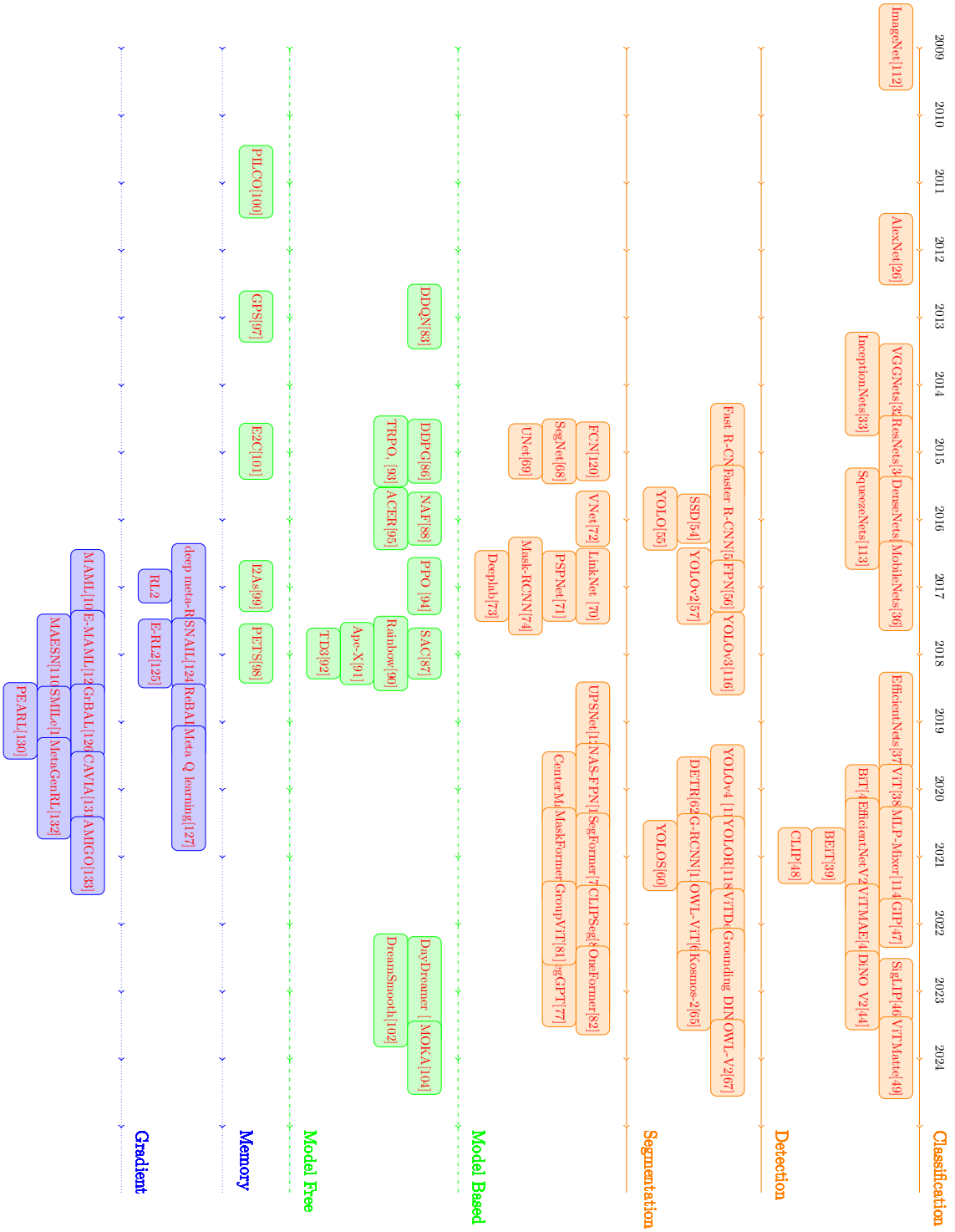


Figure 2.1: Timeline of the history of ML with a focus on deep and meta learning. Each colour represents different types of ML – orange for DL, green for RL and blue for meta-learning. Each branch indicates progress with respect to applications in that field. Dashed lines show there are more application branches including NLP, reconstruction and GANs which are not included because they are not directly related to deep meta RL

noise, and reducing overfitting by constraining the model complexity leading to robust models. Chapters 1 & 8 of Kelleher et al.'s book provide further detailed information about the overfitting issue [24]. and when these methods as described reduce overfitting, they also reduce the generalisation error, the ability of a model to generalise to new unseen data.

Sample Inefficiency in ML Algorithms: When a machine learning algorithm can get most of the information it needs to build a model from every sample in the given distribution of its training data then we can say that that algorithm is sample efficient. For instance, as humans, if we try to learn the PONG game it would take us only a few seconds and a few samples, which makes us sample efficient. On the other hand, modern RL algorithms have to look into one hundred thousand times more data than we do for this task, which makes them sample inefficient. Therefore, if an algorithm fails to learn any useful representation from many samples of experience and improves extremely slowly then it means that the algorithm has poor sample efficiency.

The issue of sample inefficiency is also related to another challenge of "time consumption for transfer learning" which means that in supervised ML and DRL each new task from outside of the data distribution model needs to perform transfer learning that could take a lot of computation time to achieve. Hence, in modern times, meta-learning is known to provide a better remedy to tackle these issues in ML algorithms [134].

Training Biases in ML: Training biases or bias in ML in general, is the phenomenon of observing results that are systematically prejudiced due to incorrect assumptions about the underlying data used to train a model although we can have bias in training data or in testing data, or in both. In many areas of ML, datasets can be unconsciously biased and in this situation a resulting model is going to learn these biases. For example, for NLP-based applications, to add diversity and avoid possible training biases, it is very sensitive to keep datasets balanced as imbalances in gender equality, in national equality, or age or colour or race equality can easily creep in. Sample bias, anchoring bias, availability bias, confirmation bias, stability bias are all common biases to check for during the development of a machine learning model, as described in chapter 1 of Kelleher et al.'s book [24].

Domain Generalisation and Domain Adaptation: When algorithms process samples of data which are taken from different distributions, ML algorithms suffer from a common problem called the domain shift. This further introduces two major issues including domain generalisation (DG) and domain adaption (DA). DG deals

with the comparatively hard situations where several different but also related domains are given, and the purpose of a machine learning algorithm is to learn a model which could be generalised on unseen test data taken from a different domain. The main goal of DG is learning a representation of its training data that could have the potential to perform well in unseen domains by leveraging more source domains during training.

The idea behind DA is different to the idea behind DG in that DA aims to maximise the performance of algorithms or models on a given target domain but using existing training source domains. The main difference between DA and DG is that DA has access to the target domain data which implies that it can see the data while DG cannot see anything from the target domain during training. This makes DG more challenging than DA but more realistic and more favourable in practical applications.

There are many generalisation-related research topics some with solutions such as DA, meta-learning, transfer learning, covariate shift, lifelong learning, and zero-shot learning [12]. In Section 2.8 of this chapter we further discuss some of the details on machine learning in domain generalisation and domain adaptation.

Explainability: Often, AI algorithms are known as black boxes when the make decisions or classifications which are not explained. Explainability, which can be interchanged with another term "interpretability", is the concept that reveals to what degree or level we can explain or justify the output of a machine learning model for some task in a way that makes sense to humans. Traditional ML algorithms are more suitable for explainability of their outputs because many of them generate rules which can be expected, or their outputs can be visualised, like with Support Vector Machines such as shown in [135]. However those traditional ML algorithms have lower performance in terms of the quality of their output compared to deep learning-based methods. DL-based algorithms on the other hand have better performance but it is extremely hard to make them explainable.

In summary, explainability in ML means to what extent we can explain to a user what is happening in our model from input to output so the user can be more informed, and comfortable with the output. Explainability enhances a model's transparency, ability to question, and ease of understanding and addresses the black box problem. There are two common methods to solve this paradigm, globally and locally. The first one explains the overall behaviour of the model and gives a big picture to a user whereas local methods try to explain features samples at each individual level.

Linear models, decision tree algorithms, and generalised additive models (GAM) are easier to explain than others however for complex models there are two main

approaches that can be used for explanations, the Model-Agnostic Approach and the Model-Specific Approach. Many techniques have been proposed to explain a model's behaviour and output and some of the most popular of them are Partial Dependence Plots (PDP), Individual Condition Expectations plots (ICE), Leave One Column Out (LOCO), Accumulated Local Effects (ALE), Local Interpretable Model-Agnostic Explanations (LIME), Anchors, SHapley Additive exPlanations (SHAP), Deep Learning Important Features (DeepLIFT), Layer-wise relevance propagation (LRP), Contrastive Explanations Method (CEM), and ProfWeight. One excellent recent reference which describes all of these is [136].

2.7 Neural Network Architectures

In this section, we introduce the essential building blocks of the ML field and this specifically refers to designs or structures of models that are used in ML. The ML architectures usually define how data is processed, transformed, and represented in an ML system. To create either simple or complex ML models, researchers have to use the same building blocks from the ML field and it is only the structural layers and their order in the models that lead to different types of ML networks/models. Similarly, the structure of such layers also defines the application for which we want to build an ML system.

Architectures play a crucial role in determining the performance, efficiency, and capabilities of machine learning systems. Moreover, most of the structural theories are based on human brain learning mechanisms like dropout and perceptron. These architectures have fundamental mathematical functions and we call these layers and with the help of these layers we can construct simpler or more complex neural network architectures. In general, we stack layers in sequential or in parallel, or even both ways, to form a ML/DL model. Therefore, this section introduces the feed-forward layer, the multi-layer perceptron, convolutional neural networks (CNN), recurrent neural networks (RNN), long shot-term memory (LSTM), adversarial layers, attention layers, transformers and vision transformers.

2.7.1 Feed-Forward Layer:

In ML, a feed-forward layer is one of the basic building blocks in neural network architectures. It is like a Lego piece that we use to create more complex structures. What it does is described below:

- **Neurons:** Imagine a feed-forward layer as a group of tiny decision-making units called neurons. These neurons work together in a single layer.

- **Connections:** Each neuron is connected to every neuron in the previous and next layer. Think of these connections as communication lines.

- **Flow:** When we give the network input data, it flows through these neurons in a forward direction. There are no feedback loops which means iinformation flows like a one-way traffic of information.

- **Computation:** Each neuron performs a small number of mathematical functions. It adds up the input it receives (weighted by certain values) and applies an activation function to the result. This function decides whether the neuron should "fire" (activate) or stay quiet. This activation function introduces non-linearities into the network and Sigmoid Function (Logistic Activation), Hyperbolic Tangent Function (Tanh), Rectified Linear Unit (ReLU), Leaky ReLU, and Exponential Linear Unit (ELU) are commonly used for this purpose.

- **Learning:** During training, the network adjusts the weights of the connections. It tries to minimise the difference between its predictions and the actual results. This helps the network to learn patterns and relationships in the training data.

- **Tasks:** With these layers, the network can perform tasks like classification (sorting input data into categories), regression (predicting values), and even feature extraction (finding important parts of the data).

  To sum up, a feed-forward layer is like a team of little decision-makers working together to understand and process information. The feed-forward layer is also known as a densely connected layer or a fully connected layer and various deep learning networks have and use this essential component.

2.7.2 Multi-layer Perceptron:

Back in 1958, Rosenblatt's work entitled "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain" is considered one of the earliest and simplest artificial neural network (ANN) algorithms [137]. After this publication, techniques based on the perceptron became very popular in the neural network field.

Therefore, a Multi-Layer perceptron (MLP) is a type of feed-forward artificial neural network that consists of multiple layers of interconnected neurons. It is one of the simplest and most common types of neural network architectures used in machine learning, enabling it to learn hierarchical and more complex representations in the data. A simple MLP consists of input layer, hidden layer, and output layer.

Key features of a Multi-layer Perceptron include:

- **Input Layer:** The first layer of the MLP receives input data. Each neuron in this layer represents a feature or input variable.

- **Hidden Layers:** Intermediate layers are located between the input and output layers. Each hidden layer consists of multiple neurons that apply non-linear transformations to the input data. The number of hidden layers and the number of neurons in each hidden layer are configurable parameters as part of the system architecture.

- **Output Layer:** The final layer of the MLP produces the network's output. The number of neurons in the output layer depends on the type of task the MLP is designed for. For example, in binary classification tasks, there may be just one neuron representing the output class probability, while in multi-class classification tasks, there may be multiple neurons representing the probabilities of different classes.

- **Activation Functions:** Non-linear functions applied to the output of each neuron in the hidden layers are included to introduce non-linearity into the model. Common activation functions include ReLU, sigmoid, tanh, and softmax.

- **Weights and Biases:** Thee are parameters that determine the strength of connections between neurons in adjacent layers and the offset or baseline activation level of each neuron, respectively. These parameters are learned during the training process through techniques like backpropagation and gradient descent.

- **Forward and Backward Propagation:** During forward propagation, data moves through the network, and each layer applies transformations and activation functions to create increasingly abstract and informative representations. To train the MLP, backpropagation and optimisation techniques like gradient descent are used. Backpropagation calculates the gradients of the loss function concerning the model's parameters, starting from the output layers and moving backwards to the input layer. The combination of forward and backward information propagation completes the cycle of the training process of a network.

  MLPs are known for their ability to approximate complex non-linear functions and are widely used in various machine learning tasks such as classification, regression, and pattern recognition. However, they may suffer from overfitting, especially when dealing with high-dimensional data or limited numbers

of training examples, and may require careful regularisation techniques to generalise well to unseen data.

2.7.3 Convolutional Neural Networks:

Even though feed-forward layers and MLPs are building blocks of ANNs because of their limitations for visual data processing like being computationally very expensive and inefficient for learning visual information, we have another important component that made the evolution of deep learning possibly faster and that is entitled Convolutional Neural Networks. The first practical example of the usage of the convolutional layer in the ML field was in 1998 when Yann LeCun designed one of first and simplest ML networks for the classification of handwritten words [25]. Then after 14 years the CNN was re-introduced by Geoffrey Hinton's team [26] for the ImageNet challenge. Because of the impact of CNNs on the development of AI, it is highly encouraged to understand the various components of CNNs.

Convolutional Neural Networks (CNNs) are a type of deep learning model specifically designed for processing and analysing visual data, such as images and videos. They are widely used in computer vision tasks, including image classification, object detection, semantic segmentation, and more. Additionally, Figure 2.2 provides an overview of key components so that we can initially understand the basic differences between the given types of architectures. The following are the most important aspects of convolutional neural networks [138]:

- **Convolutional Layers:** CNNs consist of multiple layers, with convolutional layers being the key building blocks. Each convolutional layer applies a set of learnable filters (also called kernels) to the input image. These filters slide across the input image, performing convolution operations to extract features such as edges, textures, and shapes. The convolution layer has different steps to calculate its output including feature extraction, convolution operation, learnable parameters, and Strides and Padding.

  - **Feature Extraction:** The primary function of a convolutional layer is to extract related features from input visual data by applying a set of learnable filters (also known as kernels) to the input image.
  - **Convolution Operation:** The convolutional layer performs a mathematical operation known as convolution between the input image and the filters. This operation involves sliding the filters across the input image, computing the element-wise multiplication which is a simple dot product between the filter and the corresponding region of the input image, and then summing the results to produce a single output value. This process

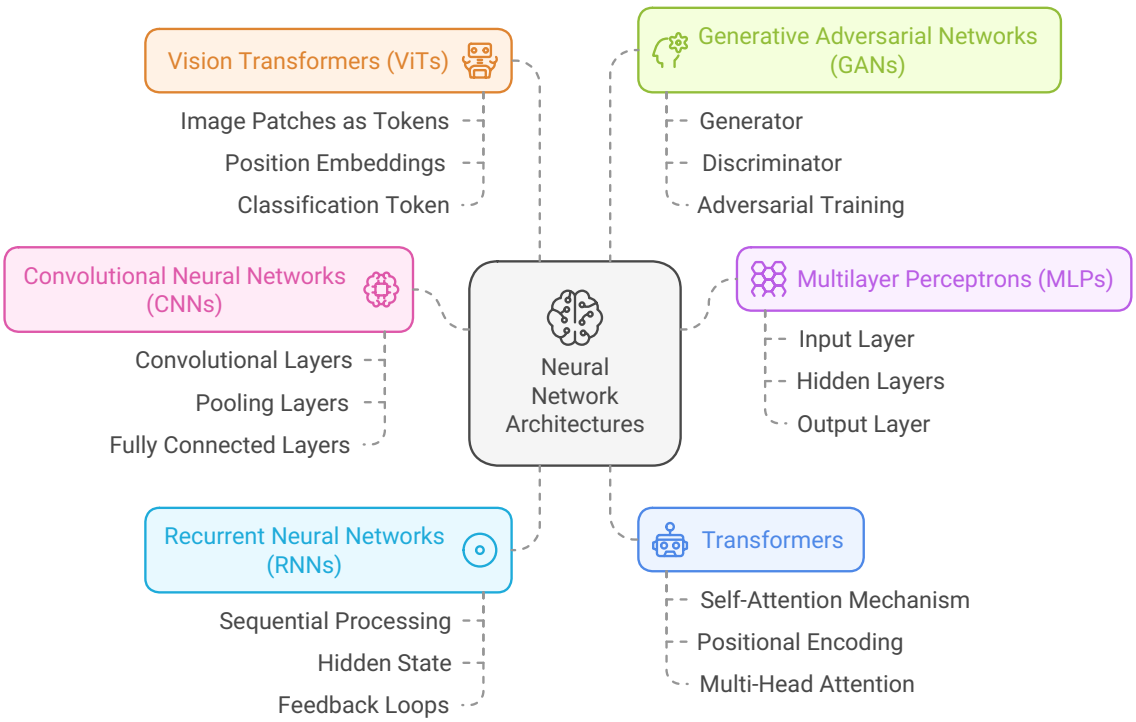


Figure 2.2: Represents the overview of the key components of different types of neural networks which we have discussed in the background chapter.

is repeated for every position of the filter across the input image, resulting in a new 2D feature map.

Mathematically, given an input volume X (visual data), a convolution layer applies a filter (kernel) K of size $f \times f$ with stride s and padding p, producing an output volume Y. The operation can be defined as:

$$Y(i,j) = \sum_{m=0}^{f-1} \sum_{n=0}^{f-1} K(m,n) \cdot X(i \cdot s + m - p, j \cdot s + n - p)$$

where:

* i, j are the spatial indices for the output volume,
* m, n are the spatial indices for the filter,
* s is the stride of the convolution,
* p is the amount of zero-padding added to the input volume.

The filter K slides over the input volume X computing the dot product at every position and producing the output volume Y. The indices i and j iterate over the output's spatial dimensions, while m and n iterate over the filter's dimensions. The stride s determines the step size of the filter as it moves across the input, and the padding p allows the filter to cover the border areas of the input to avoid dimension errors.

  - **Learnable Parameters:** The filters used in the convolutional layer are learnable parameters, meaning that their values are adjusted during the training process to minimise the difference between the predicted output of the network and the true labels in the training data. Through training, the convolutional layer learns various patterns and features in the input data.
  - **Strides and Padding:** Convolutional layers can also incorporate parameters such as stride and padding, which control how the filters are applied to the input image. Stride determines the step size of the filter's movement across the input image, while padding adds extra border pixels around the input image to ensure that the output feature map has the desired spatial dimensions.

- **Pooling Layers:** CNNs often include pooling layers to downsample the spatial dimensions of the feature maps produced by the convolutional layers. Pooling helps to reduce the computational complexity of the network and makes the learned features more invariant to small spatial variations in the input data. L1/L2-normalisation, and average pooling are considered common operations for the pooling layer.

- **Activation Functions:** Non-linear activation functions such as ReLU (Rectified Linear Unit), are applied after convolutional and pooling layers to introduce non-linearity into the network and enable it to learn complex patterns and relationships in the data.

- **Fully Connected Layers:** Usually, at the end of the CNN architecture, one or more fully connected layers are typically used to perform classification or regression tasks based on the extracted features. These fully connected layers allow the network to learn high-level representations of the input data.

- **Training and Transfer Learning:** CNNs are trained using labeled data and due to their ability to learn rich hierarchical representations of visual data, pre-trained CNN models (trained on large-scale datasets like ImageNet) are often used as feature extractors in transfer learning tasks. Fine-tuning pre-trained CNNs on smaller, task-specific datasets can lead to improved performance with less data and computation.

- **Challenges:** CNNs confront a number of common issues, including their dependence on large labelled datasets, which leads to overfitting and poor performance with small or low-quality datasets. They are computationally demanding, necessitating large resources for training and inference, and lack trans-

parency, making them difficult to comprehend in critical situations. CNNs are especially vulnerable to noise, distortions, and adversarial attacks, and their fixed receptive fields make it difficult to capture global context. Additionally, they are less efficient for sequential or temporal data, need substantial hyper-parameter adjustment, and have difficulty adjusting to new domains without major retraining. Ethical considerations, like as bias distribution through training data, hinder their use in sensitive applications.

2.7.4 Recurrent Neural Networks:

A Recurrent Neural Network (RNN) is a form of artificial neural network designed to handle sequential or time series data. It was first introduced in 1987 for machine learning by Geoffrey Hinton and his research group [139]. These advanced algorithms are commonly employed for tasks related to sequential or temporal issues, such as language translation, natural language processing (NLP), speech recognition, and image captioning

Unlike feed-forward and convolutional neural networks (CNNs), RNNs possess a unique attribute called its memory. RNNs integrate information from previous inputs to influence the current input and output. While traditional neural networks assume that inputs and outputs are independent of each other. For example, consider the phrase "feeling under the weather". To comprehend this expression, the order of the words is significant. RNNs take into account the position of each word in the sequence to predict the next word.

The unique features of RNNs are described below:

- **Shared Parameters:** RNN layers share the same weight parameters within each layer, in contrast to feed-forward networks where weights differ across nodes.

- **Backpropagation Through Time (BPTT):** RNNs utilise BPTT to compute gradients specific to sequence data and adjust model parameters during training.

- **Challenges:** RNNs may encounter problems such as exploding gradients (when gradients become excessively large) and vanishing gradients (when gradients become exceedingly small), impacting learning stability.

In essence, RNNs enable the storage of information within the network, allowing previous experiences to influence future events. Their capacity to process arbitrary input sequences makes them valuable tools in a wide range of applications. Mathematcially, RNNs can be defined by two well-known equations, the first is called the

state update equation and the second is called the output equation, which are as follows:

1. **State Update Equation:** In this equation, the hidden state (h_t) at a specific time step (t) is updated utilising the prior hidden state (h_{t-1}) and the current input (x_t):
$$h_t = \sigma(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$

2. **Output Equation:**The output (y_t) at that same time step (t) is determined based on the current hidden state (h_t):
$$y_t = \sigma(W_{hy}h_t + b_y)$$

In the equations above, the symbol (σ) denotes the activation function, the matrix (W) signifies the weight matrices, and the vector (b) represents the bias vectors.

Recurrent Neural Networks (RNNs) come in various forms, each tailored to different types of sequence-based tasks which are summarised as follows:

1. **Vanilla RNNs:** These are the simplest form of RNNs, utilising a basic neural network architecture with feedback loops to process sequence data.

2. **Long Shot-Term Memory (LSTM) [140]:** LSTMs are specifically designed to address the vanishing gradient problem encountered by vanilla RNNs. They possess memory cells that can retain information for extended periods.

3. **Gated Recurrent Units (GRUs [141]):** Similar to LSTMs, GRUs are simpler in structure and have fewer parameters. They combine the forget and input gates into a single "update gate".

4. **Bidirectional RNNs [142]:** These networks consist of two layers that process data in both forward and backward directions simultaneously. This approach allows for additional context and improved performance on specific tasks.

5. **Echo State Networks (ESNs):** ESNs belong to a category known as reservoir computing. In ESNs, the recurrent layer (reservoir) retains fixed weights, with only the readout weights being trained.

6. **Attention-based RNNs:** These RNNs incorporate attention mechanisms that enable the network to selectively focus on different parts of the input sequence. This functionality proves particularly useful in tasks like machine translation.

7. **Sequence-to-Sequence (Seq2Seq) Models:** These RNNs are employed to map input sequences to output sequences. They find widespread use in applications such as machine translation and question-answering systems.

   Each type of RNN possesses its own strengths and is chosen based on the specific requirements of the particular task at hand.

2.7.5 Long Short Term Memory:

One of the first papers related to LSTM was published in 1997 [140], it explains that a fundamental challenge is present in the behaviour of gradients as they propagate through multiple stages in recurrent neural networks (RNNs). These gradients often suffer from either vanishing (frequently) or exploding (occasionally, but significantly detrimental to optimisation) issues. To solve this challenge, the most effective sequence models utilised in real-world applications are referred to as gated RNNs. This category encompasses advanced architectures such as long short-term memory (LSTM) networks and networks based on the gated recurrent unit (GRU). These models have proven to be highly successful in handling sequential data and are widely employed in various practical applications [143].

Long Short-Term Memory (LSTM) is widely used in the field of DL as a type of recurrent neural network (RNN) architecture. The extraordinary abilities to capture long-term dependencies make it special and successful for most of the sequential applications. Compared to traditional neural networks, LSTM implements feedback connections which enables them to handle entire sequences rather than one word individually. This unique property of LSTMs makes it highly effective in comprehending and foreseeing patterns in sequential data, such as time series, text, and speech.

To get an overview of LSTM, it is important to know about individual components and their workings as in the following:

- **Architecture:** An LSTM cell consists of several essential components including a cell, an input gate, an output gate, and a forget gate. The cell retains and remembers important values over arbitrary time intervals, while the gates regulate the flow of information into and out of the cell.

- **Functionality:**

  The **forget gate** determines which information should be discarded from the previous state.

  The **input gate** determines which new information should be stored in the current state.

The **output gate** controls which information should be output from the current state. Moreover, the features like being configured to selectively output relevant information makes the LSTM networks sustainable and useful towards long-term dependencies, which are crucial for making accurate predictions.

LSTMs find applications in various areas of the real world such as speech recognition, handwriting recognition, machine translation, healthcare, video games, and robot control.

To summarise, LSTM networks effectively address the vanishing gradient problem encountered by traditional RNNs, thus allowing them to handle long-term dependencies. Their ability to preserve and utilise information over time makes LSTMs a powerful tool in the fields of AI and DL.

Mathematically, LSTM has series of equations which represent its overall functionality. An LSTM cell at time step t is defined by the following equations:

$$
\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && \text{(Forget Gate)} \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && \text{(Input Gate)} \\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && \text{(Cell Candidate)} \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t && \text{(Cell State Update)} \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && \text{(Output Gate)} \\
h_t &= o_t * \tanh(C_t) && \text{(Hidden State)}
\end{aligned}
$$

where (f_t) is the forget gate's activation vector, (i_t) is the input gate's activation vector, $(\tilde{C}_t)$ is the cell candidate vector, (C_t) is the cell state vector, (o_t) is the output gate's activation vector, (h_t) is the hidden state vector, (W) and (b) are the weights and biases for different parts of the LSTM cell, (σ) is the sigmoid function, $(*)$ denotes element-wise multiplication and $([h_{t-1}, x_t])$ denotes the concatenation of the previous hidden state and the current input.

We now explain these in more detail.

- **Forget Gate** (f_t)**:** This gate decides which information should be discarded from the cell state. It looks at the previous hidden state (h_{t-1}) and the current input (x_t), and applies a weight matrix (W_f) and a bias (b_f). The sigmoid function (σ) maps the values between 0 and 1, with values closer to 1 indicating 'keep this' and values closer to 0 indicating 'forget this'.

- **Input Gate** (i_t)**:** Simultaneously, the input gate determines which new information will be stored in the cell state. It also uses the previous hidden state

and the current input, applying its own weights (W_i) and bias (b_i), and passes them through a sigmoid function.

- **Cell Candidate** ($\tilde{C}_t$) **:** This vector creates a candidate set of values that could be added to the cell state. It combines the previous hidden state and the current input, but this time it applies a tanh function, which outputs values between -1 and 1, to create a candidate vector.

- **Cell State Update** (C_t) **:** The cell state is the memory of the LSTM. The forget gate's output (f_t), multiplies the previous cell state (C_{t-1}) to forget things it deems unnecessary. Then, it adds the product of the input gate's output (i_t) and the cell candidate ($\tilde{C}_t$), which adds new memory content.

- **Output Gate** (o_t)**:** The output gate decides what the next hidden state (h_t) should be. Like the other gates, it looks at the previous hidden state and the current input, and applies its weights (W_o) and bias (b_o). The sigmoid function determines which parts of the cell state will be output.

- **Hidden State** (h_t) **:** Finally, the hidden state for the current time step is calculated. It's the product of the output gate's result and the tanh of the cell state, which means it only outputs the parts the output gate decided on.

2.7.6 Generative Adversarial Networks:

We now provide information about generative adversarial networks and why we use them. Generative modeling is an unsupervised learning problem, although a clever property of the GAN architecture is that the training of the generative model is framed as a supervised learning problem. The first paper which proposed Generative Adversarial Networks (GANs) was published by Goodfellow in 2014 [144].

According to the original paper, generative adversarial networks are based on game theory where two separate neural networks compete with each other. One network is named as the **generator** and its role is to create new samples of data like real images and the other network is the **discriminator** which tells us whether the generated data is real or fake. During the training process, the generator tries to fool the discriminator by producing fake but as authentic as possible samples, so we can say that generator acts as a artist and the discriminator act as a critic of that artist.

The discriminator network has a structure like a model whose role is to perform classification tasks so it acts as a supervised learning method. On the other hand, the generator model is based on purely unsupervised learning where the structure of the model is like an autoencoder and decoder type network.

Mathematically, let us consider real data as x, a latent vector (input to the generator) as z, fake data generated by the generator as $G(z)$, the evaluation of discriminator for real data as $D(x)$ and the evaluation of discriminator for fake data as $D(G(z))$. Then we can write two loss functions one for each network using these terms. The loss function for the discriminator can be defined as:

$L_D = \text{Error}(D(x), 1) + \text{Error}(D(G(z)), 0)$

The goal for the discriminator is to correctly label real data as true (1) and fake data as false (0). Similarly, the loss function for the generator can be defined as:

$L_G = \text{Error}(D(G(z)), 1)$

The generator's role is to minimise the difference between the label for true data (1) and the evaluation of the discriminator for the generated fake data. Meanwhile, the training of the original GAN could be challenging as the convergence of both models is very hard to accomplish because of non-convex optimisation where the objective function has multiple local minima and saddle points. This makes it difficult to find the global optimum and leads to instability during training.

Therefore, to assist the training process, in the literature there are different types of GANs that have been proposed which could be conditional or unconditional GANs. A few examples of both types are c-conditional GANs [145], ProGAN [146], StyleGAN [147], SRGAN [148], Pix2PixGAN [149], and more. These GAN-related models are out of scope for our thesis but more details about them can be found in these two books [11, 150].

2.7.7 Attention Layers:

Recently, the concept of attention has gained popularity due to its success in deep learning for applications like computer vision and natural language processing. For example, generative AI models which are known as large language models (LLMs), are extremely popular nowadays with names like GPT, COPILOT, Claude, perplexity.ai, LLAMA, Falcon, and many more. All of them have transformer layers implemented which have attention layers as part of their architectures [6, 7, 8]. The attention mechanism is also inspired by the neuroscientific and psychological perspective of visual attention of the human brain. For instance, when a subject/human is presented with different images, their eye movements reveal the salient image parts that attract their attention. These salient features are often characterised by visual attributes such as intensity contrast, oriented edges, corners, junctions, and motion.

The attention mechanism was introduced to solve issues like bottleneck problems and to attain flexible focus in the encoder-decoder models when models have to tackle applications like machine translation where reference to long distance features are needed. **Bottleneck Problem:** In traditional fixed-length encoding vectors,

the decoder has limited access to input information. This becomes problematic for long or complex sequences, where dimensionality constraints hinder representation quality. **Flexible Focus:** The attention mechanism allows the decoder to focus on relevant parts of the input sequence. It computes a weighted sum of all encoded input vectors, emphasising the most relevant ones. **Generalisation:** Attention can be generalised beyond sequential tasks. it is not limited to sequence-related tasks only and we can use it for any task where we need flexible access to information.

According to the original paper which was published in 2014, [151], to understand the working of attention, let us break it into three parts – alignment scores, weights, and context vectors.

- **Alignment Scores:** The First step in using attention in the model is to compute the scores which will indicate how well input entries are aligned with current output positions. These alignment scores are calculated using the output of the previous decode and the encoded hidden states.

- **Weights/Attention Scores:** After implementing the sofmax operation on alignment score this produces weights. The attention scores are computed using a compatibility function (e.g., dot product, scaled dot product, or concatenation followed by a neural network layer). These weights determine the importance of each encoded input vector and relevant vectors receive higher weights.

- **Context Vector:** At each time step, the decoder receives a unique context vector. This is a weighted sum of all encoder hidden states. This context vector provides relevant information for generating the next output.

Mathematically, the attention mechanism has three main components: queries (Q), keys (K), and values (V). The attention function can be defined as:

$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

where (QK^T) represents the dot product between queries (Q) and keys (K), (d_k) is the dimension of the key vectors, and the square root of this value is used to scale the dot product. Then, the softmax function is applied to the scaled dot product of queries (Q) and keys (K) to have a probability distribution representing the attention weights. These weights are then used to create a weighted sum of the values (V), which is the output of the attention layer. Hence, the attention output is written as:

$\text{Attention Output} = \sum_{i=1}^{n} \alpha_i V_i$

where (α_i) are the attention weights computed by the attention mechanism, (V_i) are the value vectors and (n) is the number of value vectors.

This attention function allows the model to dynamically focus on different parts of the input sequence, assigning more weight to the most relevant parts for the task at hand. The use of the softmax function ensures that the weights sum up to 1, forming a valid probability distribution. There are many types of attention mechanism that have been proposed in the literature like soft, hard implicit, explicit, global and local attentions and for more details one can follow [152, 153].

2.7.8 Transformers:

The transformers were introduced in 2017 in the paper "Attention Is All You Need" where authors used it to solve NLP-related tasks where authors utlised attention layers to form transformers [154]. The idea of a transformer was a big step forward for the ML community as before researchers mostly solved sequences-to-sequence, text generation, and sentiment analyses tasks using RNNs. Compare to RNNs, the transformer-based model has clear advantages in terms of long-term memory and parallelisation. The architecture of the transformer consist of two parts the **encoder** and the **decoder**.

- **Encoder:** The encoder block has multiple identical layers (typically 6 in the original paper). Each encoder block contains:
  - **Multi-Head Self-Attention Mechanism:** This mechanism allows the model to access different parts of the input sequence simultaneously. It has a similar attention mechanism a explained in the above section but implemented in parallel to run on multiple sequences.
  - **Feed-Forward Neural Network:** This is a simple feed-forward network that processes the attention output.
  - **Residual Connection:** To avoid vanishing gradients, residual connections are added around each sub-layer.

  The purpose of the encoder is to take input sequences, process them, and transform them into context representations.

- **Decoder:** Similar to the encoder, the decoder block also has multiple identical layers, which are the following:
  - **Masked Multi-Head Self-Attention:** The decoder attends only to positions before the current position to prevent information leakage.
  - **Multi-Head Encoder-Decoder Attention:** The encoder processes the input sequence and maps it into a sequence of continuous representations, which the decoder then uses to generate an output sequence. The

Multi-Head Encoder-Decoder Attention mechanism helps the decoder to focus on relevant parts of the input sequence during this process.

  - **Feed-Forward Neural Network and Residual Connections**. The decoder generates the output sequence based on the context representation from the encoder.

- **Positional Encoding:** Since Transformers do not inherently handle sequence order, positional encodings are added to the input embeddings to provide positional information.

Transformers have become one of the most important components of ML, DL and Generative AI. To the best of our knowledge, current generative AI related chatbots like ChatGPT, Copilot, Gemini, and LLAMA are popular examples of their use which proves how successful transformers have been.

2.7.9 Vision Transformers:

Traditionally, transformers were designed to tackle NLP-related tasks however, to use the transformer architecture for computer vision tasks like classification, object detection and segmentation, a new type of model was proposed in 2020 [38] called the vision transformer (ViT). The working of vision transformers starts by splitting an image into patches, flattening the patches to produce lower-dimensional linear embeddings from the flattened patches, adding positional embeddings, feeding the sequence as an input to a standard transformer encoder, pretraining the model with image labels (fully supervised on a huge dataset), and finetuning on the downstream dataset for image classification.

Furthermore, in 2021 a paper titled "Do Vision Transformers See Like Convolutional Neural Networks" from the DeepMind team was published to explain why ViTs are better than CNNs [155]. According to that paper, ViT incorporates more global information than CNN at lower layers, leading to quantitatively different features. Skip connections in ViTs are even more influential than in CNNs, having strong effects on performance and representation similarity. CNNs required more lower layers to compute similar representations compare to VIT. Larger VIT models develop significantly stronger intermediate representations through larger pretraining datasets. Therefore, the differentiator between ViTs compared to CNNs can be summarised as differences in global context, scalability, and flexibility.

- **Global Context:** Unlike Convolutional Neural Networks (CNNs), which process local receptive fields, ViTs can capture global dependencies between any parts of the image.

- **Scalability:** ViTs can be scaled up with more data and computational resources, often leading to better downstream performance.
- **Flexibility:** The transformer architecture is flexible and can be adapted to various vision tasks beyond image classification, such as object detection and semantic segmentation.

2.8 Machine Learning in Domain Generalisation and Adaptation

Generally, in ML infrastructures it is common practice to use the same distributions of data in training and in testing. The prime purpose of domain generalisation is to develop ML models which could easily be generalised across various unseen distributions and still perform a task like classification or recognition at a significant performance level. Sometimes, DG is also referred to as out-of-distribution generalisation and it tries to solve tough conditions such as when one or many different but related domains are provided and the major aim is to develop and train a model which can be generalised to unseen testing domains.

Transfer learning, DA, multi-task learning, multiple domain learning, meta-learning, lifelong learning, and zero shot learning are popular and closely related fields to DG. DA was covered earlier in Section 2.6. This sub-section explains various algorithms which serve the purpose of DG and DA for prospective fields of ML.

2.8.1 Meta-learning in Domain Generalisation and Adaptation

During the training or learning of a meta-learning model, the tasks are different compared to other types of ML but in DG the learning tasks are always the same. In the literature, meta-learning is regarded as a general learning strategy which has been used in recent years [156, 157, 158, 159]. To increase the performance of DG, it tries to approximate the meta-training and meta-testing tasks in the training domains.

As we know, the essential idea of meta-learning is to learn a general model from multiple distributions of tasks by using methods like optimisation-based, metric-based, and model-based. These methods divide the multi-source input data into meta-training and meta-testing sets to simulate domain shift. Moreover, in [156] MLDG (meta-learning for DG) is inspired by MAML and to approximate domain shift situations and learn a general representation, MLDG also splits the data from

the source domains into meta-training and meta-testing. MetaReg [157] proposed a method to design a meta regulariser for classifiers. In [158] the authors proposed feature-critic training and extracted features by using a meta optimiser. [160] is similar to MLDG but has two additional loss functions. Recently, meta-learning is also proving its worth in semi-supervised DG and in discriminative DG as shown in [161, 162, 163, 164]. Therefore, in summary we can see that meta-learning has been used for research work in DG and it can also contribute to a few other paradigms like disentanglement [165].

2.8.2 DRL in Domain Generalisation and Adaptation

Generalisation in DRL has the goal to provide RL algorithms with generalised policies across various novel unseen situations at deployment time, avoiding overfitting to their prospective training environments. Exploring DG in DRL is important and essential if we want to deploy its model in the real world with more diverse, dynamic, and unpredictable environment scenarios. In a recent survey article, [166] provides comprehensive information about how to measure generalisation and popular benchmarks of DRL. To handle generalisation in DRL, different methods have been proposed including increasing similarities in training and test distributions, data augmentation and domain randomisation, environment generation, handling differences, fast adaptation, RL-specific problems and improvements, and many more [166].

UCB-DrAC [167] introduces a method for automatically picking the best augmentation technique to use during training while [168] proposed a method to use a randomised convolutional layer at the start of the network which improves the robustness to a large range of visual inputs. In [169] the work shows how to increase the diversity of data. All the methods described in [167, 170, 168, 169] are tested on the CoinRun data challenge described in [171]. CoinRun was introduced in 2018 as a benchmark for generalisation in reinforcement learning by the then non-profit AI research company OpenAI. This AI training environment provides a metric for an agent's ability to transfer its experience to unfamiliar scenarios.

2.8.3 Deep Learning in Domain Generalisation and Adaptation

Generalisation in DL is a well-studied area and is more mature than generalisation in RL and meta-learning. In DL, models are trained on a training set, and performance of the model is measured on an available testing set. Moreover, it is also assumed that the training and testing datasets belong to independent and identically distributed random variables (known as iid). To measure generalisation in

DL, researchers use different gap functions to measure the gap between training and testing datasets and the smaller the gap then that means better generalisation [172].

2.8.4 Transfer Learning vs. Domain Generalisation and Adaptation

While doing development in ML there is an assumption that training and testing data must be from the same kind of feature space. However, in many real time applications and situations, usually this assumption does not hold true. For instance, if we have a classification task to solve in one domain, but the dataset is available in another domain then this means that the available data may have a different feature space.

In such situations, usually it is preferred to do knowledge transfer which can potentially increase performance of ML models by avoiding expensive data labeling. This is called transfer learning and it is very common in ML. Pre-training or fine-tuning are other commonly used strategies. In transfer learning the source and target domains have different tasks and the target domain is accessed in training. In DG, the target domain cannot be accessed and the training and test tasks are often the same while they will have different data distributions.

2.9 Conclusions

In this chapter we introduced and explained the basic differences between traditional ML and its more recent fields, such as DL, RL, and meta-learning. We also discussed the motivations behind the development of early ML methods and their limitations, including the need for manual feature extraction and computational inefficiencies. In the section on DL, we described its fundamental components and summarised various DL models used in applications like classification, object detection, and segmentation. Additionally, we included an historical diagram that illustrates the evolution of different ML fields in relation to their applications and types.

Within the introduction to various types of ML methods, this chapter also discussed common challenges, including overfitting, sample inefficiency, training biases, domain generalisation, and explainability. Overfitting occurs when a model performs well on training data but poorly on unseen data, often due to learning noise rather than useful features. Techniques like cross-validation and regularisation can mitigate overfitting. Sample inefficiency refers to an algorithm's inability to learn effectively from a limited number of samples, leading to slow improvement and high computational costs. Training biases arise from imbalances in data, affecting model performance and fairness. Domain generalisation and adaptation address the issue

of domain shift, with generalisation focusing on performing well in unseen domains and adaptation optimising performance using target domain data. Explainability, or the ability to interpret model outputs, is crucial for user trust and transparency, with various methods available to explain both simple and complex models.

The architectures used for ML start from basic building blocks like feed-forward layers and go to advanced concepts like the attention mechanism, transformers, and vision transformers. In summary, feed-forward layers are basic units in neural networks where each layer's output feeds into the next without forming cycles. MLPs are such networks with multiple layers used for tasks like classification. CNNs are specialised for grid-like data such as images, using convolutional layers to learn spatial hierarchies. RNNs process sequences of data, maintaining information over time, with Long LSTM units improving long-term dependency handling. Generative Adversarial Networks (GANs) have a generator and discriminator working together to create realistic data samples. Attention Layers help models focus on specific input parts, enhancing tasks like machine translation. Transformers use attention mechanisms for parallel data processing, improving language modelling efficiency. ViTs apply transformers to image patches, achieving top performance in vision tasks.

Overall, in machine learning (ML), it is standard practice to use the same data distributions for both training and testing. DG aims to develop ML models that can perform well on unseen data distributions, maintaining significant performance in tasks like classification or recognition. DG is also known as out-of-distribution generalisation and deals with challenging scenarios where models must generalise across different but related domains. Related fields include transfer learning, domain adaptation, multi-task learning, multiple-domain learning, meta-learning, lifelong learning, and zero-shot learning. Meta-learning, a key strategy in DG, involves learning a general model from multiple task distributions using methods like optimisation-based, metric-based, and model-based approaches. This is achieved by dividing data into meta-training and meta-testing sets to simulate domain shifts. DRL generalisation seeks to create policies that perform well in diverse, novel environments, avoiding overfitting to training conditions. Methods such as data augmentation, environment generation, and fast adaptation are employed to enhance generalisation in DRL. In contrast, DL generalisation is more mature and involves training models on one set of data and testing on another, with the goal of minimising the performance gap between the two. Transfer learning differs from DG as it allows access to the target domain during training and focuses on leveraging knowledge from one domain to improve performance in another, whereas DG does not access the target domain during training and aims to generalise across different data distributions while maintaining similar tasks,

In the next chapter we look at vision-based ML algorithms which have veen

specifically designed and tested for out of distribution generalisation.

Chapter 3

Vision-Based Machine Learning Algorithms for Out-of-Distribution Generalisation

In the previous chapter, we learned about literature that has been published in the field of machine learning with respect to various applications, especially for classification, object detection, segmentation, and more. We also explored common challenges within the field and the architectures of various components of the current ML/DL field. The chapter also introduced issues related to domain generalisation-for various kind of learnings and the available benchmarks for this topic.

In this chapter, we will start an exploration into RQ1 which is related to the measurement of domain generalisation and we will present some experiments to show how dynamic learning can have better domain performance than static learning methods. To fulfil this purpose, the chapter first provides an introduction to the problems of domain generalisation and our initial contribution. Then the chapter explains some of the related work leading to descriptions of up-to-date implemented algorithms and approaches. We then address the questions of why we select the PACS and the Office-Home benchmarks for our study. The experimental methods section give information-related formulations of domain generalisation with domain-specific and domain-generalised methods. Finally a results and discussion section gives insights into the training procedures and the story behind the presented numbers.

There are many computer vision applications including object segmentation, classification, object detection, and reconstruction for which machine learning (ML) shows state-of-the-art performance. Nowadays, we can build ML tools for such applications with greater than human level accuracy. However, each of these tools normally works well only within the domain in which it has been trained and de-

veloped. Often, when we train a model on a dataset in one specific domain and test it on another unseen domain known as an out of distribution (OOD) dataset, models or ML tools show a decrease in performance. For instance, when we train a simple classifier on real-world images and apply that model on the same classes but with a different domain like cartoons, paintings or sketches then the performance of the ML tools disappoints. This presents serious challenges of domain generalisation (DG), domain adaptation (DA), and domain shifting. To enhance the power of ML tools, we can rebuild and retrain models from scratch or we can perform a technique called transfer learning. In this chapter, we present a comparison study between vision-based technologies for domain-specific and domain-generalised methods. In this chapter we highlight that simple convolutional neural network (CNN) based deep learning methods perform poorly when they have to tackle domain shifting. Experiments are conducted on two popular vision-based benchmarks, PACS and Office-Home. We introduce an implementation pipeline for domain generalisation methods and conventional deep learning models. The outcome confirms that CNN-based deep learning models show poor generalisation compared to other extensive methods.

3.1 Introduction

The field of ML has created tremendous success stories by solving many complex problems like object classification, detection, segmentation and reconstruction in videos, and other problems in areas including NLP, medical image analysis, robotics, and many more. These developments in ML algorithms and databases, and the fusion of various fields of ML help researchers to achieve high-level goals. The majority of current applications are built on what we call traditional ML where we usually have millions of example datapoints with labels to train a model under SL conditions.

Since 2011, with the help of deep learning (DL) which is a sub-domain of ML that deals with various datasets to automatically extract features, scientists are now using DL to handle both supervised and unsupervised learning problems as described in [1]. Pure DL-based models are static systems, and have many problems including overfitting, they commonly need huge datasets, have data biases, and they do not have significant potential for generalisation or for domain adaptation [173].

Previous works illustrate that the majority of the time ML tools fail to generalise when processing out of distribution (OOD) data. The main reasons for wanting to design and analyse such generalised tools are applications like vision based autonomous systems, and medical imaging [173]. For example, when only a few conditions change during an inference process in image processing such as light vari-

ations, shapes, locations, or the pose of objects, then models perform poorly because they did not have interaction with similar variations during their training phase and thus did not learn how to perform under such unpredictable circumstances [174, 175, 176]. The work in [177] conveys information about the collapse of ML tools for generalisation of OOD data which actually happens when ML models learn fake correlations instead of capturing real factors behind such variations in data. These fake correlations can be racial biases, texture statics, and object backgrounds, for example.

In the research literature, researchers have developed several methods to tackle domain generalisation. For instance, one of the first solutions that was tried is to increase the size of the training dataset with the same tasks but in different environments. The goal of the domain generalisation algorithm in this case is to learn the invariances and features for all possible domain shifts.

Before we move to the contributions of this chapter, it is important to understand the difference between domain generalisation and domain adaptation. When algorithms process samples of data from different distributions, ML algorithms suffer from a common problem called the domain shift. This introduces two major issues namely DG and DA. DG deals with the comparatively difficult situations where several different but related domains are given, and the purpose of a machine learning algorithm is to learn a model which could be generalised on unseen test data. The main goal of DG is learning a representation of its training data that could have the potential to perform well in unseen domains by leveraging more source domains during training.

The idea behind DA is different to DG in that it is to maximise the performance of algorithms or models on a given target domain using existing training source domains. The main difference between DA and DG is that DA has access to the target domain data which implies that it can see the data while DG cannot see anything from the target domain during training. This makes DG more challenging than DA but more realistic and more favourable to use in practical applications. There are many generalisation-related research topics or solutions such as domain adaptation, meta-learning, transfer learning, covariate shift, lifelong learning, and zero-shot learning.

This chapter addresses a direct comparison study between domain-specific and domain generalised methods with respect to vision-based applications, especially classification. To achieve our goal, for our experiments and to investigate the topics of this thesis we have implemented a pipeline with 9 well-known domain generalised algorithms and 7 domain-specific models. The comparison study is conducted on two popular benchmarks namely PACS and Office-Home, from which some sample images are shown in Figure 3.1. We also trained and tested 16 models by using fine-

tuning. Our research shows the learning curves of methods for both benchmarks. The result section considers accuracy as a measure of generalisation for supervised learning benchmarks.

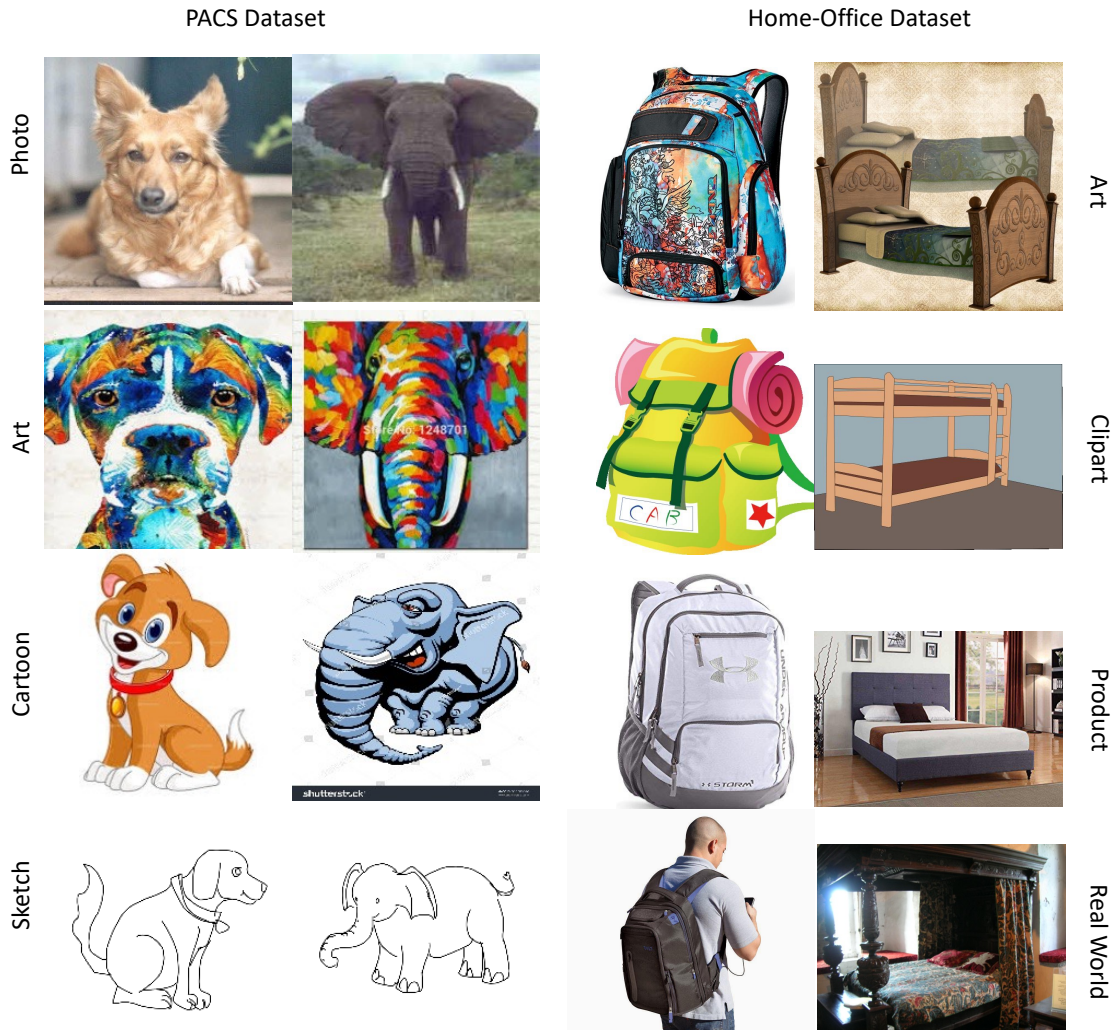


Figure 3.1: Sample images from the same classes across all domains in the PACS and Home-Office datasets.

The work in this chapter contributes to the overall thesis hypothesis and research questions in the following ways The research work and experiments in this chapter focus on the sub-hypothesis (H1) which says that a comparison between domain-specific and domain generalised methods is justified and H1 also points out that domain generalised methods are better than domain-specific methods. Moreover, the comparison in this chapter also partially highlights sub-hypothesis (H2). Meanwhile, RQ1 says there is no absolute way to measure the generalisation ability of a model. Therefore, in this chapter we consider accuracy as a measure of DG and what type of experiments should best reflect a study of better generalisation. Finally, the work in this chapter also targets RQ3 by comparing the accuracies of both types of methods.

The rest of the chapter is structured as follows: Section 2 covers some of the related work while Section 3 briefly introduces details about the algorithms we use. Section 4 is about the study of benchmarks in this area. Section 5 describes our proposed method including the experiments and formulation of DG. Results and discussion are presented in Section 6 while Section 7 presents conclusions and future work.

3.2 Related Work

To handle domain generalisation, several alternative methods have been proposed to select hyperparameters so that a model can maximise its performance for OOD applications [178]. The parameters update with the rest to a function which calculates relatedness between the different domains. Similarly, in [179] the authors illustrate ways to select models based on an algorithm-specific regularisation. These previous articles worked only for a specific kind of domain generalisation problem, although the scope for this chapter is to analyse vision-based domain generalisation for various up-to-date methods.

In the literature we find that domain generalisation has available algorithms which can be classified into data manipulation, representation learning and strategy learning algorithms [12]. One piece of information which we can extract from [12] on data generation and adversarial training is also a kind of common way to optimise ML tools for OOD. In this regard, the well-known ImagNet challenge [180] was also updated for different kinds of OOD where authors created new benchmarks and then tried to solve them with new approaches as in [181, 182, 183]. Nevertheless, these methods did not consider solving domain generalisation in vision-based applications with the most up-to-date approaches as explained in Domainbed [177]. Furthermore, [181, 182, 183] have variations in the ImageNet challenge to solve the domain generalisation paradigm even though they tried only few domain-specific methods, on the other hand totally missing out domain generalisation frameworks. Table 3.1 provides a clear picture in this regard, as well as a comparison to our own approach in the last row.

The most related of the state-of-the-art research papers are [177] and [12]. In [177], the authors implemented a framework named Domainbed which has support for various domain generalised methods to analyse vision-based domain generalisation. Domainbed has variety in model selection, training schemes and hyper parameters using Resnet as its backbone model for domain generalisation frameworks. The anaylses provided by the authors in those papers were insightful. However, [177] does not have information about domain-specific methods and does not provide training and inference analysis for domain-specific models.

Table 3.1: Comparison of our contribution to previous articles for domain generalisation and domain-specific methods where ✓ indicates whether an article has the details and ✗ means the article is missing that aspect.

Article	Model Selection	Training Framework Selection	Multiple Datasets	Domain Generalisation Frameworks	Domain-Specific Frameworks
Gulrajani and Lopez-Paz [177]	✓	✓	✓	✓	✗
Wang *et al.* [12]	✓	✓	✓	✓	✗
Hendrycks *et al.* [181]	✓	✗	✓	✗	✗
Hendrycks and Dietterich [182]	✓	✗	✗	✗	✓
Hendrycks *et al.* [183]	✓	✗	✗	✗	✓
Ours	✓	✓	✓	✓	✓

Similarly, [12] also conducted the same kind of study as [177] but improves the limitations of Domainbed in implementation and coding flexibility. However, neither of these works discussed the effect or performance of traditional deep learning methods for OOD generalisation which is the scope of this research. In this chapter, we perform an analysis of typical deep learning methods and then compare their performance with domain generalised methods. Therefore, Table 3.1 highlights the gaps in other works and improvements in terms of contributions for our research.

Most of the literature review that has been mentioned is narrowed to frameworks or pipelines for algorithms related to domain generalisation and these methods only provide solution with supervised learning. However, it is also important to highlight some domain generalisation methods in other areas of ML including, unsupervised learning, reinforcement learning, meta-learning, lifelong learning, and zero-shot or few-shot learning. Although such methods will not be directly within the scope of our research we will briefly explain them here.

We included the selected methods in Table 3.1 to compare them with our proposed framework for addressing the domain generalisation task across various machine learning models. These methods were chosen to provide fundamental insights into different aspects of DG. As highlighted in Table 3.1, prior literature such as [177, 12] largely overlooks domain-specific methods, while studies like [182, 183] focus on domain-specific approaches but neglect domain-generalised methods. Therefore, it was crucial to present a comprehensive overview of existing literature trends related to DG to better contextualise our contributions.

Unsupervised Domain Generalisation: The paper entitled "Towards Unsupervised Domain Generalisation" [184] introduces unsupervised domain generalisation (UDG) as a novel problem that focuses on learning models capable of generalising across domains using unlabeled data. The authors propose a method called domain-aware representation learning (DARLING) to address the challenges of heterogeneity in unlabeled data and distribution shifts between source and target data. The study analyses how pre-training with unlabeled data from various source domains can impact domain generalisation. DARLING demonstrates superior or comparable performance to the traditional ImageNet pre-training, even with fewer unlabeled data, highlighting its potential to improve generalisation with large-scale unlabeled datasets. Similarly, in the paper [185] entitled "Unsupervised Domain Generalisation by Learning a Bridge Across Domains", it addresses the challenge of generalising learned representations across significantly different visual domains, such as real photos, clipart, paintings, and sketches, without training supervision in source or target domains. The authors propose a self-supervised learning approach that creates an auxiliary bridge domain along with visual mappings from each train-

ing domain to Bridge Across Domains (BrAD). This helps align instances of the same class across different domains. They use a contrastive self-supervised representation model that semantically aligns each domain to its BrAD projection, driving all domains to align with each other. The edge-regularised BrAD approach shows significant gains across multiple benchmarks and tasks, including unsupervised domain generalisation, few-shot unsupervised domain adaptation, and generalisation across multi-domain datasets.

Domain Generalisation with Reinforcement Learning: There are many articles that have been published to solve domain generalisation with RL, therefore instead of covering them all we mention a survey article to summarise the progress in the field. In [166], the authors aim to address the challenge of generalisation in DRL, which is crucial for deploying RL algorithms in real-world scenarios that are diverse, dynamic, and unpredictable. It provides a unifying formalism and terminology for discussing different generalisation problems, categorises existing benchmarks for generalisation, and reviews current methods for improving generalisation in RL. The authors critically discuss the current state of the field, including the limitations of procedural content generation for benchmark design, and suggest areas for future work such as fast online adaptation and RL-specific problems. Recommendations for future research include developing benchmarks for under explored settings like offline RL generalisation and reward-function variation. This survey serves as an important overview of the recent field of generalisation in deep RL, highlighting the importance of robustness and adaptability of RL policies to unseen environments.

Similarly, in [186] the paper introduces the concept of "relay-generalisation" in RL, which assesses an RL agent's ability to handle out-of-distribution "controllable" states. The authors propose a novel evaluation method called relay evaluation. This involves starting a test agent from the midpoint of trajectories generated by other independently trained "stranger" agents. The study reveals a significant generalisation failure in current RL agents when they encounter controllable states generated by stranger agents1. For instance, a well-trained Proximal Policy Optimization (PPO) agent in the Humanoid environment had a low failure rate of 3.9 % during regular testing but failed on 81.6% of the states generated by stranger PPO agents. To address this issue, the authors introduce Self-Trajectory Augmentation, a method that resets the environment to the agent's previous states based on the Q function during training. The contributions are twofold, it provides a new perspective on evaluating RL generalisation and offers a practical solution to enhance the performance of agents in out-of-distribution scenarios. Hence, the latest articles to solve OOD-related challenges in RL are mentioned in [187, 188, 189]

Domain Generalisation with Meta learning & Zero-shot Learning: The DG problem can be addressed using meta learning based methods and a recent survey from 2024 [190] has summarised most of the up-to-date published research. The paper discusses how meta-learning can be used to improve DG in DNN which often struggles with OOD data. It introduces a new taxonomy based on feature extraction strategies and classifier learning methodologies.

The survey offers practical insights and discusses promising research directions for innovations in meta-learning for DG. Additionally, the article [156] also proposed a meta-learning procedure which has been evaluated and achieved state-of-the-art results on a cross-domain image classification benchmark and showed potential in two classic RL tasks. Moreover, for zero-shot DG, in [191], that article introduces the concept of zero-shot DG, which is an extension of DG. It addresses the challenge of training a model on multiple domains and generalising to new, unseen domains with different label spaces. The authors propose a strategy that utilises semantic information of classes to adapt existing DG methods for zero-shot DG. This allows the model to generalise across new domains and new classes within those domains. The proposed methods have been evaluated on various datasets including CIFAR-10, CIFAR-100, F-MNIST, and PACS, establishing a strong baseline for future research in this area.

3.3 Implemented Algorithms

Our research uses an implementation of 16 algorithms in total, including 9 domain generalised and 7 conventional deep learning methods. This section briefly summarises each of them.

3.3.1 Recent Algorithms for Vision-Based Generalisation

- **Empirical Risk Minimisation (ERM)** is a simple method which actually minimises the total sum of all errors in the given domains [177]. The cited paper describes the work important for the performance of DG, areas like model section criteria and proposed domainbed which is a testbed for evaluating future progress of algorithms related to DG.

- **Group Distributionally Robust Optimisation (DRO)** [192] also performs ERM but gives more focus to those domains with larger errors. We can say that it is an extension of simple ERM. The authors propose DRO to learn models that minimise the worst-case loss over a set of pre-defined groups, which helps to avoid learning models that rely on fake correlations. The authors find that overparameterised neural networks can perfectly fit the

training data and any model with a vanishing average training loss also has vanishing worst-case training loss. The authors show that regularisation is important for worst-group generalisation in the overparameterised regime, even if it is not needed for average generalisation. They use strong l_2 penalties, early stopping, and group adjustments to achieve better worst-group test accuracies. Moreover, the paper introduces a stochastic optimisation algorithm with convergence guarantees to efficiently train group DRO models.

- **Inter-Domain Mixup (Mixup)** Two published papers, [193, 194] use ERM on linear interpolations of data in domains. For instance, in [193] the authors propose a novel approach to improve the performance of DA models by incorporating domain mixup on both pixel and feature levels. The authors argue that existing adversarial DA methods are limited because of they are incapability of fully utilising the intrinsic structure of data distributions and dependence on hard labels. The paper introduces a generative-adversarial-based framework to simultaneously train a classification network and generate auxiliary source-like images from learned embeddings of both domains. Domain mixup is conducted on pixel and feature levels to improve the robustness of models and guide the domain discriminator in judging samples' differences relative to source and target domains. The authors design a framework that maps both domains to a common latent distribution, efficiently transferring knowledge learned on the supervised domain to its unsupervised counterpart.

- **Domain Adversarial Neural Networks (DANN)** is described in [195] and explores features with distribution matching in the external domains. The paper entitled "Domain-Adversarial Training of Neural Networks" by Ganin et al. [195] introduces a novel approach to domain adaptation, where data from the training and test domains come from similar but different distributions. The authors propose a DANN that combines DA and deep feature learning within one training process. DANN is designed to learn features that are both discriminative for the main learning task and invariant to the shift between the domains. This is achieved by jointly optimising the underlying features and two discriminative classifiers, one is a label predictor and the other is a domain classifier. The domain classifier is trained to maximise the loss on the source domain, while the label predictor is trained to minimise the loss on the source domain. The authors demonstrate the success of DANN for various classification problems, including sentiment analysis and image classification, and show that it can improve state-of-the-art performance on several benchmarks.

- **Class-conditional DANN** also known as C-DANN [196], is an extension of

DANN and instead of matching features in the data distributions, it matches conditional distributions across the domains and their respective data labels. The authors address the fact that this new paradigm can be achieved by ensuring the class prior does not change across training and test domains. They introduce a conditional invariant representation that can be guaranteed if the class prior remains consistent.

- **Deep CORAL** introduced in [197], utilises the matching between the mean and covariance of features across the distributions.The paper proposes a novel approach to unsupervised DA, which leverages both the deep features pre-trained on a large generic domain and the labeled source data. The authors extend the CORAL method [198], which aligns the second-order statistics of the source and target distributions with a linear transformation to learn a non-linear transformation that aligns correlations of layer activation in DNN. This approach is called Deep CORAL which is designed to minimise the difference in second-order statistics between the source and target feature activation and it ensures that the learned features are both discriminative and invariant to the difference between the domains.

- **Maximum Mean Discrepancy (MMD)** as described in [199] measures the alignment of distributions across all the domains and uses adversarial feature learning to match aligned distributions. This method leverages adversarial autoencoders to learn a generalised feature representation across multiple source domains. The authors extend adversarial autoencoders by incorporating the MMD measure to align the distributions among different domains and matching the aligned distribution to an arbitrary prior distribution via adversarial feature learning. This approach shows that the learned feature representations are both domain-invariant and discriminative so that the trained model could be generalised well on unseen target domains.

- **Invariant Risk Minimisation (IRM)** [200] learns a linear classifier on top of representation matching techniques. This article introduces the concept of IRM which aims to learn a data representation that extracts an invariant predictor across multiple environments or domains. This is achieved by minimising a weighted average of training errors. The authors argue that existing DA methods often fail to capture the true invariance in the data which leads to poor performance on unseen domains. They propose a learning paradigm that extracts nonlinear invariant predictors across multiple domains, enabling OOD generalisation.

3.3.2 Conventional Deep Learning Algorithms for Vision-Based Applications

To have a fair comparison with other approaches, the processingpipeline developed to explore the research hypotheses in this thesis, also includes domain-specific conventional deep learning methods where we implemented 7 well-known DL networks including AlexNet [26], VGGNet16 [32], ResNet18 [34], ResNet50 [34], InceptionV3 [201], DenseNet [35], and SqueezeNet [113], each of which are covered in an overview article [202]. These DL models have different architectures based on their number of DL layers and formation and structures. Initially, these models were designed to solve classification-related problems. These networks are popular in computer vision applications, therefore further details on them may not be needed.

The most related state-of-the-art research papers are [177] and [12]. In [177], the authors implemented a framework named Domainbed which has support for various domain generalised methods to analyse vision-based domain generalisation. Similarly, [12] also conducted the same kind of study but improved the limitations of Domainbed in implementation and coding flexibility. However, neither of these works discussed the effect and performance of traditional deep learning methods for OOD generalisation which is the scope of the research in this thesis. In this chapter, we will perform an analysis of typical deep learning methods and then compare their performance with domain generalised methods.

3.4 Out-of-Distribution Benchmarks

Before going into further explanation about the problem statements and research questions for the thesis, it is crucial to understand what is the meaning of out-of-distribution or OOD benchmarks, and what makes them different from other large available datasets like ImageNet or PASCAL.

By conducting a detailed literature review and exploration work for DG, we understand that DG is a vast area of research that can be considered in any kind of machine learning problems. Therefore, to work on the research hypotheses and to carry out experiments on DG, we only consider vision-based benchmarks for supervised learning.

What is Out-of-Distribution data: In ML, Out-of-Distribution (OOD) detection is considered very important to achieve general purpose AI performance as it provides reliability and safety to ML systems. By definition, when we talk about the term "out-of-distribution" in ML, it means that such data samples belong to a different distribution and this distribution is unseen to the ML model during the

training process. The goal of OOD generalisation is to build such models which can be reliable and resilient to changes in the time of inference on real-world applications.

Moreover, to tackle OOD generalisation issues, many benchmarks have been proposed. The main difference between a common visual benchmark and DG benchmark is that conventional benchmarks or datasets have classes/objects/categories in a single domain/environment/setting. On the other hand, a DG benchmarks have classes/ objects/categories in various settings/domains/ environments.

Table 3.2: Benchmarks

Datasets	**Domains**	**Classes**	**Samples**	**Descriptions**	**References**
Office-Caltech	4	10	2,533	Caltech, Amazon, Webcam, DSLR	[203]
Office-31	3	32	4,110	Amazon, Webcam, DSLR	[203]
PACS	4	7	9,991	Art, Cartoon, Photos, Sketches	[204]
VLCS	4	5	10,729	Caltech101, LabelMe, SUN09, VOC2007	[205]
Office-Home	4	65	15,588	Art, Clipart, Product, Real World	[206]
Terra Incognita	4	10	24,788	Wild animal images recoded at four different locations L100, L38, L43, L46	[207]
Rotated MNIST	6	10	70,000	Rotated Hand written Digits	[208]
DomainNet	6	345	586,575	Clipart, Infograph, Painting, Quickdraw, Real, Sketch	[209]

Benchmarks Table 3.2 presents a summary of some of the open DG benchmarks used in the literature for supervised learning. For our initial experiments, we explored two of these, PACS and Office-Home benchmarks. In the case of reinforcement learning, RoboSuite, DMC-Remastered, DMC-GB, DCS, KitchenShift, NaturalEnvs MuJoCo , CausalWorld, RLBench, Meta-world and others are commonly used benchmarks [166].

We restrict this work to vision-based benchmarks to narrow the scope because if we were to work with benchmark datasets across multiple domains covering vision, robotics, language processing, etc. then our results would have an overhanging question of whether results would have had as much to do with the domains chosen.

Of the two benchmarks we use, one is a relatively simple dataset (PACS) with 4 different domains including images presented as Art, Cartoon, Photos, and Sketches. Each domain has 7 classes and there are 9,991 sample images in total. The second benchmark is Office-Home, also consisting of images. This also has 4 domains namely Art, Clipart, Product, and Real World with 65 classes in each domain. The main idea behind choosing the first is to work on a benchmark which could have comparatively less complex classification tasks so that we can observe the behaviours of domain-specific and domain-generic models. We select Office-Home as a second benchmark because of the higher number of classes (supervised tasks) which adds complexity into the tasks for each domain. The visual difference in the domains of these benchmarks can be observed in Figure 3.1 shown earlier in Section 3.1.

3.5 Experimental Methods

To investigate our hypotheses and research questions, we created a processing pipeline consisting of both types of algorithms and Figure 3.2 provides an overview of our approach.

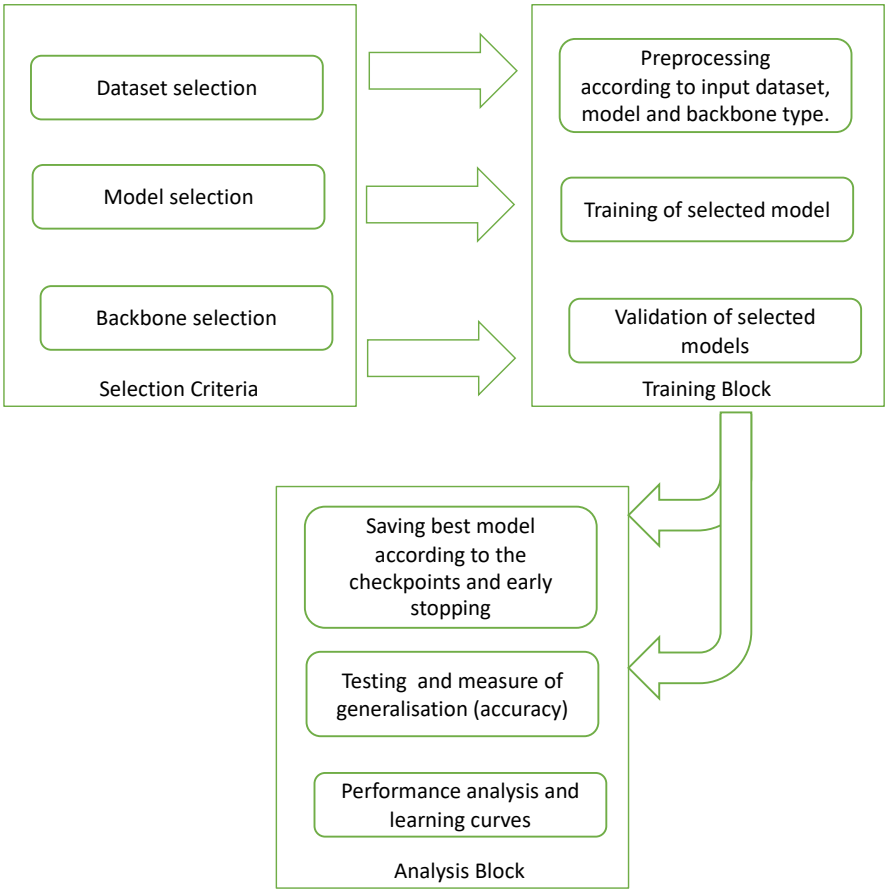


Figure 3.2: Overview diagram summarising our processing pipeline.

Figure 3.2 contains three blocks namely selection criteria, training, and analysis. In the first block, we choose the dataset or benchmark for which we want to try the proposed pipeline. This block also includes the model selection step in which we identify which type of learning method our pipeline will use, either conventional deep learning like VGGNet16 [32], ResNet18 [34] or one of the more recent vision-based domain generalised methods like ERM[177], DRO[192], etc. The training block is the second block and includes data pre-processing according to selected conditions, training and validation of models. The third block, analysis, save the best model according to checkpoints and early stopping. It also measures the generalisation in the form of accuracy and loss metrics and compares the performance by computing learning curves.

3.5.1 Formulation of Domain Generalisation and Experiments

In domain generalisation, let us assume we are given $\mathcal{N}$ training (source) domains, $S_{train} = \{S^i | i = 1, ..., N\}$ where $S^i = \{x_j^i, y_j^i\}$ denotes the i-th domains. The joint distributions between each domain are different with $D_{XY}^i \neq D_{XY}^j$, $N \geq j \neq i \geq 1$. The objective of domain generalisation is to learn a robust and comparatively generalised predictive function $f : X \rightarrow Y$ by using N training domains to get minimum error on an unseen test domain $D_X \rightarrow S_{test}$ where S_{test} cannot be accessed in training and $D_{XY}^{test} \neq D_{XY}^i$. Therefore, the model's goal is to minimise the loss function L on S_{train}

$$\min_f E_{(x,y)} \in S_{train}[L(f(x), y)]$$

where E is the expectation and y are the labels. Figure 3.3 presents a graphical representation of domain generalisation. The above equation highlights a fundamental principle for machine learning models: they should be trained to optimise an objective function that minimises the loss on the training distribution. Furthermore, this learned objective should generalise effectively, ensuring the model exhibits similar performance on the testing distribution S_{test}. This approach underscores the importance of designing models capable of capturing patterns that extend beyond the training data, thereby supporting robust generalisation.

3.5.2 Domain Generalisation Model Experiments

Here we look at the model pipeline for the training and the inference domain generalisation. The current version of our approach has been executed on two benchmarks PACS and Office-Home and for each of these, 9 models which were commonly used in the recent literature have been used, these models being GroupDRO, ANDMask,

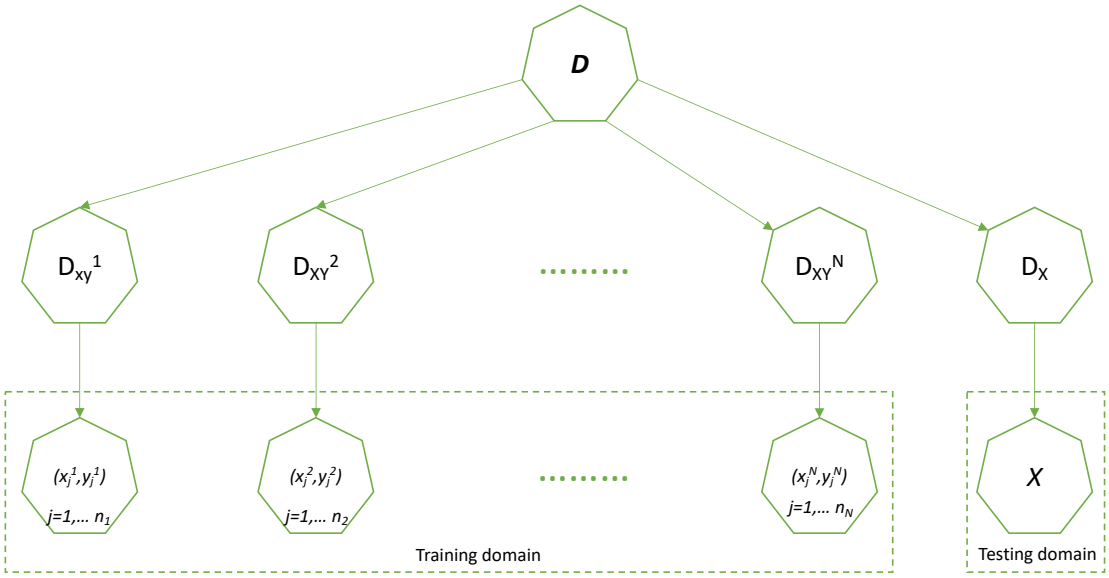


Figure 3.3: A graphical representation of domain generalisation.

Mixup, MMD, DANN, CORAL, VREx, RSC, and ERM. Experiments were performed using Pytorch as a backend with an Nvidia GPU RTX 3090, with 24 GB memory.

Experiments were conducted with original images from the benchmarks without any additional data augmentation and in the pre-processing phase only re-sizing of the images was implemented. Each model has its own set of hyperparameters but the common parameters are batch size 32, epochs 120, momentum 0.9, learning rate 0.01, weight decay 0.0005, input size (3, 224, 224), and baseline model Resnet-18. From each source domain of each of the two datasets, PACS and Office-Home, models utilise 80% of the data in training and validation, and keep 20% of the data as the unseen or target domain.

3.5.3 Domain-Specific Model Experiments

Our domain-specific processing pipeline has different settings to the domain generalisation pipeline and it also supports the same two benchmarks as well as 7 domain-specific models namely AlexNet, VGGNet16, ResNet18, ResNet50, InceptionV3, DenseNet121, and SqueezeNet. In this system, we use the same Pytorch environment with the same Nvidia GPU RTX 3090, with 24 GB memory. These models use a fine tuning technique in which pre-trained weights can be used as a feature extractor and the last fully-connected layers could be re-initialised and trained.

For these experiments, a model does training and validation in one domain and then performs inference in another target domain. For example, in the case of PACS,

models explore the domain of "art painting" (with 1,638 samples) in training and in validation, and then use 20% of the target domain's "cartoon images" (with 410 samples). Similarly, for the Office-Home dataset, models use the domain of "clipart" (with 3,492 samples) as the source, and then images categorised as "real world" (with 873 samples) as the target.

During training, models do not have any access to the target domain and initially, models use only one domain as a source. The common hyperparameters are batch size 64, epochs 120 with early stopping 20, momentum 0.9, learning rate 0.0001, weight decay 0.0005, input size (3, 224, 224), and cross-entropy loss.

3.6 Experimental Results

Table 3.3 presents the results for ther accuracy metric for 9 domain generalisation frameworks and 7 conventional deep learning or domain-specific models shown in blue font.

Table 3.3: Experimental accuracy results with domain generalisation and domain-specific methods

	PACS		Office-Home	
Models	Validation	Target	Validation	Target
GroupDRO	0.95	0.73	0.82	0.52
ANDMask	0.95	0.72	0.81	0.44
Mixup	0.97	0.72	0.83	0.53
MMD	0.94	0.69	0.82	0.52
DANN	0.94	0.73	0.83	0.51
CORAL	0.95	0.77	0.84	0.55
VREx	0.97	0.80	0.76	0.49
RSC	0.97	0.77	0.83	0.50
ERM	0.97	0.78	0.84	0.57
AlexNet	0.74	0.45	0.56	0.30
VGGNet16	0.80	0.47	0.50	0.23
ResNet18	0.86	0.51	0.65	0.52
ResNet50	0.89	0.57	0.70	0.62
InceptionV3	0.90	0.55	0.68	0.66
DenseNet121	0.86	0.44	0.62	0.35
SqueezeNet	0.80	0.50	0.54	0.29

In Table 3.3, the columns marked "Validation" and "Target" represent the accuracy figures for the validation and for the unseen or target testing sets respectively with results presented for two different benchmarks. Well-trained models will have

adequately high validation accuracy and target accuracy always tries to follow validation accuracy. According to various types of data distribution, a model can have different values of validation and target accuracy but for a balanced dataset, an accuracy figure close to 90% can be considered good enough to deploy in some application domains.

Table 3.3 has five columns and the rows include the names of the models used and the validation and target accuracy performance figures for each dataset. The first 9 models belong to the domain generalisation method and the remaining 7 models in blue font relate to domain-specific methods. Overall, if we examine the PACS

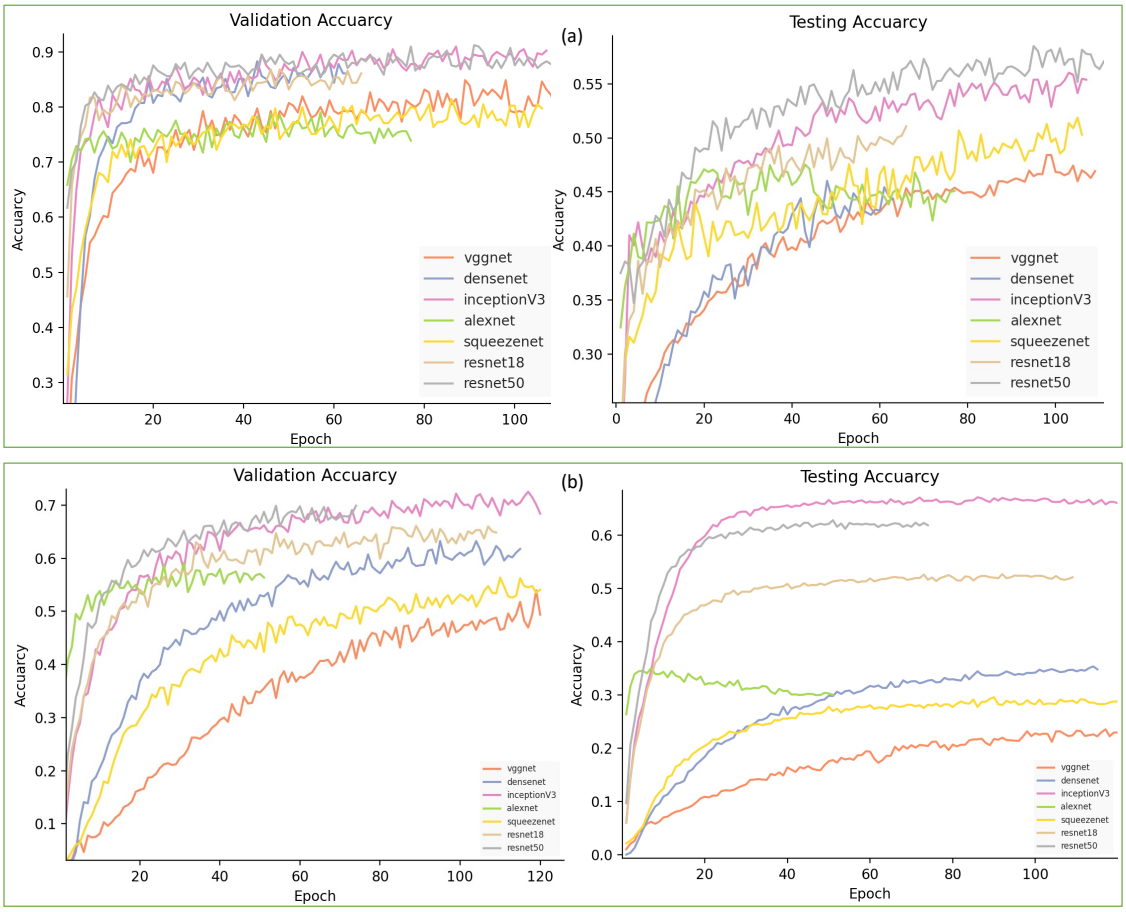


Figure 3.4: Accuracy analysis for the PACS and the Office-Home benchmarks. The length of the curves indicate the stopping points.

dataset, domain generalisation methods have clearly higher validation and target accuracy compared to domain-specific methods and VREx shows the best performance compared to the others. In the case of domain-specific models, InceptionV3 performs better than the others. Similarly, for the Office-Home dataset, both types of models have identical behaviour.

In the case of domain-specific models, they perform close to the performance levels of the domain generalisation methods for the PACS dataset but we need to

consider that during the training, they use a single domain and are tested on another single domain and results on the remaining domains will be different.

On the other hand, the Office-Home benchmark has more complex tasks than PACS therefore domain-specific models perform comparatively poorly. From this we can conclude that with more variation in tasks and domains, the performance of conventional deep learning methods is not stable and fluctuates rapidly.

Even though we try to explain DG with the help of accuracy metrics, there is no absolute way to measure the performance of domain generalisation. For example, sometimes, a model can have low scores during the training and validation but that model can still have better generalisation because the model will be more stable towards unseen data domains.

The poor performance of domain-specific models, indicated in blue text in Table 3.3, can largely be attributed to their reliance on CNNs alone. Unlike domain-generalised models, which incorporate advanced mechanisms such as adversarial training and domain-invariant feature extraction, domain-specific models lack these specialised techniques tailored for handling DG tasks. The inclusion of models in Table 3.3 serves to highlight structural differences, complexity, size, and novelty among methods. This allows for a comparative analysis of the performance of simple CNN-based models (in blue text) versus more advanced CNN-based methods designed for DG. This approach helps to better understand the strengths and limitations of each type of model in addressing DG challenges.

We now present a more detailed analysis of accuracy and loss for both benchmarks. Figure 3.4 represents the validation and testing accuracy across the datasets. Figure 3.4(a) highlights the validation and testing accuracy curves for PACS and Figure 3.4(b) presents validation and testing accuracy for Office-Home. Each graph shows information for 7 conventional deep learning models. In Figure 3.4(a) Alexnet shows the lowest accuracy, Resnet50 and InceptionV3 show the highest but almost the same accuracy level, which is around 0.9 for the validation case. Moreover, in the case of testing/unseen accuracy, Alexnet and VGGNet have almost the same but lowest accuracy from among the others and Resnet50 clearly outperforms other models. From Figure 3.4(a), for the PACS benchmark, the key information which we can extract is that less deep or smaller models show low accuracy relative to larger models that give high accuracy. Therefore, we can also say that larger models have better generalisation properties for out-of-distribution datasets.

Figure 3.4(b) illustrates accuracy analysis for the Office-Home benchmark dataset and in the case of validation curves, InceptionV3 and Resnet50 show the highest results and VGGNet performs poorly. On the other hand, if we look at Table 3.3, even though Resnet50 has 0.7 and InceptionV3 0.68 validation accuracy and based upon these numbers we can not possibly say that overall Resnet50 has better generalisa-

tion. The reason behind this argument is again numbers in Table 3.3 for testing or target set which means that on the target or unseen domains it is InceptionV3 that has higher accuracy among all. Figure 3.4(b) testing accuracy curves also contains similar trends like validations curves but one vital piece of information which we can clearly see in it that the gaps between InceptionV3 and Resnet50 increased. Therefore, for Office-Home, InceptionV3 has better domain generalisation than the other models.

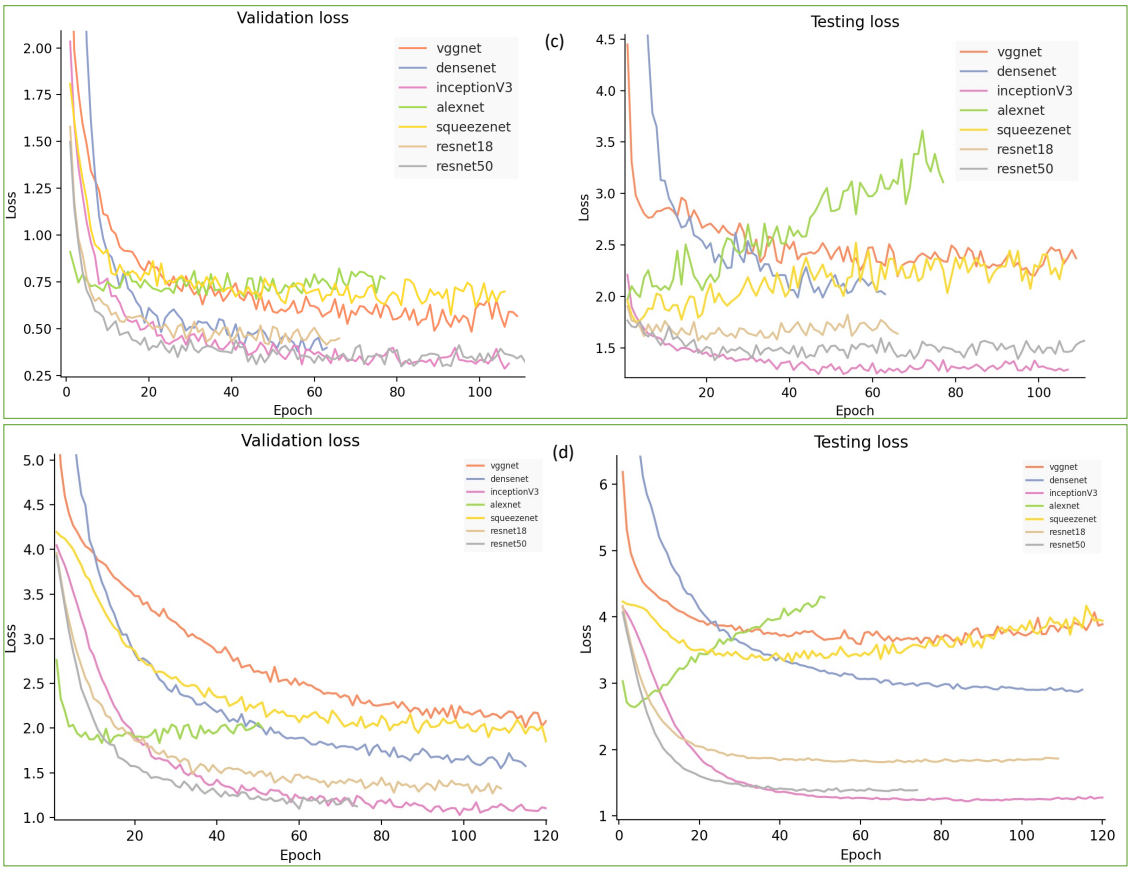


Figure 3.5: Loss analysis of conventional models for the PACS (top) and the Office-Home (bottom) benchmarks.

Figure 3.5(c) shows the validation and testing losses for the PACS benchmark and Alexnet performs worst in both cases and InceptionV3 perform best in both cases. Similar to the accuracy pattern in Office-Home, Figure 3.5(c) also shows increments in gaps between the losses of Resnet50 and InceptionV3. Figure 3.5(d) illustrates losses for Office-Home. Likewise the accuracy analysis of Office-Home, for the validation loss, VGGNet give high losses, Resnet50 and InceptionV3 are close to each other but have lowest losses. On unseen data, overall VGGNet shows relatively stable behaviour and Alexnet crosses VGGNet around 40 epochs and becomes a higher loss-giving network. Correspondingly, Resnet50 performs well at the start of analysis but around 35 epochs InceptionV3 crosses Resnet50 and becomes the lowest loss-giving network. Hence, based on losses curves, InceptionV3 has lower losses than

other networks and it supports our above-mentioned hypothesis that InceptionV3 has better domain generalisation ability than other conventional models.

3.7 Conclusions and Lessons Learned

This chapter presented performance analysis for popular benchmarks used in evaluation of domain generalisation. It also presented experiments for conventional domain-specific deep learning methods and recent domain generalisation training frameworks. The results section especially Table 3.3, Figure 3.4 and Figure 3.5 convey the vital message that domain-specific models perform poorly most of the time if we try to explain generalisation with accuracy and loss metrics.We also tried to highlight another parameter in the graphs in Figure 3.4 and Figure 3.5 which show the gaps between validation and testing accuracy. Higher gaps mean the target model has poor domain generalisation and lower gaps mean comparatively higher generalisation.

Another outcome which we can extract from the findings is that even domain-specific models perform less effectively when we compare them with domain generalisation frameworks but larger models have better domain generalisation. For example, in Table 3.3 ResNet50 has the best domain generalisation results for both benchmarks including PACS and Office-Home compared to other domain-specific models in blue colour. Furthermore, models having skip connection like ResNets and DenseNets are better for domain generalisation compared to models without skip connections like AlexNet and VGGNets. Therefore, it is safe to say that larger models, with skip connection have better DG compared to models which do not have.

The chapter presents important information in the form of a summary of the performances of various vision-based machine learning tools and connects these results with the emerging areas of domain generalisation and domain adaptation. The two base pipelines presented help us to understand that for the PACS and Office-Home benchmarks, domain-specific methods perform poorly.

The chapter demonstrates that in the field of supervised learning, domain generalised learning is better than domain-specific learning for some kinds of benchmarks. Our evaluation is performed on relatively complex benchmarks and by determining their accuracy, we try to explain generalisation.

Other ways for measuring domain generalisation have been proposed in the literature like measuring the gap between source and target domains, therefore, to have better understanding in Chapter 5 we will also introduce gap metrics. In Chapter 5 we will extend our experiments to cover attention-based vision transformers as it would be insightful to introduce an attention mechanism for such benchmarks.

Furthermore, this chapter, Chapter 3 has implemented and investigated traditional DL models and some of the very latest domain generalised models but we did not explore transformer-based methods for domain generalisation therefore, in Chapter 4 we will first learn about vision transformers and then Chapter 5 will focus on the implementation of vision transformers on domain generalisation benchmarks. Meanwhile our present results are for benchmarks which have 4 domains, and as next steps we will analyse a comparatively larger benchmark in Chapter 5 which is known as DomainNet [209] and which has 6 domains and 345 categories of objects. This means that we will increase the number of domains by using benchmarks like DomainNet[209] which has 345 classes.

In summary, this chapter 3 {explains how a pipeline of two different types of algorithm can help us to a draw a comparison study. This chapter is our first contribution to work on our hypothesis ($H\phi$) and according to which domain generalised methods (dynamic learning) are better than domain-specific (static learning). Here we focus on sub-hypotheses H1 and H2. The results describe that the domain generalisation study is totally justified as the outcomes are extremely insightful. This also demonstrates that domain generalised models have better accuracy then domain-specific models and which also convey an important message that domain generalised methods are better for handling generalisation problems. Additionally, RQ1 provides the motivation to conduct an investigation into various methods to measure the DG of any model and encourages us to consider accuracy for initial experiments which further helps us to work on RQ2. Our research in this chapter has answered sub-hypotheses H1 and H2 by contributing that DG study is justified and domain generalised methods have better and faster performance than domain-specific. By presenting this research in this chapter, we also noticed that transformers (which are backbone of current form of modern AI) based methods were missing from the literature to work on DG where one can introduce an attention mechanism to solve DG issues. Therefore, the next chapter 4 of thesis will mainly focus on different types of vision transformers and their operation.

Chapter 4

An Overview of Vision Transformers and Related Methods

The previous chapter, Chapter 3, tried to address the first research question in the thesis (RQ1) where it explains the measurement of DG with the help of the accuracy metric and it also provides a unique comparison between the traditional domain-specific DL based models with current domain-generalised DL models and the results are shown in Table 3.2. As a recap of what RQ1 says, the literature is missing an absolute measurement method for DG. Therefore, for our initial experiments we considered accuracy as a measurement metric. Moreover, Chapter 3 also focused on sub hypothesis (H1) where it first describes the general problem of DG with different available benchmarks. H1 says that a general study for DG should be conducted which reflects the behaviour of both types of model including domain-specific and domain-generalised. The results of the initial experiments shown in Table 3.2 highlight that domain-generalised learning methods could perform better than domain-specific learning methods especially when we have to solve DG or OOD related challenges.

The chapter, Chapter 3, addresses the important question of why domain-specific models are not able to perform as well as domain-generalised models which means another exploration into factors or reasons behind this phenomenon is needed. Prior works highlight that such models which learn spurious correlations by memorisation of racial biases [210], textual statistics [211], and object backgrounds [212] could be among the factors blocking the achievement of a stable generalised model for OOD challenges. Moreover, our previous Chapter 3 and paper [177] illustrate the essential parameters for DG including dataset types, architectures of models, and selection criteria. However, this chapter also unifies other factors which could be critical for

better DG. For instance, our previous research did not address transformer-based models for vision DG benchmarks. Therefore, before going into experimental and implementation details, it is important to understand the basics of the transformer architecture for vision-related datasets known as vision transformers.

The main motivation behind the selection and study of vision transformers is those factors which are highlighted in the literature. For example, self-supervised learning improves domain generalisation by learning generalised features, which have been proved in the following papers [213, 214]. It is stated there that removing the background or textural information from images can improve the DG [215] which is related to the denoising ability of a model. The article in [216] demonstrates that training using adversarial data augmentation on a single domain is sufficient to improve DG. The work in [217, 218] promotes such representations of data that ignore texture and focus on shape which also increases the DG. Moreover, the results from research in our previous chapter 3 also points out that larger models are better for DG. Therefore, vision transformers are the models which seem to have most of essential factors and thus check more boxes. Hence, our hypothesis is that exploring vision transformers will improve the DG for OOD benchmarks.

This chapter will describe the working of various vision transformers including ViT [38], ViTMAE [45], ViTMSN [219], DeiT [41], DINO [220], SwinTransformer [42], PVT [221], and BEIT [39] and we will also explain in what ways they are different from each other. The chapter will illustrate such models which combine vision transformers and CNNs to get the best advantages from both types of architectures. In the end, the chapter also emphasises a combination of vision and language problems in one unified model which is known as VLM where we will explain the basic workings of VLM with learning strategies and datasets for VLM.

4.1 Vision Transformers

This section explains detailed working of ViT [38] especially how the basic components play their role in ViT [38] methods like patch embedding, Multi-Head Self Attention Layer (MHSA), positional encoding, classification head, and training and performance. Then, this section will explain different variations of ViT [38] like ViTMAE [45], and ViTMSN [219]. In addition, this section explains the operation of vision transformers like SwinTransformer [42], DINO [220], BEIT [39], DeiT [41], and PVT [221].

4.1.1 Overview of the ViT Model

Just like the simple transformer architecture, which we have described earlier in Section 2.7.8 of the previous chapter, the components of ViT [38] are the following:

1. **Patch Embedding**

   - **Image patches:** An input image ($\mathbf{x} \in \mathbb{R}^{H \times W \times C}$) is divided into fixed-size patches, typically ($P \times P$) pixels.
   - **Flattening:** Each patch is flattened into a vector ($\mathbf{x}_p \in \mathbb{R}P2 \cdot C$)
   - **Linear Embedding:** These vectors are then linearly embedded into a lower-dimensional space using a trainable linear projection ($\mathbf{E} \in \mathbb{R}(P2 \cdot C) \times D$), creating a sequence of patch embeddings: $\mathbf{z}_0 = [\mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \ldots; \mathbf{x}_p^N\mathbf{E}] + \mathbf{E}_{pos}$ In this equation ($\mathbf{x}_p^i$) represents the (i)th image patch. ($\mathbf{E}$) is the trainable linear projection matrix. ($\mathbf{E}_{pos}$) is the positional encoding added to the patch embeddings.

2. **Transformer Architecture**

   - **Standard Transformer:** The sequence of patch embeddings ($\mathbf{z}_0$) is fed into a standard transformer block. The major part of transformer block for images also consists of a MHSA.
   - **Multi-Head Self Attention Layer (MHSA)**
     - **Self-Attention Mechanism:** The self-attention mechanism computes the attention scores between different patches of the image. For each patch, it calculates three vectors: Query ($\mathbf{Q}$), Key ($\mathbf{K}$), and Value ($\mathbf{V}$). The attention score is computed as:
       $\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$
       where (d_k) is the dimensionality of the key vectors.
     - **Multi-Head Mechanism:** Instead of computing a single attention score, the MHSA layer splits the Query, Key, and Value vectors into multiple heads. Each head performs the self-attention operation independently, allowing the model to capture different types of relationships between patches. The outputs of all heads are then concatenated and linearly transformed:
       $\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \text{head}_2, \ldots, \text{head}_h)\mathbf{W}^O$
       where each head is computed as:
       $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$

       Here, ($\mathbf{W}_i^Q$), ($\mathbf{W}_i^K$), and ($\mathbf{W}_i^V$) are the projection matrices for the (i)th head, and ($\mathbf{W}^O$) is the output projection matrix.

In the Vision Transformer, the MHSA processes the sequence of patch embeddings and by looking at the different regions of interests, it helps the model to understand the complex patterns and relationships within the image. Having multiple heads improves the efficiency using parallel processing. Furthermore, by capturing various projection and perspectives of an image, the MHSA enhances the model's ability to recognise complex details and patterns.

3. **Positional Encoding Adding Positional Information:** Positional encodings ($\mathbf{E}_{pos}$) are added to the patch embeddings to retain spatial information. These encodings can be either learned or fixed.

4. **Classification Head**

   - **CLS Token:** A special classification token (CLS) is added to the sequence of patch embeddings.
   - **Final Layer:** After passing through the Transformer layers, the output corresponding to the CLS token is used for classification:
     $\mathbf{y} = \text{MLP}(\mathbf{z}_{\text{CLS}})$
     where ($\mathbf{z}_{\text{CLS}}$) is the output embedding of the CLS token and MLP is a multi-layer perceptron.

5. **Training & Performance**

   - **Large Datasets:** The ViT [38] requires large-scale datasets for training in order to achieve state-of-the-art performance.
   - **Pre-training and Fine-tuning:** The model is often pre-trained on huge datasets and then fine-tuned on a smaller or task-specific dataset.
   - **Performance:** ViT [38] has shown to achieve state-of-the-art results on various image recognition benchmarks, especially when trained on large-scale datasets like ImageNet [26].

To put it simply, the ViT [38] model divides an image into patches, which are then flattened to yield lower-dimensional linear embeddings. Subsequently, it incorporates positional embeddings and feeds the sequence into a conventional transformer encoder. The model is fully supervised on a large dataset to pre-train with image labels, and it is then fine-tuned on a downstream dataset for image classification.

Figure 4.1(a) gives an overview of the ViT [38] method and explains the blockwise understanding of model. The figure is derived from an original paper and modified according to our requirements. The input image is from the PACS dataset [204]. Similarly, ViT also has different versions in the literature which actually extended the work of ViT in terms of efficiency, resources, computationally or structurally.

4.1.2 Workings of Masked Autoencoder-Based ViT (ViT-MAE)

A state-of-the-art self-supervised learning vision transformer technique known as Masked Autoencoder (MAE) is presented in the publication "Masked Autoencoders Are Scalable Vision Learners" [45], published in 2022. The main concept here is to train the model to reconstruct the missing areas of the input image by masking a significant amount of it. This approach forces the model to learn meaningful representations from the visible parts of the image.

How the Masked Autoencoder (MAE) Works. The main part of the masked autoencoder consists of three components including input masking, asymmetric encoder-decoder architecture, and reconstruction. In **input masking**, images are divided into patches like ViT [38] and then a large proportion of these patches are randomly masked out. The **Asymmetric encoder-decoder architecture** has an encoder and a decoder based structure where the encoder utilisez the visible or remaining patches (such patches without mask tokens) which increases the efficiency by reducing the computational complexity of the model. Contrarily, a lightweight and shallow decoder reconstructs the original image from the latent representation and the mask tokens. In the **reconstruction** part, the model is trained to predict missing pixels in each of the masked patches and this self-supervised method forces the model to learn robust visual representations.

- Training is accelerated by three times or more when MAE processes only the visible patches in the encoder, hence reducing computational complexity.
- Through the process of reconstructing the missing patches, MAE creates more resilient and significant visual depictions that smoothly transition into later tasks.
- Better performance for applications like image classification, object recognition, and semantic segmentation might be achieved by expanding MAE to larger models and datasets.
- MAE beats other self-supervised systems, including supervised pre-training, on a range of benchmarks.

The architectural details of ViTMAE is shown in Figure 4.1(b) where the input image is modified and masked into the original image as described in the paper [45]. Moreover, the encoder-decoder architecture predicts the missing patches to reconstruct the original input image.

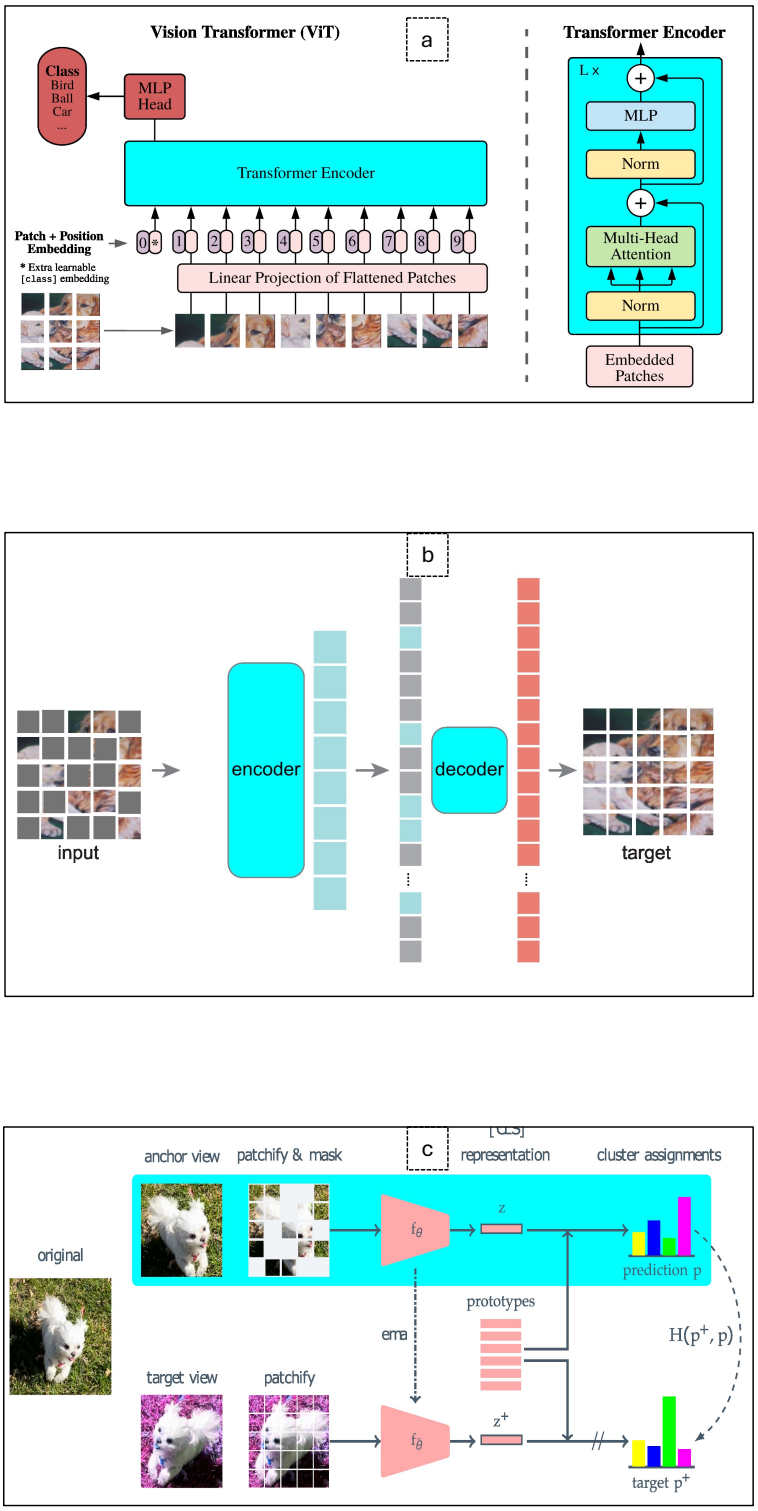


Figure 4.1: Original image taken from three articles [38, 45, 219] and modified according to our requirements. Figure 4.1(a) presents the overview diagram for the first ViT model. Figure 4.1(b) denotes the basic structural overview for ViTMAE which is a basic encoder-decoder architecture for ViT, Figure 4.1(c) describes the architectural diagram for ViTMSN.

4.1.3 Masked Siamese Network-Based ViT (ViTMSN)

Masked Siamese Networks (MSN) [219] is a new self-supervised learning framework that is introduced in the paper "Masked Siamese Networks for Label-Efficient Learning" which is similar to ViTMAE. The goal of this approach is to build strong image representations with little or no labelled input. Ensuring that the representation of an image view with randomly masked patches matches the representation of the original unmasked image is the main principle. Figure 4.1(c) shows a basic overview of the ViTMSN method where the figure is modified from the original paper [219].

Self-supervised learning, the Siamense network architecture, representation matching, and scalability are the key components of MSN, which we now expand on.

1. **Self-Supervised Learning:** By projecting the masked regions of the image, the model gains knowledge from unlabelled data. Without needing pixel- or token-level reconstruction, this method makes use of the concept of mask-denoising.

2. **Siamense Network Architecture:** By using two similar networks to interpret two distinct perspectives of the same image, the framework creates a Siamese network. There are areas in one view that are randomly masked, and unmasked in the other. The Siamese network is trained to predict if the masked patches are successfully reconstructed or not.

3. **Representation Matching:** The model matches the representation of the masked view to the representation of the unmasked view. This strategy helps the model to learn high-level semantic features that are useful for downstream tasks like classification or object detection etc.

4. **Scalability:** The method is particularly scalable when using ViT as a backbone because the network only processes the unmasked patches. This reduces computational complexity and increases productivity.

Significance of ViTMSN over ViT: In term of **efficiency**, as we know MSN processes only the visible patches in the masked view, reducing computational complexity. However, ViT uses the entire image, which can be computationally intensive. **Robust Representation:** MSN learns robust representations by matching masked and unmasked views, which generalise well to downstream tasks. On the other hand, ViT learns representations from the entire image which may require more labelled data for effective training. **Performance:** ViTMSN achieves competitive performance on low-shot image classification tasks with minimal labelled data, MSN sets new benchmarks for self-supervised learning on datasets like ImageNet-1K. Sim-

ilarly, ViT also achieves state-of-the-art performance on various benchmarks but typically requires more labelled data.

4.1.4 Understanding Distillation Based Vision Transformers like DINO & DeiT

Distillation is a process in which a "student" model learns to copy the behaviour of a "teacher" model, which is usually a more sophisticated or high-performing model. The student model seeks to attain comparable performance using a simpler or more efficient architecture.

Self-Distillation with no labels is a distillation variation in which the student and teacher models are the same and no labelled data is required during the training process. In this case, the model learns from diverse augmentation perspectives of the same input data.

DINO

The study published as "Emerging Properties in Self-Supervised Vision Transformers" investigates the capabilities of ViT taught with self-supervised learning approaches [220]. The authors show that self-supervised ViT have emergent qualities such as excellent performance on downstream tasks, resilience to distribution shifts, and improved representation learning. The research emphasises the efficacy of a self-distillation strategy without labels and achieves outstanding results across a variety of benchmarks. By using this method, a new self supervised learning ViT model with self distillation is referred to as DINO.

Architectural Details of DINO

1. **Vision Transformer ViT Backbone:** DINO [220] improves on the typical ViT architecture which divides an image into fixed-size patches that are linearly embedded into tokens. The transformer encoder then processes these tokens.

2. **Self-Supervised Learning (SSL):** The emphasis here is on training the ViT using self-supervised learning, which allows the model to learn representations from the data without the need for annotated labels.

3. **Self-Distillation without Labels:** The main method introduced here is self-distillation without labels, which allows the model to learn robust representations by distilling knowledge from various augmentations of the same image.

For the significance of the DINO model, it is vital to notice that conventional ViT is supervised and strongly reliant on large-scale labelled datasets. In contrast, the self-supervised ViT presented in this research learns representations from unlabelled data via a self-distillation strategy which is shown in Figure 4.2(b). While supervised ViT perform well on the tasks for which they are trained, they may not generalise well to new tasks or distribution shifts. However, the self-supervised ViT improved generalisation and resilience. DINO is a less data hungry model compared to others ViT.

DeiT

Similar to the DINO model, the paper DeiT "Training data-efficient image transformers & distillation through attention" [41] also uses knowledge distillation and introduces approaches that increases the data efficiency and performance of traditional ViT. The authors suggest a new distillation strategy that uses "attention processes" to improve the learning process by shifting knowledge from a teacher model to a student model. This distillation through attention improves performance dramatically, particularly in data-scarce settings. The research uses numerous trials to illustrate the usefulness of this approach producing results that outperform standard ViT.

Architectural Details of DeiT

1. **Data-Efficient Vision Transformers DeiT:** The proposed method which is known as DeiT [41] it modifies the standard ViT architecture to enhance data efficiency. This entails including a distillation token with the class token in the transformer.

2. **Distillation Through Attention:** A unique distillation approach is developed that uses the attention processes inside the transformer architecture to transfer knowledge. This strategy increases the learning efficiency and representation quality of the student model. Unlike traditional distillation methods, this novel approach uses the internal attention maps of the teacher model to guide the student model and produces more effective and refined learning.

3. **Tokens in DeiT:** DeiT [41] has two type of token in the architecture one is class token like ViT and the other is a distillation token. Similar to ViT, a class token is used for the image classification task. Comparing that to that a distillation token is contribution by the authors which is a novel contribution that helps the teacher model to transfer information to the student model during training.

4. **Training Process:** The model is trained using both labelled data and the outputs from the teacher model and it ends up increasing the ability of the student model to learn effectively from limited data.

To illustrate the significance of DeiT the traditional ViT models require large-scale labelled datasets to function well whereas for DeiT distillation via attention produces equivalent or higher performance with substantially less labelled data. ViT does not offer a mechanism for utilising pre-trained teacher models. In contrast, DeiT includes a distillation token and uses attention-based distillation to successfully transfer information from a teaching model. While DeiT and ViT use the same fundamental transformer architecture yet the inclusion of distillation-specific components (distillation token and attention-based distillation) in DeiT adds a layer of complexity targeted at enhancing data efficiency and improving speed [41].

Distillation Through Attention and new distillation tokens are novel contributions of this method therefore, it is important to understand what that means and how it actually works which is shown in Figure 4.2(a). Distillation via attention is the technique of employing attention mappings from a teacher model to assist the learning of a student model. This method assures that the student model not only duplicates the behaviour/output of the teacher model but also learns to focus on comparable parts of the input data. This method has two distillation techniques including soft distillation and hard label distillation. In the **soft distillation** strategy the student model is trained to match the probability distribution of output logits of the teacher model. It offers a milder kind of supervision than rigid labelling. The loss function for soft distillation is defined as:

$$\mathcal{L}_{\text{soft}} = \text{KL}(p_{\text{teacher}} \parallel p_{\text{student}})$$

where KL denotes the Kullback-Leibler divergence, p_{teacher} is the probability distribution from the teacher model, and p_{student} is the probability distribution from the student model.

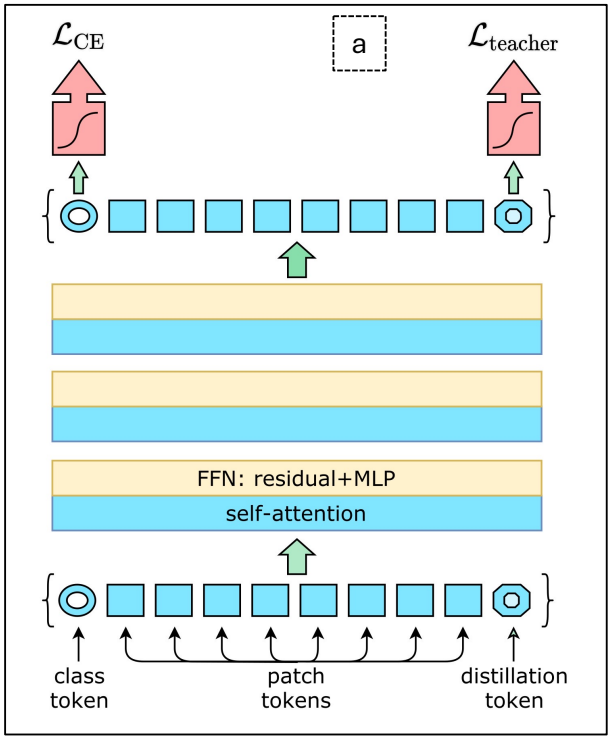
In the **hard label distillation** strategy it involves training the student model to match the dataset's hard labels (ground truth labels) by implementing a cross-entropy loss. The loss function for hard-label distillation is the following:

$$\mathcal{L}_{\text{hard}} = \mathcal{H}(y, p_{\text{student}})$$

where $\mathcal{H}$ represents the cross-entropy loss, y represents the ground truth labels. Hence, the final loss function is the combination of both types of attention which could be described as:

$$\mathcal{L} = \alpha\mathcal{L}_{\text{hard}} + (1 - \alpha)\mathcal{L}_{\text{soft}}$$

where α is a weighting factor balancing the contributions of hard-label and soft distillation losses. To have better understanding, Figure 4.2(a) contains a graphical explanation of the DeiT model.



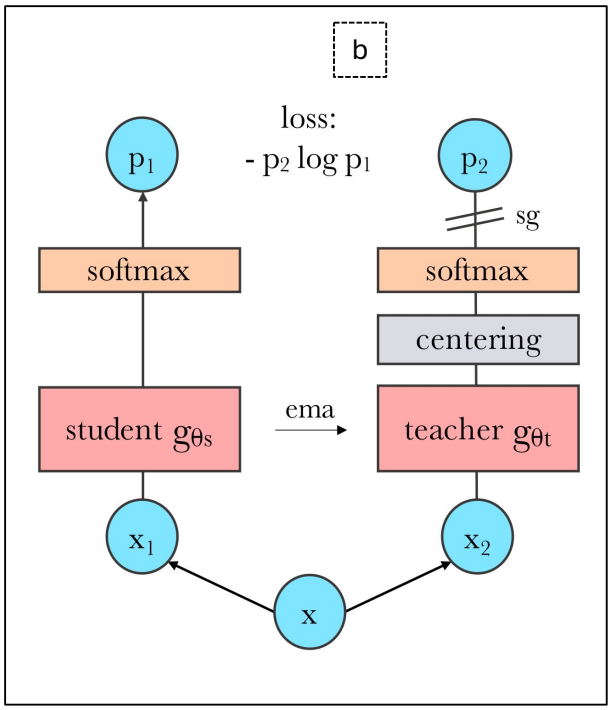


Figure 4.2: This figure summarises both distillation based methods DeiT [41] (Figure 4.2(a)) and DINO [220] (Figure 4.2(b)) into one figure where original images are taken from [41, 220] and modified.

4.1.5 SwinTransformer

The SwinTransformer aims to overcome the limitations of traditional ViT by introducing an hierarchical architecture for computing visual representations using a shifted windowing scheme. This model achieves state-of-the-art performance on various benchmarks, demonstrating its effectiveness in dense prediction tasks such as object detection and semantic segmentation [42]. Figure 4.3(a) shows the overview diagram of SwinTransformer.

Architectural Details

1. **Hierarchical Design:** Unlike ViT, which analyses images which are at a fixed resolution, SwinTransformer takes an hierarchical approach to produce different resolutions. It starts with small patches that gradually blend into bigger ones. This enables the model to generate multi-scale feature representations effectively.

2. **Shifted Window Mechanism:** The main novelty here is the shifting window concept. Traditional window-based transformers may struggle to capture cross-window relationships. The SwinTransformer establishes links between distinct areas by moving the windows between successive layers, hence improving the model's ability to grasp global context.

3. **Local Window Attention:** Within each window, typical multi-head self-attention is used, but the computational cost is greatly reduced due to window partitioning. This concentrated attention mechanism balances efficiency with performance.

4. **Patch Merging:** Patch merging layers maintain the hierarchical nature by gradually decreasing the number of tokens (patches) while increasing feature dimensions. This allows the model to process high-resolution inputs using reasonable computational resources.

Advantages of Using SwinTransformer: SwinTransformer's hierarchical architecture and local window attention minimise computational cost significantly when compared to non-hierarchical ViT models, making it more suitable for high-resolution vision tasks. SwinTransformer solves the problem of fixed grid attention in regular ViT by using the shifted window method, effectively capturing cross-window dependencies and enhancing performance on dense prediction tasks. The model's adaptability is demonstrated by its ability to be used on a variety of vision tasks, including classification, object detection, and semantic segmentation.

ViT employs a simple transformer model applied to fixed-size and non-overlapping patches from images which results in a significant increase in computing costs for large images. ViT performed well on image classification tasks but failed on dense prediction tasks like object detection and segmentation. For the ViT model, the computational cost increases quadratically with image size beacuse of the global self-attention mechanism. Conversely, SwinTransformer minimises complexity by employing local self-attention within shifting windows which yields greater efficiency for high-resolution images [42].

4.1.6 PVT

The Pyramid Vision Transformer PVT method [221] described in the paper "Pyramid Vision Transformer: A Versatile Backbone for Dense Prediction without Convolutions" is another state-of-the-art vision transformer architecture for dense prediction tasks including object recognition and semantic segmentation, similar to SwinTransformer. PVT overcomes the constraints of classic ViT, which struggle with high-resolution image processing and dense predictions because of their limited patch size and quadratic complexity in patch number. The PVT achieves higher performance by integrating a pyramid structure with a feature hierarchy providing an adaptable and strong backbone for many vision applications [221].

Architectural Details The PVT method introduces several architectural innovations that enhance its performance and efficiency which can be seen in Figure 4.3(b).

1. **Pyramid Structure:** PVT uses a pyramid structure to create multi-scale feature representations. This enables the model to detect both small and coarse visual information, which is critical for dense prediction tasks.

2. **Feature Pyramid for Transformer:** The architecture of the transformer processes visual information at many scales by using a feature pyramid network (FPN)-inspired design but without convolutions. This architecture helps in properly collecting contextual data from various resolutions.

3. **Transformer Encoder:** PVT employs an hierarchical transformer encoder with decreasing resolution at each level. The input features are processed at each level using typical MHSA and feed-forward networks.

4. **Efficient Attention Mechanism:** To handle computational complexity, PVT applies a spatial reduction attention (SRA) mechanism which decreases the spatial dimensions of attention maps like the pooling layer in CNNs and considerably reducing computational cost while maintaining performance.

To get a deeper view of this method, Figure 4.3(b) focus on these aspects

4.1.7 BEIT

By reviewing and attempting to understand different available vision transformer models from the literature, we have extracted a number of unique factors which is our motivation to use BEIT for DG in this thesis. For instance, models with transformers have a better understanding of an image than CNNs because of having a more global view/information of the image which could lead to better chances for DG. This can be seen in the ViT paper [38]. We have observed that models with self-attention including DeiT [41] and SwinTransformer [42], and models based on self-supervised learning including DINO [220], ViTMAE [45] and ViTMSN [219] are more robust when shifting domain. In addition, we also observed that models with denoising capability like ViTMAE [45] and ViTMSN‘[219] have reconstruction potential during the training which could improve the generalisation property of a model.

As a result of the above factors, BEIT [39] is the model with the greatest number of properties essential for better DG. The work described in the paper “BEIT: BERT Pre-Training of Image Transformers” proposes Bidirectional Encoder Representation from Image Transformers, a new pre-training technique for vision transformers inspired from BERT which is a well-known model in natural language processing. BEIT uses masked image modeling (MIM) to learn visual representations from large-scale unlabelled vision datasets. The strategy outperforms earlier vision transformer models by effectively capturing visual semantics and obtains cutting-edge performance on a variety of downstream applications.

Architectural Details

BEIT presents a novel approach to pre-training vision transformers using principles from BERT. Furthermore, Figure 4.3(c) provides a preview diagram of the model's architecture, extracted from the original paper [39]. The key features of the architecture include:

1. **ViT Backbone:** BEIT is based on the ViT architecture, which divides images into fixed-size patches and applies a series of transformer layers to each patch.

2. **Discrete Visual Tokens:** BEIT uses a tokeniser to convert image patches into discrete visual tokens which are similar to subword tokens in NLP. Then, these visual tokens are used in the MIM pre-training task.

   $\mathbf{X} = \text{Tokenizer}(\text{Patchify}(\mathbf{I}))$

3. **Mask Image Modelling (MIM):** Inspired by BERT's masked language modeling, BEIT randomly masks a subset of image patches and predicts their

visual tokens. This self-supervised task assists the model in learning contextual information.

$\mathcal{L}_{\text{MIM}} = \mathbb{E}_{\mathbf{x}\sim D}\left[\sum_{i\in M} \log P(\mathbf{X}_i|\mathbf{X}_{\setminus M})\right]$

where D is the data distribution, M denotes the set of masked tokens, and $\mathbf{X}_{\setminus M}$ represents the unmasked tokens.

4. **Transformer Encoder:** The masked and visible tokens are then passed into the transformer encoder, which records the relationships between tokens. The encoder's functioning can be stated as:

   $\mathbf{Z} = \text{TransformerEncoder}(\mathbf{X}_{\setminus M})$

   where$\mathbf{Z}$ is the contextualised representation of the input tokens.

5. **Pre-training & fine-tuning:** The model is first pre-trained on large-scale unlabeled image datasets using the MIM objective and then fine-tuned for downstream tasks like image classification, object detection, and segmentation. For example, in image classification, the loss function used during fine-tuning is the cross-entropy loss:

   $\mathcal{L}_{\text{CE}} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c)$

   where C is the number of classes, y_c is the ground truth label, and $\hat{y}_c$ is the predicted probability for class C.

Working Principle

The fundamental working principle of BEIT is to pre-train a vision transformer with a self-supervised MIM objective. First, the input image is separated into non-overlapping patches, which are then turned into a series of visual tokens via a discrete tokenizer. Then, a subset of visual tokens is randomly masked and the BEIT encoder transformer processes the remaining visible tokens to create contextualised representations. In the end, the model predicts the masked visual tokens using the context provided by the visible tokens.

4.2 Vision Transformers and Convolutional Neural Networks

In this section, we will describe some models of AI which are combinations of vision transformers and CNNs. Both types of model have their own unique advantages

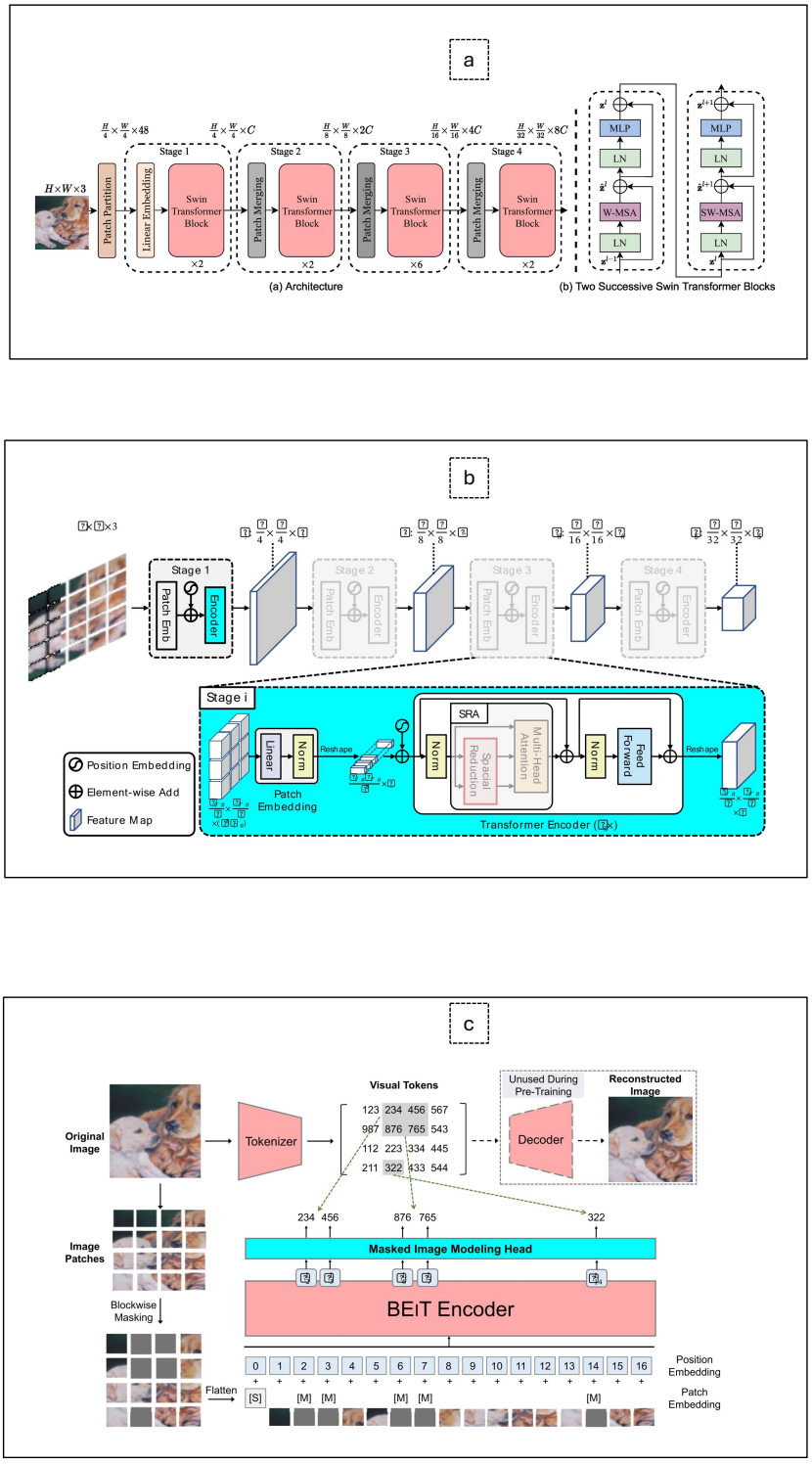


Figure 4.3: Architectures of three vision transformers including SwinTransformer [42], PVT [221] and BEIT [39]. Each image is taken an original paper [42, 221, 39] and modified according to our dataset. For instance, we used our own input images and colour schemes. Figure 4.3(a), Figure 4.3(b), and Figure 4.3(c) is for SwinTransformer [42], PVT [221] and BEIT [39] respectively.

and limitations towards each other. For instance, global and local feature extraction, computational efficiency, and improved performance are the motivation factors behind development of such models.

W know that the convolutional operation of CNNs enables them to capture local patterns and features in images with great accuracy while ViT are excellent for capturing global context and long-term dependencies via self-attention techniques. Combining the two results in a comprehensive feature extraction approach that incorporates both local and global perspectives.

Because of their reliance on global self-attention, ViTs frequently require huge datasets and larger computational resources to train. Therefore, CNNs can minimise computational cost by first extracting significant local features which are then subsequently fed into transformers.

Hybrid models have demonstrated greater performance in a variety of visual tasks by combining the complimentary characteristics of CNNs and transformers, which leads to better generalisation, robustness, and accuracy. We used one such kind of transformer named LeViT [222] for our feasibility study as described in the next chapter, Chapter 5. This section will briefly summarise the workings of some well-known hybrid methods includingLeViT [222], CvT [223], and CoAtNet [224].

4.2.1 LeViT

The paper "LeViT: A Vision Transformer in ConvNet's Clothing for Faster Inference" describes LeViT, a hybrid model that combines ViT with CNNs to provide efficient and fast inference in computer vision applications. LeViT combines convolutional layers with transformer blocks to improve both local and global feature extraction, reducing computing complexity and making it suited for resource-constrained situations [222].

The LeViT model uses a multi-stage processing method starting with convolutional layers to quickly collect low-level characteristics and minimise spatial dimensions. The process then moves on to transformer blocks which process these features to capture global context and long-range relationships. LeViT deploys a lightweight self-attention technique to reduce computational cost and cross-attention in order to successfully integrate information from various spatial locations and feature maps.

The architecture is specifically designed for fast inference which makes it applicable in real-world scenarios. LeViT also achieves high performance with fewer parameters than pure ViT [38]. Technical information about the architecture includes patch embedding using convolutional layers, a hierarchical framework for processing information at different resolutions, and improved attention layers to decrease computational complexity. LeViT achieves competitive accuracy on a number of image

classification benchmarks, with performance equivalent to cutting-edge models and substantially quicker inference appropriate for real-time applications.

4.2.2 CvT

In the paper, "CvT: Introducing Convolutions to Vision Transformer", the authors introduce convolutional vision transformers (CvT), a new method that incorporates convolutional operations into ViT architecture to improve image classification performance and efficiency. CvT integrates CNNs in two ways: token embedding and projection in self-attention using CNNs. By adding convolutions, CvT improves local spatial context while reducing computational complexity [223].

Convolutional token embedding replaces classic linear embedding in ViTs and allows the model to access a greater local feature extraction, whilst convolutional projection in self-attention layers increases the capacity of the model to focus on key spatial regions. This hybrid technique overcomes the disadvantages of pure ViTs such as high processing demands and inefficiencies in capturing local structures.

CvT [223] outperforms typical ViTs in terms of accuracy and efficiency across a range of benchmarks. The design of this model preserves transformer's global context modelling strengths while exploiting CNNs' local feature extraction, yielding a robust and efficient architecture for vision applications.

4.2.3 CoAtNet

The publication "CoAtNet: Marrying Convolution and Attention for All Data Sizes" describes a method named as CoAtNet, a unified architecture that successfully handles a broad variety of data sizes by combining the strengths of CNNs and ViT [224]. CoAtNet also combines convolutional layers with a self-attention mechanism in a coherent architecture and enable the model for both local feature extraction and global context mapping.

In the architecture of this hybrid technique, the beginning layers using convolutions to capture local patterns and later layers using self-attention to simulate long-term relationships. This combination helps CoAtNet to retain excellent efficiency and performance over a range of datasets, from small to large sizes. When compared to current designs such as pure CNNs and transformers, the model outperforms them in terms of accuracy and computing efficiency. CoAtNet's [224] adaptability and strong performance make it acceptable.

4.3 Vision-Language Models

Human learning is intrinsically multimodal since the integration of various senses assist for a deeper understanding and processing of new information. Recent advances in multimodal learning have unsurprisingly drawn inspiration from this natural process and as a result developed models which are capable of processing and interlinking information across multiple modalities, such as images, videos, text, audio, body gestures, facial expressions, and physiological signals.

Due to the exceptional success and the extraordinary performance of DL models like transformers in dense vision applications including classification, object detection, segmentation and more, and in textual data applications like NLP, motivated researchers to combine both type of modalities. Therefore, since 2021, there has been a growing curiosity in combining vision-language models, such as OpenAI's CLIP [48] model. These models have shown outstanding performance in complicated tasks including image captioning, text-guided image generation and modification, and visual question answering (QA). This research discipline is constantly evolving and improving its potency in zero-shot generalisation and extending its range of practical applications. Current AI models or services like ChatGPT, Claude, Copilot etc, [6, 7, 8] all are directly or indirectly inspired by CLIP like backbones.

This section address the basic working of vision-language models and their building blocks in order to present an overview of them. As vision-language models are more complicated and larger models compare other available DL models therefore, we will not include the deeper concepts here.

As we know, a vision-language model has two types of input, images and textual prompts or captions. To train a multimodal-based model the major task is to build relationships between these two data types. For instance, during training, the model learns to score the relationship between objects in the images and words/phrases in the given captions by learning concept vectors. This is done by using two types of model including a vision model and a language model. CNNs or vision transformers-based models could be used as a vision model because they are recommended for feature extraction such as shapes, colours, and objects. Similarly, for language model transformer-based models are used to understand the structure and meaning of input prompts by capturing the semantic relationships between the words. By combining these two models, VLM can align visual features with their corresponding textual descriptions.

A vision-language model typically includes three key components: an image encoder, a text encoder, and a strategy for fusing information from the two encoders. For instance, to solve the zero-shot classification task, Figure 4.4 presents an overview of the procedure of a VLM where input textual prompts go into a text

encoder and visual input image go into an image encoder. Both encoders will extract corresponding features and fuse them using various metrics like cosine similarity to generate final predictions as shown in Figure 4.4. The next section will highlight learning strategies for VLM.

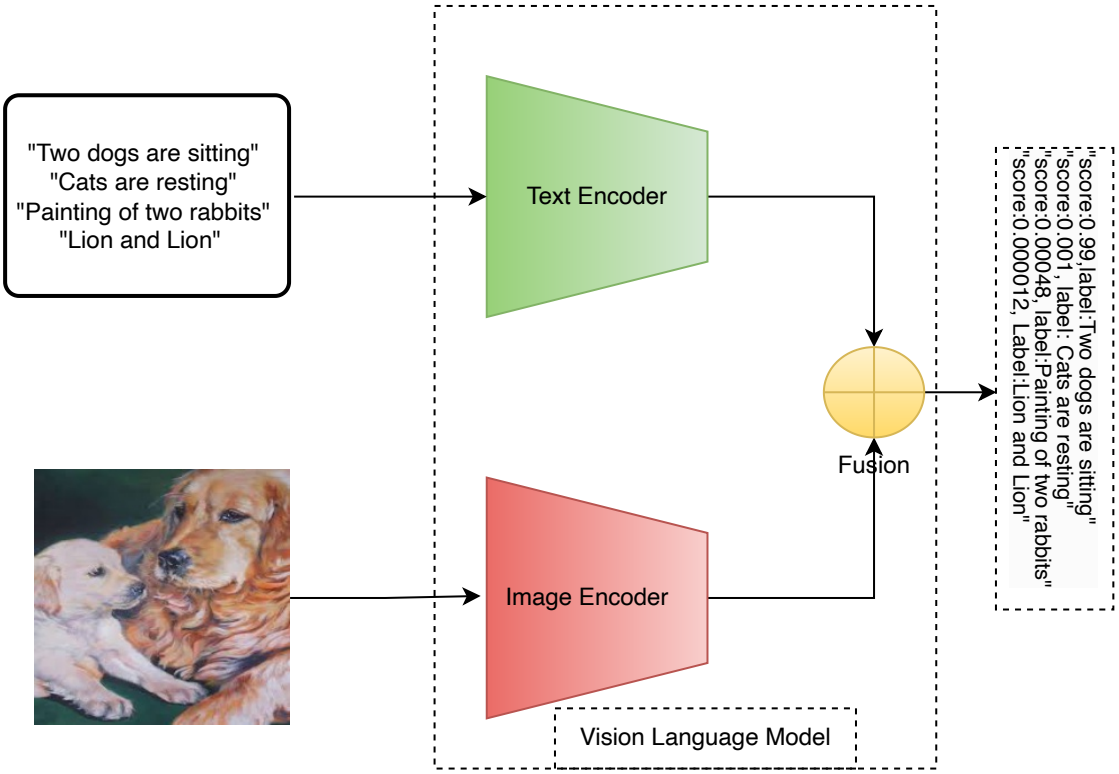


Figure 4.4: The basic working of a VLM for a simple task of zero-shot classification where prompts go into a text encoder which could be a textual transformer model and the input image will be processed by an image encoder and VLM will perform fusion of both modalities using evaluation metrics like cosine similarity.

4.3.1 Learning Strategies

This section addresses common pre-training goals and strategies for VLM which have been demonstrated to perform well in terms of performance. Therefore, we will explain methods incldluing contrastive learning, masked language modelling, multimodal fusing with cross attention, prefixLM, and no traning.

Contrastive Learning

The primary goal of contrastive learning is to learn an embedding space in which similar data points are closer together and dissimilar data points are further away. This allows the model to better analyse and categorise incoming input based on its previously learned representations. Current works in VLM such as CLIP [48],

CLOOB [225], ALIGN [226], and DeCLIP [227] bridge the vision and language modalities by learning a text and image encoder together with a contrastive loss by using large datasets consisting of [image, caption] pairs.

Contrastive learning seeks to map input images and word pairs to the same feature space so that the distance between image-text embeddings is minimised when they match and increased when unmatched. CLIP [48] basically calculates the cosine distance between the text and image embeddings. On the other hand, methods like ALIGN [226] and DeCLIP [227] create their own distance metrics to handle noisy datasets.

In another paper, LiT [228] describes a straightforward strategy for fine-tuning the text encoder with the CLIP pre-training target while keeping the image encoder frozen. The authors consider this concept as a method for teaching the text encoder to better understand image embeddings and LiT [228] is more sample-efficient than CLIP. Other approaches such as FLAVA [229], employ contrastive learning and other pre-training procedures to match visual and language embeddings.

Multimodal Fusing With Cross Attention

Another method for utilising pre-trained language models (transformer) for multimodal tasks is to directly fuse visual information into the layers of the decoder of same language model using a cross-attention mechanism. This pre-training method is used by models such as VisualGPT [230], and Flamingo [231] which are trained on image captioning and visual question-answering (QA) tasks. The primary purpose of such models is to optimally balance text-generating capability and visual information.

VisualGPT models apply a visual encoder to embed images and create an embedding which then fed into the cross-attention layers of a pre-trained language decoder module to create acceptable captions. Similarly, another model FIBER [232] introduces cross-attention layers with a gating mechanism into both vision and language backbones for efficient multimodal fusing and supports different downstream tasks like image-text retrieval and open vocabulary object detection.

PrefixLM

Another learning strategy to train VLM is known as PrefixLM. Such language models which have a prefix goal to predict the next token if prefix input text is given. Similarly, vision transformers like ViT use the same concept of the prefix on images and split each image into small patches, which are then fed progressively to the model as inputs. SimVLM [233] and VirTex [234] are example of models in this type of learning strategy.

Frozen PrefixLM: One further idea related to the area of vision language models is Frozen PrefixLM where integrating vision information into a language model can be extremely successful. Implementing such a pre-trained language model that does not require any fine-tuning, it would be far more efficient. As a result, this type of method learns image embeddings that align to a frozen language model.

The Frozen [235] and ClipCap [236] models implement the Frozen PrefixLM method into their learning process. During their training, both models update parameters of the image encoder and create embeddings which further feed to frozen prefix language models like PrefixLM. Coversely, models like MAPL [237, 231] keep both the pre-trained visual encoder and the language model frozen.

Masked Language Modelling/Image-Text Matching

Table 4.1: Large scale databases available for pre-training of VLMs

Dataset Name	**Description**	**Size**	**Tasks**	**Ref**
COCO Captions	Image dataset with captions describing each image.	330K images, 5 captions per image	Image captioning, retrieval	[238]
Visual Genome	Images with region descriptions, objects, attributes, and relationships.	108K images	Object detection, scene understanding	[239]
Conceptual Captions	Images with captions sourced from web pages.	3.3M images and captions	Image caption annotations, Conceptual Captions,	[240]
Flickr30k	Images with multiple captions describing different aspects.	31K images, 5 captions per image	Image-text matching, retrieval	[241]
SBU Captions	Image-caption pairs obtained from Flickr.	1M image-caption pairs	Image captioning	[242]
YFCC100M	A diverse collection of images and videos with metadata from Flickr.	100M images	Multimodal tasks	[243]
PMD (Public Multimodal Dataset)	The dataset contains pairs from Conceptual Captions,RedCaps, COCO, WIT, etc.	70M image-text pairs	Image-text tasks	[229]
LAION-5B	Large-scale CLIP-filtered image-text pairs, gathered from various online sources.	5.85B image-text pairs	Pre-training image captions	[244]

The Masked Language Modelling (MLM) / Image-Text Matching (ITM) method utilises both approaches together and rather than matching whole image it aligns specific parts of images with text prompts which makes it effective for downstream tasks. The goal of MLM is to predict masked words by using related images if a partially masked caption is provided to MLM while in ITM if an image and text pair is given then the goal is to determine if the caption matches the image

or not. VisualBERT [245], FLAVA [229], ViLBERT [246], LXMERT [247], and BridgeTower [248] are examples of models that follow similar sittings during their training process.

No Training

In addition to the aforementioned models whic have a training component, there are also methods where different optimisation approaches are implemented to map the image and text representations using pre-trained models for both type of modalities. Unlike the previous models, this helpss the VLM to adapt for new downstream tasks without any additional training. For instance, the MaGiC model [249] uses interactive optimisation to create captions for input images which is possible by generating a CLIP-based "Magic score".

4.3.2 Vision Language Model Datasets

VLMs are usually trained with large image and text datasets with various architectures, depending on the pre-training objectives. VLMs are first pre-trained and then fine-tuned for distinct downstream tasks utilising task-specific datasets. This section gives an overview of some of the most common pre-training and downstream datasets for training and testing vision-language models.

In this thesis we divided these benchmark datasets into two tables including pre-train datasets shown in Table 4.1 and downstream task datasets shown in Table 4.2. Both tables have information related to the names of the datasets, a brief description of each, their size, tasks that they have been used for, and a reference citation to each.

Table 4.2: Datasets for downstream vision tasks which are commonly used for fine tuning of VLMs

Dataset Name	**Description**	**Size**	**Tasks**	**Ref**
VQA (Visual Question Answering)	Images with associated questions and answers.	200k images, 1.1M (image,question) pairs with 13M answers	Visual question answering	[250]
GQA (Graph Question Answering)	Real-world images with compositional questions and answers.	113k images, 22M questions	Visual question answering	[251]
RefCOCO / RefCOCO+	Images with phrases referring to objects in the image.	20k images, 142k expressions	Referring expression comprehension	[252]
Flickr30k Entities	Augmented Flickr30k with entity annotations linking phrases to image regions.	31K images, 243k coreference chains, 276k bounding boxes	Phrase localisation, visual grounding	[253]
Visual7W	Images with questions requiring different types of reasoning (what, where, how, etc.).	47k images, 327k QA, 1.31M human-generated multiple-choices & 561k object groundings	Visual question answering, reasoning	[254]
CLEVR	Synthetic images with complex questions requiring logical reasoning.	100k images, 1M QA	Compositional QA, reasoning	[255]

NLVR2	Pairs of images with natural language statements to verify.	107k examples, 29k unique sentences, 127k unique images	Visual reasoning, language verification	[256]
MSR-VTT (Microsoft Research Video to Text)	Open domain video captioning.	10K video clips, 20 categories, 200K clip-sentence pairs	Video captioning, retrieval	[257]
TRECVID	Large-scale video dataset for video information retrieval.	Multiple benchmarks annually	Video retrieval, event detection	[258, 259]
HowTo100M	Video clips with subtitles and instructional content.	136M video clips, 23k activities	Video understanding, action recognition	[260]
OKVQA (Outside Knowledge VQA)	Questions about images requiring external knowledge to answer.	14k questions	Visual question answering	[261]
TextVQA	Questions about images requiring understanding of textual information within the image.	45k questions, 28k images	Visual question answering	[262]
TextCaps	Image captions requiring understanding of text within the image.	145k image-caption pairs, 28k images	Image captioning	[263]
VizWiz	Visual question answering for blind users.	31k VQA	Visual question answering	[264]

LSVTD (Large Scale Video-Text Dataset)	Video dataset with textual descriptions.	129 video clips	Video understanding with diverse scenarios, video captioning	[265]
WebVid	Large-scale video dataset with captions sourced from the web.	2.5M videos	Video understanding, video captioning	[266]
Hateful Memes	Multimodal dataset for detecting hateful content in memes.	10K memes	Hate speech detection	[267]
SNLI-VE (SNLI-Visual Entailment)	Pairs of images with natural language statements to verify entailment.	565k image-sentence pairs	Visual entailment, reasoning	[268]
Winoground	Evaluate the ability of VLM to conduct visio-linguistic compositional reasoning.	1.6k image-text pairs	Compositional reasoning	[269]

These datasets collectively provide a solid foundation for training and evaluating vision-language models across a wide range of tasks, allowing for advances in multimodal content understanding and generation.

4.4 Conclusions and Lessons Learned

This chapter covered the developing topics of vision transformers and VLM with detailed investigations into some aspects. The main focus of the chapter was to understand the basic workings of various transformer models and the motivation behind the need for exploring different vision transformer models for DG. The chapter concluded that factors like self-supervised learning [213, 214], removing the background or textural information from images (denoising) [215], adversarial data augmentation [216], ignoring texture and focus on shape [217, 218], larger models 3, and multi-head self-attention [41], SwinTransformer [42] can each effect and improve DG capabilities.

As a result of this conclusion, vision transformers are the models which have most of the factors that we wish to include in this thesis. Hence, our hypothesis is that using vision transformers will improve the DG for OOD benchmarks. To address this research question and sub-hypotheses it was important to present and to discuss these methods. In the next chapter we will report on the implementation of vision transformers for DG benchmarks.

Chapter 5

Domain Generalisation with Bidirectional Encoder Representations from a Vision Transformer

The previous chapter 4 explained the foundations of different transformer-based models for vision applications and it also provided background fundamentals behind the motivation for exploring vision transformers for domain generalisation. It also provided factors which could possibly impact the effectiveness of domain generalisation (DG). In terms of research questions and sub hypothesis within the thesis, the previous chapter, Chapter 4, helped to construct the concepts of the latest transformer-based domain generalised models.

In this chapter, Chapter 5, we will train our own transformers after performing an initial feasibility study for DG. As we know that most of the vision transformers need large scale datasets and computing resources for training, therefore in our case the training will be achieved using fine-tuning on selected DG benchmarks. After considering the factors which we learned in the previous chapter, the results of our trained models for each benchmark have outperformed the previously trained models which we have highlighted in Section 3.3. Furthermore, this chapter focuses on RQ3 which explored whether domain-generalised learning can have better understanding by having better accuracy. This chapter can also focus towards H1 and H2 which are about better domain generalisation, less distracted models with better accuracy and being more robust to domain shifting.

In this chapter we will address characteristics and methods which can help us to achieve a better domain generalised model as an introduction part of the chapter. Then follows a brief related work section on methods even though we have already

presented a comprehensive chapter 4 which gave details on various similar methods. Rather than blindly starting an exploration study of vision transformers, we first conducted a feasibility study where we run inference of already trained models on OOD benchamrks which is presented in Section 5.3 and based upon the results we select the BEIT model for further exploration. The methodology section has detailed information on the model development. The results section of the chapter highlights key improvements in term of DG for overall test benchmarks. Similarly, the results section also shows performance per domain analysis and a comparison study across various domains. This chapter tries to go deeper into the reasons and to investigate the possible factors like why we could have better DG, by exploring the latent attention space. Finally, attention heat maps are presented which describe the behaviour of trained models visually.

As we already stated, domain generalisation involves pooling knowledge from a number of source domain(s) into a single model that can generalise well to unseen target domain(s). Successful domain generalisation depends on three factors: dataset, network architecture, and model selection criteria. Recent research in domain generalisation has faced challenges when using deep learning models when they interact with unseen data distributions which differ from the distributions they are trained on. Issues occur when domain shifting happens and models usually perform poorly for out-of-distribution data.

In computer vision research, the progressive success of vision transformers over CNNs in simple tasks like object classification, detection and segmentation, now shows state-of-the-art results. The focus of this chapter is to perform DG inference of trained models on OOD) benchmarks using vision transformers.

We use four of the most recent vision transformer architectures including ViT [38], LeVit [222], DeiT [41], and BEIT [39] for out-of-distribution data. We explore the bidirectional encoder representation from the BEIT [39] image transformer which is a pre-training of image transformer based upon BERT. BEIT has a denoising autoencoder, and this property is a useful sign for domain generalisation and needs further investigation. Moreover, BEIT has a mask image modelling technique to train the vision transformer in a self-supervised manner which produces a self-attention mechanism for input images.

Our work exploits the fact that BEIT has three major advantages for generalisation. First, because of self-attention, it can separate objects from background noise. Second, self-supervised fine-tuning helps to perform better generalisations which ignore texture information in images and promote the shapes of objects. The third advantage is the denoising of corrupted images using MIM. For our investigation into vision transformers for generalisation we conduct experiments on three benchmarks namely PACS, Home-Office and DomainNet. Our results highlight the

significant improvements in validation and test accuracy for all and models significantly overcome the gap between Independent and Identically Distribution (IID) and OOD data.

5.1 Introduction

Computer vision in numerous scenarios completely fails to generalise when tried on OOD data. This shows when we inject randomness and speculation during prediction when trained models are tested on unseen domain distributions. Often machine learning systems rely on training distributions which make them vulnerable to use. For example, real-world systems like autonomous vehicle navigation show a huge decline in performance when interacting with even partially different conditions and settings during model inference, compared to their training distributions. Different weather [175], day vs. nighttime lighting conditions [270], poses and positioning of objects [271] are recurring reasons which affect such systems.

In robotics when introducing visual distractions to agents, some methods are known to significantly decrease performance [272]. A few possible reasons why ML models are learning poor generalisation could be racial biases, texture statistics, and object backgrounds [273]. These factors reduces a model's ability to capture causal factors related to potential variations in the data but instead fit the model to fake representations. Hence, to tackle OOD and for the progressive development of real-world ML systems, these issues have been partially addressed in the literature.

As we know, domain generalisation can be controlled by three factors: dataset types, network architectures, and model selection criteria. To overcome OOD related challenges, much work has been done including solutions like additional data collection for different domains or environments, adversarial learning, and data augmentation for the purpose of learning generalised invariances from the training domain [274, 275]. Prior research has also proposed techniques to find hyperparameters which maximise performance on an external domain by measuring relatedness [178]. In summary, invariant feature learning, data augmentation, meta learning, life long learning, and generative adversarial learning are related methods frequently found in the literature from which we can extract some highlights.

1. An extensive study in [273] indicates that larger models have greater generalisation compared to smaller architectures.

2. [276, 214, 277] show how self supervised learning helps to improve domain generalisation of a model for OOD by learning generalised and transferable features.

3. The removal of texture information also boosts domain generalisation [215].

4. Methods to ignore the texture of images and focus on shapes and ignore background noise could lead to better generalisation [218].

These pointers from the literature encourage us to explore the latest vision transformers for domain generalisation because vision transformers have most of these characteristics. The success of transformers in natural language processing motivates us to use transformers for vision based tasks [155, 38]. For instance, ViT contains more uniform representations in all the network layers, self attention creates global information during the early layers, and residual connections help to propagate features [38].

In [278] the authors introduced a teacher-student based learning strategy for efficient training on a single computer. Similarly, LeViT [222] also focused on the computational advancements by using few CNN functions in transformers and creating hybrid architectures. BEIT [279] is another state-of-the-art transformer-based method which has a denoising auto-encoder implementation to pre-train a vision transformer which we use. During pre-training, it randomly masks a few patches with a proportion of the image and feed this corrupted input to the transformer which is one of the key points to tackle in OOD.

To address issues related to OOD, this chapter investigates various aspects of DG using vision transformers. For proof of concept, we implemented a pipeline to check the OOD capability of four available pre-trained vision transformers. Originally each of these models were pre-trained and then fine-tuned on ImageNet-21k and ImageNet1k [180] respectively. We then run inference on unseen benchmarks including the ImageNet-Sketch [215], ImageNet-R (endition) [181], ImageNet Adversarial [183], and ImageNet Corrupted [182] benchmarks. This work discovers that BEIT outperforms all other vision transformers and to the best of our knowledge, properties like denoising, the self-attention mechanism, and self supervised fine-tuning could be the main reasons.

Based upon initial results, we choose BEIT for further analysis where pre-trained weights are used as feature extractors along with attention masks. We then fine-tune three separate models on three popular benchmarks for domain generalisation namely PACS, Home-Office, and DomainNet. During the training process, our method implements the *training-domain validation set*, inspired by [273] which means that 80% of data in each domain will be used for training and validation, and the remaining 20% from each domain will be combined and used for testing. We compute detailed analysis of the self-attention mechanism using attention distance metrics where we explore the latent space of models. Our method tries to explain where models which learn early global information then have better domain gen-

eralisation and where this is not effective. To measure generalisation, the results section considers metrics like accuracy, gap and precision, which show significant improvement. We also provide graphical visualisations of attention maps for OOD cases.

5.2 Related Work

In the literature many methods have been proposed to address the challenges of OOD, including collection of extra data, invariant feature learning, data augmentation, meta learning, life long learning, and generative adversarial learning. For instance, [192] is a group distributionally robust optimisation method which uses empirical risk minimisation and explains the importance of domains with error rates. Similarly [196, 197] involves invariant feature learning across domains which learns features which are invariant to external data variability. However, such methods were critcised in [280], who highlighted reasons why invariant feature representation is insufficient for domain generalisation.

Approaches based on invariant risk minimisation also exist where the network learns an optimal classifier on top of invariant feature representations [200]. A popular meta learning based method known as Meta-Learning for Domain Generalisation (MLDG) builds on model-agnostic meta learning where a meta learner is generalised on various domains. To enhance generalisation, [216] explains that training on a single domain is sufficient when using adversarial data augmentation training. All the above methods have their own strengths and weaknesses, but no single approach is best. Another concern is that they use a CNN as their backbone with pre-trained weights which means models will have a limited respective field compared to vision transformers.

Vision transformers are more appropriate for generalisation than CNNs because of factors like global understanding, handling of variable-length inputs, fewer parameters, an attention mechanism, and pre-training. The initial idea of the ViT [38] is simple to understand. First, the image's patches and positional embeddings after flattening goes to a transformer encoder where multi-head attention helps the transformer with coverage with the respective loss function. In the same way, ViT and other vision transformers including LeViT, DeiT, BEIT [38, 278, 222, 279] followed similar architectures but pre-training requires much computational resources and large datasets. Similarly, [281, 282] also investigated domain generalisation with vision transformers especially, ViT used as a backbone but in this work we are using BEIT.

5.3 OOD Inference Experiments With Available Pre-trained Vision Transformers

To conduct initial OOD experiments as proof of concept, we use pre-trained weights of four baseline vision transformers namely ViT [38], LeVit [222], DeiT [41], and BEIT [39]. These models are fine-tuned on ImageNet 2012 1K classes with 224×224 input resolution and 16×16 patch size except LeViT which has 256×256 input resolution. For performance of these models on OOD datasets, we exploits different variations of ImageNet as OOD examples namely ImageNet-Sketch [215], ImageNet-R (endition) [181], ImageNet Adversarial [183], and ImageNet Corrupted [182].

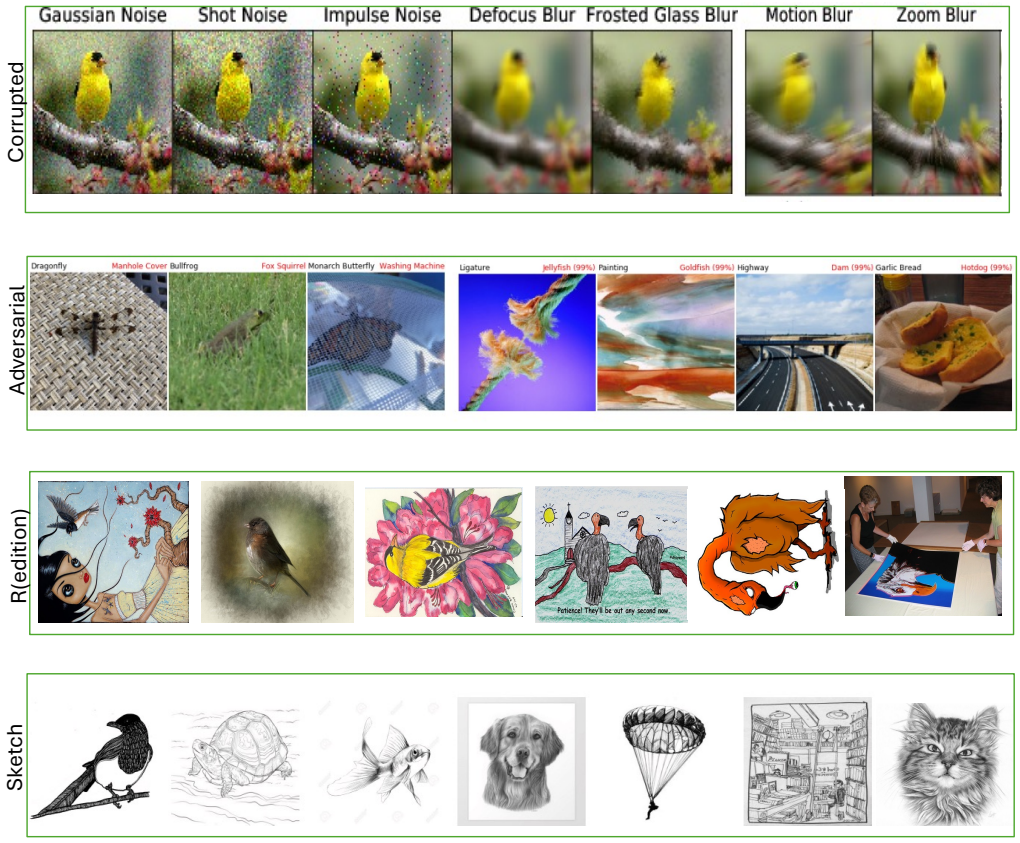


Figure 5.1: Visual definition of selected OOD benchmarks, different versions of ImageNet, to test the robustness of models.

1. **ImageNet-Sketch** dataset has 50,000 images with 1K classes, 50 images for each of the 1,000 ImageNet classes

2. **ImageNet-R (endition)** contains 30,000 image renditions for 200 ImageNet classes which is a subset of ImageNet-1K. In addition, ImageNet-R (endition) has distributions including art, cartoons, deviantart, graffiti, embroidery, graphics, origami, paintings, patterns, plastic objects, plush objects, sculptures, sketches, tattoos, toys, and video game renditions of ImageNet classes.

3. **ImageNet-adversarial** consists of adversarially filtered real-world images to fool ImageNet classifiers and it also contains 200 classes as a subset of ImageNet-1K.

4. **Imagenet Corrupted** consists of images with 75 common visual distractions and the goal was to improve and evaluate the robustness of models, it has 1,000 classes. ImageNet-Corrupted (ImageNet-C) is a dataset created to evaluate the robustness of image classification models against common corruptions and perturbations. The dataset includes 15 different types of corruption, each applied at 5 severity levels. These corruptions can be grouped into four categories: **Noise:** Gaussian noise, shot noise, impulse noise. **Blur:** Defocus blur, frosted glass blur, motion blur, zoom blur. **Weather:** Snow, frost, fog, brightness. **Digital:** Contrast, elastic transformation, pixelation, JPEG compression

To get a clearer idea about the variations in these benchmarks we have created a visual summary as Figure 5.1. Each row indicates the type of dataset and the figure also denote that each data distribution is quite different with respect to others which explains the variety for measuring the DG. Additionally, selected vision transformer models have never seen these distribution during the pre-training and fine-tuning procedures.

Table 5.1: Results of using vision transformers on OOD-related benchmarks

	ImageNet-Sketch		ImageNet-R(edition)		ImageNet Adversarial		ImageNet	
Models	Top1 Acc	Top5 Acc	Top1 Acc	Top5 Acc	Top1 Acc	Top5 Acc	Top1 Acc	Top5 Acc
ViT	35.43	57.29	32.82	47.54	12.97	30.04	78.06	94.43
LeViT	0.95	0.72	0.81	0.44	9.13	27.14	73.67	90.98
DeiT	32.58	50.21	31.04	44.42	9.97	24.31	77.95	92.56
BEIT	47.55	71.01	44.72	62.13	22.60	47.74	81.88	96.41

Table 5.1 contains 9 columns which present the top 1 and top 5 accuracy of selected transformers for four OOD-related benchmarks. The results indicate that BEIT outperforms other transforms with a notable difference in evaluation metrics for each benchmark. The main reason BEIT surpasses others are its properties including MIM with self-supervised learning of large models, a self-attention mechanism, and denoising of corrupted inputs. Thus, we selected BEIT for analysis of domain generalisation benchmarks. The next section presents the methodology in our approach.

The comparatively low performance of LeViT is due to its design priorities. LeViT is optimised for efficiency and speed, with fewer parameters and lower computational complexity which makes it much faster during inference but restricts its capacity to extract robust features, especially in difficult OOD circumstances. Furthermore, its shallow depth is defined by fewer transformer layers and attention heads compared to models such as ViT and BEIT, and this leads to poorer representation learning and generalisation. LeViT may potentially use fewer or less diversified pre-training datasets which limits its capacity to adapt to adversarial inputs or stylised data. Furthermore, because LeViT is intended for deployment on resource-constrained devices, its focus on lightweight architecture loses the capability required to perform on OOD benchmarks effectively and it reduces contextual understanding.

DeiT and ViT underperform BEIT mostly because of the lack of advanced pre-training approaches. BEIT uses Masked Image Modelling (MIM), a pre-training method inspired by BERT that allows the model to acquire richer, more robust features by reconstructing masked patches of images. In contrast, DeiT and ViT use traditional SL objectives, making their learnt representations less resilient for OOD tasks. BEIT also excels in capturing domain-invariant features, which are important for generalisation, whereas DeiT and ViT are more sensitive to domain changes. BEIT's self-supervised fine-tuning improves its capacity to adapt to a variety of datasets, unlike DeiT and ViT, which rely largely on labelled data, risking overfitting to certain distributions. Finally, the training method of BEIT prioritises high-level object features such as shapes over textures and produces improved performance on benchmarks such as ImageNet-Sketch and ImageNet-Rendition, where abstract representations are critical. However, DeiT and ViT may rely more on textural features which make them less effective in such situations.

5.4 Methodology

This methodology section has three parts including preliminary data processing, feature extraction and fine-tunning of the BEIT model and finally inference.

5.4.1 Preliminary Data Processing

Based on the numbers of domains, images, and classes, we chose to use PACS [204], Office-Home [206], and DomainNet [209] as benchmarks so that we can test the domain generalisation capability of BEIT for small, medium and large scale datasets. Further details of benchmarks can be found in [283]. We used hugging face with Pytorch in the backend processing. During the training and validation

steps, pre-processing includes image resizing, random horizontal flip, and normalisation. Similarly, testing includes re-sizing, centre cropping, and normalisation as pre-processing procedures.

Inspired by the original paper [279], we used the based version of the BEIT transformer in this research. This has 12 transformer layers with 768 hidden and 3,072 feed-forward networks. Each attention layer has 12 attention heads of size 64 and these are responsible for learning self-attention masks. Each image was divided into 14×14 patches of 16×16 pixels. BEIT is trained with 8,192 visual tokens.

5.4.2 Feature Extraction and Fine-Tuning of Models

We use the base BEIT model from the hugging face repository as a feature extractor. To extract features from input images, the model re-sizes images into $224 \times 224 \times 3$ resolution and the output features also have the same size. The model first converts images into 14×14 patches with a resolution of 16×16 for each and then flattens the patches and feeds them into the transformer with positional embeddings. The model is pre-trained and fine-tuned on ImageNet21k. After extracting features our method normalises feature maps with respect to the mean and standard deviation of images. With initial features we also extract an attention mask to ensure the self-attention mechanism of self-supervised BEIT is able to separate semantic regions and object boundaries.

After normalising features for each benchmark, we configure the BEIT transformer model [279] to fine-tune it. It is clear that pre-training of a vision transformer is expensive so we freeze most of the model and retrain only the last layers where we used Adam as the optimiser function with learning rate of 0.00005, weight decay of 0.05 and cross-entropy as the loss function. These benchmarks have vital differences in classes, complexity and samples [283], hence for a fair analysis we trained three separate models for each benchmark. To avoid over-fitting, we add early stopping to training by monitoring losses with patience 5. Our method saves the best weights during each checkpoint. Each model has 85.8 million trainable parameters, and training and inference are executed using one RTX 3090 GPU and all experiments were performed with batch size of 8.

During the model development process, we adopt the *training-domain validation set* method to divide datasets into training, validation, and testing sets. Our method used 80% of the data from each domain for training and validation and the remaining 20% from each domain will be combined as an overall testing set. Ideally, to test the OOD generalisation of a model, the testing set should be from other unseen domains but this overall test set is also unique and unseen to the BEIT transformer model.

The version of pre-trained weights which we used in our research were pre-trained and fine-tuned on ImageNet 21k. For fine-tuning on a downstream task like classification, we append task layers in the BEIT model and fine-tune parameters on the specific benchmarks for OOD domain generalisation. Like the original work, our method also uses average pooling to aggregate information, and then feed it to a softmax-based classifier. As an overview, our approach first does image preprocessing and extracts feature maps along with attention masks from a frozen BEIT transformer layers for PACS [204], Office-Home [206], and DomainNet [209]. To perform a downstream task of classification we further fine-tune the last layers of the model in addition to the pooling and classification layer for domain generalisation. Figures 5.2 and 5.3 provides an overview of our results.

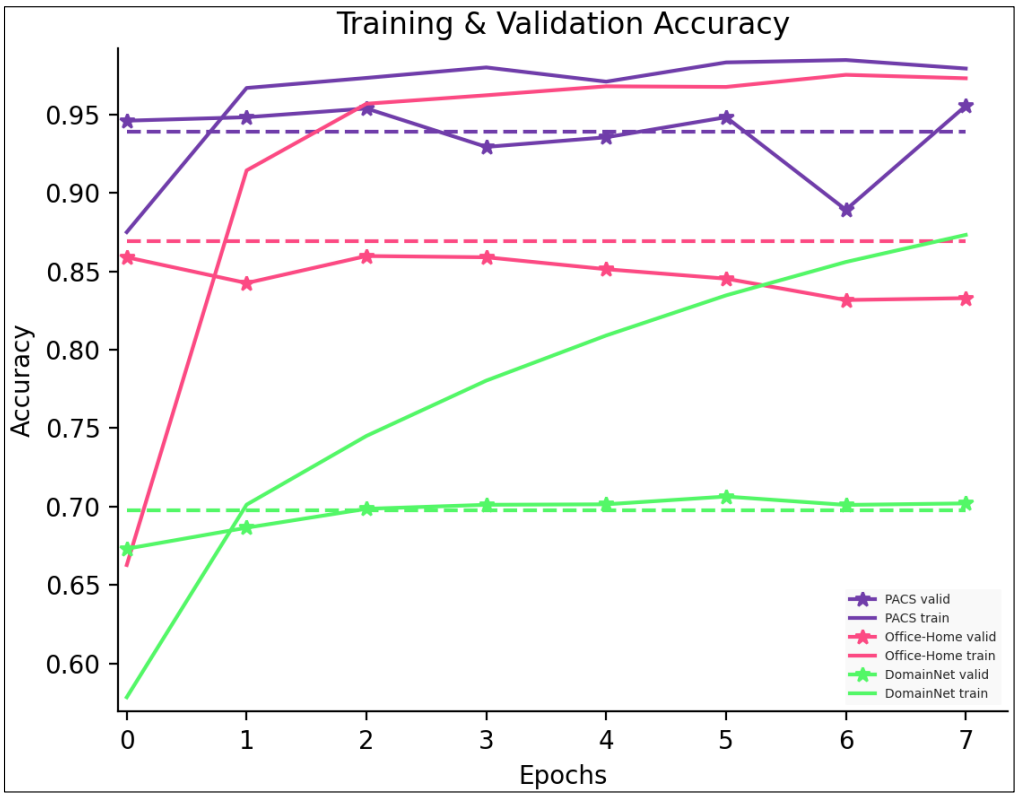


Figure 5.2: Accuracy curve for three benchmarks when training and inference with the BEIT transformer model

5.4.3 Inference on OOD Benchmarks

Following fine-tuning of hyperparameters for OOD datasets, our approach runs inference on a well-trained and shallow network on the unseen testing set from each benchmark. PACS has 9,991 images with 4 domains and 7 classes which is a comparatively smaller dataset. Office-Home has 15,588 images also with 4 domains but it has 65 classes. DomainNet is one of the largest benchmarks for domain gener-

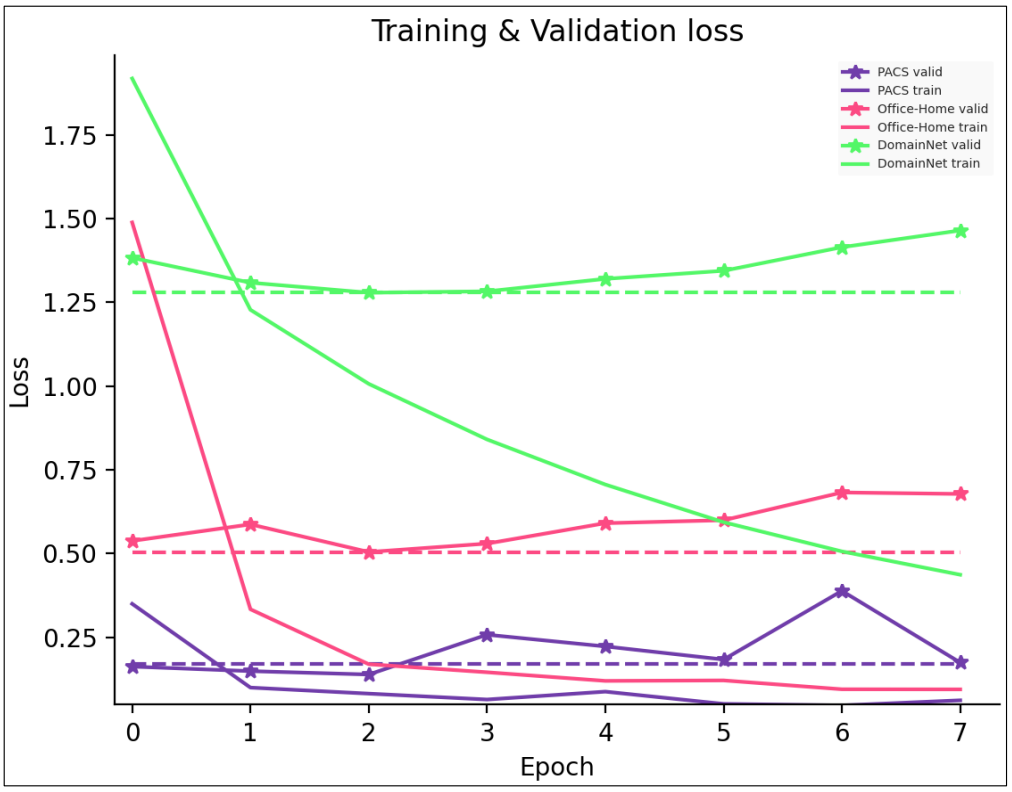


Figure 5.3: Loss curves for three benchmarks

alisation with more than 0.5 million images, 6 domains and 365 classes. Although fine-tuning of any vision transformer is a relatively less time-consuming process than pre-training from scratch, this also depends on the size of the dataset. For instance, PACS and Office-Home take almost 4-6 hours for fine-tuning but in the case of DomainNet our model takes almost 3 days on the same equipment.

During the development of our model, we monitored accuracy and loss metrics for all three benchmarks and Figures 5.2 and 5.3 show our model fine-tuned for only 7 epochs because of our conditioning like early stopping and checkpoints with saving the best weights. The three different colours in the Figures represent the performance of all three models and different line styles describe loss and accuracy for train, validation, and testing sets. In addition, another important factor to consider in the graphs is the gap between the target and validation curves and this gap could directly describe the domain generalisation ability of a model. For example, PACS shows a stable trend because the gap between target and validation curves is small for both metrics which means the model has better domain generalisation for PACS. On the other hand, DomainNet has larger differences between train and validation curves but the gap between target and validation is small which means that in this case the BEIT vision transformer needs more investigation. The numerical results in Table 5.2 also reflect the same information.

5.5 Results

This section presents quantitative results from experiments with various algorithms which were conducted during the exploration of domain generalisation. To conduct a fair comparison, this chapter also includes the performance of some other CNN-based approaches to domain generalisation taken from our published work as well as the work of others [283, 273].

Table 5.2: Fine-tuning using BEIT on three benchmarks – PACS, Office-Home, DomainNet

	Validation		**Target**		**Gap**	**Precision**
Benchmarks	***Top1 Acc***	***Top5 Acc***	***Top1 Acc***	***Top5 Acc***		
PACS	0.96	1.0	0.94	0.9980	0.02	0.9464
Office-Home	0.8597	0.9948	0.8691	0.9679	-0.0094	0.8754
DomainNet	0.7019	0.9347	0.6978	0.8793	0.0041	0.7111

Table 5.2 presents results of the vision transformer-based domain generalised model. The table includes the top-1 and top-5 scores for validation and target data distributions. Ideally, in domain generalisation, the gap is the difference between the score metrics like accuracy, loss or precision for IID and OOD. In our case, we consider the validation distribution as an IID and the target distribution as OOD and the gap is the difference between both terms. The graphs in Figure 5.2, 5.3 and numeric results in Table 5.2 support each other. For all three benchmarks, our vision transformer-based model shows state-of-the-art performance. For instance, in the case of PACS, it has 94% accuracy and the gap is only 2% which is a sign of good domain generalisation. For Office-Home, the OOD or target accuracy is 0.94% which is higher than the validation accuracy of IID which made the gap negative. Although our model has relatively low performance for DomainNet and Office-Home compared to PACS, the gap remains small which means the model performs effectively in terms of DG. The decline in performance figures for Office-Home and DomainNet are because of the complexity of the sample space.

We also determined the loss and accuracy metrics domain-wise. The main purpose of Table 5.3 is to examine hidden factors like which specific domain affects overall loss and accuracy and in what way. Thus we further break down the metrics into domain levels. The first part of Table 5.3 presents the performance on PACS which has four domains namely photos, art, cartoon, and sketch. It is clear from Table 5.3 that the model has good performance for the samples of photos. Conversely, BEIT-PACS shows low accuracy and loss for samples related to the artwork domain. The second

Table 5.3: Accuracy and loss for PACS, Office-Home, and DomainNet for each domain independently

PACS	***Photos***	***Art***	***Cartoon***	***Sketch***	***Metrics***
BEIT	0.9766	0.9183	0.9578	0.9371	accuracy
	0.0493	0.2507	0.1206	0.2227	loss

Office-Home	**Art**	**Clipart**	**Product**	**Real World**	**metrics**
BEIT	0.7979	0.8488	0.9324	0.8645	accuracy
	0.7443	0.5947	0.2502	0.5339	loss

DomainNet	**Clipart**	**Infograph**	**Painting**	**Quickdraw**	**Real World**	**Sketch**	**metrics**
BEIT	0.7822	0.3812	0.6893	0.6727	0.8073	0.6764	accuracy
	0.9129	3.0639	1.4036	1.2023	0.7915	1.4661	loss

part of Table 5.3 has details of the performance of the BEIT model on the domain of Office-Home. The domains are art, clipart, product, and real world. Like BEIT-PACS, BEIT-Office-Home also shows low scores for the artwork domain. A possible reason for this could be that pre-trained weights do not have essential high-level features because they may not have used enough artwork images in training.

In the case of DomainNet, the model has 6 domains to tackle, namely clipart, infogragh, painting, quickdraw, real world, and sketch. As mentioned earlier, DomainNet has 365 classes and more than 0.5 million samples from across different domains. The model has poor performance for samples of infograph as it is a very different domain to the others. Table 5.3 indicates that infograph is actually affecting overall accuracy and loss.

To compare our method with the state-of-the-art, Table 5.4 presents various CNN-based domain generalised algorithms from different areas of machine learning. We performed our own experiments for PACS and Office-Home but in case of DomainNet, we present performance figures from the literature [273]. The columns in the table are model names, IID accuracy, OOD accuracy, and gap for PACS and for Office-Home. For DomainNet, our work in this chapter only consider target accuracy. It is clear from the table that our method outperforms the state-of-the-art approaches in all benchmarks with a substantial difference. In the case of PACS and Office-Home, these CNN back-boned algorithms have few things in common. For example, their in-domain accuracy is comparatively high which could be better for tasks where models have to tackle partial domain shifting. Nevertheless, the performance of these methods on OOD accuracy is poor and as a result, such methods have expansion in their gaps. For instance, to be specific other algorithms have on

Table 5.4: Comparison between our trained model and other state-of-the-art methods for OOD generalisation

	PACS			Office-Home			DomainNet
Models	IID Accuracy	OOD Accuracy	Gap	IID Accuracy	OOD Accuracy	Gap	Target Accuracy
GroupDRO	0.95	0.73	0.22	0.82	0.52	0.30	0.337
ANDMask	0.95	0.72	0.23	0.81	0.44	0.37	*
Mixup	0.97	0.72	0.25	0.83	0.53	0.30	0.396
MMD	0.94	0.69	0.25	0.82	0.52	0.30	0.394
DANN	0.94	0.73	0.21	0.83	0.51	0.32	0.384
CORAL	0.95	0.77	0.18	0.84	0.55	0.29	0.418
VREx	0.97	0.80	0.17	0.76	0.49	0.27	0.336
RSC	0.97	0.77	0.20	0.83	0.50	0.33	0.389
ERM	0.97	0.78	0.19	0.84	0.57	0.27	0.412
Ours	0.96	0.94	0.02	0.86	0.87	-0.0094	0.70

average 21.1% and 30.5% gaps for PACS and Office-Home respectively.

Our method accomplishes its task by reducing the gap and also maintaining IID and OOD accuracy. If we examine the performance figures for PACS, our method obtains overall 96% and 94% for IID and OOD accuracy respectively and the gap shrinks from 21.1% to 2%. Our method also shows similar trends for Office-Home and reduces the gap parameter. The gap is negative because our model perform slightly better on OOD than on IID which is also a good sign for domain generalisation capabilities. Table 5.4 also shows that overall the target accuracy is not that high for all approaches but still our method outperforms all exiting approaches with a difference of 37.98% if we consider 31.8% as average accuracy.

In summary, in this results section, our work shows three tables to convey various dept information and analysis which could be important for domain generalisation. The first table discusses simple evaluation metrics, in the second table, we analyse the accuracy parameter of the model at the domain level individually which was helpful to understand the effectiveness of each domain on overall accuracy and to understand the behaviour of the model towards a specific domain. Furthermore, the third table is about the comparison between our method and other approaches in which we discussed IID and OOD accuracy with gap metrics.

5.6 Analysis of Latent Attention Space

We now focus on a detailed analysis of domain generalisation with the BEIT type vision transformer for the selected datasets. We consider self-attention distance metrics to explore different data distributions, using the learned latent space of each model for individual domains, and we investigate which domain is not effectively learned by the models. We also illustrate some sample attention maps with token masks and their respective attention heat maps.

5.6.1 Self-Attention Distance Analysis for Domain Generalisation

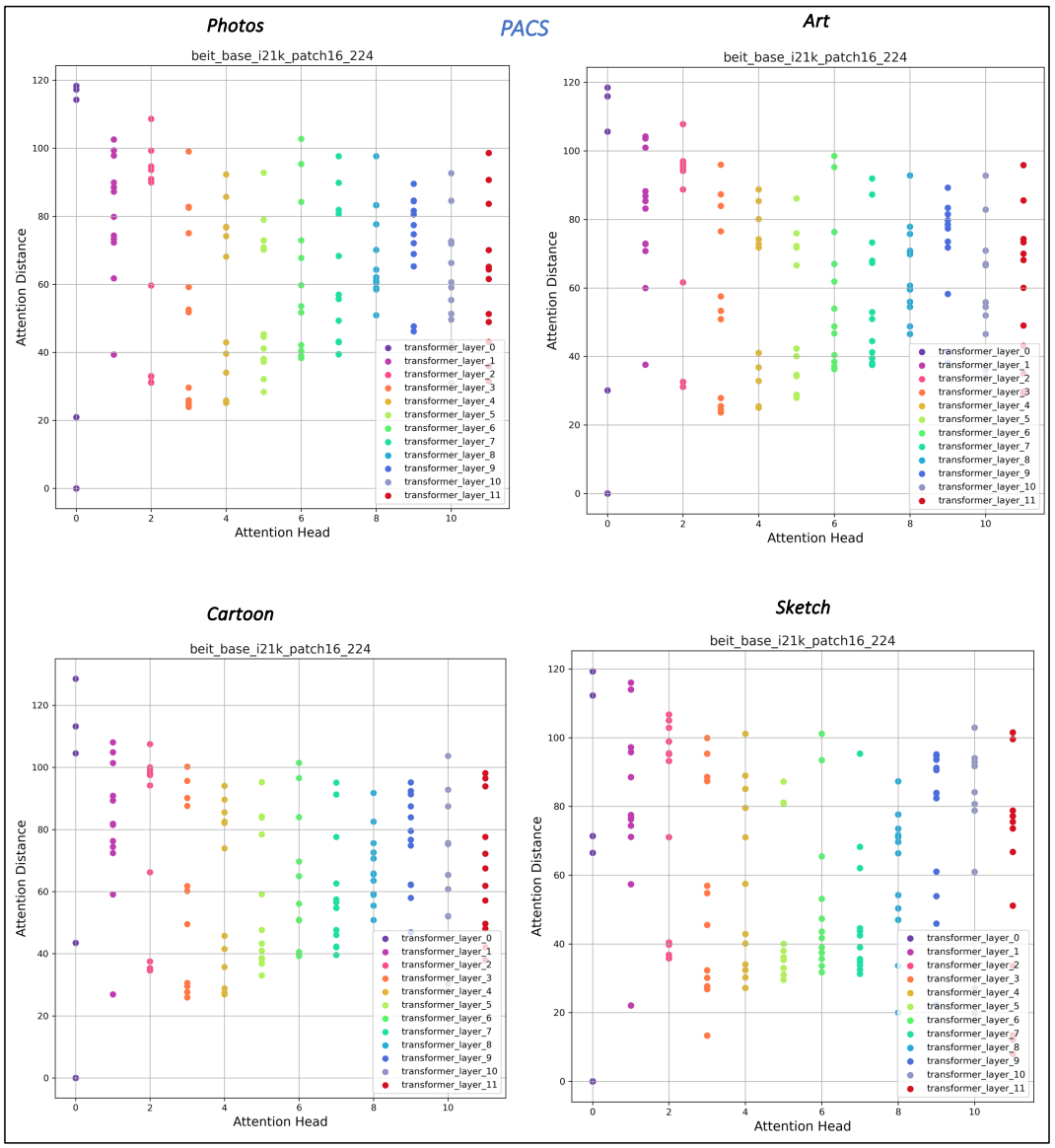


Figure 5.4: Mean self-attention distance analysis computed by our model when tested on the PACS benchmark.

After fine-tuning a model, it is important to calculate the mean attention distance of the learned latent space because it provides insights about layers of the model such as which layer learns most about local and global information. Moreover, in [215] the authors emphasise that if a model focuses on global features of images rather than local details it could increase robustness of that model.

Similar to that work, our interest here comes from the literature which shows that if a model has mixture of local and global spatial information in the early layers, this is a positive sign for the model to perform well for DG [155] . In addition, ideally in later layers the vision transformer model should only focus on global information which point out towards insights of article[215]. In this way, the learned features will be a combination of very strong and diversified local and global features. Similar to [155], our method follows steps to calculate the mean attention distance. This is defined as the distance between the query pixel and the rest of the patch multiplied by attention weights. We compute an attention distance for each head and then average these over all OOD testing benchmarks. Figures 5.4 and 5.5 show the mean self-attention head distance analysis for PACS and for Office-Home, respectively. We calculate distances for each domain separately.

In vision transformers, self-attention distance is defined as the spatial range across which a token (or patch) attends to other tokens in an image. This gives useful information about whether the model focusses on local information (short distances) or global patterns (large distances). In the context of DG, models that achieve a balance between local and global attention are frequently better at dealing with domain shifting issues. For example, the BEIT model outperforms on DG benchmarks because its attention layers successfully aggregate input across diverse spatial regions by learning robust and domain-invariant features. Self-attention distances also provide an interpretable method for assessing a model's generalisation capabilities. Short attention distances in DG tasks may reflect a reliance on texture-based and domain-specific elements, whereas longer distances represent focus on object shapes or structural patterns which are more transferable across domains. Furthermore, attention distance may be used to compare how models process OOD input across multiple layers and heads, providing a new metric for measuring DG performance.

However, self-attention distance has certain drawbacks when used as a DG metric. Long attention distances typically indicate global representation which does not always guarantee superior generalisation and may not be appropriate for tasks that need localised focus. The behaviour of attention distances is also significantly influenced by the training datasets, with models trained on texture-focused datasets possibly generating biased findings. Furthermore, the dynamics of attention distances vary across layers, with early layers often exhibiting mixed attention and

deeper layers tending toward global attention. This heterogeneity makes it difficult to capture the complex relationships necessary for DG. Furthermore, measuring self-attention distances across all layers and attention heads in large datasets can be computationally demanding if we consider scaling challenges.

Self-attention distance has potential applications beyond DG. In adversarial robustness, it can help evaluate whether a model maintains consistent attention patterns under adversarial attacks. For object detection and segmentation, attention distances can reveal whether a model focuses on relevant informations which is crucial for spatial tasks. The metric also improves the explainability of vision models by providing information about which areas of an image the model focusses on, which is useful for applications such as medical imaging. To analyse cross-modal interactions in multi-modal learning tasks such as image captioning, attention distances in the visual domain can be used in conjunction with text-based attention metrics. Furthermore, attention distances may be utilised in transfer learning to evaluate a model's adaptability to new domains by comparing attention patterns before and after fine-tuning.

Self Attention Distance Analysis For The Latent Space of PACS, Office-Home & DomainNet

Figure 5.4 presents the four graphs in the black box on PACS where each graph represents each domain. Similarly, Figure 5.5 represent four graphs for Office-Home according to its prospective domains. The axis of each graph is attention head and the y-axis is attention distance. Overall, the model provides a similar kind of trend in the graphs as the first layer model has a combination of both local and global distances. Higher distances relate to global information and lower distances correspond to local information. In both benchmarks, for every domain our models start fine-tuning from a mixture of both types of attention and then tries to shift learning focus onto the global side during the later layers. For this reason we can correlate the results of the experiment from Table 5.4 with this understanding of the graphs. As the final layers of the model mainly access higher global distances and this means stable and better generalisation ability. The graphs highlight a possible reason behind the state-of-the-art performance of our fine-tuned model.

In the same manner, Figure 5.6 indicates the mean self-attention distances for DomainNet. Overall, the model shows a downward trend for all domains of the DomainNet dataset which means that the model starts fine-tuning the same as PACS and Office-Home but in the later layers the model is unable to properly shift towards the global side of the latent space which we can see by low mean distances in the graphs. Consequently, rather than the model paying attention at global distances it struggles to learn global features and focuses on the local parts. However, from

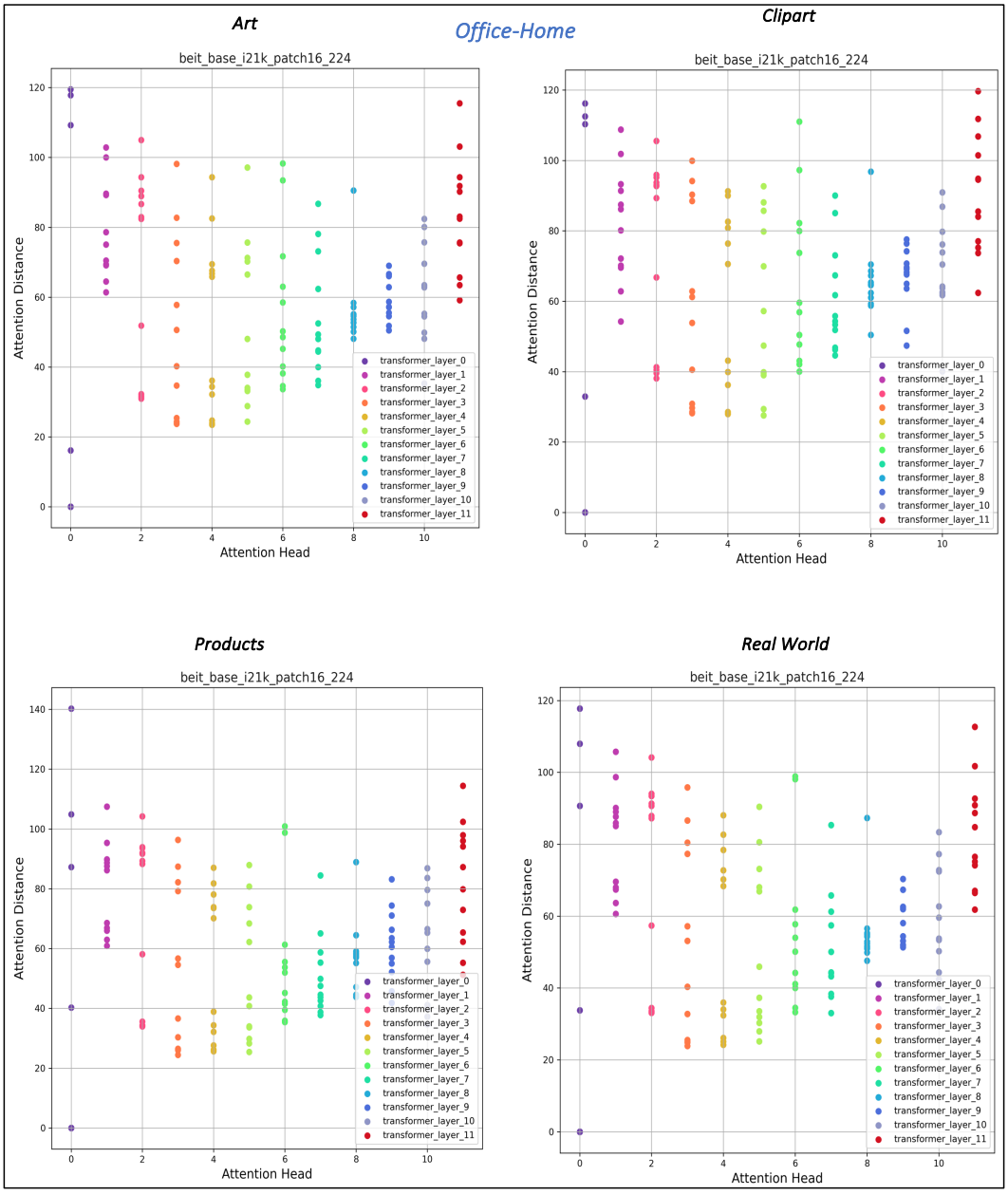


Figure 5.5: Mean self-attention distance analysis computed on the Office-Home benchmark.

Table 5.4 it is clear that our model still has better accuracy than other available methods but the generalisation ability of our fine-tuned model is not that effective compared to PACS or Office-Home. One of the reasons our models outperforms others is because the early layers and middle layers have mixture of both the local and global feature space.

During our analysis of DomainNet, we also observe that after the first few layers, mean attention distance starts decreasing which is a common trend across all domains. This means we actually do not need to train all layers of the transformers because the model is not learning generalised features and thus we could reduce the number of training layers from the transformers. Following the results in Figure 5.6, this work also explores the literature related to more efficient training which could also increase the generalisation of vision transformers. Inspired from the results we mentioned about pruning in the literature this is a future area to explore. According to our understanding, it will be interesting and helpful for the research community if we can implement pruning methods on vision transformers and further explore which method is better and will have better DG.

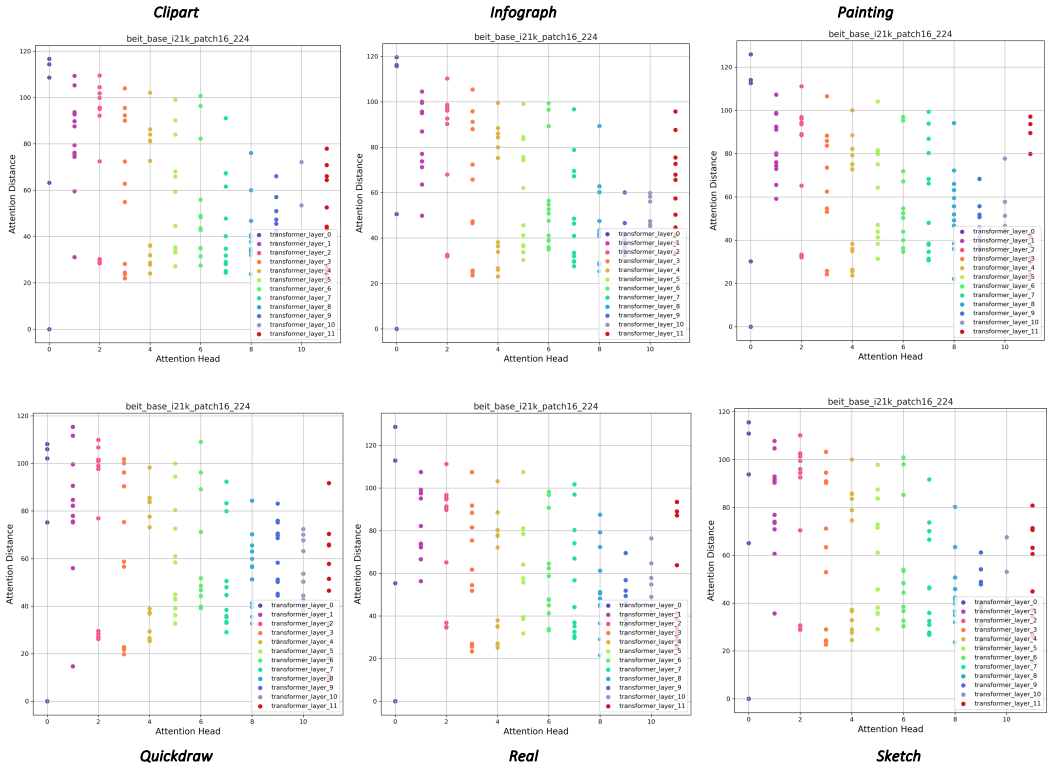


Figure 5.6: Latent space of self-attention distance for our model learned when tested on DomainNet

In summary, the attention distance analysis conveys extremely important insights. For instance, if attention distances are low it means that the model is too focused on local features/information which could lead towards poor DG. On

the contrary, high attention distances means more global features or information is learned by the model which usually leads models towards better and more robust DG. Therefore, Figures5.4, 5.5, and 5.6 also deliver similar messagea across each domain.

5.6.2 Analysis of Attention Maps

For our research it is important to visualise the attention maps learned by models for various domains. In Figure 5.7, we present a number of original images (12 of them) with their prospective token mask extracted directly from the latent space of the fine-tuned model and heatmaps of attention on the original images. It can be clearly seen from the results that model is paying attention to the shapes rather than backgrounds and is ignoring noise.

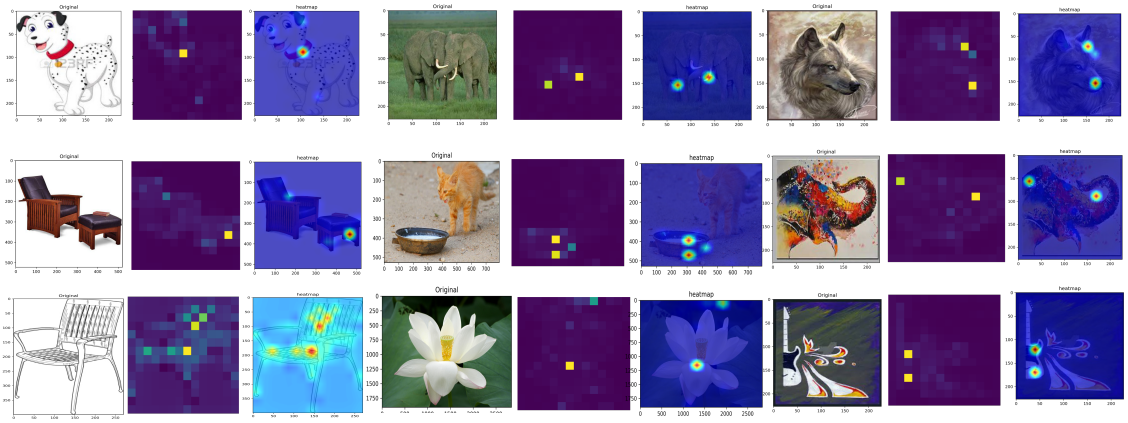


Figure 5.7: Attention map analysis for for sample images for various domains.

5.7 Conclusions and Lessons Learned

This chapter presents an exploration of vision transformers to achieve the goal of domain generalisation. Our work successfully highlights the important key points we should consider for the improvement of domain generalisation including the model's length, self-attention mechanism, self-supervised learning, the demonising ability of the model, removal of texture information, and the focus on the shapes of objects. We fine-tuned a base version of the BEIT model for domain generalisation benchmarks. Because pre-trained BEIT has most of the key points, therefore our model yields state-of-the-art results for all benchmarks. Moreover, we also investigate the generalisation of other pre-trained vision transformers on OOD versions of ImageNet.

Additionally, our method also provides quantitative performance figures for metrics like gap, precision, IID and OOD accuracy. This chapter also presents a per-

domain analysis and then provides a detailed comparison with other popular algorithms in the area of domain generalisation. Our method is able to significantly reduce the gaps between IID and OOD scores with big margins in all selected benchmarks. As we know that gap is directly related to domain generalisation, our model is able to reduce this from 21.1% to 2% for PACS and provides similar trends for Office-Home and DomainNet.

This chapter also emphasises the analysis of the learned latent space and we especially considered the mean self attention distance for each attention head of our fine-tuned vision transformer. The mean attention distance is one mechanism for measuring information flow and provides insights as to where a model is not learning any significant domain knowledge. We also relate global distances with domain generalisation and explain reasons why DomainNet does not perform well. Similarly, attention maps help us with visualisation of claims such as whether a model is really paying attention to the objects rather than on backgrounds. To address the issues with DomainNet, we also proposed interesting future research directions including implementing the idea of pruning for efficient training and better domain generalisation in Chapter 7.

The original paper of BEIT claims that because of MIM it denoises the image and which means accuracy will be less affected in this model. Therefore, to verify this factor, in the next chapter, Chapter 6 we will generate our own OOD benchmarks and investigate this factor. Even though feature maps are indicating that model is focused on shapes but we want to explore the behaviour in the presence of external noise.

This chapter focused most significantly on addressing research question RQ3 which is related to getting improved accuracy by having a better understanding of data distribution with domain generalised models compared to domain specific methods. Therefore, the state-of-the-art results we obtained from BEIT models to handle DG contribute to this goal. This chapter also explores sub-hypotheses H1 and H2 by delivering a more robust model with better domain generalisation ability.

Chapter 6

Measuring the Resilience of Domain Generalisation in the Presence of Newly Generated Out-of-Distribution Noisy Images

In the previous chapter, Chapter 5, we presented a detailed analysis of vision transformers like BEIT for DG. The results of the experiments in that chapter showed the validity of our exploration by fulfilling the requirements of research question 3 and sub-hypotheses H1 and H2. The results shown earlier in Table 5.4 also highlighted the clearly superior performance of the fine-tuned BEIT model compared to other available methods. Moreover, we also tried to describe the relationship between domain generalisation and mean attention distance of each attention layer by building connections with local and global information flow.

In similar manner, in this chapter we will test a crucial claim of an already trained BEIT model which says that the model will also have denoising ability. Towards this aim, we design a paradigm in which we add noise to images which will be a periodic masking grid overlaid on images in our test benchmarks including PACS, Office-Home and DomainNet. Additionally, the idea of grid masking is inspired from the original paper [284] and details about this approach are discussed in the methodology section of this chapter. This leads to the generation of new datasets which models will have never seen previously and such datasets could serve as an OOD test set. Briefly, for each selected benchmark, our method creates four different variations of periodic grid markings based on the number of grids and the locations of these grids with respect to class objects in the images. Similarly, from these variations of generated image data which can be seen later in Figures 6.3, 6.4 and 6.5, each image will have three varietiations based upon the mask and object occlusion ratio which

will be set at 25%, 50%, and 75%. Therefore, our method will create 36 different kinds of benchmark to measure the behaviour of a model.

In a broader view our designed mechanism to measure the resilience of DG has two main tasks. The first is to generate a new periodic grid masking around or outside the targeted objects or shapes within a test image where the most important information for correct classification of that image already exits. The second task consists of the inference on these generated benchmarks in order to get summary results.

It is important for this research to measure the ratio between the targeted shapes of given class objects within images and the grid masks so that we can investigate performance changes by blocking different proportions of the object area and we call this the occlusion ratio. Hence, to obtain the area of masks of objects present in the given images, our method uses a combination of pre-trained models like SAM [76] and Grounding DINO [64] to extract the required masks with text-prompts in a zero-shot instance segmentation manner. Furthermore, we can use these masks to calculate the overlapping regions between the grid masks and objects, as will be explained in Section 6.3.

More importantly, by conducting such experiments as these, our research focuses on research question RQ2 which addresses the measurement of variations or stretchiness we can induce into a data distribution and what type of effects we can expect in terms of domain generalisation. Similarly, this chapter also addresses sub-hypotheses H3 according to which when we trained a domain generalised model under all the conditions which we have described earlier then we do not need transfer learning or fine-tuning to adapt it for new benchmarks. The experimental results in Tables 6.1, 6.2, and 6.3 wil illustrate this point.

The structure of this chapter is that it contains a brief introduction and a related work of our proposed method and explains how we achieve our motivation by cascading two models with grid masking. Then the methodology section has details about grid masking, SAM, Grounding DINO, and other blocks of our proposed idea. This chapter also has a details section about the new data generation and we will explain how we obtained our experimental results in Section 6.4. By the end of the chapter we have presented and discussed the results of our comprehensive experiments on these newly generated benchmarks.

6.1 Introduction

The main motivation behind the research reported in this chapter is to test the claim of BEIT and according to previous papers it also has a denoising ability that could means better domain generalisation. Therefore, we designed a new paradigm

for the generation of different OOD benchmarks to test the DG capability of our fine-tuned models. This will also help us to understand the robustness of vision transformer-based models like BEIT. At the same time, the generation of such new benchmarks will add more diversity and we will have more extensive testsets for OOD inference.

Another key idea behind our research is related to measurement which means that rather than creating datasets randomly it is vital to take a small numbed of steps of change in a deterministic manner. Then, after obtaining or creating each new distribution, the methods and models we proposed run inference on it and we can determine evaluation metrics like accuracy and gaps. In addition, this research also provides insights to building relationships between distractions in datasets and the performance of models.

To the best of our knowledge, our proposed method is a unique way to generate new benchmarks to test the OOD detection capabilities of any model by using zero shot instance masking. To design the basic framework, we implemented simple grid masking functions to datasets and created grid masks of the same size as the original images and then used the popular SAM [76] and Grounding DINO [64] to obtain precise masks of objects appearing in the images using text-prompts. WE then find overlapping regions between the grid mask and the object masks to determine the occlusion ratios. The details about the design method are given in Section 6.3. We have four variations of benchmarks based on the number of grids and their locations. To be specific, three variations are with 2, 5, and 9 grids within the shapes of objects and one variation is with 5 grids but outside the masks of shapes which is illustrated in Figures 6.3, 6.4 and 6.5. Moreover, the grids mask outside of shapes are calculated by finding rectangles or bounding boxes.

6.2 Related Work

Although the previous chapters have presented most of the related work in terms of the overall thesis and hypotheses and research questions, in this section we will present the specific related methods which we have used directly within our framework. Hence, we will explain the simple working of SAM and then the advantages of using SAM and Grounding DINO together. In addition, it is also important for the reader to learn about various masking methods so we cover that as well.

Segment Anything (SAM): The paper "Segment Anything" describes a versatile and powerful model that can execute segmentation tasks in a variety of contexts without requiring task-specific training. The authors describe a unique technique known as the Segment Anything Model (SAM), which uses a highly generalisable

framework to effectively segment items in photos [76]. SAM's primary notion is to accomplish high-performance segmentation through a mix of prompt engineering and a unique transformer-based architecture. The model operates by receiving prompts such as points, boxes, or masks as input and then creates the segmentation masks accordingly. SAM's architecture consists of an image encoder, a flexible prompt encoder, and a mask decoder, which process the input picture and prompt to create the necessary segmentation output. The primary benefit of SAM is its capacity to adapt to multiple segmentation tasks without requiring explicit fine-tuning for each job, considerably increasing its usefulness. Figure 6.1 in the methodology section of this chapter describes the basic block diagram of SAM.

SAM+Grounding DINO: The original method of SAM only works with three types of prompt modes including, bounding boxes, points and masks. Moreover, we also notice that SAM is more effective and accurate with its bounding boxes prompt mode. Therefore, by combining with another method like Grounding DINO [64] which takes image and text prompts as a input and outputs the bounding boxes according to the text prompts, then these generated bounding boxes are given into the SAM model as a input prompts, along with the image and tasked to segment the objects in the image.

To describe Grounding DINO [64], the paper "Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection" describes a novel approach that combines the DINO [44] framework with grounded pre-training techniques to improve open-set object detection capabilities. Grounding DINO tries to overcome the limits of classic object identification models including SAM, which often perform well only on preset categories, by allowing for the detection of objects outside of the training set. The essential innovation is the combination of DINO's strong implicit neural representations with grounded pre-training, which takes into account extra contextual information and different training data. This integration enables the model to generalise more effectively and recognise a broader range of items in different circumstances. Figure 6.1 presents an overview of the two methods and also explains the combination of both in outline.

Masking Related Methods: In this chapter, another component of our research is to conduct an investigation into various available masking methods and then to select the most optimal solution for our research. In the literature, Grid Masking [284], Random Erasing [285], Cutout [286], Mixup [287], CutMix [288], Patch Masking [45], DropBlock [289], Image Impainting [290], Segmantic Masking [73], and Hide-and-Seek [291], are the most famous methods which perform different types of masking. We now explain each of them breifly.

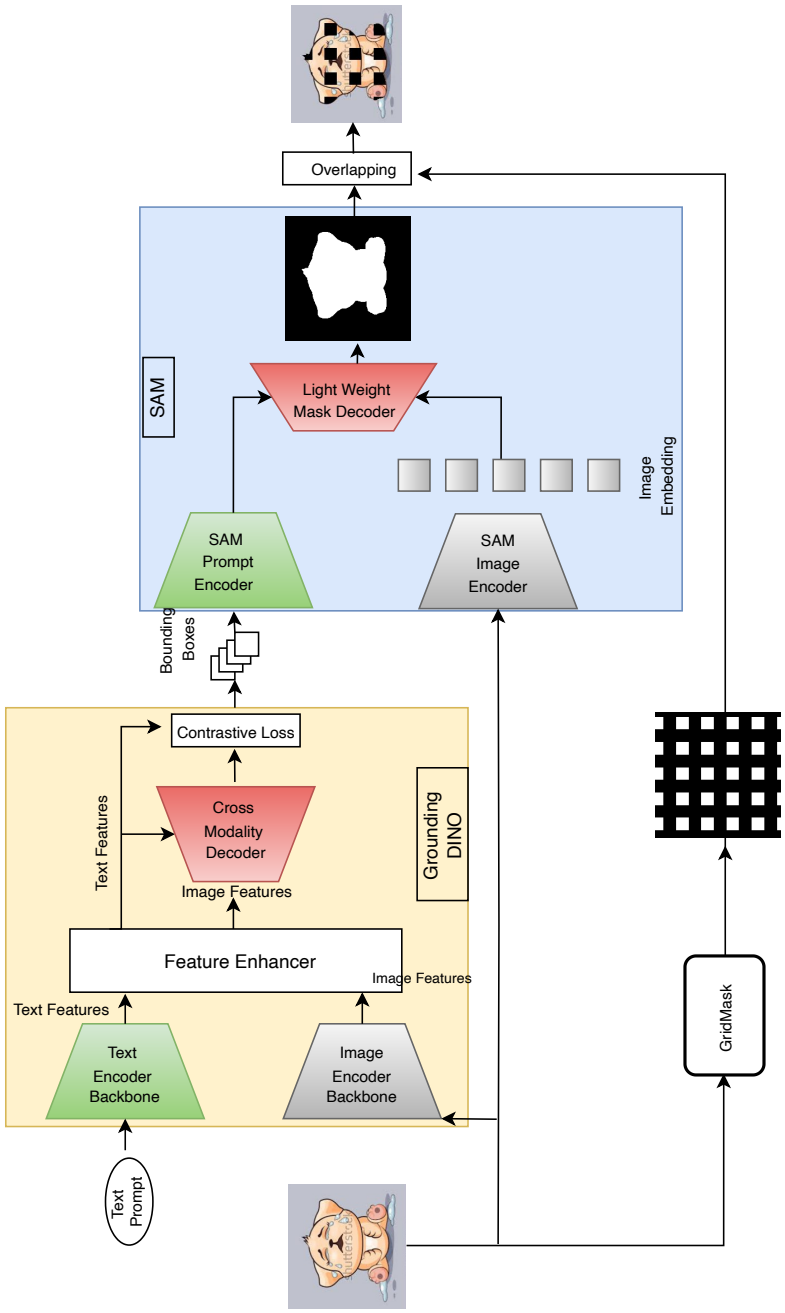


Figure 6.1: Overview diagram illustrating how SAM and DINO combine to segment objects in images in our test sets.

- **Grid masking** involves overlaying a grid on an image and randomly masking (obscuring) some of the grid cells [284].
- The **Random erasing** method randomly selects a rectangular region in an image and replaces it with random values, zeros, or a constant value [285].
- **Cutout** is similar to random erasing but specifically zeros out (or replaces with a fixed value) a randomly chosen square region in an image [286].
- **Mixup** combines two images and their labels by taking a weighted sum of

both. The pixels of the two images are mixed according to a mixing parameter, and the labels are mixed in the same proportion [287].

- **CutMix** combines elements of Cutout and Mixup. It replaces a rectangular region of one image with a patch from another image while also mixing their labels accordingly [288].

- **Patch masking** involves masking out random patches in an image. This method is particularly useful in self-supervised learning, where the model is trained to predict the masked patches, encouraging it to learn robust and meaningful representations of the data [45].

- **DropBlock** is a structured form of dropout where contiguous regions of the feature map are dropped during training [289].

- **Image Inpainting** masks out a portion of an image and trains the model to reconstruct the missing parts [290].

- **Semantic masking** involves masking specific objects or regions in an image based on their semantic meaning [73].

- **Hide-and-Seek** is a data augmentation technique where parts of an image are randomly hidden during training [291].

What all these masking methods have in common is that they have been developed to enhance the generalisation, diversity, robustness, and regularisation of any model during the time of training.

6.3 Methodology

6.3.1 Implementation of Grid Masking

Grid masking is a data augmentation approach in computer vision that improves neural network stability and generalisation. GridMask is a novel information removal method that employs structured dropping of uniformly distributed square regions, which contrasts with previous methods like Cutout and Hide-and-Seek. Unlike Cutout, which removes large continuous regions, or Hide-and-Seek, which randomly selects squares, GridMask deletes spatially uniform squares, allowing for better statistical balance between preserving and removing information. The GridMask which we implemented in our proposed method is inspired by the original work in [284] and has the following components:

- **Grid Overlay:** The image is covered with a grid of fixed-size cells. Each cell represents a tiny rectangle or square portion of the picture.
- **Random Masking:** A fraction of these grid cells are randomly chosen to be masked. The selection might be based on a preset probability or a predetermined number of maskable cells.
- **Masking Operation:** The selected grid cells are then hidden or "masked." This may be done in one of a number of different ways including **Zero Masking:** Set the pixel values in the chosen cells to zero. **Random Values:** Replace the pixel values with random values that are evenly distributed or derived from a normal distribution. **Constant Value:** Replace the pixel values with a constant value (such as the dataset's mean pixel value).
- **Augmented Image:** The end result of this is an enhanced image with specific sections defined by grid cells blocked off. This image is then utilised as input while training the neural network.

Mathematically the setting of GridMask method can be written as,

$$\tilde{x} = x \times M$$

where $x \in \mathbb{R}^{H \times W \times C}$ represents the input image, $M \in \{0,1\}^{H \times W}$ is the binary mask that stores pixels to be removed, and $\tilde{x} \in \mathbb{R}^{H \times W \times C}$ is the result produced by the algorithm. For the binary mask M, if $M_{i,j} = 1$ it keeps pixel (i, j) in the input image otherwise it removes it. Figure 6.2 explains the fundamental concepts behind the GridMask method which we implemented in our approach.

Four numbers are used by this method. r is the ratio of the shorter white edge in a unit. d is the length of one unit. δx and δy are the distances between the first intact unit and boundary of the image.

$$k = \frac{\sum(M)}{H \times W}$$

As r computes the keep ratio of the input image therefore we first calculate k for a given mask M and then r can be calculated using the following equation.

$$k = 1 - (1 - r)^2 = 2r - r^2$$

After finding the keep ratio, the algorithm determines l. Therefore, the length of one unit d does not impact the keep ratio. But it determines the size of a single dropped square. When r is fixed, the relationship between one dropped square's side length (l) and d is

$$l = r \times d$$

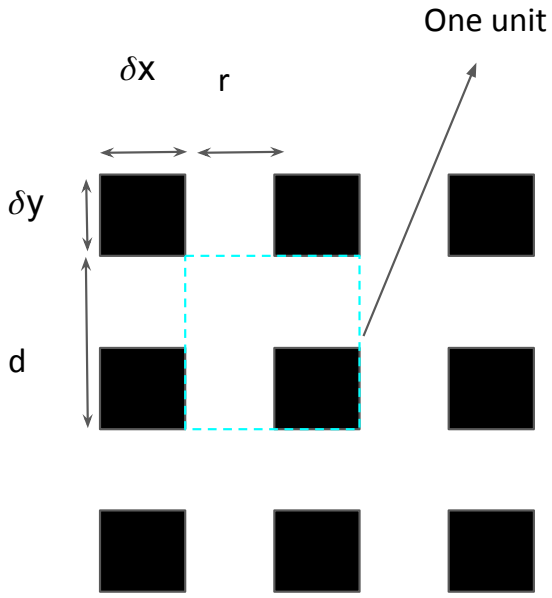


Figure 6.2: The basic diagram for GridMask and how one unit of the grid appears.

The larger d produces larger l. Hence to add randomness, d is selected randomly using the given equation.

$$d = \text{random}(d_{\text{min}}, d_{\text{max}})$$

In the end, δx and δy can shift the mask according to given r, and d and covers all possible situations. δx and δy are also selected randomly.

$$\delta x(\delta y) = \text{random}(0, d - 1)$$

6.3.2 Workings of SAM and Grounding DINO for Object Masking

As we know, our goal is to generate an object occlusion based grid masking which is itself a unique concept to block regions of interest with most relevant information. To fulfil this goal, the first step will be to calculate the grid marking for whole input image as shown in Figure 6.1 by using fundamentals from Section 6.3.

In parallel to the above, our proposed method with cascade settings also takes an input image with a corresponding text label as a prompt. In this cascade of two models, first we have the Grounding DINO model which will boost the power of the original SAM model. The main motivation for using Grounding DINO is to add a more interactive mode into SAM which is the use of text prompts. In the original paper describing SAM, although the text prompt mode is mentioned, it is not implemented in the official code base for the technique. Therefore, we designed

our own mechanism to get the best of both models in such cascade settings.

At the start of this process, each of the input images in the benchmark dataset is given to backbone image encoder and related text prompts are given to the text encoder backbone as shown earlierin Figure 6.1. Each of the two encoders extract basic image and text features respectively and we thenapply another important function which is known as feature enhancer. In addition, the cross modality decoder in Grounding DINO provides the first bridge to correlate the textual information to image features. The contrastive loss function takes input from the cross modality decoder and the new text features and finally yields localisation output in the form of bounding boxes. In Figure 6.1, the processes shown in the yellow colour relates to Grounding DINO.

As we have already established that SAM works better in bounding box mode which means that bounding boxes will be given to SAM as a input prompts, we use the first model to get the bounding boxes of a given class and the input images will again pass into the SAM image encoder. This will generate image embeddings and in the end a lightweight decoder will accept the embeddings of bounding boxes and embeddings of images to produce segmentation masks which can be seen in Figure 6.1. This mask will be the final output of the cascade network with a zero shot instance segmentation. Therefore, by combining both methods in a cascade manner can enable generalised zero shot settings and we can deliver a segmentation model based on zero shot text-prompts.

6.3.3 Finding Overlapping Region of Interests

After determining the object masks in all the images in each of our benchmark datasets, the next step in the process is to compute overlapping regions between grid masks and object masks. In the method we developed, to calculate and keep only the grids from a grid mask that overlap with an object mask, we need to perform a number of mathematical steps. For example, let G be the grid mask and O be the object mask, where $G, O \in \{0,1\}^{H \times W}$, which means both will be binary masks. We can calculate the overlapping region mask:

$$G_O = G \odot O$$

where $\odot$ denotes the element-wise multiplication, keeping only the grids present in the overlapping regions G_O. To get our final results, this new grid mask G_O will be further multiplied by the original image and the end result is shown in Figure 6.1.

Let I be the original image where $I \in \mathbb{R}^{H \times W \times C}$, to extend the mask across channels:

$$G_O^{(ext)} = G_O \otimes \mathbf{1}_C$$

where $\otimes$ denotes the outer product with a vector of 1's of length C. Then, apply the mask to the original image

$$I_{masked} = I \odot G_O^{(ext)}$$

6.3.4 Inference and Testing

At the final step, this method is applied on all benchmarks one by one with 12 variations and yields 36 new benchmarks for all three distributions. In the inference stage, the algorithm simply runs on each dataset and generates a new benchmark in a specific selected setting with an end-to-end solution. In the testing, we run already trained BEIT models for the original PACS, Office-Home, and DomainNet benchmarks and on these newly generated ones and produce the target results. All the data generation and testing is done on 1 GPU (RTX 3090) with 24 GBs memory capacity.

6.4 Generation of New OOD Benchmarks

This section explains the different variations of newly generated OOD benchmarks which we created in order to carry out further experiments. For example, Figures 6.3, 6.4 and 6.5 each show two types of variation. On the y-axis, these figures have variations according to the number of grids and the locations or positionings of those grids. On the x-axis we have changes according to occlusion ratios which are 25%, 50%, and 75%.

Moreover, for 25% occlusion ratio between object masks and grid masks, we considered the original grid pattern which is shown in Figure 6.2. This means that we will keep one grid per unit or window of grids. On the contrary, for 50% occlusion ratio, instead of getting simple grids, we use a checkerboard pattern. This means that we activate opposite but connecting grids and get 50% blocking of objects. Similar to the 50%, the 75% occlusion ratio also has a checkerboard pattern for grids but it allows overlapping between the grids of each unit or window.

Another important variations lies on the end of the y-axis of thee Figures which is related to generating grids outside the shapes of objects. This will distract models as with the occlusion ratio, shapes of objects or silhouettes are also deformations with the given style of grids. To calculate the grids outside the shapes, after getting the masks of objects, we simple find bounding box coordinates according to masks and then generate the implemented grids styles within the rectangle area. Next, we

will highlight how these data generation approaches look like for each of the three benchmarks.

6.4.1 PACS Occlusion Benchmarks

This part of the chapter shows OOD data generation for the PACS benchmark. We run our method on the testing set with only 1,014 images because the idea is to generate unseen data distributions which means that this data is unseen by the model and not used in training. Therefore, by adding these distractions to a dataset can also produce new unseen distributions. Figure 6.3 describes the visual results of each type of variation on some of the different domains of the PACS dataset.

For instance, in the case of a 2 grid system, we will have larger sized grids to block the information and by increasing the number of grids, the size of grid masks decreases. Hence our end goal is to explore the behaviour of already trained vision transformers on such new generated benchmarks and investigate the domain generalisation like how such distractions will effect the performance of models.

6.4.2 Office-Home Occlusion Benchmarks

When our proposed method is implemented on the second benchmark which is Office-Home, it has a much greater number of classes to work on. The results of some sample variations are shown in Figure 6.4. Our data generation method follows the same settings as in PACS for Office-Home data generation, therefore, for different domains we highlight the visual definitions in Figure 6.4. The details about the original dataset are described in Table 3.2. The size of each unit depends on the number of grids and overall height and width of the input image. Our proposed method is implemented on the test-set given by the original dataset paper and the number of images this is applied to is 3,117.

6.4.3 DomainNet Occlusion Benchmarks

We know that DomainNet is already a large scale domain generalisation dataset with 6 different domains and more than 0.5 million images. The method we designed for creating additional testing images was implemented on 119,202 images and generated 12 more versions of the DomainNet testset. Figure 6.5 illustrates how each variation blocks or masks the information.

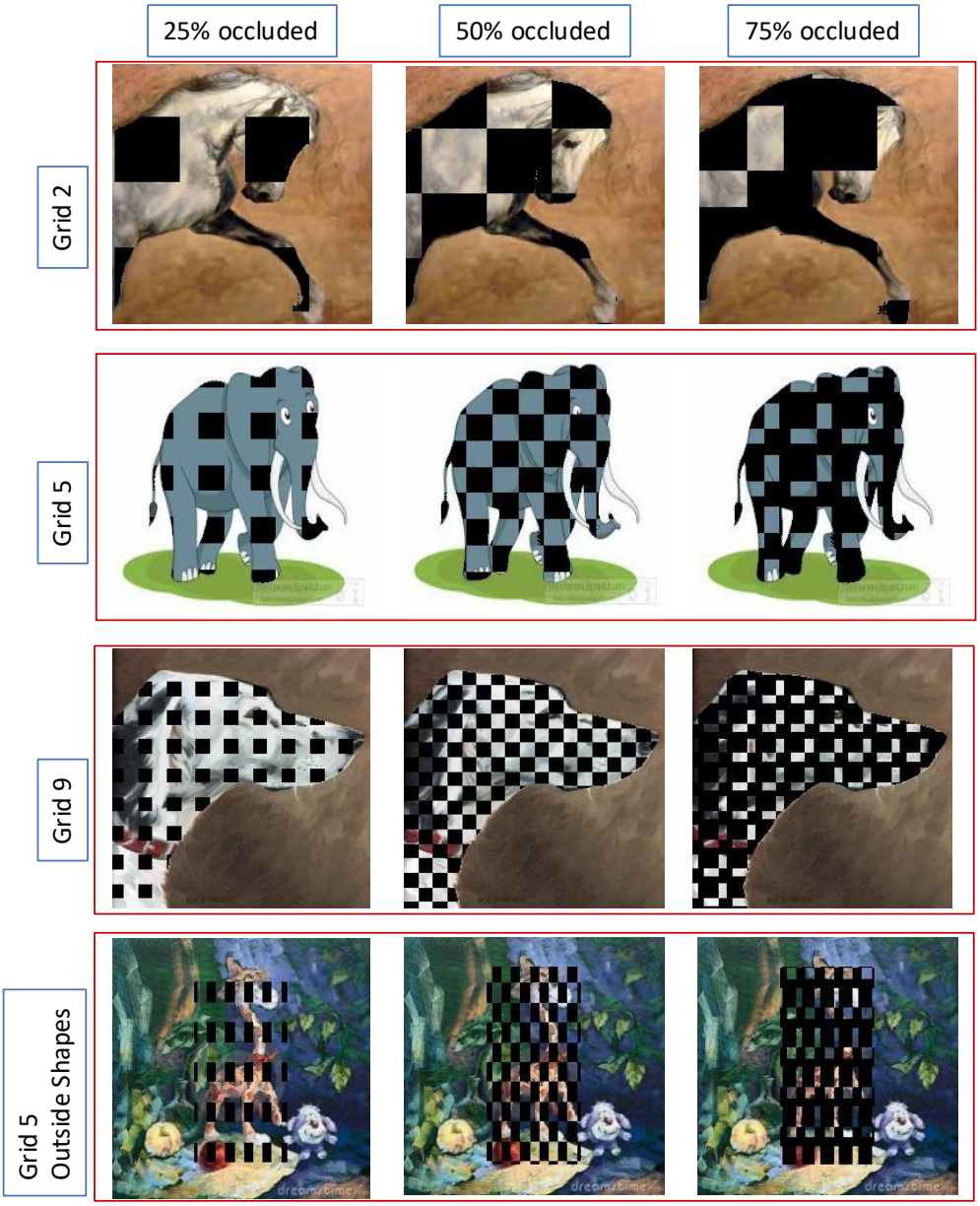


Figure 6.3: Generated data distributions for PACS with 12 different variations to measure the resilience of the BEIT model. The y-axis shows types of newly generated distributions based on number of grids and x-axis has different occlusion ratios.

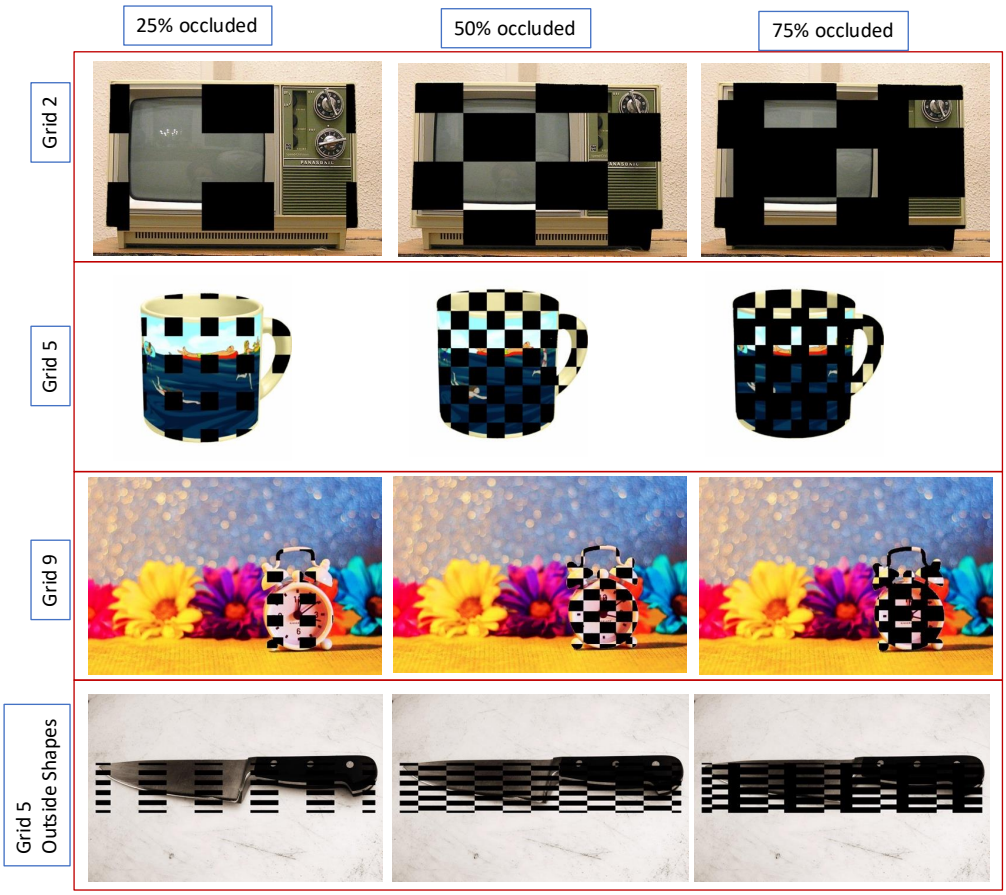


Figure 6.4: Generated data distributions for Office-Home with 12 different variations to measure the resilience of the BEIT model. The 25% means simple grids, 50% means checkerboard pattern, and 75% means checkerboard with overlapping between the units.

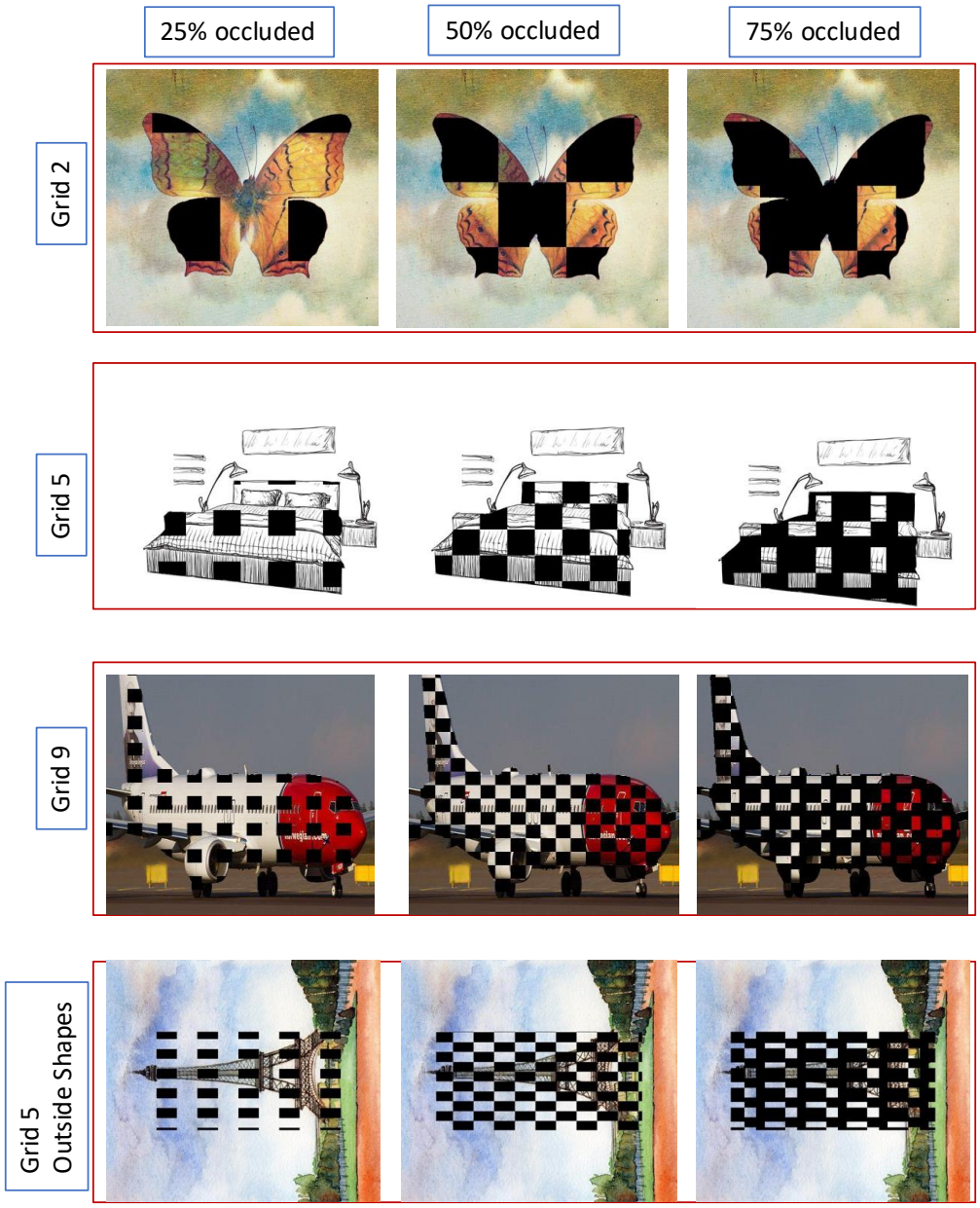


Figure 6.5: Generated data distributions for DomainNet with 12 different variations to measure the resilience of the BEIT model.

6.5 Results and Discussion

In Chapter 5 the results in Table 5.2 were of the fine-tuned BEIT model for PACS, Office-Home, and DomainNet respectively and it is worth re-capping and summarising those results at this point before presenting results on the new benchmarks we generated. According to the analysis, the major reasons behind better performance for PACS and Office-Home are that the BEIT-based fine-tuned vision transformer models know about similar classes during the time of pre-training in the original work. Additionally, PACS and Office-Home are relatively less complex and smaller benchmarks compared to DomainNet. In our study, based on the features of benchmarks [292] like complexity, number of domains, and sample sizes, we consider PACS as a basic level benchmark, Office-Home as a medium level benchmark, and DomainNet as a high-level benchmark. We know that to get better domain generalisation, a model's goal is to decrease the gap between IID and OOD. Therefore, for PACS, we observe only 2% gap and for Office-Home Gap is negative meaning the model performs even better on OOD distributions.

In the case of DomainNet, even though precision was lower than for the other two benchmarks, the gap between IID and OOD remains smaller which could be proof of the stability and better generalisation potential of vision transformer-based methods. Moreover, metrics in Table 5.2 described the overall results of models on benchmarks and did not provide a complete picture of for which specific domain(s) in a benchmark a model achieves better accuracy and precision. Therefore, in Table 5.3 we computed similar metrics for each domain individually.

Table 5.3 expressed accuracy and loss for OOD benchmarks according to each domain. The first section of the Table highlighted numbers for BEIT-PACS which has a lower accuracy of 91% and higher loss of 25% for the artwork domain compared to photos, cartoon, and sketch. Moreover, BEIT-PACS shoed highest accuracy of 97% for the real photos domain which is because the original baseline model of BEIT was trained with similar real images and weights of the pre-trained model could already have better representation for such domains. In the same way, Table 5.3 also had information about the BEIT-Office-Home model, and the second row of the table focused on that. Office-Home also had four different domains including art, clipart, product, and real world.

Interestingly, the BEIT-Office-Home model achieved the highest accuracy of 93% for the product domain and the real-world domain had the second highest accuracy of 86%. The loss metrics also indicated the same trend in the table. As a result, we can ask the question of why BEIT-Office-Home did not have the highest accuracy for the real world domain like BEIT-PACS did. The answer is straightforward, the product domain contains images of various products classes with white backgrounds

meaning fewer distractions for models. On the other hand, in the real world domain, images are relatively complex, which means more distraction and closer to reality. As a result, the BEIT-Office-Home had more refined accuracy for the product domain than for the real world. Meanwhile, BEIT-Office-Home also exhibited lower accuracy of 79% and higher loss for the art domain which means it followed the same trends as the BEIT-PACS model.

Finally, the third row of Table 5.3 had information related to DomainNet with six domains including clipart, infograph, painting, quickdraw, real world, and sketch. BEIT-DomainNet highlighted similar patterns to the previous two models and our observation stands correct as this model also performed well on real world images with higher accuracy of 80% for the real-world domain for the same reason we described earlier. Conversely, the BEIT-DomainNet model performed poorly on the infograph domain as this data distribution is much more diverse and complex than the original baseline model and even after fine-tuning of the model it was still extremely challenging to get a better result.

Overall, this analysis of individual domains further encourages us to go into the root cause of the problem as to why the BEIT-DomainNet was not that well learned like the other models which we have highlighted in the results. As was stated in this chapter earlier, our goal is to test the denoise ability of already trained models and to measure the resilience of our models, hence the next section will focus on the results of new experiments.

6.5.1 Results for PACS, Office-Home and DomainNet on Newly OOD Benchmarks

To extend the domain generalisation analysis and measure OOD potential of our model, Tables 6.1, 6.2, and 6.3 indicate the results on newly generated OOD benchmarks using GroundingDino and SAM methods on three benchmark datasets. By using these two model's architectures, we calculated the approximate segment mask of objects present in the image and created a periodic grid masking in the area of shape as shown in Figure 6.3. To test the OOD potential of the trained model, the method created 9 new data distributions with different occlusion ratios like 25% masking of objects, 50% masking of objects, and 75% masking of objects which can be seen in Figure 6.3. We performed this transformation on all three benchmarks and create OOD distributions that models have not seen before.

Table 6.1, like the other results tables in this chapter, has six columns explaining the number of grids, the benchmark, the top 1 and top 5 accuracy, and the gaps in accuracy between the PACS benchmark with occlusion, and PACS-original. The top 1 gap is calculated with the difference between the original top 1 score and

Table 6.1: Results on newly generated OOD PACS benchmarks using GroundingDino and SAM for image augmentation.

Number of Grids	**Benchmarks**	***Top1 Acc***	***Top5 Acc***	***Top1 Gap***	***Top5 Gap***
	PACS-original	0.9400	0.9980	-	-
2 Grids	PACS-occluded$_{25\%}$	0.9008	0.9921	-0.04	-0.01
	PACS-occluded$_{50\%}$	0.7804	0.9822	-0.16	-0.02
	PACS-occluded$_{75\%}$	0.7156	0.9744	-0.22	-0.02
5 Grids	PACS-occluded$_{25\%}$	0.8307	0.9813	-0.11	-0.02
	PACS-occluded$_{50\%}$	0.6616	0.9379	-0.28	-0.06
	PACS-occluded$_{75\%}$	0.6589	0.9517	-0.29	-0.05
9 Grids	PACS-occluded$_{25\%}$	0.6295	0.9576	-0.31	-0.04
	PACS-occluded$_{50\%}$	0.4103	0.9546	-0.53	-0.04
	PACS-occluded$_{75\%}$	0.5182	0.9428	-0.42	-0.06
5 Grids outside segments edges	PACS-occluded$_{25\%}$	0.5699	0.9379	-0.3701	-0.0601
	PACS-occluded$_{50\%}$	0.2800	0.7959	-0.6600	-0.2021
	PACS-occluded$_{75\%}$	0.2527	0.7800	-0.6873	-0.218

occluded data distribution scores. Similarly, the top 5 gap is calculated between the original top 5 accuracy metrics of the testing benchmark and the top 5 accuracy of newly generated benchmarks. Additionally, Table 6.1 demonstrates that when we occluded 25% of shape which is shown in the first row of Figure 6.3, our PACS-BEIT model still performs well on the PACS-occluded$_{25\%}$ OOD benchmark. The top 1 gap only increased by 8% from the original score and the top 5 gap is at 2%. This is a significant achievement for the model to show its resilience against outside noise/distractions. For PACS-occluded$_{50\%}$ and PACS-occluded$_{75\%}$, examples can be seen from Figure 6.3 that the transformer-based PACS-BEIT model decreases its accuracy to 28% to 29% from the original benchmark and it is fascinating to explore that even if 75% of the shapes are blocked by grid masks the model is stabilising itself at around 65% accuracy. Even though the top 1 accuracy has a declining trend for 50% and 75% occluded images in the dataset, the top 5 accuracy is still close to the original top 5 scores.

Table 6.1 also shows the transition of accuracy for the PACS benchmark when we increase the number of grids. Ultimately, this means reducing the grid size, and Figure 6.3 shows what each occluded ratio looks like in each case of the experiment. The main reason we conducted experiments based on a number of grids is that we would like to explore the overall trend of the model and what would be the optimal grid size to most affect the performance of the model and in which scenario generated PACS benchmark provides stable results regardless of blockages of visual information. Therefore, for 2 grids the PACS-BEIT model shows the most significant results compared to other implemented scenarios. Even though we have occluded 75% of shapes we still get 72% accuracy and only a 22% decrease compared to the original scores. When we block 25% of shapes a 4% decrease is observed in the top 1 score.

Hence, to expand the analysis, we increased the number of grids to 5 which means reducing the overall size of the grid unit model slightly dropping performance compared to the 2 grid benchmark. For 25%, 50%, and 75% occlusion ratios it drop 11%, 28% and 29% accuracy, respectively. To understand the reasons behind this drop, we further the decrease size of the grids and the findings are unique to observe. For example, when we block smaller chunks of information in the images, the PACS-BEIT model seems to suffer and significantly drop the accuracy parameter which means domain generalisation is most affected when we block smaller chunks of information with respect to bigger chunks. Therefore, for 9 grids, a 31% drop is observed for the 25% occluded benchmark images. Interestingly, another trend appears namely that when we increase the number of grids, PACS-occluded$_{50\%}$ performs poorer than PACS-occluded$_{75\%}$.

Table 6.2 offers similar results for the Office-Home benchmark. First of all, in the case of 2 grids, Office-Home is also less affected by the masking of shapes. It also provides proof of concept that when we corrupt big chunks of information, at the same time we also allow the model to see other big chunks to make the final predictions and remain less affected and utilise the available information best way. For instance, with 2 grids, the Office-BEIT model also shows similar patterns to PACS-BEIT and shows 5%, 14% and 20% drop in performance compared to the original baseline results.

For 5 grids, the Office-Home-occluded$_{25\%}$ has a 7% top 1 gap meaning the model decreases 7% accuracy compared to the original baseline results and 3% for the top 5 gap score. In the case of masking 50% and 75%, the Office-Home-BEIT model remains at 60% and 58% top 1 accuracy results which are surprising because even though Office-Home is a medium-level benchmark compared to PACS, and the model is performing well, it explains the potential of the model to have better generalisation. The top 1 gap is increased from 7% to 27% for 50% occluded benchmark

Table 6.2: Results on newly generated OOD benchmarks using zero shot instance masking for Office-Home.

Number of Grids	Benchmarks	*Top1 Acc*	*Top5 Acc*	*Top1 Gap*	*Top5 Gap*
	Office-Home-original	0.8691	0.9679	-	-
2 Grids	Office-Home-occluded$_{25\%}$	0.8203	0.9442	-0.05	-0.02
	Office-Home-occluded$_{50\%}$	0.7244	0.9034	-0.14	-0.06
	Office-Home-occluded$_{75\%}$	0.6692	0.8556	-0.20	-0.11
5 Grids	Office-Home-occluded$_{25\%}$	0.7987	0.9355	-0.07	-0.03
	Office-Home-occluded$_{50\%}$	0.6007	0.7956	-0.27	-0.17
	Office-Home-occluded$_{75\%}$	0.5810	0.7777	-0.29	-0.19
9 Grids	Office-Home-occluded$_{25\%}$	0.6297	0.8347	-0.18	-0.13
	Office-Home-occluded$_{50\%}$	0.5655	0.7796	-0.30	-0.19
	Office-Home-occluded$_{75\%}$	0.5338	0.7530	-0.33	-0.21
5 Grids outside segments edges	Office-Home-occluded$_{25\%}$	0.5484	0.7844	-0.3207	-0.1835
	Office-Home-occluded$_{50\%}$	0.2523	0.4472	-0.6168	-0.5207
	Office-Home-occluded$_{75\%}$	0.1799	0.3820	-0.6892	-0.5859

and even if we increase occlusion rate by 25% till the Office-Home-occluded$_{75\%}$, the model only shows a 29% drop which is 2% more than the Office-Home-occluded$_{50\%}$. A similar trend can be seen in the top 5 gap scores. The overall behaviour conveys an important message about learning generalised connections for the classes present in the images and being less affected by domain-shifting mechanisms.

Similar to the benchmarking results from PACS, when we enhance the number of grids to 9, a similar decline trend can be observed from Table 6.2. To the best of our understanding, this means that when we block bigger chunks of input image, the model is still able to use the relationship between different part of an image at feature levels and when we increase the number of grids, it becomes harder for the model to maintain its performance.

The results for DomainNet-occluded$_{25\%}$ shown in Table 6.3 show an approximately 10% reduction in top 1 accuracy compared to baseline scores and when we increase the occlusion rate to the 50% and 75% model shows 26% overall reduction in top 1 accuracy. The top 5 gap remains at 24% for both cases. As we know, DomainNet is one of the more complex and larger domain generalisation benchmarks

Table 6.3: Results on newly generated OOD benchmarks using zero shot instance masking using SAM and Grounding DINO for DomainNet

Number of Grids	Benchmarks	*Top1 Acc*	*Top5 Acc*	*Top1 Gap*	*Top5 Gap*
	DomainNet-original	0.6978	0.8793	-	-
2 Grids	DomainNet-occluded$_{25\%}$	0.6197	0.8283	-0.0781	-0.051
	DomainNet-occluded$_{50\%}$	0.5226	0.7407	-0.1752	-0.1386
	DomainNet-occluded$_{75\%}$	0.4805	0.6916	-0.2173	-0.1877
5 Grids	DomainNet-occluded$_{25\%}$	0.5898	0.8050	-0.10	-0.07
	DomainNet-occluded$_{50\%}$	0.4322	0.6361	-0.26	-0.24
	DomainNet-occluded$_{75\%}$	0.4326	0.6365	-0.26	-0.24
9 Grids	DomainNet-occluded$_{25\%}$	0.4882	0.7031	-0.21	-0.18
	DomainNet-occluded$_{50\%}$	0.3943	0.5840	-0.30	-0.3
	DomainNet-occluded$_{75\%}$	0.4094	0.6038	-0.29	-0.28
5 Grids outside segments edges	DomainNet-occluded$_{25\%}$	0.3527	0.5759	-0.3451	-0.3034
	DomainNet-occluded$_{50\%}$	0.1596	0.2714	-0.5382	-0.6079
	DomainNet-occluded$_{75\%}$	0.1542	0.2533	-0.5436	-0.6260

but in terms of generalisation concerns, the performance of the trained model did not drop regardless of domain shifting.

At the last rows of each table, it can observed that for such data distributions with shapes outside the shapes or edges, all three models show decreases in their performance. This also highlights another important point which is that our learned models focus on shapes as we stated in the claims. Therefore, when our methods create distributions with grids outside the segmented edges it deforms the shapes of objects and this has the greatest effect on the generalisation of models.

6.6 Conclusions and Lessons Learned

This chapter presented an investigation study incorporating two main aspects. The first was generating new OOD benchmarks by using PACS, Office-Home, and DomainNet and the development of a cascade network using Grounding DINO and SAM. This chapter also extended the original SAM Segment Anything Model to

work with text-prompts and also to boost the potential of SAM by computing bounding boxes for objects in images using another model namely Grounding DINO as shown in Figure 6.1 . The second aspect of the chapter was computing numeric scores in terms of accuracy and gaps to measure the DG of already trained models. The main idea behind this chapter was to measure the resilience of our fine-tuned models to explore how much noise or stretching of a distribution causes the performance got decease significantly.

The results of this chapter delivers extremely important pointers. For instance, Tables 6.1, 6.2 and 6.3 indicate that larger grids or chucks of masks blocking the objects in images have less distraction meaning there is less effect on the performance of the models. This also means that models are resilient to larger grid marks. Similar to this, when we gradually increase the number grids and decrease the size of the grids, the performance of models starts tp decrease meaning models are less resilient to smaller chunks or masks. One of the main reasons behind this finding is that models get distracted at the higher feature levels by having less relationship links between the different neurons.

We also discovered that for outside segment edge benchmarks, in summary, all the fine-tuned BEIT models perform poorly and this has the greatest distraction to models. Our previous claims were in Chapter 5 tjat to have a model focused on shapes of objects, and to the best of our understanding, when we apply outside segmented edge grid masking, it actually disturbs the shapes of objects with other features. As a result, all the models have less accuracy. It is also a clear indication that our models learn to focus on the shapes of objects as they got better results for the other three variations with occlusion inside the shapes. This can be interpret as a better DG models.

In this chapter, we have verified our previous claims about the denoising ability of the BEIT model and observed that in the case of all the new benchmarks, in the presence of 25% noise or masking, our model still yields state-of-the-art results. The performance of models for 50% and 75% occlusion ratios is largely effected only when we decreases the grid sizes.

In summary, in this chapter we explored research question 2 which is related to the measurement of variations and the stretcihness we can add to any data distribution and how such transitions will effect the overall accuracy or DG of a model. This chapter also targeted sub-hypothesis H3 according to which if a model is trained enough or generalised enough then we can possibly eliminate transfer learning. Our method did not have any transfer learning or new fine-tuning and in this chapter we calculated all the performance figures on previously trained models. This also highlights to what extent we can avoid transfer learning for unseen data distributions.

Chapter 7

Pruning of Vision Transformers & its Effects on Domain Generalisation

In chapter 6 we investigated the generation of noisy or masked data distributions which we considered as OOD benchmarks and the focus of the study there was to measure how much variation there is among benchmarks and how much this can have effects on the performance of trained models. This is related to the resilience of models and the generalisation to new data distributions which occurs in the zero shot manner using SAM and Grounding DINO. In chapter 6 the aim was to tackle RQ2 which raised a question about designing mechanisms to measure the stretchiness and its effects on the DG ability of models. Moreover, it also tries to answer sub-hypothesis H3, according to which if any model M is generalised enough then on new unseen but related benchmarks it may not need transfer learning or fine tuning.

This chapter 7 illustrates the understanding and a set of experimental works for term "pruning for neural networks". The chapter is not directly related to any specific research question from the thesis however it focuses on sub hypothesis H2 which is related to the exploration of dynamic learning methods that can have better and faster DG. Moreover, motivated from the conclusion of chapter 5, for larger and versatile benchmarks like DomainNet, a BEIT fine-tuned model does not learn new or unique features in the last 6 layers or attention heads. Therefore, compare to our previous experiments, our motivation is to train the vision transformers more efficiently thus saving time and computational resources. Hence, this chapter investigates methods of pruning in general and how can we prune vision transformers and what effect this will have on the DG of models.

By definition "Pruning" in neural networks initially refers to the process of elim-

inating specific neurons or connections from a trained model to reduce its size, complexity, or computational demands while striving to preserve its performance. This technique is commonly employed in neural network optimisation to enhance a model's efficiency, speed, and memory usage. Generally as a pruning target style, it can be divided into weight pruning, neuron pruning, filter pruning or channel selection/pruning. In **weight pruning**, connections with small weight magnitudes are pruned, or set to zero, based on a specific threshold or criterion. This process can occur either during or after the training of a neural network. The goal of weight pruning is to reduce the number of parameters in the network, thereby decreasing its memory footprint and computational demands during inference. In **neuron pruning**, entire neurons or units are removed from a neural network based on specific criteria, such as having low activity during training or not significantly contributing to the network's output. **Filter pruning** involves removing whole convolutional filters or channels from a CNNs depending on certain criteria, such as low relevance or activation during training. Filter pruning is a frequently used technique in CNNs to decrease the computational cost associated with convolutional operations [293].

Inspired from the results of chapter 5, in this chapter we will explore the concepts of pruning with vision transformers to address the issue of domain generalisation. To achieve our goal, we implement structural pruning methods including on vision transformers for PACS, Office-Home, and DomainNet, the three datasets used throughout the thesis. Moreover, there are many classifications within the pruning literature. For instance, pruning of neural networks can be divided into before training, during training, and after training. Similar to such methods, pruning of neural networks could also be structural and unstructural [293] and we will explore these alternatives.

The structure of this chapter has a comprehensive introduction and related work, then the unfolding significance of pruning and its various types as themotivation of using structural pruning for our experiments. Then a brief methodology is introduced to implement structural pruning for different vision transformers to explore the effects it has on domain generalisation benchmarks. The results sections will then discuss the trade off between speed, time and accuracy, with the overall relationship to the DG capabilities of models.

7.1 Introduction

As a consequence of the recent major developments in the field of AI especially in generative models and large language models, the compression of deep learning models has became an important area of interest. To deploy large models on edge devices, compression methods for neural networks have been developed as shown

in the literature [294, 295, 296, 297, 298, 299, 300, 301, 302, 303] which sometimes could be referred to as optimisation or quantisation to given hardware. Pruning has also become highly effective and practical among the various other compression paradigms [304, 305, 306, 307, 308, 309, 310, 311]. As we know, the goal of pruning a neural network is to remove the redundant parameters from it, which reduces the size and thus increases the inference speed of models. For example, the widely used ResNet-50 [34] requires about 95 MB of RAM and has over 23 million parameters [312]. For models like $BERT_{BASE}$ [313], the size is around 440 MB with 110 million parameters, GPT-3 contains up to 175 billion parameters [314], GPT-4 has even more and GPT-5 will have more again. The trend of developing and releasing larger models has become a race between big technology companies and research groups like Google, Meta, OpenAI and others. This creates the persistent need for pruning methods which could be used during the development to save the resources or after the development of ML models.

It is clear that DNN with higher parameters often require significant time and memory for processing, posing challenges for deployment on devices with limited computational resources (CPU, GPU and memory), such as those needed in real-time applications like autonomous driving. Furthermore, DNN with redundant features might reduce their resilience, increasing the danger of adversarial attacks. For example, the high-dimensional feature spaces generated by these networks might present more entry points for adversarial attacks, compromising the network's capacity to generalise beyond its initial training data [315]. All of this has led to increased interest in neural network compression techniques, such as pruning [316, 317], low rank factorisation [318], quantisation [319, 320], neural architecture search [321, 322], and knowledge distillation [323, 324] which aim to create lightweight models that reduce memory and computational demands while maintaining or improving performance. Pruning, in particular, has shown effectiveness in saving resources without sacrificing the accuracy of the original models.

Although pruning approaches can be divided into different schemes, first we are going to consider mainstream pruning approaches like structural pruning [325, 326, 327] and unstructural pruning [308, 328, 329]. Unstructured pruning is a strategy for lowering the size of a neural network by deleting unnecessary individual connections (weights). Unstructured pruning eliminates individual weights rather than whole neurons, filters, channels, or transformer attention heads or whole layers. The primary distinction between the two is that structural pruning alters the structure of neural networks by physically deleting grouped parameters, whereas unstructural pruning zeros partial weights without modifying the network structure. In reality, for small or medium models, unstructured pruning often sets their respective masks (or indicators) m to 0, rather than the weights themselves. In this situation, un-

structured pruning is defined as assigning a binary mask to each weight. In general assigning a mask to each weight in large models such as LLMs, is difficult because of the enormous number of weights [330, 331]. One of the main motivations for using structural pruning for our analysis is that it does not need any specific extra support from external hardware or software to deploy in practice as it can reduce size and speed up networks directly [332, 333, 326].

In this chapter, we implemented structural pruning methods on different vision transformers models like ViT, BEIT and DeiT for domain generalisation benchmarks. Inspired from [334], the pruning mechanisms consist of finding a dependency graph, then grouping and group level pruning for the selected model. The implemented structural pruning removes sets of parameters scattered over many levels. The parameters within each group are interdependent because of layer-to-layer linkages, therefore they must be pruned simultaneously to maintain the model's structural integrity. The method employs a technology called "DependencyGraph" to automatically recognise these relationships and collect parameter groupings for pruning [334]. Then it prunes the pre-trained models from base settings into new models which will be structurally different from the original models and we can then fine-tune the new pruned models for domain generalisation benchmarks. In the end, the fine-tuned models are tested on testsets and OOD masked benchmarks which we generated earlier in chapter 6.

7.2 Related Work

In this section, we explain the major literature contributions in pruning across the topic of "when to prune the models or networks". Even though we can find plenty of related work in various fields of pruning like Pruning Criteria, Learn to Prune, and more, in this chapter we only focus and divide the previous works into basic structure according to types namely pruning after training, pruning during training, and before training.

7.2.1 Pruning Before training

Pruning before training is a strategy for sparsifying the structure of a neural network before starting the training process. Instead of training a whole network and then pruning it after or during training, the model is trimmed at the start or early phases of training. This strategy minimises the model's size from the start, which may result in more efficient training and inference while preserving model performance.

Given an initial weight matrix $W_0 \in \mathbb{R}^{n \times m}$, the goal is to prune certain connections to reduce the computational cost and memory footprint without degrading

the model's performance. We define a binary mask matrix $M \in \{0,1\}^{n \times m}$ which is applied to the weight matrix at initialisation. The pruned weight matrix $\tilde{W}_0$ can be computed as:

$$\tilde{W}_0 = W_0 \odot M$$

Here, $\odot$ denotes element-wise multiplication, and the binary mask M is generated based on a predefined criterion (e.g., random, magnitude-based, or learned masks). The pruned weight matrix $\tilde{W}_0$ is then used to start the training process. In this approach, certain weights are permanently zeroed out, meaning they do not participate in the learning process. The goal is to ensure that the sparse network performs comparably to a dense network.

Moreover, one of the important work in early stage pruning of neural network is known as the "Lottery Ticket Hypothesis" [335]. The "Lottery Ticket Hypothesis" proposes that within a randomly initialised dense neural network, there exists a smaller, sparse subnetwork (a "winning ticket") that can be trained to match the performance of the original dense network/models when trained in isolation. This approach includes training the network once, pruning, resetting the pruned weights to their initial values, and then retraining the sparse network to determine the winning ticket. This technique has demonstrated that sparse networks can perform as equal to dense networks. This method of the winning ticket requires iterative pruning and retraining which can be computationally costly. In the series of lottery ticket hypothesis, another famous article called single-shot network pruning (SNIP) has been proposed [336]. The idea of SNIP is simple: we have some randomly initialised mesh and we leave the most promising connections in it. The most promising are determined by the so-called connection sensitivity means how much the removal of a specific weight affects the loss, so that the least influential ones are removed.

Furthermore, the importance of each weight is determined by computing its gradient in relation to the loss function during the initialisation phase. While computationally efficient, SNIP struggles to prune large percentages of weights, resulting in a performance loss [336]. To extend the functionality of SNIP, the gradient signal preservation (GraSP) method was developed [330]. When we trim a model before training, it generate sparse structure of neural networks which are challenging to train. Therefore, GraSP enhances SNIP by focusing not just on weight significance but also on preserving the gradient flow after pruning, which assists in training stability. It establishes a pruning criterion depending on how much the gradient signal changes after pruning. The purpose is to eliminate weights that have little effect on gradient flow during training. This method outperforms SNIP in terms of training stability and performance because it preserves the gradient signal, particularly in

severely pruned networks. Because of its emphasis on gradient preservation, GraSP requires more processing resources than SNIP does [330]. The authors of the current work argue that at the beginning of training, it is more important to maintain the overall dynamics of training than the loss itself. SNIP explicitly does nothing of the kind and can disrupt the flow of information over the network.

Existing gradient-based pruning strategies, which eliminate portions of a neural network at the start of training, can occasionally result in layer collapse. Layer collapse occurs when an entire layer of the network is mistakenly pruned (removed), rendering the network incapable of learning or functioning properly. The literature has introduced a new pruning method called iterative synaptic flow pruning (SynFlow), which avoids layer-collapse [337]. SynFlow prunes a network by examining the synaptic flow of information throughout network. It focuses on treating all parameters equally regardless of layer depth and avoids layer collapse during pruning. This provides importance scores to weights depending on how they contribute to the flow of information across the network rather than just gradients. This results in a more balanced pruning technique throughout the network. Despite being successful in preventing layer collapse, it can be computationally costly to execute [337].

7.2.2 Pruning During Training

Pruning during training refers to progressively removing the most unimportant weights or neurons from a neural network while it is still being trained, unlike post-training pruning where pruning is applied after the model is fully trained. Such methods allow the models to dynamically adjust and recover from pruning during the training process, helping it maintain performance while becoming more efficient.

Mathematically, let us consider the weight matrix at a given training step t as W_t. A binary mask $M_t \in \{0,1\}^{d_1 \times d_2}$ is applied to the weight matrix, where the mask determines which weights remain active. The effective weight matrix at step t, after pruning, is:

$$\tilde{W}_t = W_t \odot M_t$$

where $\odot$ also denotes element-wise multiplication. During training, the mask M_t is updated based on a pruning criterion such as weight magnitude, gradients, or other performance metrics. The training process aims to minimise the loss function like commonly use ML:

$$L(W_t \odot M_t)$$

subject to a sparsity constraint:

$$\sum_{i,j} M_{i,j} \leq s$$

where s denotes the desired level of sparsity, i.e. the number of active weights to maintain during training. Typically, it begins with a randomly initialised dense network as the input model. It simultaneously trains and prunes the neural network by updating both the weights and the associated masks (which may represent weights, filters, channels, etc. and the approaches described in [338, 339, 340, 341, 342] are well-known for pruning during training. In addition, according to the literature, during pruning the methods can be further divided into four types including 1) sparsity regularisation based [343, 344, 345], 2) dynamic spare training based [338, 346, 347, 348, 349, 350, 351], 3) score-based [340, 330, 352], and 4) differentiable pruning based methods [353, 342, 354].

7.2.3 Pruning After Training

Pruning after training is the practice of eliminating redundant or less significant parameters from a neural network once it has been fully trained. Most pruning methods [355, 326, 294, 328, 356] work with a pre-trained network. The fundamental concept here is to find which weights are most redundant and, as a result, will have the least impact on performance when removed. Magnitude-based pruning algorithms eliminate weights that are less than a threshold, which may miscalculate the relevance of each weight [294]. In contrast, Hessian-based pruning algorithms determine the significance of each weight by calculating how its removal would effect the loss [356] . However, all of the aforementioned solutions need pre-training and hence are not relevant at initialisation.

Pruning after training is a popular strategy in neural network optimisation that reduces model size while improving inference performance. This type of approach can also be divided into four types including magnitude-based, iterative, structured, and dynamic pruning. Each has its own set of advantages and disadvantages. Magnitude-based approaches are straightforward but they have an unevenly effect on layers whereas iterative pruning allows for performance recovery. Structured pruning maximises hardware efficiency, whereas dynamic pruning provides flexibility during inference. According to the literature, the proper pruning strategy is determined by the unique use case, model architecture, and deployment environment [293].

7.3 Methodology

This section explains the procedure that our method follows to prune vision transformers. The method is inspired by original paper [334] which we used as a tool to design our own framework. As we implemented the structural pruning to minimise the size of neural networks it deals with whole structures such as filters, channels, or layers, rather than individual weights. Therefore, the idea needs to handle structural dependencies in neural networks and ensuring that after finding the dependency and pruning the networks it should maintain the overall structural and stability of the network.

7.3.1 Finding Dependency for Networks

There are several steps involve in the process of finding the dependencies before the pruning can commence, such as the following:

Dependency Graph Construction: Here, the neural network is represented as a Dependency Graph (DepGraph) [334]. Each layer, filter, or channel in the network is viewed as a node in the graph, and the relationships between them (such as layer dependencies) are shown as edges in the graph. These links demonstrate how one node depends on another for information or functionality.

Identifying Dependencies: In this approach the first stage in the process is to automatically analyse the neural network to discover which components (nodes) are interdependent. This dependence analysis is crucial since trimming one element of the network may influence another. If two layers are connected, removing one without taking into account its relationship to the other can cause the model to fail.

Let us consider a neural network N which can be represented as a set of layers, filters, or channels:

$$N = \{L_1, L_2, ..., L_n\}.$$

Each element of the network has dependencies on other elements in terms of connections, which can be fully connected layers, skip connections, concatenation or residual connections. This can be presented as a dependency graph $G = (V, E)$, where:

- $V = \{v_1, v_2, ..., v_n\}$ refers as a set of nodes of network components (like filters, channels, etc.),
- E is the set of directed edges representing dependencies between nodes.

Pruning Strategy Once the dependency graph has been constructed, a pruning approach is implemented. The essential insight here is that groups of dependent parameters should be trimmed simultaneously. If one filter/layer is pruned and has dependencies on other filters/layers, the other filters/layers must also be trimmed in order to preserve structural consistency. In **Dynamic Pruning** the designed framework which is initially motivated by [334] supports dynamic pruning across many structures (filters, channels, and layers) and modifies the network structure accordingly. Pruning is not confined to a particular type of structure and may be applied to several designs, making it very adaptable. However, in our case, we mostly prune the structure of pre-trained models and then perform fine-tuning for domain generalisation. In future, after new modifications, dynamic pruning or sparse structural pruning can also be implemented for vision transformers.

Each node v_i has a score $S(v_i)$, also known as important score in our case, which quantifies its importance (e.g., weight magnitude, or gradient). In our experimental approach, we use 5 types of importance score metrics including, L_1 norm, L_2 norm, Taylor, random, and Hessian. These metrics set a threshold θ, and nodes with scores below this threshold are pruned:

$$\text{Prune node } v_i \text{ if } S(v_i) < \theta.$$

However, due to dependencies when a node v_i is pruned all nodes v_j dependent on v_i must also be pruned. This relationship is defined by the edges in the graph $(v_i, v_j) \in E$. The pruning process is formalised as:

$$\text{For each node } v_i \in V, \text{ if } S(v_i) < \theta, \text{ prune } v_i \text{ and all nodes } v_j \text{ such that } (v_i, v_j) \in E.$$

The above equation ensures that if a component/element is selected to prune then all dependent components/elements also are pruned, preserving the structural consistency of the network.

Moreover, for dynamic pruning this applies to different types of structure in the network. For example, a filter F, a channel C, or an entire layer L can be treated as nodes in the dependency graph. The pruning criterion applies to each structure s:

$$\text{Prune structure } s \text{ if } S(s) < \theta,$$

where $s \in \{F, C, L\}$.

Graph Optimisation The approach we implemented refines the dependency graph once it has been created and pruned after performing the above process. The objective is to reduce network size while maintaining performance. Fine-tuning

involves retraining the model after pruning to recover accuracy lost throughout the process.

Let the pruned network be represented by N_p. The training objective is to minimise the loss $Loss$ over N_p:

$$\min_{W_p} Loss(N_p(W_p)) \quad \text{subject to} \quad \|W_p\|_0 \leq \kappa,$$

where W_p expresses the weights of N_p and κ is a constraint on the number of remaining parameters.

7.3.2 Pruned Vision Transformer Models Fine-Tuning on Domain Generalisation Benchmarks

In the framework which was designed and built for these experiments, we implemented after training structural pruning on pre-trained models from the open source community. For this purpose, first we load and convert the pre-trained weights according to our own benchmarks which are PACS, Office-Home and DomainNet. For the investigation of pruning of a vision transformer, we perform experiments with three models namely ViT DeiT and BEIT with different variations which can be found in the experiments details.

Our framework runs an importance score criterion on each model with selected hyperparameters and benchmarks. The pruned weights and structures are then stored for each one and the pruning weights will be further fined-tuned for evaluation on domain generalisation benchmarks. In addition, it is important to mention that the pruning step and the fine-tuning of the models will have different sets of hyperparameters.

7.3.3 Inference of New Fine-Tuned and Pruned Models

After preforming pruning of pre-trained weights and fine-tuning of these weights according to our DG benchmarks, the final step is to evaluate the performance of the pruned models on test distributions. Like our previous research in earlier chapters, we divide the each DG benchmark into a 80% / 20% ratio of training+validation and testing. Tables 7.1, 7.2, 7.3, and 7.4 will contain our experimental results. Figure 7.1 illustrates the working of our structural pruning.

7.4 Experiment Details

In the experimental setup for our structural group pruning for vision transformers, our method first loads any specific custom new data distributions and converts

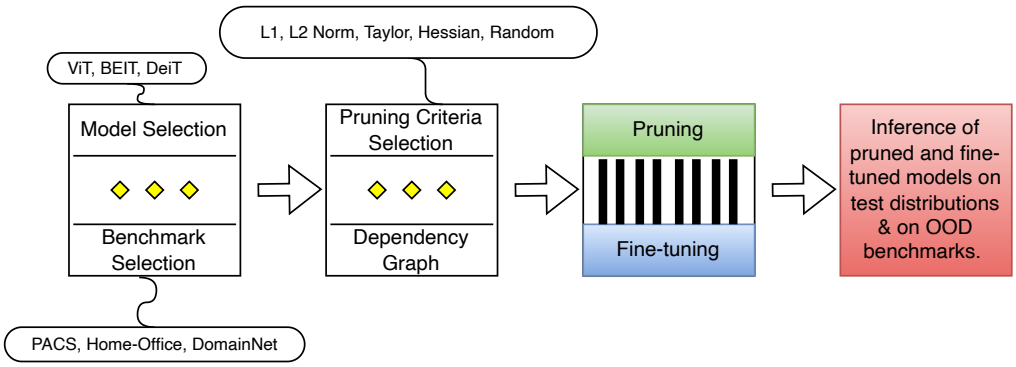


Figure 7.1: Overview of group structural pruning for vision transformers.

them into the ImageNet data format so that we can avoid the data related issues and construct the framework in a generalised manner. Our method then converts the domain generalised benchmarks PACS, Office-Home, and DomainNet into ImageNet data format. Then, the method prunes the pre-trained models including ViT, BEIT, and DeiT. There are many open source versions of these model available however we specifically use base versions of models namely vit-base-patch16-224, beit-base-patch16-224, and deit-base-distilled-patch16-224 respectively at the first initialisation of model weights.

After initialisation of weights, we apply group pruning based on importance score metrics namely L1 & L2 Norm [326], Taylor [357], Hessian [356], and random. In the group structural pruning, we remove groups of filters, attention heads, channels, or layers of models and the elimination rate will be decided by the importance metrics. For instance, L1&L2 Norm are motivated from this paper [326] where the authors explain that usually both norms have similar response in the selection process with only small differences. Similarly, the Taylor importance estimation is inspired by the original paper [357] which implemented first-order and second-order of Taylor importance on filters or neurons. Moreover, the results of [357] shows that both methods are extremely close in terms of performance therefore we only implemented the first-order Taylor importance method to trim groups of layers. In addition, the Hessian group pruning is also known as optimal brain damage (OBD) in the literature [356]. OBD use information-theoretic principles to determine which weights in the network may be deleted with lowest effect on performance. The approach involves determining the second derivative of the error function with respect to the weights in order to discover those that contribute least to the network's performance [356]. Finally, in the random group selection, we pick and drop groups of (neurons) randomly without any importance selection mechanism to check the general performance of sub-networks already present in the pre-trained weights.

	ViT	BEIT	DeiT	ViT	BEIT	DeiT	ViT	BEIT	DeiT
Hessian	1.59	3.11	2.12	1.15	2.42	0.57	0.42	2.16	0.38
Taylor	2	3.12	2.12	0.55	2.41	0.5	0.35	2.17	0.35
Random	1.32	3.11	1.28	1.02	2.42	1	0.46	2.17	0.37
L1	2.04	3.11	1.58	1.49	2.42	0.39	1.24	2.16	0.36
L2	1.58	3.12	1.59	0.55	2.42	0.39	0.41	2.16	0.36
	50%			75%			95%		
	×	×	×	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	×
	50%			75%			95%		
	×	×	×	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	×
	×	×	×	×	×	×	×	×	×
	50%			75%			95%		

Figure 7.2: Summary of experiments.

Figure 7.2 illustrates the types of experiments performed in this chapter. Each row illustrates the details for each DG benchmark including PACS, Office-Home and DomainNet, respectively. The first three tables show the experiments and give the time needed to fine-tune each model across individual variations. This explains the configurations for the execution plan for our experiments on pruning. Each table has 15 values of fine-tuning time for each model including ViT, BEIT, DeiT with respect

to pruning criteria namely Hessian, Taylor, random, L1, L2 and pruning ratios of 50%, 75%, and 95%. The first table has timing information for experiments for the PACS benchmark while the second and third tables could be filled with fine-tune timings for the Office-Home and DomainNet benchmarks which could be compiled if needed in future work.

As mentioned in Section 7.3, there are two major steps involved in the development including pruning of vision transformer and fine-tuning the pruned models. The pruning process has hyperparameters like pruning ratio, number of heads, dimensions of output heads, and global pruning. The term "global pruning" means it takes the entire network into account and prunes weights across all layers by applying a more balanced reduction of network size. On the other hand, local pruning focuses on the elimination of individual layers or specific sections of the network by frequently using criteria such as weight magnitude. There is another type of pruning method which has been introduced recently called isomorphic pruning [358]. This approach preserves the structural similarity between the pruned and original networks and guarantees that the pruned network keeps the same architecture but with fewer parameters.

For our experiments on vision transformers, we implemented global pruning with three variation in pruning ratio namely 50%, 75%, and 95%. The number of heads are reduced to 50% from 144 to 72 with each head's dimentionality as 64. This settings is applied on all the models with selected importance score metrics and for each base model we have 15 pruned variations and the performance results for this are shown later.

In the fine-tuning of stored pruned architectures of models, we have a range of different sets of hyperparameters such as epoch, batch size, optimiser, initial learning rate, momentum, weight-decay, learning rate scheduler, learning rate warm up method, and learning rate warm up-decay. The fine-tuning process will continue in each experiment untll 300 epochs, with AdamW as the optimiser, 0.00015 as the initial learning rate, 0.9 as momentum, 0.3 as weight-decay, CosineAnnealingLR as the learning rate scheduler, linear as tlearning rate warm up method, and 0.033 as the learning rate warm up decay. At this stage of the development process, we did not implement sparse or dynamic training and pruning methods for vision transformers to explore DG. Hence, after the fine-tuning of pruned models, the performance of best weights is given in Tables 7.2, 7.3 amd 7.4. Similarly, Figures 7.3, 7.4 and 7.5 illustrate the validation accuracies of each model across all the variations of pruning setups.

Figure 7.3 shows validation accuracy for the **ViT** model under different pruning ratios (50%, 75%, and 95%) with importance score metrics including Hessian, Taylor, random, L1 and L2 Norm. In 50% half of the parameters are pruned, in 75 % three-

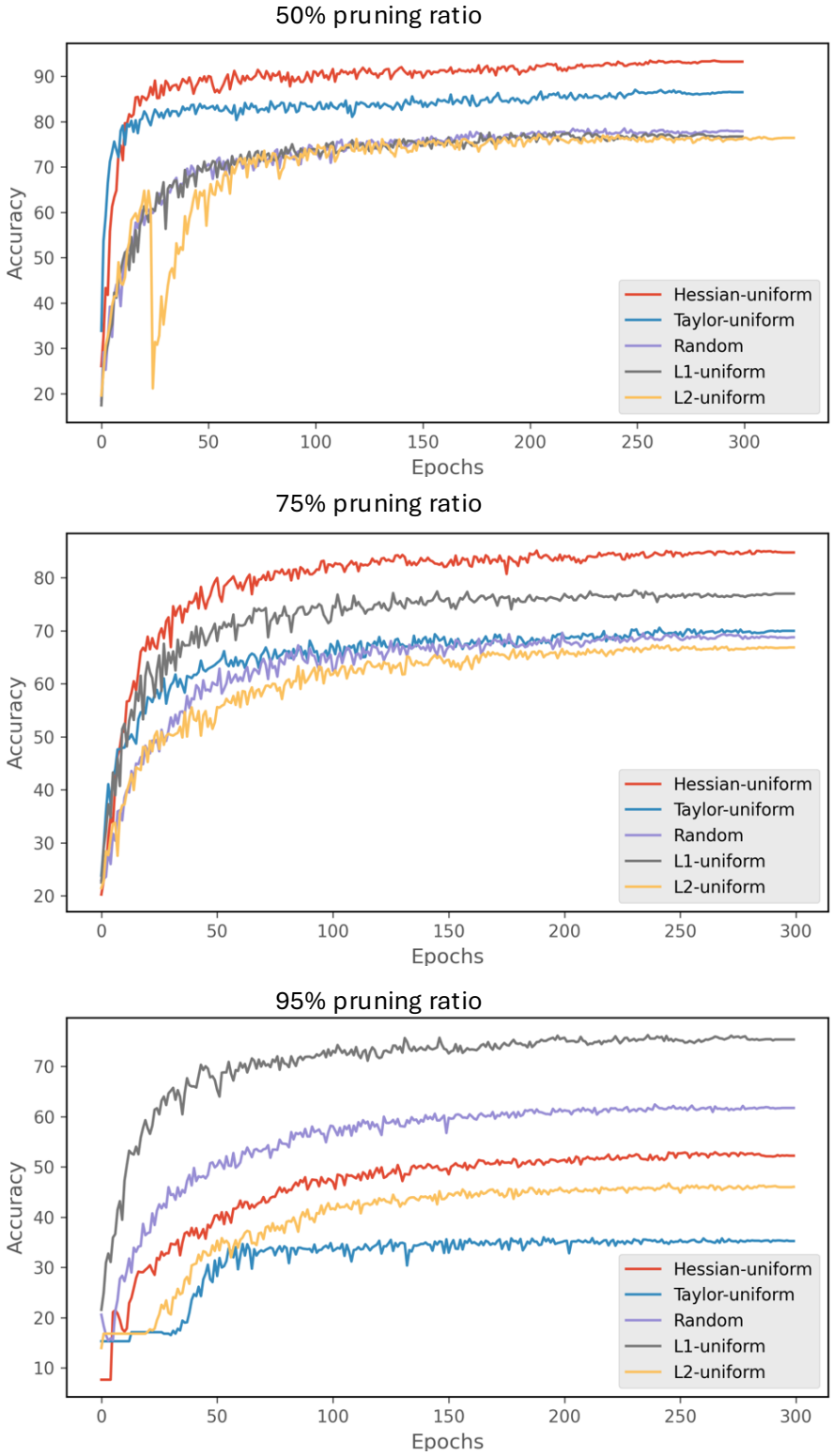


Figure 7.3: Accuracy for 3 different pruning ratios, 50%, 75%, and 95%, for ViT model on PACS benchmark.

quarters of the parameters are pruned, and in 95% pruning nearly the entire model is pruned, keeping only 5% of the original parameters. The graph shows that 50% pruning has overall better performance compared to other two because all 5 versions

of ViT models have more original structures.

In the 50% pruning ratio, the Hessian performs better with 90% accuracy and Taylor is at 80% accuracy, but the other three are at almost 70% after 300 epochs of training. Moreover, the L1 and random importance metrics indicates almost similar behaviour during the training process.

Interestingly, when we further increase the pruning ratio to 75% the Hessian importance metric still shows higher performance which is slightly over 80% but for Taylor the model drastically decreases the accuracy and L1 norm metric still remains at 70% and shows the second highest performance for a model. Random and Taylor metrics show similar behaviour and L2 norm shows lowest accuracy.

In the same way, after raising the pruning ratio to 95% the ViT model shows highest accuracy for L1 Norm which is still around 70%. The random selection becomes second best and Hessian's accuracy declined to 50% and Taylor shows the lowest scores. In summary, L1 Norm shows stability in terms of accuracy in all cases however, for 50% and 75% the Hessian selection has better results.

In the case of the **BEIT** model whose results are shown in Figure 7.4, for 50% all the models highlight a similar pattern for accuracy with the performance at around 90%. Nevertheless, at a pruning ratio of 75% the BEIT model provides unique insights. For example, in the case of 75% pruning, when we keep only 25% of the original model and fine-tune this remaining structure of networks for 300 epochs it can actually increase the potential of the model. The second graph in Figure 7.4 demonstrates a boost in the performance of all the models under each importance score criteria. The Hessian crossed the overall accuracy of 90% which is surprising, even with 25% of model size and operations compared to the base model. The other four models also have an accuracy score between 80% to 90%. At 95% pruning, Taylor metrics becomes highly efficient in terms of accuracy with overall score more than 80% and random selection becomes the lowest. The other three models show an almost similar structure of accuracy.

The graphs in Figure 7.5 report the accuracy performance figures for the DeiT model. According to the first graph with a 50% pruning rate, during the fine-tuning the Hessian based pruning DeiT model adapted to the PACS benchmark after first epoch with 90% accuracy which is a good sign for domain generalisation of the model. The Taylor based model has the second best performance from among the others and L2 Norm performs better than random and L1 Norm. After enhancing the pruning ratio to 75% and 95%, it is observed that L1 Norm and L2 Norm based models display an immediate decline in their accuracy. It is also important to note that at the highest pruning rate which is 95%, random selection is the better option compared to others. This means that for DeiT base networks, the other importance score metrics fail to keep the most important links or neurons in the models.

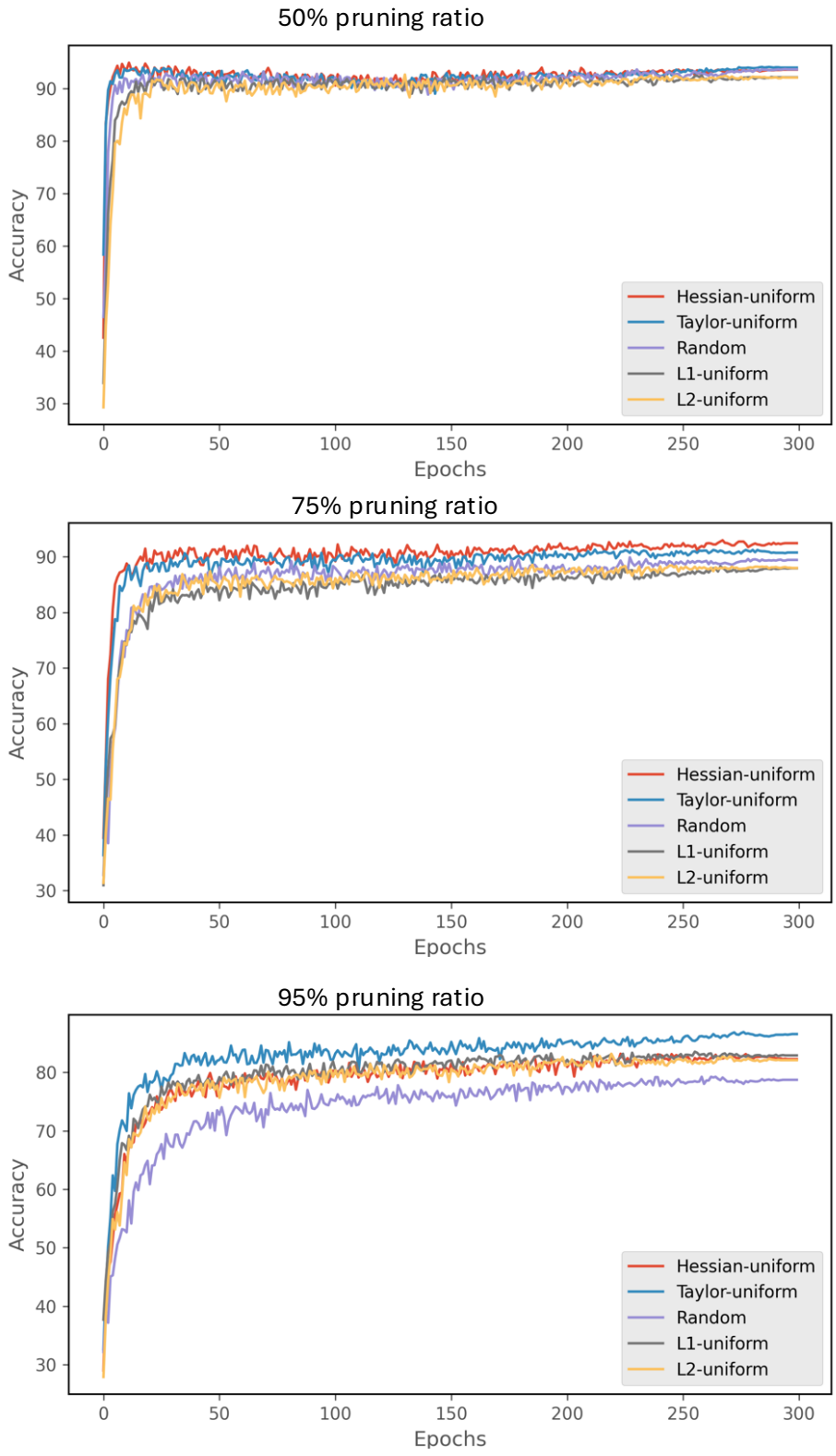


Figure 7.4: Accuracy for 3 different pruning ratios, 50%, 75%, and 95%, for BEIT model on PACS benchmark.

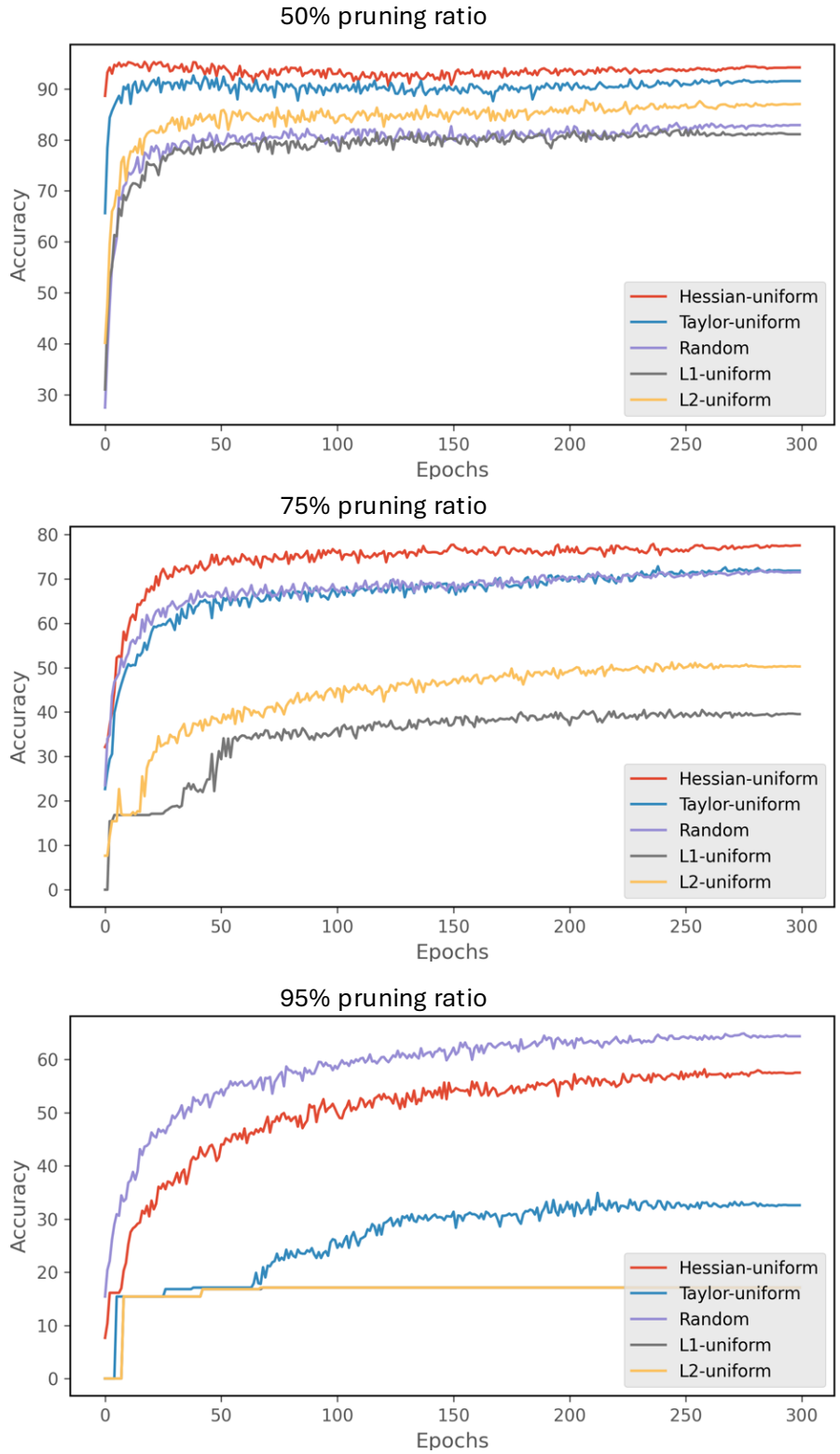


Figure 7.5: Accuracy for 3 different pruning ratios, 50%, 75%, and 95%, for Deit vision transformer on PACS benchmark.

7.5 Results and Discussion

In this section we focus on the experimental results for grouped structural pruning on vision transformers namely ViT, BEIT, DeiT for the PACS benchmark. The

results section has an analysis of the PACS benchmark by showing accuracy learning graphs for each model and here we present detailed tables to further understand their behaviour in general.

Table 7.1: Comparison of base model performances on key metrics namely the number of trainable parameters, computational cost (MACs), accuracy, and fine-tuning time for PACS.

Model Name	***# Params.***	***MACs***	***Valid Acc. (%)***	***Test Acc. (%)***	***FT-Time***
ViT-Base	86.57 M	17.59G	94.43	94.33	04:00:54
BEIT-Base	86.53 M	12.67G	96.41	95.64	04:10:24
Deit-Base	87.34 M	17.69G	96.87	96.56	04:01:35

Our first evaluation figures are for the base versions of vision transformers because it is important to know the standard behaviour of these vision transformers for their base models before any pruning. Table 7.1 has six columns with the headings of model name, # Params which is the total number of trainable parameters in each model which is expressed in millions (M), Multiply-Accumulate Operations (MACs) shows the number of operations required to perform a single forward pass as measured in gigas (G) which represents billions of operations, validation and test accuracy and fine-tuning time measured in hours, minutes, and seconds (HH:MM:SS). We selected the three vision transformers because they are unique and different to each other. ViT-Base (Vision Transformer) means ViT specificially vit-base-patch16-224, BEIT-Base (Bidirectional Encoder Representation from Image Transformers) means beit-base-patch16-224, and Deit-Base (Data-efficient image transformers) means deit-base-distilled-patch16-224 model.

ViT-Base contains 86.57 million parameters, BEIT-Base has 86.53 million and Deit-Base has 87.34 million. More parameters often suggests a bigger model which may have more capacity but with higher computing requirements. In the same way, for MACs ViT-Base requires 17.59G MACs, BEIT-Base requires 12.67G, and Deit-Base needs 17.69G. Lower MACs indicates that the model is more efficient in terms of computing, whereas higher MACs indicates that it uses additional resources. For performance, ViT-Base scored 94.43%, BEIT-Base 96.41%, and Deit-Base 96.87% in the validation accuracy. Higher validation accuracy means higher generalisation to unseen data after training. Similarly, test accuracy measures the performance of models on the test dataset/distribution and it is another indicator of how effec-

tively a model generalises. In Test Acc metrics, ViT-Base scored 94.33%, BEIT-Base 95.64%, and Deit-Base 96.56%. The test accuracy is slightly lower than the validation accuracy which is usual because the test dataset is often more difficult. FT-Time means the time required to fine-tune the model and Table 7.1 highlights that Deit-Base performs the best in terms of validation and test accuracy, while having the highest number of parameters and MACs. This results reflects another vital insight which we have already established during the previous chapters, that bigger models usually have better domain generalisation capabilities. On the other hand, BEIT-Base is the most computationally efficient (lowest MACs) while yet providing excellent performance. ViT-Base has less accuracy than the others although it is comparable to Deit-Base in terms of size and processing needs.

7.5.1 Group Structural Pruning Results for ViT Transformer

We now present detailed analysis of our experiments to explore the domain generation with pruning on vision transformers. Table 7.2 shows pruning results for the ViT model at three distinct pruning ratios, 50%, 75%, and 95%. The table compares the impact of several pruning strategies (Hessian, Taylor, Random, L1-Norm, and L2-Norm) and has 9 columns including, pruning ratio, group importance score, pruned parameters, pruned MACs, pruned validation and testing accuracy, δAcc which is the measure of change in accuracy and can be calculated by subtracting pruned test accuracy in Figure 7.3 from base test accuracy in Figure 7.1, speedup and fine-tuning time. Theoretically, the term speedup can be calculated by Speed Up $= \frac{\text{MACs(base)}}{\text{MACs(pruned)}}$.

Firstly at 50% pruning, the Hessian based ViT model produces the best results with a validation accuracy of 93.5% and a test score of 91.39%, with just -2.94% decline. It also provides a reasonable 2.5x speedup with a fine-tuning time of 01:59:08, this also means that the base ViT model has original parameters of 86.57 millions which were reduced to 35.03 million and MACs decreased from 17.59G to 6.64G. On the other hand, for the Taylor method, we notice a significant reduction in performance with a testing accuracy of 84.70% and an accuracy loss of -9.63%. Taylor compensates partially by delivering a 2.6x speedup over Hessian. Random pruning and L1-Norm techniques exhibit the significant decreases in performance with testing accuracies of 78.22% and 73.76%, respectively. For both, the accuracy losses reach -16.11% and -20.57%. However random pruning delivers the highest speedup (3.95x), but at the expense of a significant loss of accuracy. L2-Norm operates similarly to L1-Norm in terms of validation results. For 50% pruning experiments, random pruning takes less time to train compared to others.

Moving towards 75% pruning experiments for the ViT model, we observe that

Table 7.2: Pruning methods on the ViT model at various pruning ratios.

Pruning Ratio	**Group Importance Score**	***Pruned Params***	***Pruned MACs***	***Pruned Valid Acc (%)***	***Pruned Test Acc (%)***	**Δ*Acc (%)***	***Speed Up***	***FT-Time***
ViT-50%	Hessian	35.03M	6.94G	93.5	91.39	-2.94	2.5x	01:59:08
	Taylor	33.68M	6.66G	87.01	84.70	-9.63	2.6x	02:00:45
	Random	21.01M	4.45G	78.49	78.22	-16.11	3.95x	01:32:15
	L1-Norm	34.85M	6.88G	77.62	73.76	-20.57	2.55x	02:04:31
	L2-Norm	31.03M	6.21G	77.22	75.38	-18.95	2.83x	01:58:12
ViT-75%	Hessian	12M	2.72G	85.10	83.79	-10.54	6.5x	01:15:18
	Taylor	6.19M	1.75G	70.55	70.11	-24.22	10.05x	00:55:23
	Random	7.02M	1.91G	69.62	66.87	-27.46	9.21x	01:02:32
	L1-Norm	30.45M	5.84G	77.56	75.18	-19.15	3.01x	01:49:16
	L2-Norm	4.98M	1.56G	67.25	66.87	-27.46	11.28x	00:55:22
ViT-95%	Hessian	0.60M	0.83G	52.93	50.15	-44.17	21.19x	00:42:12
	Taylor	0.11M	0.73G	36	32.22	-62.11	24.10x	00:35:21
	Random	1.03M	0.89G	62.44	62.01	-32.32	19.76x	00:46:00
	L1-Norm	18.68M	3.52G	76.23	73.76	-20.57	5.0x	01:24:32
	L2-Norm	0.21M	0.76G	46.73	43.97	-50.35	23.14x	00:41:28

pruned parameters and MACs are removed even more when compared to the original base model in Table 7.1. For example, Hessian still maintains the best accuracy due to pruned test accuracy of 83.79% and it only shows performance decline of -10.54% with a huge 6.5x speedup. Table 7.2 also indicates Hessian is most efficient when compare to the second best model L1 Norm as it has fewer parameters and MACs. The Taylor selection method experiences substantially steeper reduction in accuracy with pruned testing accuracy of 70.11% and -24.22% accuracy loss, but it compensates with an speedup of 10.05x. Meanwhile, Taylor and L2 Norm experience almost the same fine-tuning time of 55 minutes. Both Random and L2-Norm perform poorly as their pruned test accuracies are at only 66%. However, L1-Norm outperforms Random and L2-Norm because its pruned test score 75.18%

but has a slowest speedup of 3.01x when compared to all other models.

Finally, for 95% pruning ratio of the ViT, Hessian retains comparable performance although testing accuracy after pruning and fine-tuning decreases dramatically to 50.15% with -44.17% δAcc. The Taylor method has the most severe accuracy reduction in testing metrics at 32.22% and has accuracy loss of -62.11%, however it achieves the greatest speedup at 24.10x. Random pruning outperforms Taylor, preserving a score of 62.01% at 19.76x speedup and this suggests that despite its unpredictability, it maintains some level of precision. L1-Norm maintains a comparatively smaller accuracy loss of -20.57% with the pruned testing accuracy of 73.76% although it gives a slower speedup of 5.0x than the other approaches. One of the other reasons behind the better performance of L1 Norm could be more clear if we examine the pruned parameters and MACs columns in Table 7.2 which indicate that even though the size of L1 based reduces compared to 75% pruning but it still has a larger structure when unpruned compared to others. The number of parameters for all models except L1-Norm which is 18.68M becomes less or equal to 1M and MACs goes to less than 1G, hence random selection is not performing so badly after all giving us second best model at higher pruning rates. It also conveys another message to us that at higher pruning rates with the help of random selection we can try to get sub-networks which still have better potential to learn compared to other selection methods.

In summary, for the ViT model across all pruning ratios, Hessian performs better in most cases except at extremely higher pruning rates where it immediately displays large losses. L1-Norm provides more balance and stable performance for domain generalisation benchmarks by always maintaining its accuracy above 70%.

7.5.2 Group Structural Pruning Results for BEIT Transformer

Table 7.3 also has the same evaluation metrics to those in 7.2. In case of 50% pruning, the Hessian technique maintains the maximum pruned test accuracy of 94.23%, with a minor loss of -1.42% compared to its base model on the PACS benchmark, which is good performance given that the model is trimmed about 57.79M parameters and reduced MACs to 7.01G. The speedup is moderate (1.81x) which is still faster than the original model and take less fine-tuning time of 03:11:36 when compared to the baseline. This implies that Hessian prioritises accuracy retention above speedup, making it suited for applications where preserving accuracy is more crucial than speeding up calculations. The Taylor pruning follows closely and has pruned test accuracy of 93.52%. The speedup is equivalent to 1.80x, indicating that the Hessian and Taylor techniques perform similarly in terms of computing benefits,

Table 7.3: Pruning methods on the BEIT model at various pruning ratios.

Pruning Ratio	**Group Importance Score**	***Pruned Params***	***Pruned MACs***	***Pruned Valid Acc (%)***	***Pruned Test Acc (%)***	**Δ*Acc (%)***	***Speed Up***	***FT-Time***
BEIT-50%	Hessian	57.79M	7.01G	94.96	94.23	-1.42	1.81x	03:11:36
	Taylor	58.03M	7.06G	94.15	93.52	-2.13	1.80x	03:12:06
	Random	58.17M	7.09G	93.74	92.91	-2.74	1.79x	03:11:37
	L1-Norm	57.88M	7.03G	92.52	90.58	-5.07	1.80x	03:11:53
	L2-Norm	58.43M	7.14G	92.70	89.97	-5.67	1.77x	03:12:38
BEIT-75%	Hessian	44.13M	4.32G	92.99	92.10	-3.55	2.93x	02:42:14
	Taylor	44.01M	4.3G	91.30	90.78	-4.86	2.95x	02:41:42
	Random	44.05M	4.3G	89.91	89.36	-6.28	2.95x	02:42:04
	L1-Norm	44.29M	4.35G	88.29	87.44	-8.21	2.91x	02:42:19
	L2-Norm	44.07M	4.31G	88.58	87.94	-7.7	2.94x	02:42:09
BEIT-95%	Hessian	32.71M	2.07G	83.19	82.78	-12.87	6.12x	02:16:57
	Taylor	32.74M	2.08G	86.90	85.11	-10.54	6.09x	02:17:00
	Random	32.68M	2.06G	79.25	78.32	-17.33	6.15x	02:17:01
	L1-Norm	32.75M	2.08G	83.54	83.08	-12.56	6.09x	02:16:47
	L2-Norm	32.61M	2.05G	83.19	80.14	-15.50	6.18x	02:16:34

while Hessian has an advantage in terms of accuracy. On the other hand, random pruning gives significantly lower pruned test accuracy (92.91%) and has accuracy loss of -2.74% which indicates that this strategy is less effective in selecting the most essential values for retention. L1-Norm and L2-Norm perform less effective than other three models in this circumstance. Both have test accuracies of 90.58% and 89.97%, respectively, with accuracy losses greater than -5%. Both L1-Norm and L2-Norm pruning algorithms are dependent on the magnitude of pruning weights, which may be insufficient for finding the most important weights in a transformer model such as BEIT.

As the pruning ratio rises to 75%, again the Hessian approach remains the most successful in maintaining accuracy, providing the highest pruned test accuracy of

92.10% and an acceptable accuracy reduction of -3.55%. The speedup is more evident at 2.93x and the number of trimmed parameters is substantially larger at 44.13 million. This mix of pruning efficacy and accuracy retention implies that Hessian pruning is still dependable, even at greater pruning ratios. All other models follow same pattern as for 50% and have overall accuacies between 87%-91% which are still very close to the baseline version of BEIT.

At the extreme pruning ratio of 95%, surprisingly, Taylor expresses better scores in terms of accuracy and has highest score of 85.11% and speed is also acceptable 6.09x faster than the original model. However, this performance suggests that Taylor becomes more competitive at extreme pruning rates which is possibly because of its gradient-based importance scores which better capture essential weights when a vast number of parameters are being removed.

In summary, for all pruning ratios, the BEIT model produces similar groups of reduced parameters and MACs as shown in Table7.3. If we summarise the analysis of the BEIT model and its effect of pruning on a domain generalisation benchmark like PACS, we can say that **Hessian-based** pruning consistently outperforms other approaches because it captures first and second-order information (curvature of the loss function), allowing for a better understanding of the network's sensitivity to parameter changes. This allows it to keep the most important weights even after rigorous pruning that explains its improved accuracy across varied pruning ratios. **Taylor-based** pruning is less successful than Hessian pruning at lower pruning levels, but it becomes competitive at greater pruning rates. This might be because Taylor-based approaches rely on gradient information which becomes increasingly useful as the network shrinks and each remaining parameter has a greater influence on performance. **Random pruning** is computationally efficient and simple to apply but it lacks any type of importance scoring and this is the reason for its elimination of critical parameters. As a result, its accuracy deteriorates faster, particularly at greater pruning levels. **L1-Norm** and **L2-Norm** pruning approaches suffer because they rely entirely on the magnitude of the weights which is not necessarily a suitable signal of importance, especially in models like BEIT with complicated parameter interactions. As pruning gets more harsh, these approaches fail to maintain key network connections, resulting in increased accuracy loss.

7.5.3 Group Structural Pruning Results for DeiT Transformer

Table 7.4 indicates that at 50% pruning, the Hessian approach surpasses the other strategies by keeping its test accuracy at 94.83% while incurring just -1.72% decrease in accuracy. With 43.02M trimmed parameters and 8.24G MACs, it achieves a 2.15x

Table 7.4: Pruning methods on the DeiT model at various pruning ratios.

Pruning Ratio	**Group Importance Score**	***Pruned Params***	***Pruned MACs***	***Pruned Valid Acc (%)***	***Pruned Test Acc (%)***	**Δ*Acc (%)***	***Speed Up***	***FT-Time***
DEIT-50%	Hessian	43.02M	8.24G	95.30	94.83	-1.72	2.15x	02:12:50
	Taylor	38.05M	7.36G	92.64	90.27	-6.28	2.40x	02:12:06
	Random	22.20M	4.62G	83.30	82.98	-13.58	3.83x	01:28:22
	L1-Norm	32.13M	6.34G	81.97	80.85	-15.70	2.79x	01:58:15
	L2-Norm	34.62M	6.77G	87.77	86.12	-10.44	2.61x	01:59:14
DEIT-75%	Hessian	8.01M	2.04G	77.86	75.89	-20.67	8.67x	00:57:21
	Taylor	4.94M	1.54G	72.81	72.14	-24.42	11.49x	00:50:31
	Random	8.47M	2.16G	71.88	70.21	-26.34	8.19x	01:00:57
	L1-Norm	0.35M	0.79G	40.52	37.29	-59.27	22.39x	00:39:05
	L2-Norm	0.67M	0.84G	51.19	47.92	-48.63	21.06x	00:39:10
DEIT-95%	Hessian	0.47M	0.81G	58.15	54.41	-42.15	21.84x	00:38:13
	Taylor	0.12M	0.75G	34.90	32.02	-64.54	23.59x	00:35:37
	Random	1.03M	0.88G	64.87	64.13	-32.42	20.10x	00:37:49
	L1-Norm	0.07M	0.74G	17.10	17.33	-79.23	23.91x	00:36:59
	L2-Norm	0.07M	0.74G	17.10	17.33	-79.23	23.91x	00:36:53

speedup for this accuracy. When we compare it with Hessian in BEIT at a similar pruning ratio it actually performs better in terms to speed and accuracy while the Taylor approach is similarly successful but suffers -6.28% accuracy loss which reduces test accuracy to 90.27%. It prunes 38.05M parameters and decreases MACs to 7.36G, modestly enhancing speedup by 2.40x. Random pruning shows a significant loss in performance and has a pruned test accuracy of 82.98% and an accuracy drop of -13.58%. L1-Norm and L2-Norm perform worse than the others when we consider accuracies of 80.85% and 86.12% and speed up metrics, respectively.

For 75% pruning the DeiT model displays a huge reduction in the parameters and MACs reductions compared to 50% on the original implementation. At 75% pruning, the Hessian technique retains its advantage by showing highest performance

of 75.89% test score and -20.67% accuracy loss. Again Taylor is the second best option followed by random pruning selection. The L1 and L2 Norms show great decreases in performances.

Similarly, the behaviour of DeiT model at an extreme pruning ratios changes rapidly. For example, at 95% pruning, performance degrades across all approaches. Hessian maintains its lead but only scores 54.41% test accuracy, a decline of -42.15%. Table 7.4 also illustrates that under extreme pruning rates, the DeiT model has removed almost all the parameters and MACs are less than 1G for all models which is why their speed becomes even higher by decreasing accuracy with larger margins. It is also clear that under extreme pruning conditions random selection performs even better than sophisticated methods.

In summary, the Hessian approach regularly outperforms other pruning strategies, especially at low and moderate pruning levels (50% and 75%). This is because Hessian-based pruning uses first/second-order derivatives, which provide more detailed information on how each parameter affects the model's performance. This new information allows it to more effectively target and prune less significant parameters while conserving those critical to the model's structure and function. On the other hand, methods such as Taylor and random pruning, produce less consistent outcomes. Taylor pruning is based on gradient information and it struggles at higher pruning levels (75% and 95%) because it fails to capture complicated relationships of the network. Random pruning is less successful at lower pruning levels however it has surprising resilience at high pruning (95%) since the randomisation helps avoid overfitting. Meanwhile, L1-Norm and L2-Norm pruning perform badly at all levels because they rely on magnitude which does not consider the complexities in the parameter of the DeiT model.

As another prospective of these results, we explore the relationship between the pruning ratios of the models and their resulting accuracy reduction. The graphs in Figure 7.6 attempt to provide an explanation for the behaviour of each model (ViT, BEIT, DeiT) with respective pruning criteria and ratios. The x-axes in the graphs represent the pruning ratio (50, 75 or 95)% and ΔAcc presents the accuracy drop after pruning the models compared to baseline accuracy of same model which was shown earlier in Table 7.1. All the graphs show that overall with the increase in pruning rates the ΔAcc also increases. However, as we already observed that when we drop 50% of any model using pruning methods, its parameters and MACs also drop accordingly which is shown in Tables 7.2, 7.3 and 7.4. Theoretically, by dropping 50% connections, filters, links, or layers accuracy should also be decreased by 50%. Similarly, by accelerating pruning to 75% and 95%, the expected accuracy drop should be close to those same ratios. In reality the ΔAcc is far less than the expected behaviour. For example, in Figure 7.6(a), for Hessian the curve the ΔAcc

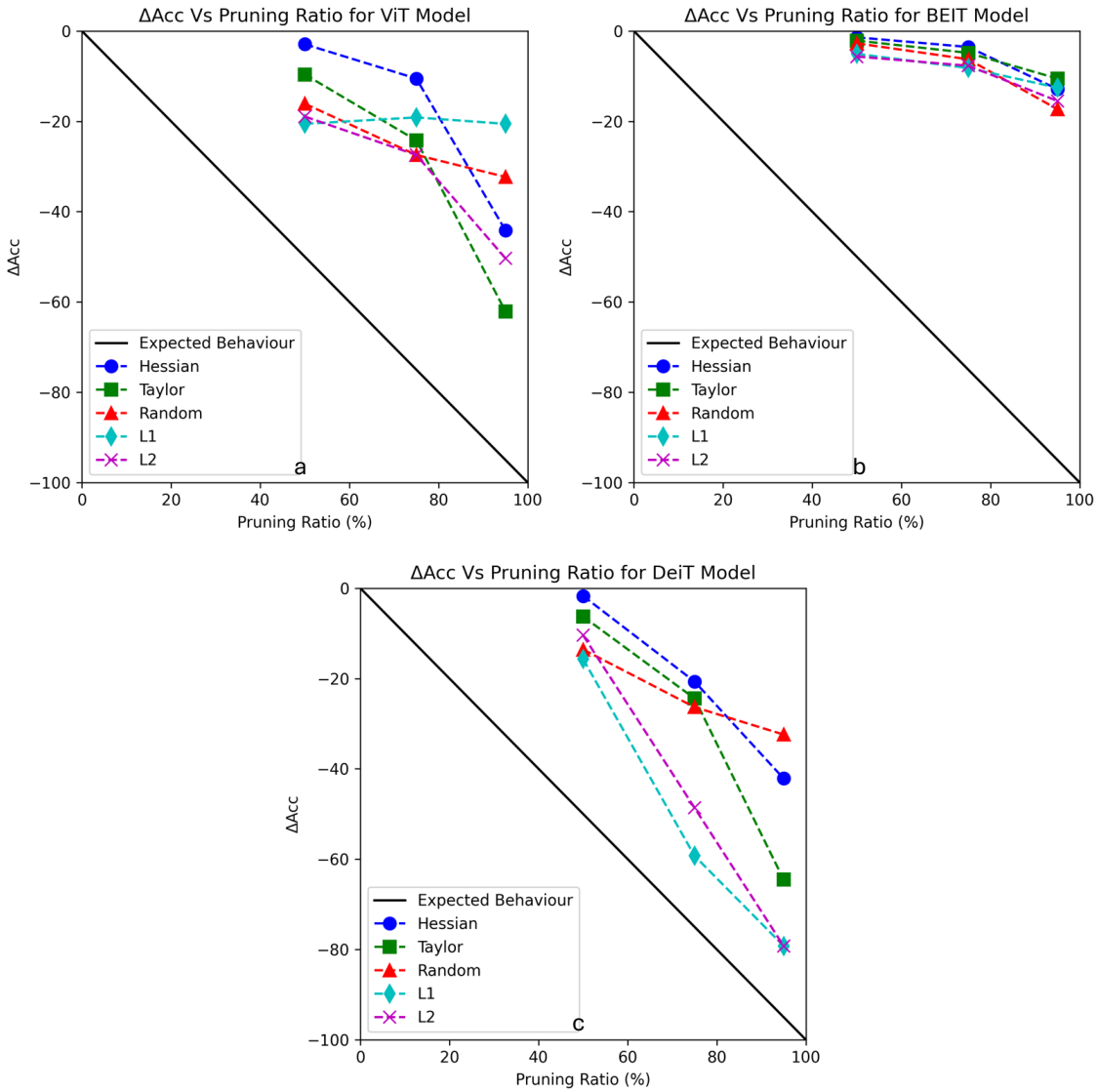


Figure 7.6: Relationship between pruning ratios and ΔAcc which represents the reduction rate in accuracy after pruning the models (a) ViT, (b) BEIT (c) DeiT.

is at 2.94% drop with 50% removal of network and 44.14% drop for the extreme case of a 95% pruning ratio. This behaviour can be observed for other curves in Figure 7.6(a). The graphs in Figures 7.6(b) and 7.6(c) also highlight the same trend of less accuracy dropping from the theoretical expectation which is a linear line in black colour in all the graphs.

Figure 7.6 gives us a general idea of performance of models under the influence of pruning. The second graph in Figure 7.6(b) has information about the BEIT model and illustrates that even in extreme conditions the ΔAcc is less than compared to others models for all the curves and this also means that BEIT has better DG. Moreover, the Figure 7.6(a) displays that L1 norm has stable performance for all pruning ratios.

In summary, all the graphs show that the performances of models for each prun-

ing criteria and following grouped structured pruning give better than expected behaviour and this behaviour may be caused by an interesting phenomena called "grokking". Grokking is the phenomenon in which LLM or any models are extensively trained, often to the point of overfitting but then their performance abruptly shifts into a stage of generalisation in which they begin to perform well on new and previously unexplored datasets. This transformation occurs unexpectedly and we still do not completely grasp the underlying mechanisms that drive it. It's analogous to a "lightbulb moment" in human learning — when after much effort where we are straining to understand something complex, such as quantum computing, and then it suddenly makes sense [359]. Grokking is the computational equivalent of that.

This phenomenon of grokking is related to the concept of "emergence" which occurs when a complex system (such as a neural network or a large population) shows behaviours or features that its individual elements do not exhibit on their own. In the field of AI, this has sparked discussions and enthusiasm about the possibility for Artificial General Intelligence (AGI) where a system could learn and understand any task that a human can, perhaps resulting from the iterative training and interaction of various models [359].

7.5.4 Pruning a Pre-trained larger Model vs Training a Smaller Model

We also noticed that it was important to discuss the significance of pruning a pre-trained tranied model compare to traning a smaller model.

Pruning a Pre-trained larger Model

By pruning pre-trained models (e.g., reducing 50%, 75% & 95% of the parameters), the pre-existing knowledge from large-scale pretraining is retained. This is especially useful for DG benchmarks, as pre-trained representations already capture robust and transferable features. For example, in the proposed grouped structural pruning, selection algorithms like Hessian, Taylor, and more, effectively eliminate unnecessary weights while keeping important pathways and consequently significantly improve accuracy even after extensive pruning. Furthermore, fine-tuning after pruning boosts the performance by adapting the remaining structure to the unique DG task.

Pruning and fine-tuning a pre-trained model are done in two steps: initial pre-training (once) which is already done by the ML community and subsequent fine-tuning after pruning which is done here. While this method may be computationally costly initially, during the fine-tuning the final model benefits from efficient inference due to the reduced number of parameters and computations (e.g., lower MACs).

The pruned version of a pre-trained model preserves the benefits of large-scale pretraining such as the capacity to learn domain-invariant characteristics. This is visible in vision transformers like BEIT, which use pretraining techniques like MIM to build robust representations. The preserved weights after pruning are more likely to capture the important properties required for generalisation across domains.

Training a Smaller Model

Training a smaller model from the start does not have the same advantages as large-scale pretraining. A smaller model begins with fewer parameters and less capacity, which may limit its ability to acquire rich and generalisable representations. While it may produce equivalent results for domain-specific tasks, its performance often decreases dramatically on OOD datasets or DG benchmarks, where large-scale pretraining offers a substantial solution.

Training a smaller model from scratch eliminates the requirement for pruning and fine-tuning, lowering the overall computational cost of training. However, to achieve comparable performance, smaller models might require longer training cyles or additional iterations, due to their limited capacity for complex tasks like DG.

Smaller models lack the pretraining advantage, which makes them more prone to overfitting on the training domain or failing to generalise to OOD data. This limitation is more obvious in DG benchmarks, where accumulating both local and global features is critical.

7.6 Conclusions and Lessons Learned

This chapter presented another investigation and exploration study into the analysis of DG using grouped structural pruning of neural networks with different vision transformers. The framework to perform the given experiments can support any data distributions and vision transformer but we only focussed on ViT, BEIT, and DeiT with the PACS benchmark even though in our previous research we have used other more complicated benchmarks namely Office-Home and DomainNet. However, in this chapter, as a proof of concept, we only used the PACS benchmark as it is widely acceptable in the research community. The more complex benchmarks like Office-Home and DomainNet can be tried as part of future work.

The chapter illustrated the procedure to remove structures (layers, filters, etc.) from pre-trained vision transformers based upon the pruning importance score methods. Our proposed method pruned the models in a grouped structural manner to make it a hardware-friendly method when it comes to deployment of the models. The results of this chapter highlight a crucial aspect of vision transformers under the

condition of pruning and then its relationships to domain generalisation. For example, in Table 7.1, we presented the performance of base models under same training conditions and hyperparameters we showed that the DeiT model performs better with slightly larger base trainable parameters and MACs and both performance levels are close to each other which is an indication of good domain generalisation. Tables 7.2, 7.3, and 7.4 showed that under the influence of different pruning ratios and methods the BEIT model delivers more stable and efficient performance compared to ViT or DeiT. This is another validation of our findings from the previous chapters 5 and 6 where we explained the uniqueness of BEIT over the other vision transformers when we have to handle domain generalisation.

This chapter also described another vital behaviour of vision transformers, for instance, under normal and medium level pruning (50%, 75%) all vision transformers behave normally and where Hessian perform better than all other pruning selection criteria because of its second order calculations. However, under high pruning rates 95% Talyor and random selection perform better meaning they have good generalisation capability. Based upon the results, we have future motivations to apply sparse training and pruning together on vision transformers. The results under hihg pruning rates also give us another intuition which is that it may be randomness that has potential to explore such sub networks which can have even better performance if we fine-tune the network in a special way. As part of future research directions we can expend this idea by applying it on Office-Home and DomainNet in addition to measuring the performance of OOD on other, newly-generated benchmarks.

In this chapter, we observed that in case of structural group pruning for vision transformers, it did not show a decrease in the performance in line with the the drop in pruning ratios. All the models are far away from their expected behaviour which could be attributed to the phenomenon of grokking as is shown in the graphs in Figure 7.6. Our results express that for smaller models with fewer parameters, obtaining this level of generalisation (or grokking) sometimes requires much more lengthy training meaning many more iterations than for bigger models. Over-training these smaller models can eventually result in grokking in which the model suddenly converges and begins to function better and displays emergent behaviours. Even though the grokking phenomena needs much more training time, if we can reflect back to the validation graphs for different pruning ratios for ViT, BEIT, DeiT in Figures 7.3, 7.4 and 7.5, we observe that most of the models got adequately better in their first few epochs. However, the process of over-training even at this level can induce some form of grokking in the models.

In summary, the chapter focused on research sub-hypothesis H2 which explains that dynamic learning methods can have better and faster domain adaptation and generalisation which can be seen from the results of base Table 7.1 and pruning

Tables 7.2, 7.3 and 7.4. . One of the best of these models is BEIT Hessian with 50% ratio which only drops by -1.42% but increase sthe speed by 1.81x compared to its base model. A similar pattern can be found in the other models which is a clear indication of support for our initial research sub hypothesis H2.

Chapter 8

Conclusions

Regardless of recent developments, popularity and public visibility in the field of AI, the research in this thesis has been motivated by the fundamental need to improve the resilience and generalisation capabilities of machine learning models notably in the field of computer vision. The value of this study stems from its thorough investigation of domain generalisation using novel strategies such as benchmarking, masking, and pruning. By tackling the crucial issues of OOD generalisation, this thesis adds to the boarder objective of creating more resilient and adaptable AI systems. The findings not only improve our theoretical knowledge of domain generalisation, but they also provide practical insights that may be useful to real-world applications ranging from autonomous systems to medical imaging. This research underscores the importance of creating AI models that can perform reliably across diverse and unseen environments, ultimately pushing the boundaries of what is possible in the field of artificial intelligence.

The study tries to solve the difficulty of when models perform badly when evaluated on OOD data which is data from domains other than the one the model was trained on. For this purpose, we mainly focus on DG benchmarks namely PACS, Office-Home, and DomainNet and we proposed innovative and unique methods to tackle DG based on the factors which we established after the initial investigation into the topic. In summary, the key focus of the thesis was around understanding domain generalisation in computer vision applications. To drive the idea, first of all, we examined conventional and relatively modern ML models to check their behaviour for DG. After that we conducted research on vision transformers namely ViT, BEIT, DeiT, self-Supervised learning, attention mechanisms, benchmarking, masking techniques, and pruning.

This chapter re-caps on the proposed methods and their contributions according to a set of research questions and hypotheses which we previously explained in Chapter 1. In this concluding chapter we will revisit the RQs and the inspiration behind each proposed method with individual contributions and outcomes from

each method. We will then explain the limitations of the work and future research directions.

8.1 Dissertation Overview: Research Questions and Hypotheses

In this thesis, especially in Chapter 1, we have stated three research question and three research hypotheses and we have stated them in Chapters 3, 5, 6 and 7. All the RQs (RQ1, RQ2, RQ3) and $H\phi$ (H_1, H_2 *and* H_3) tried to point towards the principle motivation which is to explore domain generalisation for vision based machine learning) and to discuss the main hypotheses which says that domain generalised (dynamic learning) methods can handle domain generalisation better than domain specific (static learning) methods. According to that, we constructed different scenarios where we discuss the conditions to perform studies with domain generalisation benchmarks so that we can work on the RQs.

8.1.1 Revisiting RQ1

RQ1 has two parts, which say that the **exploration of the literature to investigate how can we measure domain generalisation, and second what type of experiments we can perform to highlight that dynamic learning can handle generalisation better than static learning.** Chapter 2 explained the overall background of the field and literature related to the problem statement topics which led us to start the experiments stated in Chapter 3. Chapter 3 provided a comprehensive review of how common domain generalisation (DG) frameworks performed against domain-specific models on benchmarks such as PACS and Office-Home. The experiments demonstrated that domain-specific models perform poorly when generalisation is tested using accuracy and loss criteria. Furthermore, the results show that models with skip connections (e.g., ResNet and DenseNet) outperform those without (e.g., AlexNet, VGG), and that bigger models generalise better.

Chapter 3 also discussed the disparity between validation and test accuracy as indicators of DG. Models with narrower gaps perform better in DG. While the chapter concentrated on classic deep learning approaches, it also highlighted the importance of investigating attention-based vision transformers for domain generalisation in subsequent chapters. Finally, the study showed that domain-generalised approaches (dynamic learning) outperform domain-specific methods (static learning), especially on complicated benchmarks.

In Chapter 3, we stated RQ1 as exploring ways to measure domain generalisation and evaluating the generalisation capability of different models using accuracy metrics. The chapter compared domain-specific and domain-generalised models, explaining why the latter outperforms the former, particularly on benchmarks such as PACS and Office-Home. It emphasised the significance of skip links and model size in enhancing domain generalisation and described validation-test accuracy gaps as a measure for determining generalisation. In addition, the conclusion emphasised the need for experimenting with vision transformers and suggested that future research should expand the study to larger benchmarks like DomainNet.

8.1.2 Revisiting RQ3

In order to address the proposed RQs in a logical order, we worked on RQ3 first and then we will explain the work done on RQ2. RQ3 is related to the accuracy of models and states that if we are able to work on H1 and H2 and successfully find/develop more effective and efficient domain generalised methods then can such models also have better accuracy? Inspired by the results from Chapter 3 which addressed RQ1, we performedfurther investigations to highlight the factors which could effect a model's capability for DG. These factors were explained with details in Chapter 4 which presented the ourcome that factors like self-supervised learning [213, 214], removing the background or textural information from images (denoising) [215], adversarial data augmentation [216], ignoring texture and focusing on shape [217, 218] and using larger models 3, and multi-head self-attention [41] could each effect DG. If any model has these factors or even a subset of these factors there is a good chance to get better DG. We know that vision transformers are the models with most of these factors and if we consider vision transformers as relatively modern and very recent compared to models based on CNNs then it follows that such models should have good potential for DG.

To address RQ3, in Chapter 5 we conducted further experiments with vision transformers for DG with three different benchmarks of different complexity. Based on the results of the feasibility study in Table 5.1, we found BEIT outperformed other transformers on unseen OOD benchmarks.

Further study then showed that the BEIT models have most of the features which we stated as desirable in Chapter 4. The reason why the BEIT model has better results than others was because of the model's length, self-attention mechanism, self-supervised learning, the demonising ability of the model, removal of texture information, and the focus on the shapes of objects. Chapter 5 included fine-tuning details of the base BEIT model for PACS, Office-Home and DomainNet. The underlying BEIT model was fine-tuned for DG benchmarks resulting in cutting-edge

performance across all metrics and decreased the gap between IID and OOD accuracy, specifically from 21.1% to 2% on PACS and showed similar trends in the other two benchmarks. The results can be seen in Tables 5.2, 5.3 and 5.4.

Chapter 5 emphasised the importance of understanding the latent space in a model by using attention mechanisms like mean self-attention distance to gauge the model's learning of domain knowledge. Attention maps also helped to visualise whether the model truly focuses on objects rather than backgrounds or not. Chapter 5 also built the relationship between DG and global distances with the conclusion that models with high global distances have better DG. This also provides us with the insights to further test the robustness of the trained models in the presence of external noise, particularly focusing on masked image modelling and its denoising effects.

8.1.3 Revisiting RQ2

RQ2 asks "for real world data, how much elasticity or variation in the datasets is needed in order for domain generalised methods to break." To address RQ2, Chapter 6 gave a thorough examination focused on two essential aspects: the generation of new OOD benchmarks utilising PACS, Office-Home, and DomainNet, and the establishment of a cascade network integrating Grounding DINO and SAM, the Segment Anything Model. We specifically updated the SAM model to operate with text prompts and improved its capabilities by including bounding box implementation using Grounding DINO, as shown in Figure 6.2. The second section of this chapter focused on assessing DG by computing accuracy and gap metrics to determine the durability of our fine-tuned models. The major purpose was to investigate how noise and distribution shifts affect the performance of these models.

The findings reported in this chapter provide crucial insights. Tables 6.1, 6.2 and 6.3 showed that larger grids or masks obscuring items give less attention to the model which implies that the models are more resilient to larger grid masks. However, as the grid size decreases and the number of grids grows, the performance of the model suffers indicating that smaller grid masks are more disruptive. This behaviour can be explained by interference at higher feature levels where fewer connections between neurons impair the network's capacity to generalise.

Another significant discovery was observed in the outer segment edge benchmarks, where all fine-tuned BEIT models fared badly. This result supports previous claims made in Chapter 5 about the role of object shapes in DG. Outside segmented edge masking distorts the shape of objects and introduces new kinds of noise into features severely reducing model performance. This shows that our models have learned to focus largely on object shapes as seen by better performance when mask-

ing generation occurs within the object bounds.

Chapter 6 also tested the denoising capacity of the BEIT model. Even with 25% noise or occlusion ratio, the model continues to produce cutting-edge performance in the newly developed benchmarks. However, performance suffers significantly when occlusion ratios exceed 50% or 75%, especially when grid size becomes smaller. These results support the resilience of the BEIT model under moderate noise settings.

In answering RQ2, Chapter 3.4 investigated how different levels of noise and distortion impact a model's accuracy and DG. Furthermore, the chapter supported sub-hypothesis H3, which states that a well-trained or generalised model can reduce or remove the requirement for transfer learning. Our study did not include any fresh fine-tuning or transfer learning but instead we relied on already trained models which demonstrates the ability to bypass transfer learning for unknown data distributions.

In summary, RQ2 contributed the following:

- Developed an infrastructure to generate new OOD benchmarks using already available PACS, Office-Home, and DomainNet.
- Enhanced SAM by integrating text prompts and bounding box calculations with Grounding DINO.
- Conducted a comprehensive analysis of the resilience of fine-tuned BEIT models under different noise and occlusion conditions.
- Confirmed the denoising capability of BEIT and its superior performance under moderate noise.
- Demonstrated the possibility of eliminating transfer learning for unseen data distributions by showing how well fine-tuned models performed without additional fine-tuning.

8.1.4 Brief Summary of Chapters with Respect to H1, H2 and H3

This sub-section provides a brief summary of all the chapters of the thesis and highlights corresponding sub hypothesis addressed by respective chapters.

1. **Chapter 1:** described the general introduction to field of ML and defined the problem statement with three exciting research questions and three sub-hypothesis.
2. **Chapter 2:** gave the background knowledge related to ML and tried to explain DG in different fields of ML like SL, USL, and RL. The chapter also

summarised the literature of the field according to applications in a form of a timeline graph as Figure 2.1.

3. **Chapter 3:** addressed research sub-hypothesis $H1$ which is related to hypothesis $H\phi$ that domain-generalised methods outperform domain-specific methods.

4. **Chapter 4:** focused on the theoretical background related to vision transformers of different types and vision language models. This chapter conducted a detailed investigation on finding the common factors which have greatest impact on enhancing or effecting the DG of a model.

5. **Chapter 5:** Based on the fundamental findings from 4, which says BEIT model could be a better choice and then after the feasibility study in Chapter 5, this chapter conducted experiments with BEIT for DG benchmarks. This chapter also tried to find the reasons behind the good performance of BEIT by building the relationship between global attention distances and the DG potential of model. This chapter also explored sub-hypotheses H1 and H2 by delivering a more robust model with better domain generalisation ability.

6. **Chapter 6:** illustrated how we can measure the variations in data distributions in a deterministic way and proposed a unique cascade network which is a combinations of enhanced SAM and Grounding DINO to determine the masks of objects in an image and applied grid masking on test images to find out how much such manipulations reduced or effected the DG of already trained models. This chapter also targeted sub-hypothesis H3 according to which if a model is trained enough or generalised enough then we can possibly eliminate transfer learning.

7. **Chapter 7:** defined a pruning of neural networks and implemented grouped structural pruning for vision transformer and domain generalisation benchmarks, specifically for PACS. This chapter focused on sub-hypothesis H2, which proposes that dynamic learning approaches can provide faster and better domain adaptation and generalisation. This notion is supported by the findings of Table 7.1, as well as the pruning trials in Tables 7.2, 7.3, and 7.4. The chapter also looked at significant behavioural features in vision transformers. Under modest pruning rates (50-75%), all transformers worked reasonably well with Hessian-based pruning outperforming other approaches due to its second-order calculations. However, at high pruning rates (95%), Taylor and random selection approaches perform better in terms of generalisation. The main contributions were the following:

(a) The creation of a grouped structural pruning framework for vision transformers, optimised for hardware deployment.

(b) Empirical proof that BEIT beats the ViT and DeiT models in domain generalisation under different pruning settings.

(c) Hessian-based pruning was shown to be the most successful strategy at intermediate pruning levels, whereas Taylor and random selection methods outperformed excessive pruning.

(d) Future directions were proposed, such as sparse training and examining randomness in pruning for sub-network identification and optimisation.

(e) The investigation of how pruning approaches effect domain generalisation and the validation of the H2 hypothesis that dynamic learning methods improved DG performance.

8. **Chapter8:** This concluding chapter summarised the overall dissertation by revisiting the research questions and sub-hypotheses and how each chapter contributes to them.

8.2 Limitations

In this thesis, the main focus was to explore domain generalisation in machine learning models for DG benchmarks like PACS, Office-Home, and DomainNet, especially, conducting research to find out the different factors which can effect or improve the DG ability of ML models. Initially, this task ws challenging because most of the ML and DL models lack generalisation on OOD data which can been seen in Tables 3.3 and 5.1.

Similarly, when simulating the results for DG, we realised that relying on only accuracy will not be enough, regardless of other metrics which we have described in this thesis. Therefore, may be we need to use measurement metrics which measure feature level contributions to redefine the DG capability of models.

Another drawback of our research is related to versatility. For example, in this thesis, we only considered vision based applications for DG benchmarks. However, we do not have observations and experimental results for other modalities like textual or audio data. Hence, we cannot be sure if this thesis will stand in such modalities or not.

Chapter 6 proposed a method to generate new and rare OOD grid mask based benchmarks which are created on already available data distributions like PACS, Office-Home, and DomainNet. Nevertheless, the development of more complex benchmarks is missing from the research, such as using generative AI models like

diffusion [360] or Dall-E [361] to create prompt-based complex benchmarks to measure DG for OOD. Moreover, in Chapter 6, we utilised SAM and Grounding DINO for creation of new OOD benchmarks, hence any limitations of SAM or Grounding DINO will have impact on the quality of our results.

The pruning work in Chapter 7 only considered the implementation of methods after training on the models which we have explained in Figure 7.1. Hence, we are not sure will these findings hold in other pruning types like before training or sparse training methods as it was out of scope for our research.

8.3 Future Research Directions

In this research we have explored domain generalisation for vision based benchmarks like PACS, Office-Home, and DomainNet with different range of ML models and after foundation of vision transformers we investigates on benchmarking, masking, and pruning for DG of models.

One of the future directions related to our work could be redefining an absolute measurement metrics other than accuracy to represent domain generalisation of AI. This method could be based on features of objects rather than the final results of probabilities of neural networks. For example, to be specific, a pre-defined dictionary of features of each category will be needed which could be matched or aligned to any unseen distribution and will show new insights about the potential of models.

Another challenge we faced during our research was related to the generation of new OOD benchmarks which we did in Chapter 3.4. We did not implement generative AI, hence in future it would be interesting to develop such a framework which can generate more detailed and complex OOD benchmarks to measure or test the OOD ability of any model.

The findings in Chapter 7 suggest that future study should combine sparse training and pruning as well as investigate randomness as a technique to locate subnetworks that, when properly fine-tuned, might lead to better performance. Future study might apply these concepts to more complicated benchmarks such as Office-Home and DomainNet, as well as to evaluate performance on newly generated OOD benchmarks in Chapter 3.4.

Similarly, the results in Chapter 7 also highlight another important insight namely that the relationship between pruning ratio and reduction is not linear which points towards the grokking phenomenon in the model. This also known as emergence in small AI model and further investigation methods could be needed.

To enhance the scope of our thesis, the pruning work in Chapter 7 also highlights another interesting future research direction, to make pruning more efficient and also preserve the information flow in the neural networks which can also represent global

links or common links for any category of object in different domains that can help us to obtain smaller and generalised models which could achieve better performance in future. This idea can be related to a term called "mechanistic interpretability" [362] which could be explored in future for vision based applications.

Bibliography

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[2] Hao-nan Wang et al. "Deep reinforcement learning: a survey". In: *Frontiers of Information Technology & Electronic Engineering* 21.12 (Dec. 2020), pp. 1726–1744. ISSN: 2095-9230. DOI: 10.1631/FITEE.1900533. URL: https://doi.org/10.1631/FITEE.1900533.

[3] Pedro A Tsividis et al. "Human learning in Atari". In: *AAAI Spring Symposia*. 2017.

[4] Sandra Vieira, Walter Hugo Lopez Pinaya, and Andrea Mechelli. "Chapter 1 - Introduction to machine learning". In: *Machine Learning*. Ed. by Andrea Mechelli and Sandra Vieira. Academic Press, Jan. 2020, pp. 1–20. ISBN: 978-0-12-815739-8. DOI: 10.1016/B978-0-12-815739-8.00001-8. URL: https://www.sciencedirect.com/science/article/pii/B9780128157398000018.

[5] "Skinner, B. F. Science and human behavior. New York: The Macmillan Company, 1953. 461 P. $4.00". In: *Science Education* 38.5 (Dec. 1954). Publisher: John Wiley & Sons, Ltd, pp. 436–436. ISSN: 0036-8326. DOI: 10.1002/sce.37303805120. URL: https://doi.org/10.1002/sce.37303805120 (visited on 02/11/2022).

[6] Josh Achiam et al. "GPT-4 technical report". In: *arXiv:2303.08774* (2023).

[7] Hugo Touvron et al. "Llama: Open and efficient foundation language models". In: *arXiv:2302.13971* (2023).

[8] Ebtesam Almazrouei et al. "The Falcon series of open language models". In: *arXiv:2311.16867* (2023).

[9] Karan Singhal et al. "Towards expert-level medical question answering with large language models". In: *arXiv:2305.09617* (2023).

[10] Karan Singhal et al. "Publisher Correction: Large language models encode clinical knowledge". In: *Nature* 620.7973 (2023), E19.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[12] Jindong Wang et al. "Generalizing to unseen domains: A survey on domain generalization". In: *IEEE transactions on knowledge and data engineering* 35.8 (2022), pp. 8052–8072.

[13] Jonathan Frankle and Michael Carbin. "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=rJl-b3RcF7.

[14] Warren S. McCulloch and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259. URL: https://doi.org/10.1007/BF02478259.

[15] N. Kanopoulos, N. Vasanthavada, and R.L. Baker. "Design of an image edge detection filter using the Sobel operator". In: *IEEE Journal of Solid-State Circuits* 23.2 (1988), pp. 358–367. DOI: 10.1109/4.996.

[16] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[17] I. Daubechies. "The wavelet transform, time-frequency localization and signal analysis". In: *IEEE Transactions on Information Theory* 36.5 (1990), pp. 961–1005. DOI: 10.1109/18.57199.

[18] Andrzej Maćkiewicz and Waldemar Ratajczak. "Principal components analysis (PCA)". In: *Computers & Geosciences* 19.3 (1993), pp. 303–342. ISSN: 0098-3004. DOI: https://doi.org/10.1016/0098-3004(93)90090-R. URL: https://www.sciencedirect.com/science/article/pii/009830049390090R.

[19] Niall O'Mahony et al. "Deep learning vs. traditional computer vision". In: *Advances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1 1*. Springer. 2020, pp. 128–144.

[20] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[21] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[22] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[23] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[24] John D Kelleher, Brian Mac Namee, and Aoife D'arcy. *Fundamentals of machine learning for predictive data analytics: algorithms, worked examples, and case studies*. 2020. ISBN: 0-262-36110-8.

[25] Yann LeCun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998). ISBN: 0018-9219 Publisher: Ieee, pp. 2278–2324.

[26] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems* 25 (2012), pp. 1097–1105.

[27] Francois Chollet. *Deep Learning with Python*. 1st. USA: Manning Publications Co., 2017. ISBN: 1617294438.

[28] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, pp. 807–814. ISBN: 9781605589077.

[29] Jun Han and Claudio Moraga. “The influence of the sigmoid function parameters on the speed of backpropagation learning”. In: *From Natural to Artificial Neural Computation*. Ed. by José Mira and Francisco Sandoval. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 195–201. ISBN: 978-3-540-49288-7.

[30] Herbert Robbins and Sutton Monro. “A stochastic approximation method”. In: *The Annals of Mathematical Statistics* (1951), pp. 400–407.

[31] Vittorio Mazzia, Francesco Salvetti, and Marcello Chiaberge. “Efficient CapsNet: Capsule Network with Self-Attention Routing”. In: *arXiv:2101.12491* (2021).

[32] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv:1409.1556* (2014).

[33] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[34] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[35] Gao Huang et al. “Densely connected convolutional networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4700–4708.

[36] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: *arXiv:1704.04861* (2017).

[37] Mingxing Tan and Quoc Le. “Efficientnet: Rethinking model scaling for convolutional neural networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[38] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv:2010.11929* (2020).

[39] Hangbo Bao et al. “BEiT: BERT Pre-Training of Image Transformers”. In: *International Conference on Learning Representations*. 2021.

[40] Alexander Kolesnikov et al. “Big transfer (bit): General visual representation learning”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*. Springer. 2020, pp. 491–507.

[41] Hugo Touvron et al. “Training data-efficient image transformers & distillation through attention”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10347–10357.

[42] Ze Liu et al. “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 10012–10022.

[43] Sanghyun Woo et al. “Convnext v2: Co-designing and scaling convnets with masked autoencoders”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16133–16142.

[44] Maxime Oquab et al. “Dinov2: Learning robust visual features without supervision”. In: *arXiv:2304.07193* (2023).

[45] Kaiming He et al. “Masked autoencoders are scalable vision learners”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 16000–16009.

[46] Xiaohua Zhai et al. “Sigmoid loss for language image pre-training”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 11975–11986.

[47] Jianfeng Wang et al. “Git: A generative image-to-text transformer for vision and language”. In: *arXiv:2205.14100* (2022).

[48] Alec Radford et al. “Learning transferable visual models from natural language supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 8748–8763.

[49] Jingfeng Yao et al. “ViTMatte: Boosting image matting with pre-trained plain vision transformers”. In: *Information Fusion* 103 (2024), p. 102091.

[50] Wenhai Wang et al. “Pvt v2: Improved baselines with pyramid vision transformer”. In: *Computational Visual Media* 8.3 (2022), pp. 415–424.

[51] Ross Girshick et al. “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation”. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[52] Ross Girshick. “Fast R-CNN”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[53] Shaoqing Ren et al. “Faster R-CNN: Towards real-time object detection with region proposal networks”. In: *Advances in Neural Information Processing Systems* 28 (2015), pp. 91–99.

[54] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, Proceedings, Part I 14*. Springer, 2016, pp. 21–37.

[55] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788.

[56] Tsung-Yi Lin et al. “Feature Pyramid Networks for Object Detection”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[57] Joseph Redmon and Ali Farhadi. “YOLO9000: better, faster, stronger”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7263–7271.

[58] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. “YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information”. In: *arXiv:2402.13616* (2024).

[59] Juan Terven and Diana Cordova-Esparza. “A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond”. In: *arXiv:2304.00501* (2023).

[60] Yuxin Fang et al. “You only look at one sequence: Rethinking transformer in vision through object detection”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 26183–26197.

[61] Yanghao Li et al. “Exploring plain vision transformer backbones for object detection”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 280–296.

[62] Nicolas Carion et al. “End-to-end object detection with transformers”. In: *European Conference on Computer Vision*. Springer. 2020, pp. 213–229.

[63] Xizhou Zhu et al. “Deformable detr: Deformable transformers for end-to-end object detection”. In: *arXiv:2010.04159* (2020).

[64] Shilong Liu et al. “Grounding dino: Marrying dino with grounded pre-training for open-set object detection”. In: *arXiv:2303.05499* (2023).

[65] Zhiliang Peng et al. “Kosmos-2: Grounding multimodal large language models to the world”. In: *arXiv:2306.14824* (2023).

[66] Matthias Minderer et al. “Simple open-vocabulary object detection”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 728–755.

[67] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. “Scaling open vocabulary object detection”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[68] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017). Publisher: IEEE, pp. 2481–2495. ISSN: 0162-8828.

[69] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer. 2015, pp. 234–241.

[70] Abhishek Chaurasia and Eugenio Culurciello. “Linknet: Exploiting encoder representations for efficient semantic segmentation”. In: *2017 IEEE Visual Communications and Image Processing (VCIP)*. IEEE. 2017, pp. 1–4.

[71] Hengshuang Zhao et al. “Pyramid scene parsing network”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2881–2890.

[72] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. "V-net: Fully convolutional neural networks for volumetric medical image segmentation". In: *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE. 2016, pp. 565–571.

[73] Liang-Chieh Chen et al. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.4 (2017), pp. 834–848.

[74] Kaiming He et al. "Mask R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2961–2969.

[75] Shervin Minaee et al. "Image segmentation using deep learning: A survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[76] Alexander Kirillov et al. "Segment anything". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 4015–4026.

[77] Xinlong Wang et al. "Seggpt: Segmenting everything in context". In: *arXiv: 2304.03284* (2023).

[78] Enze Xie et al. "SegFormer: Simple and efficient design for semantic segmentation with transformers". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12077–12090.

[79] Bowen Cheng, Alex Schwing, and Alexander Kirillov. "Per-pixel classification is not all you need for semantic segmentation". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 17864–17875.

[80] Timo Lüddecke and Alexander Ecker. "Image segmentation using text and image prompts". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 7086–7096.

[81] Jiarui Xu et al. "Groupvit: Semantic segmentation emerges from text supervision". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18134–18144.

[82] Jitesh Jain et al. "Oneformer: One transformer to rule universal image segmentation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 2989–2998.

[83] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv:1312.5602* (2013).

[84] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016). Publisher: Nature Publishing Group, pp. 484–489. ISSN: 1476-4687.

[85] David Silver et al. “Mastering the game of go without human knowledge”. In: *nature* 550.7676 (2017). Publisher: Nature Publishing Group, pp. 354–359. ISSN: 1476-4687.

[86] Timothy P Lillicrap et al. “Continuous control with deep reinforcement learning”. In: *arXiv:1509.02971* (2015).

[87] Tuomas Haarnoja et al. “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor”. In: PMLR, 2018, pp. 1861–1870.

[88] Shixiang Gu et al. “Continuous deep q-learning with model-based acceleration”. In: *International conference on machine learning*. PMLR. 2016, pp. 2829–2838.

[89] Tuomas Haarnoja et al. “Reinforcement learning with deep energy-based policies”. In: *International conference on machine learning*. PMLR. 2017, pp. 1352–1361.

[90] Matteo Hessel et al. “Rainbow: Combining improvements in deep reinforcement learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[91] Dan Horgan et al. “Distributed prioritized experience replay”. In: *arXiv: 1803.00933* (2018).

[92] Scott Fujimoto, Herke Hoof, and David Meger. “Addressing function approximation error in actor-critic methods”. In: PMLR, 2018, pp. 1587–1596.

[93] John Schulman et al. “Trust region policy optimization”. In: PMLR, 2015, pp. 1889–1897.

[94] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv: 1707.06347* (2017).

[95] Ziyu Wang et al. “Sample efficient actor-critic with experience replay”. In: *arXiv:1611.01224* (2016).

[96] Volodymyr Mnih et al. “Asynchronous methods for deep reinforcement learning”. In: PMLR, 2016, pp. 1928–1937.

[97] Sergey Levine and Vladlen Koltun. “Guided Policy Search”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Issue: 3. PMLR, May 2013, pp. 1–9. URL: `https://proceedings.mlr.press/v28/levine13.html`.

[98] Kurtland Chua et al. “Deep reinforcement learning in a handful of trials using probabilistic dynamics models”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[99] Théophane Weber et al. “Imagination-augmented agents for deep reinforcement learning”. In: *arXiv:1707.06203* (2017).

[100] Marc Deisenroth and Carl E Rasmussen. “PILCO: A model-based and data-efficient approach to policy search”. In: Citeseer, 2011, pp. 465–472.

[101] Manuel Watter et al. “Embed to control: A locally linear latent dynamics model for control from raw images”. In: *Advances in Neural Information Processing Systems* 28 (2015).

[102] Vint Lee, Pieter Abbeel, and Youngwoon Lee. “DreamSmooth: Improving Model-based Reinforcement Learning via Reward Smoothing”. In: (2023). arXiv: 2311.01450.

[103] Philipp Wu et al. “DayDreamer: World Models for Physical Robot Learning”. In: *Proceedings of The 6th Conference on Robot Learning*. Ed. by Karen Liu, Dana Kulic, and Jeff Ichnowski. Vol. 205. Proceedings of Machine Learning Research. PMLR, Dec. 2023, pp. 2226–2240. URL: https://proceedings.mlr.press/v205/wu23c.html.

[104] Fangchen Liu et al. “MOKA: Open-Vocabulary Robotic Manipulation through Mark-Based Visual Prompting”. In: *arXiv:2403.03174* (2024).

[105] Han Altae-Tran et al. “Low data drug discovery with one-shot learning”. In: *ACS Central Science* 3.4 (2017). Publisher: ACS Publications, pp. 283–293. ISSN: 2374-7943.

[106] Gary Marcus. “Deep learning: A critical appraisal”. In: *arXiv:1801.00631* (2018).

[107] Andrey Ignatov et al. “Ai benchmark: All about deep learning on smartphones in 2019”. In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. IEEE. 2019, pp. 3617–3635.

[108] Allan M. Schrier. “Learning how to learn: The significance and current status of learning set formation”. In: *Primates* 25.1 (Jan. 1984), pp. 95–102. ISSN: 1610-7365. URL: https://doi.org/10.1007/BF02382299.

[109] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta learning for fast adaptation of deep networks”. In: PMLR, 2017, pp. 1126–1135.

[110] Abhishek Gupta et al. “Meta-reinforcement learning of structured exploration strategies”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[111] Keyu Chen, Di Zhuang, and J Morris Chang. “Discriminative adversarial domain generalization with meta-learning based cross-domain validation”. In: *Neurocomputing* 467 (2022), pp. 418–426.

[112] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[113] Forrest N Iandola et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and¡ 0.5 MB model size”. In: *arXiv:1602.07360* (2016).

[114] Ilya O Tolstikhin et al. “Mlp-mixer: An all-mlp architecture for vision”. In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272.

[115] Mingxing Tan and Quoc Le. “Efficientnetv2: Smaller models and faster training”. In: *International conference on machine learning*. PMLR. 2021, pp. 10096–10106.

[116] Joseph Redmon. “Yolov3: An incremental improvement”. In: *arXiv:1804.02767* (2018).

[117] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. “Yolov4: Optimal speed and accuracy of object detection”. In: *arXiv:2004.10934* (2020).

[118] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. “You only learn one representation: Unified network for multiple tasks”. In: *arXiv:2105.04206* (2021).

[119] Anima Pramanik et al. “Granulated RCNN and Multi-Class Deep SORT for Multi-Object Detection and Tracking”. In: *IEEE Transactions on Emerging Topics in Computational Intelligence* 6.1 (2022), pp. 171–181. DOI: 10.1109/TETCI.2020.3041019.

[120] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[121] Jingdong Wang et al. “Deep high-resolution representation learning for visual recognition”. In: *IEEE transactions on pattern analysis and machine intelligence* 43.10 (2020), pp. 3349–3364.

[122] Yuwen Xiong et al. “Upsnet: A unified panoptic segmentation network”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8818–8826.

[123] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. “Nas-fpn: Learning scalable feature pyramid architecture for object detection”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 7036–7045.

[124] Nikhil Mishra et al. “A simple neural attentive meta-learner”. In: *arXiv:1707.03141* (2017).

[125] Bradly C Stadie et al. “Some considerations on learning to explore via meta-reinforcement learning”. In: *arXiv:1803.01118* (2018).

[126] Anusha Nagabandi et al. “Learning to adapt in dynamic, real-world environments through meta-reinforcement learning”. In: *arXiv:1803.11347* (2018).

[127] Rasool Fakoor et al. “Meta-q-learning”. In: *arXiv:1910.00125* (2019).

[128] Chelsea Finn, Pieter Abbeel, and Sergey Levine. “Model-agnostic meta-learning for fast adaptation of deep networks”. In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[129] Seyed Kamyar Seyed Ghasemipour, Shixiang (Shane) Gu, and Richard Zemel. “SMILe: Scalable Meta Inverse Reinforcement Learning through Context-Conditional Policies”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/2b8f621e9244cea5007bac8f5d50e476-Paper.pdf.

[130] Kate Rakelly et al. “Efficient off-policy meta-reinforcement learning via probabilistic context variables”. In: *International conference on machine learning*. PMLR. 2019, pp. 5331–5340.

[131] Luisa Zintgraf et al. “Fast context adaptation via meta-learning”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7693–7702.

[132] Louis Kirsch, Sjoerd van Steenkiste, and Jürgen Schmidhuber. “Improving generalization in meta reinforcement learning using learned objectives”. In: *arXiv:1910.04098* (2019).

[133] Andres Campero et al. “Learning with amigo: Adversarially motivated intrinsic goals”. In: *arXiv:2006.12122* (2020).

[134] Amina Adadi. “A survey on data-efficient algorithms in big data era”. In: *Journal of Big Data* 8.1 (Jan. 2021), p. 24. ISSN: 2196-1115. DOI: 10.1186/s40537-021-00419-9. URL: https://doi.org/10.1186/s40537-021-00419-9.

[135] Vanya Van Belle et al. “Explaining support vector machines: a color based nomogram”. In: *PloS one* 11.10 (2016), e0164568.

[136] R. Roscher et al. “Explainable Machine Learning for Scientific Insights and Discoveries”. In: *IEEE Access* 8 (2020), pp. 42200–42216. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.2976199.

[137] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." In: *Psychological review* 65 6 (1958), pp. 386–408. URL: https://api.semanticscholar.org/CorpusID:12781225.

[138] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv:1511.08458* (2015).

[139] David E. Rumelhart and James L. McClelland. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. 1987, pp. 318–362.

[140] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[141] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv:1406.1078* (2014).

[142] M. Schuster and K.K. Paliwal. "Bidirectional recurrent neural networks". In: *IEEE Transactions on Signal Processing* 45.11 (1997), pp. 2673–2681. DOI: 10.1109/78.650093.

[143] Alex Sherstinsky. "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network". In: *Physica D: Nonlinear Phenomena* 404 (2020), p. 132306. ISSN: 0167-2789.

[144] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in Neural Information Processing Systems* 27 (2014).

[145] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv:1411.1784* (2014).

[146] Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation". In: *arXiv:1710.10196* (2017).

[147] Tero Karras, Samuli Laine, and Timo Aila. "A style-based generator architecture for generative adversarial networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4401–4410.

[148] Christian Ledig et al. "Photo-realistic single image super-resolution using a generative adversarial network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 4681–4690.

[149] Phillip Isola et al. "Image-to-image translation with conditional adversarial networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1125–1134.

[150] D. Foster and K.J. Friston. *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly Media, Incorporated, 2023. ISBN: 9781098134181. URL: https://books.google.ie/books?id=tZxazwEACAAJ.

[151] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *arXiv:1409.0473* (2014).

[152] Mohammed Hassanin et al. "Visual attention methods in deep learning: An in-depth survey". In: *Information Fusion* 108 (2024), p. 102417.

[153] Nikolas Adaloglou and Sergios Karagiannakos. "How attention works in deep learning: understanding the attention mechanism in sequence models". In: *https://theaisummer.com/* (2020). Last Accessed 15 October 2024.

[154] Ashish Vaswani et al. "Attention is all you need". In: *Advances in Neural Information Processing Systems* 30 (2017).

[155] Maithra Raghu et al. "Do Vision Transformers See Like Convolutional Neural Networks?" In: *Advances in Neural Information Processing Systems*. Vol. 34. 2021, pp. 12116–12128.

[156] Da Li et al. "Learning to generalize: Meta-learning for domain generalization". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[157] Yogesh Balaji, Swami Sankaranarayanan, and Rama Chellappa. "MetaReg: Towards Domain Generalization using Meta-Regularization". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/647bba344396e7c8170902bcf2e15551-Paper.pdf.

[158] Yiying Li et al. "Feature-critic networks for heterogeneous domain generalization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3915–3924.

[159] Yingjun Du et al. "Learning to learn with variational information bottleneck for domain generalization". In: *European Conference on Computer Vision*. Springer. 2020, pp. 200–216.

[160] Qi Dou et al. "Domain Generalization via Model-Agnostic Learning of Semantic Features". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[161] Keyu Chen, Di Zhuang, and J. Morris Chang. "Discriminative adversarial domain generalization with meta-learning based cross-domain validation". In: *Neurocomputing* 467 (2022), pp. 418–426. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2021.09.046. URL: https://www.sciencedirect.com/science/article/pii/S0925231221014247.

[162] Hossein Sharifi-Noghabi et al. "Domain generalization via semi-supervised meta learning". In: *arXiv:2009.12658* (2020).

[163] Bailin Wang, Mirella Lapata, and Ivan Titov. "Meta-Learning for Domain Generalization in Semantic Parsing". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 366–379. DOI: 10.18653/v1/2021.naacl-main.33. URL: https://aclanthology.org/2021.naacl-main.33.

[164] Yuyang Zhao et al. "Learning to Generalize Unseen Domains via Memory-based Multi-Source Meta-Learning for Person Re-Identification". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 6277–6286.

[165] Ha Bui et al. "Exploiting Domain-Specific Features to Enhance Domain Generalization". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer et al. 2021. URL: https://openreview.net/forum?id=vKxFYApxBjr.

[166] Robert Kirk et al. "A survey of zero-shot generalisation in deep reinforcement learning". In: *Journal of Artificial Intelligence Research* 76 (2023), pp. 201–264.

[167] Roberta Raileanu et al. "Automatic Data Augmentation for Generalization in Reinforcement Learning". In: *Advances in Neural Information Processing Systems* 34 (2021).

[168] Kimin Lee et al. "Network Randomization: A Simple Technique for Generalization in Deep Reinforcement Learning". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HJgcvJBFvB.

[169] Kaiyang Zhou et al. "Domain generalization with mixstyle". In: (2021). arXiv: 2104.02008.

[170] Kaixin Wang et al. "Improving generalization in reinforcement learning with mixture regularization". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 7968–7978.

[171] Karl Cobbe et al. “Quantifying Generalization in Reinforcement Learning”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 1282–1289. URL: https://proceedings.mlr.press/v97/cobbe19a.html.

[172] Behnam Neyshabur et al. “Exploring generalization in deep learning”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[173] Antonio Torralba and Alexei A. Efros. “Unbiased look at dataset bias”. In: *CVPR 2011*. 2011, pp. 1521–1528. DOI: 10.1109/CVPR.2011.5995347.

[174] Dengxin Dai and Luc Van Gool. “Dark model adaptation: Semantic image segmentation from daytime to nighttime”. In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 3819–3824.

[175] Georg Volk et al. “Towards Robust CNN-based Object Detection through Augmentation with Synthetic Rain Variations”. In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 285–292. DOI: 10.1109/ITSC.2019.8917269.

[176] Michael A Alcorn et al. “Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4845–4854.

[177] Ishaan Gulrajani and David Lopez-Paz. “In search of lost domain generalization”. In: *arXiv:2007.01434* (2020).

[178] David Krueger et al. “Out-of-Distribution Generalization via Risk Extrapolation (REx)”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 5815–5826. URL: https://proceedings.mlr.press/v139/krueger21a.html.

[179] Isabela Albuquerque et al. “Generalizing to unseen domains via distribution matching”. In: *arXiv:1911.00804* (2019).

[180] Olga Russakovsky et al. “Imagenet large scale visual recognition challenge”. In: *International Journal of Computer Vision* 115 (2015), pp. 211–252.

[181] Dan Hendrycks et al. “The many faces of robustness: A critical analysis of out-of-distribution generalization”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8340–8349.

[182] Dan Hendrycks and Thomas Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: *International Conference on Learning Representations*. 2018.

[183] Dan Hendrycks et al. “Natural adversarial examples”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 15262–15271.

[184] Xingxuan Zhang et al. “Towards Unsupervised Domain Generalization”. In: *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 4900–4910. DOI: 10.1109/CVPR52688.2022.00486.

[185] Sivan Harary et al. “Unsupervised domain generalization by learning a bridge across domains”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5280–5290.

[186] Li-Cheng Lan, Huan Zhang, and Cho-Jui Hsieh. “Can Agents Run Relay Race with Strangers? Generalization of RL to Out-of-Distribution Trajectories”. In: *The Eleventh International Conference on Learning Representations*. 2022.

[187] Beining Han et al. “Learning Domain Invariant Representations in Goal-conditioned Block MDPs”. In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 764–776. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/06d172404821f7d01060cc9629171b2e-Paper.pdf.

[188] Akhilan Boopathy et al. “Model-agnostic measure of generalization difficulty”. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 2857–2884.

[189] Adrien Ali Taiga et al. “Investigating Multi-task Pretraining and Generalization in Reinforcement Learning”. In: *Deep Reinforcement Learning Workshop NeurIPS 2022*. 2022. URL: https://openreview.net/forum?id=NEtI9o7gtPL.

[190] Arsham Gholamzadeh Khoee, Yinan Yu, and Robert Feldt. “Domain Generalization through Meta-Learning: A Survey”. In: *arXiv:2404.02785* (2024).

[191] Udit Maniyar et al. “Zero shot domain generalization”. In: *arXiv:2008.07443* (2020).

[192] Shiori Sagawa* et al. “Distributionally Robust Neural Networks”. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=ryxGuJrFvS.

[193] Minghao Xu et al. “Adversarial domain adaptation with domain mixup”. In: *Proceedings of the AAAI Conference on artificial intelligence*. Vol. 34. 04. 2020, pp. 6502–6509.

[194] Shen Yan et al. “Improve unsupervised domain adaptation with mixup training”. In: *arXiv:2001.00677* (2020).

[195] Yaroslav Ganin et al. “Domain-adversarial training of neural networks”. In: *Journal of machine learning research* 17.59 (2016), pp. 1–35.

[196] Ya Li et al. “Domain generalization via conditional invariant representations”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[197] Baochen Sun and Kate Saenko. “Deep coral: Correlation alignment for deep domain adaptation”. In: *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*. Springer. 2016, pp. 443–450.

[198] Baochen Sun, Jiashi Feng, and Kate Saenko. “Return of frustratingly easy domain adaptation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 30. 1. 2016.

[199] Haoliang Li et al. “Domain Generalization with Adversarial Feature Learning”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5400–5409. DOI: 10.1109/CVPR.2018.00566.

[200] Martin Arjovsky et al. “Invariant risk minimization”. In: *arXiv:1907.02893* (2019).

[201] Christian Szegedy et al. “Rethinking the inception architecture for computer vision”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2818–2826.

[202] Iqbal H. Sarker. “Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions”. In: *SN Computer Science* 2.6 (Aug. 2021), p. 420. ISSN: 2661-8907. URL: https://doi.org/10.1007/s42979-021-00815-1.

[203] Kate Saenko et al. “Adapting Visual Category Models to New Domains”. In: *Computer Vision – ECCV 2010*. Ed. by Kostas Daniilidis, Petros Maragos, and Nikos Paragios. Springer Berlin Heidelberg, 2010, pp. 213–226. ISBN: 978-3-642-15561-1.

[204] Da Li et al. “Deeper, broader and artier domain generalization”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5542–5550.

[205] Chen Fang, Ye Xu, and Daniel N. Rockmore. "Unbiased Metric Learning: On the Utilization of Multiple Datasets and Web Images for Softening Bias". In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 1657–1664. DOI: 10.1109/ICCV.2013.208.

[206] Hemanth Venkateswara et al. "Deep Hashing Network for Unsupervised Domain Adaptation". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[207] Sara Beery, Grant Van Horn, and Pietro Perona. "Recognition in Terra Incognita". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

[208] Muhammad Ghifary et al. "Domain generalization for object recognition with multi-task autoencoders". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2551–2559.

[209] Xingchao Peng et al. "Moment Matching for Multi-Source Domain Adaptation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[210] Pierre Stock and Moustapha Cisse. "Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 498–512.

[211] Robert Geirhos et al. "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: (2018). arXiv: 1811.12231.

[212] Sara Beery, Grant Van Horn, and Pietro Perona. "Recognition in terra incognita". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 456–473.

[213] Fabio M Carlucci et al. "Domain generalization by solving jigsaw puzzles". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 2229–2238.

[214] Isabela Albuquerque et al. "Improving out-of-distribution generalization via multi-task self-supervised pretraining". In: *arXiv:2003.13525* (2020).

[215] Haohan Wang et al. "Learning Robust Global Representations by Penalizing Local Predictive Power". In: *Advances in Neural Information Processing Systems*. 2019, pp. 10506–10518.

[216] Riccardo Volpi et al. "Generalizing to Unseen Domains via Adversarial Data Augmentation". In: *Advances in Neural Information Processing Systems*. Vol. 31. 2018.

[217] Hyeonseob Nam et al. “Reducing domain gap by reducing style bias”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8690–8699.

[218] Nader Asadi et al. “Towards shape biased unsupervised representation learning for domain generalization”. In: *arXiv:1909.08245* (2019).

[219] Mahmoud Assran et al. “Masked Siamese networks for label-efficient learning”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 456–473.

[220] Mathilde Caron et al. “Emerging properties in self-supervised vision transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 9650–9660.

[221] Wenhai Wang et al. “Pyramid vision transformer: A versatile backbone for dense prediction without convolutions”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 568–578.

[222] Benjamin Graham et al. “LeViT: A Vision Transformer in ConvNet’s Clothing for Faster Inference”. In: *Proc of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 12259–12269.

[223] Haiping Wu et al. “Cvt: Introducing convolutions to vision transformers”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 22–31.

[224] Zihang Dai et al. “Coatnet: Marrying convolution and attention for all data sizes”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 3965–3977.

[225] Andreas Fürst et al. “Cloob: Modern hopfield networks with infoloob outperform clip”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 20450–20468.

[226] Chao Jia et al. “Scaling up visual and vision-language representation learning with noisy text supervision”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 4904–4916.

[227] Yangguang Li et al. “Supervision exists everywhere: A data efficient contrastive language-image pre-training paradigm”. In: *arXiv:2110.05208* (2021).

[228] Xiaohua Zhai et al. “Lit: Zero-shot transfer with locked-image text tuning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18123–18133.

[229] Amanpreet Singh et al. “Flava: A foundational language and vision alignment model”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 15638–15650.

[230] Jun Chen et al. “Visualgpt: Data-efficient adaptation of pretrained language models for image captioning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 18030–18040.

[231] Jean-Baptiste Alayrac et al. “Flamingo: a visual language model for few-shot learning”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 23716–23736.

[232] Zi-Yi Dou et al. “Coarse-to-fine vision-language pre-training with fusion in the backbone”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 32942–32956.

[233] Zirui Wang et al. “Simvlm: Simple visual language model pretraining with weak supervision”. In: *arXiv:2108.10904* (2021).

[234] Karan Desai and Justin Johnson. “Virtex: Learning visual representations from textual annotations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11162–11173.

[235] Maria Tsimpoukelli et al. “Multimodal few-shot learning with frozen language models”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 200–212.

[236] Ron Mokady, Amir Hertz, and Amit H Bermano. “Clipcap: Clip prefix for image captioning”. In: *arXiv:2111.09734* (2021).

[237] Oscar Mañas et al. “Mapl: Parameter-efficient adaptation of unimodal pre-trained models for vision-language few-shot prompting”. In: *arXiv:2210.07179* (2022).

[238] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer. 2014, pp. 740–755.

[239] Ranjay Krishna et al. “Visual genome: Connecting language and vision using crowdsourced dense image annotations”. In: *International Journal of Computer Vision* 123 (2017), pp. 32–73.

[240] Piyush Sharma et al. “Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Ed. by Iryna Gurevych and Yusuke Miyao. Melbourne,

Australia: Association for Computational Linguistics, July 2018, pp. 2556–2565. URL: https://aclanthology.org/P18-1238.

[241] Peter Young et al. "From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions". In: *Transactions of the Association for Computational Linguistics* 2 (2014). Ed. by Dekang Lin, Michael Collins, and Lillian Lee, pp. 67–78. DOI: 10.1162/tacl_a_00166. URL: https://aclanthology.org/Q14-1006.

[242] Vicente Ordonez, Girish Kulkarni, and Tamara Berg. "Im2Text: Describing Images Using 1 Million Captioned Photographs". In: *Advances in Neural Information Processing Systems*. Ed. by J. Shawe-Taylor et al. Vol. 24. Curran Associates, Inc., 2011. URL: https://proceedings.neurips.cc/paper_files/paper/2011/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[243] Bart Thomee et al. "Yfcc100m: The new data in multimedia research". In: *Communications of the ACM* 59.2 (2016), pp. 64–73.

[244] Christoph Schuhmann et al. "Laion-5b: An open large-scale dataset for training next generation image-text models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 25278–25294.

[245] Liunian Harold Li et al. "Visualbert: A simple and performant baseline for vision and language". In: *arXiv:1908.03557* (2019).

[246] Jiasen Lu et al. "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks". In: *Advances in Neural Information Processing Systems* 32 (2019).

[247] Hao Tan and Mohit Bansal. "Lxmert: Learning cross-modality encoder representations from transformers". In: *arXiv:1908.07490* (2019).

[248] Xiao Xu et al. "Bridgetower: Building bridges between encoders in vision-language representation learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 37. 9. 2023, pp. 10637–10647.

[249] Yixuan Su et al. "Language models can see: Plugging visual controls in text generation". In: *arXiv:2205.02655* (2022).

[250] Yash Goyal et al. "Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[251] Drew A Hudson and Christopher D Manning. "Gqa: A new dataset for real-world visual reasoning and compositional question answering". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 6700–6709.

[252] Sahar Kazemzadeh et al. "ReferItGame: Referring to Objects in Photographs of Natural Scenes". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by Alessandro Moschitti, Bo Pang, and Walter Daelemans. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 787–798. DOI: 10.3115/v1/D14-1086. URL: https://aclanthology.org/D14-1086.

[253] Bryan A Plummer et al. "Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2641–2649.

[254] Yuke Zhu et al. "Visual7w: Grounded question answering in images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4995–5004.

[255] Justin Johnson et al. "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2901–2910.

[256] Alane Suhr et al. "A corpus for reasoning about natural language grounded in photographs". In: *arXiv:1811.00491* (2018).

[257] Jun Xu et al. "MSR-VTT: A Large Video Description Dataset for Bridging Video and Language". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 5288–5296. DOI: 10.1109/CVPR.2016.571.

[258] George Awad et al. "TRECVID 2023 - A series of evaluation tracks in video understanding". In: *Proceedings of TRECVID 2023*. NIST, USA. 2023.

[259] George Awad et al. "TRECVid 2018: Benchmarking video activity detection, video captioning and matching, video storytelling linking and video search". In: *Proceedings of TRECVID 2018*. 2018.

[260] Antoine Miech et al. "Howto100m: Learning a text-video embedding by watching hundred million narrated video clips". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2630–2640.

[261] Kenneth Marino et al. "OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[262] Amanpreet Singh et al. “Towards VQA Models That Can Read”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8317–8326.

[263] Oleksii Sidorov et al. “Textcaps: a dataset for image captioning with reading comprehension”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer. 2020, pp. 742–758.

[264] Danna Gurari et al. “Vizwiz grand challenge: Answering visual questions from blind people”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 3608–3617.

[265] Zhanzhan Cheng et al. “You Only Recognize Once: Towards Fast Video Text Spotting”. In: *Proceedings of the 27th ACM International Conference on Multimedia*. ACM. 2019, pp. 855–863.

[266] Max Bain et al. “Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval”. In: *IEEE International Conference on Computer Vision*. 2021.

[267] Douwe Kiela et al. “The hateful memes challenge: Detecting hate speech in multimodal memes”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2611–2624.

[268] Ning Xie et al. “Visual entailment: A novel task for fine-grained image understanding”. In: *arXiv:1901.06706* (2019).

[269] Tristan Thrush et al. “Winoground: Probing vision and language models for visio-linguistic compositionality”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5238–5248.

[270] Dengxin Dai and Luc van Gool. “Dark Model Adaptation: Semantic Image Segmentation from Daytime to Nighttime”. In: *2018 21st International Conference on Intelligent Transportation Systems*. 2018, pp. 3819–3824. DOI: 10.1109/ITSC.2018.8569387.

[271] Michael A. Alcorn et al. “Strike (With) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects”. In: *2019 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 4840–4849. DOI: 10.1109/CVPR.2019.00498.

[272] Austin Stone et al. “The Distracting Control Suite–A Challenging Benchmark for Reinforcement Learning from Pixels”. In: *arXiv:2101.02722* (2021).

[273] Ishaan Gulrajani and David Lopez-Paz. “In Search of Lost Domain Generalization”. In: *International Conference on Learning Representations*. 2021.

[274] Yaroslav Ganin et al. “Domain-Adversarial Training of Neural Networks”. In: *Domain Adaptation in Computer Vision Applications*. Ed. by Gabriela Csurka. Springer International Publishing, 2017, pp. 189–209.

[275] Kei Akuzawa, Yusuke Iwasawa, and Yutaka Matsuo. “Adversarial invariant feature learning with accuracy constraint for domain generalization”. In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2020, pp. 315–331.

[276] Fabio M. Carlucci et al. “Domain Generalization by Solving Jigsaw Puzzles”. In: *2019 IEEE/CVF Conf on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2224–2233.

[277] Xiao Liu et al. “Self-Supervised Learning: Generative or Contrastive”. In: *IEEE Transactions on Knowledge and Data Engineering* 35.1 (2023), pp. 857–876. DOI: 10.1109/TKDE.2021.3090866.

[278] Hugo Touvron et al. “Training data-efficient image transformers & distillation through attention”. In: *Proceedings of the 38th International Conference on Machine Learning*. Vol. 139. PMLR, 2021, pp. 10347–10357.

[279] Hangbo Bao et al. “BEiT: BERT Pre-Training of Image Transformers”. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=p-BhZSz59o4.

[280] Han Zhao et al. “On Learning Invariant Representations for Domain Adaptation”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. PMLR, 2019, pp. 7523–7532.

[281] Zangwei Zheng et al. “Prompt vision transformer for domain generalization”. In: *arXiv:2208.08914* (2022).

[282] Maryam Sultana et al. “Self-Distilled Vision Transformer for Domain Generalization”. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. 2022, pp. 3068–3085.

[283] Hamza Riaz and Alan F. Smeaton. “Vision Based Machine Learning Algorithms for Out-of-Distribution Generalisation”. In: *Intelligent Computing*. Ed. by Kohei Arai. Cham: Springer Nature Switzerland, 2023, pp. 870–884. ISBN: 978-3-031-37963-5.

[284] Pengguang Chen et al. “Gridmask data augmentation”. In: *arXiv:2001.04086* (2020).

[285] Zhun Zhong et al. “Random erasing data augmentation”. In: *Proceedings of the AAAI Conference on artificial intelligence*. Vol. 34. 07. 2020, pp. 13001–13008.

[286] Terrance DeVries and Graham W Taylor. "Improved regularization of convolutional neural networks with cutout". In: *arXiv:1708.04552* (2017).

[287] Hongyi Zhang et al. "mixup: Beyond empirical risk minimization". In: *arXiv: 1710.09412* (2017).

[288] Sangdoo Yun et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 6023–6032.

[289] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. "Dropblock: A regularization method for convolutional networks". In: *Advances in Neural Information Processing Systems* 31 (2018).

[290] Deepak Pathak et al. "Context encoders: Feature learning by inpainting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2536–2544.

[291] Krishna Kumar Singh and Yong Jae Lee. "Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3524–3533.

[292] Hamza Riaz and Alan F Smeaton. "Domain generalisation with bidirectional encoder representations from vision transformers". In: *Irish Machine Vision and Image Processing Conference (IMVIP), Galway, August*. 2023.

[293] Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. "A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2024), pp. 1–20. DOI: 10.1109/TPAMI.2024.3447085.

[294] Song Han et al. "Learning both weights and connections for efficient neural network". In: *Advances in neural information processing systems* 28 (2015).

[295] Geoffrey Hinton. "Distilling the Knowledge in a Neural Network". In: *arXiv: 1503.02531* (2015).

[296] Yongcheng Jing et al. "Meta-aggregator: Learning to aggregate for 1-bit graph neural networks". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 5301–5310.

[297] Songhua Liu et al. "Dataset distillation via factorization". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 1100–1113.

[298] Songhua Liu et al. "Slimmable Dataset Condensation". In: *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2023, pp. 3759–3768. DOI: 10.1109/CVPR52729.2023.00366.

[299] Jiaxiang Wu et al. “Quantized convolutional neural networks for mobile devices”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4820–4828.

[300] Xingyi Yang, Jingwen Ye, and Xinchao Wang. “Factorizing knowledge in neural networks”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 73–91.

[301] Yiding Yang et al. “Distilling knowledge from graph convolutional networks”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 7074–7083.

[302] Jingwen Ye, Songhua Liu, and Xinchao Wang. “Partial network cloning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 20137–20146.

[303] Ruonan Yu, Songhua Liu, and Xinchao Wang. “Dataset distillation: A comprehensive review”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).

[304] Xiaohan Ding et al. “Resrep: Lossless CNN pruning via decoupling remembering and forgetting”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 4510–4520.

[305] Shangqian Gao et al. “Network Pruning via Performance Maximization”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 9266–9276. DOI: 10.1109/CVPR46437.2021.00915.

[306] Tailin Liang et al. “Pruning and quantization for deep neural network acceleration: A survey”. In: *Neurocomputing* 461 (2021), pp. 370–403.

[307] Mingbao Lin et al. “Hrank: Filter pruning using high-rank feature map”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 1529–1538.

[308] Sejun Park et al. “Lookahead: A far-sighted alternative of magnitude-based pruning”. In: *arXiv:2002.04809* (2020).

[309] Huan Wang et al. “Neural pruning via growing regularization”. In: *arXiv: 2012.09243* (2020).

[310] Wenxiao Wang et al. “Accelerate CNNs from three dimensions: A comprehensive pruning framework”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10717–10726.

[311] Ruichi Yu et al. “Nisp: Pruning networks using neuron importance score propagation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 9194–9203.

[312] Haoran You et al. “Drawing Early-Bird Tickets: Toward More Efficient Training of Deep Networks”. In: *International Conference on Learning Representations*.

[313] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Ed. by Jill Burstein, Christy Doran, and Thamar Solorio. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[314] Jiayang Wu et al. “AI-generated content (aigc): A survey”. In: (2023). arXiv: 2304.06632.

[315] Vikash Sehwag et al. “Hydra: Pruning adversarially robust neural networks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 19655–19666.

[316] Saleh Ashkboos et al. “Slicegpt: Compress large language models by deleting rows and columns”. In: *arXiv:2401.15024* (2024).

[317] Xinyin Ma, Gongfan Fang, and Xinchao Wang. “Llm-pruner: On the structural pruning of large language models”. In: *Advances in Neural Information Processing Systems* 36 (2023), pp. 21702–21720.

[318] Emily L Denton et al. “Exploiting linear structure within convolutional networks for efficient evaluation”. In: *Advances in Neural Information Processing Systems* 27 (2014).

[319] Tim Dettmers et al. “Qlora: Efficient finetuning of quantized llms”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[320] Wenqi Shao et al. “Omniquant: Omnidirectionally calibrated quantization for large language models”. In: *arXiv:2308.13137* (2023).

[321] Hanxiao Liu, Karen Simonyan, and Yiming Yang. “Darts: Differentiable architecture search”. In: *arXiv:1806.09055* (2018).

[322] Miao Zhang et al. “iDARTS: Differentiable Architecture Search with Stochastic Implicit Gradients”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 12557–12566. URL: https://proceedings.mlr.press/v139/zhang21s.html.

[323] Yuxian Gu et al. “Knowledge distillation of large language models”. In: *arXiv:2306.08543* (2023).

[324] Xiaohan Xu et al. “A survey on knowledge distillation of large language models”. In: *arXiv:2402.13116* (2024).

[325] Xiaohan Ding et al. “Centripetal SGD for pruning very deep convolutional networks with complicated structure”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4943–4953.

[326] Hao Li et al. “Pruning filters for efficient convnets”. In: *arXiv:1608.08710* (2016).

[327] Zhonghui You et al. “Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[328] Xin Dong, Shangyu Chen, and Sinno Pan. “Learning to prune deep neural networks via layer-wise optimal brain surgeon”. In: *Advances in Neural Information Processing Systems* 30 (2017).

[329] Yiwen Guo, Anbang Yao, and Yurong Chen. “Dynamic network surgery for efficient DNNs”. In: *Advances in Neural Information Processing Systems* 29 (2016).

[330] Chaoqi Wang, Guodong Zhang, and Roger Grosse. “Picking winning tickets before training by preserving gradient flow”. In: *arXiv:2002.07376* (2020).

[331] Mingjie Sun et al. “A simple and effective pruning approach for large language models”. In: *arXiv:2306.11695* (2023).

[332] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. “Thinet: A filter level pruning method for deep neural network compression”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5058–5066.

[333] Liyang Liu et al. “Group Fisher Pruning for Practical Network Compression”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 7021–7032. URL: `https://proceedings.mlr.press/v139/liu21ab.html`.

[334] Gongfan Fang et al. “Depgraph: Towards any structural pruning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 16091–16101.

[335] Jonathan Frankle and Michael Carbin. “The lottery ticket hypothesis: Finding sparse, trainable neural networks”. In: *arXiv:1803.03635* (2018).

[336] Namhoon Lee, Thalaiyasingam Ajanthan, and Philip HS Torr. "Snip: Single-shot network pruning based on connection sensitivity". In: *arXiv:1810.02340* (2018).

[337] Hidenori Tanaka et al. "Pruning neural networks without any data by iteratively conserving synaptic flow". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6377–6389.

[338] Utku Evci et al. "Rigging the lottery: Making all tickets winners". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 2943–2952.

[339] Chenglong Zhao et al. "Variational Convolutional Neural Network Pruning". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 2775–2784. DOI: 10.1109/CVPR.2019.00289.

[340] Yang He et al. "Soft filter pruning for accelerating deep convolutional neural networks". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 2018, pp. 2234–2240.

[341] Zechun Liu et al. "Metapruning: Meta learning for automatic neural network channel pruning". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3296–3305.

[342] Minsik Cho, Saurabh Adya, and Devang Naik. "PDP: parameter-free differentiable pruning is all you need". In: *Advances in Neural Information Processing Systems* 36 (2024).

[343] Wei Wen et al. "Learning structured sparsity in deep neural networks". In: *Advances in Neural Information Processing Systems* 29 (2016).

[344] Zehao Huang and Naiyan Wang. "Data-driven sparse structure selection for deep neural networks". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 304–320.

[345] Ariel Gordon et al. "Morphnet: Fast & simple resource-constrained structure learning of deep networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1586–1595.

[346] Decebal Constantin Mocanu et al. "Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science". In: *Nature Communications* 9.1 (2018), p. 2383.

[347] Shiwei Liu et al. "Sparse training via boosting pruning plasticity with neuroregeneration". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 9908–9922.

[348] Xiaoliang Dai, Hongxu Yin, and Niraj K. Jha. "NeST: A Neural Network Synthesis Tool Based on a Grow-and-Prune Paradigm". In: *IEEE Transactions on Computers* 68.10 (2019), pp. 1487–1497. DOI: 10.1109/TC.2019.2914438.

[349] Hesham Mostafa and Xin Wang. "Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 4646–4655.

[350] Tao Lin et al. "Dynamic Model Pruning with Feedback". In: *ICLR-International Conference on Learning Representations*. 2020.

[351] Tim Dettmers and Luke Zettlemoyer. "Sparse networks from scratch: Faster training without losing performance". In: *arXiv:1907.04840* (2019).

[352] Yang He et al. "Filter pruning via geometric median for deep convolutional neural networks acceleration". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4340–4349.

[353] Xuefei Ning et al. "Dsa: More efficient budgeted pruning via differentiable sparsity allocation". In: *European Conference on Computer Vision*. Springer. 2020, pp. 592–607.

[354] Shaopeng Guo et al. "DMCP: Differentiable Markov Channel Pruning for Neural Networks". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020), pp. 1536–1544. URL: https://api.semanticscholar.org/CorpusID:218537870.

[355] Pavlo Molchanov et al. "Pruning convolutional neural networks for resource efficient inference". In: *arXiv:1611.06440* (2016).

[356] Yann LeCun, John Denker, and Sara Solla. "Optimal Brain Damage". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989. URL: https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf.

[357] Pavlo Molchanov et al. "Importance estimation for neural network pruning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 11264–11272.

[358] Gongfan Fang et al. "Isomorphic Pruning for Vision Models". In: *arXiv: 2407.04616* (2024).

[359] Alan F Smeaton. "Understanding Foundation Models: Are We Back in 1924?" In: *arXiv:2409.07618* (2024).

[360] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[361] Aditya Ramesh et al. “Zero-shot text-to-image generation”. In: *International Conference on Machine Learning*. Pmlr. 2021, pp. 8821–8831.

[362] Daking Rai et al. “A practical review of mechanistic interpretability for transformer-based language models”. In: *arXiv:2407.02646* (2024).