Lancaster University

Interpretable Deep Learning and its application to Earth Observation

Ziyang Zhang, BS, B.Mgmt., MSc

School of Computing and Communications

Lancaster University

A thesis submitted for the degree of

Doctor of Philosophy

December, 2025

Interpretable Deep Learning and its application to Earth Observation

Ziyang Zhang, BS, B.Mgmt., MSc.

School of Computing and Communications, Lancaster University

A thesis submitted for the degree of *Doctor of Philosophy*. December, 2025.

Abstract

Deep learning algorithms have revolutionized almost every industry over the last 10 years. They have been used in a wide range of different domains, including image classification and generation, language translation, robotics, health care, and earth observation. However, with the benefit of the promising performance that such deep learning algorithms provides, the models have become larger and more complex. This has led to the current state-of-the-art models becoming 'back box' models where people do not understand the reasoning behind a particular decision made by the model. This can be fatal in areas where human life and property are at stake.

Floods, as one of the most common natural disasters, cause significant damage to people's lives and property. Therefore, flood detection has been a key research area in remote sensing. Recently, deep learning based techniques have been extensively used to solve this problem because of their ability to provide high accuracy. However, their black box nature remains controversial and has been criticized by human users. In the case of flood detection, where the safety of a large number of people is at stake, the interpretability and transparency of the algorithm are even more critical.

In conclusion, the dissertation aims to provide general interpretable deep learning methods and their application to flood detection. The thesis can be summarized by the following key contributions:

- **Interpretable method for general image classification**. An interpretable-by-design algorithm (IDEAL) for image classification has been proposed. IDEAL was shown to provide comparable results compared to standard deep

learning architectures without fine-tuning while providing an interpretable decision-making process.

- **Interpretable methods for semantic segmentation**. An interpretable semantic segmentation method (IDSS) for flood mapping has been proposed. It is a prototype-based method that provides both high accuracy and interpretability. Linguistic IF ... THEN rules together with the prototypes layer, which is the combination of the latent feature space and the raw feature space, have been used to explain the algorithm's decision-making process.

- **Interpretable methods for flood detection and flood mapping**. An Interpretable Multi-stage Approach to Flood Detection from time series Multispectral Data (IMAFD) has been proposed. IMAFD progressively narrow down the research problem from the time series sequence level to the multi-image level to the image level, providing an automatic, efficient and interpretable approach to flood detection.

- **Interpretable ensemble method with geoscience foundation models for flood mapping**. An Interpretable Ensemble Geoscience Foundation Model for flood mapping (IEGF) has been proposed. IEGF introduces an ensemble foundation model framework that achieves state-of-the-art performance on the flood mapping task while offering an interpretable decision-making process for human users.

Together, these contributions advance the development of interpretable deep learning models for Earth observation and demonstrate that interpretability and high performance can be achieved simultaneously.

Acknowledgements

Four years have flown by, and looking back on my PhD, I am glad to have had this enjoyable and challenging journey in my life. I am honoured to have met so many talented people during this time, and I would like to thank them all for their support.

First of all, I would like to thank my supervisor, Prof. Plamen Angelov, who has always supported and guided me throughout my PhD career. Not only did I learn from him how to plan and solve one academic challenge after another. At the same time, his passion for science has always inspired me.

Second, I would like to thank the European Space Agency's Phi Lab for supporting my PhD project and for hosting me during my visit to Phi Lab. In particular, I would also like to thank Dr Pierre Philippe Mathieu and Dr Nicolas Longépé for their guidance and support throughout my project. Furthermore, I would also like to thank Dr Nicolas Longépé, Dr Bertrand Le Saux and Sabrina Ricci for hosting me during my stay at the Phi Lab. In addition, I would like to thank Dr. Nikolaos Dionelis and Dr. Casper Fibaek, who worked and collaborated with me during my stay. Finally, I would like to thank everyone I had the pleasure of meeting at Phi Lab.

Third, I would like to thank a number of co-authors with whom I had the opportunity to collaborate. They are Dr Eduardo Soares, Dr Dmitry Kangin, Dr Yi Li and Dr Valerio Marsocci. It has been a great pleasure to work and collaborate with them.

Finally, I would like to thank my parents, family, and friends for their continuous support throughout my PhD career. In particular, I would like to thank my uncle Rongzu Yang (in memoriam), I hope there is no sickness in heaven and that all is well with you there!

Declaration

I declare that the work presented in this thesis is, to the best of my knowledge and belief, original and my own work. The material has not been submitted, either in whole or in part, for a degree at this, or any other university. This thesis does not exceed the maximum permitted word length of 80,000 words including appendices and footnotes, but excluding the bibliography. A rough estimate of the word count is: 24153

Ziyang Zhang

Publications

Zhang, Ziyang, Plamen Angelov, Eduardo Soares, et al. (2022). "An interpretable deep semantic segmentation method for earth observation". In: *2022 IEEE 11th International Conference on Intelligent Systems (IS)*. IEEE, pp. 1–8.

Zhang, Ziyang, Plamen Angelov, Dmitry Kangin, et al. (2024). "IMAFD: An Interpretable Multi-stage Approach to Flood Detection from time series Multispectral Data". In: *arXiv preprint arXiv:2405.07916*. Major revision with Applied Soft Computing journal.

Zhang, Ziyang, Plamen Angelov, Casper Fibaek, et al. (2025). "An Interpretable Ensemble Geoscience Foundation Model for Flood Mapping". Manuscript in preparation.

Angelov, Plamen, Dmitry Kangin, and **Ziyang Zhang** (2025). "IDEAL: Interpretable-by-Design ALgorithms for learning from foundation feature spaces". In: *Neurocomputing*, p. 129464.

Soares, Eduardo, Plamen Angelov, and **Ziyang Zhang** (2024). "An explainable approach to deep learning from CT-scans for covid identification". In: *Evolving Systems* 15.6, pp. 2159–2168.

Contents

Publications **vi**

1 Introduction **1**

1.1 Motivation . . . 1

1.2 Objectives . . . 2

1.3 Contributions . . . 2

1.3.1 Interpretable-by-design deep learning algorithms . . . 3

1.3.2 Interpretable deep semantic segmentation method . . . 3

1.3.3 Interpretable multi-stage approach to Flood detection . . . 4

1.3.4 Interpretable ensemble geoscience foundation model for flood mapping . . . 5

1.4 Thesis Outline . . . 5

2 Background and Related Work **6**

2.1 Machine learning . . . 6

2.1.1 K-means and its variants . . . 7

2.1.2 Support Vector Machines . . . 8

2.1.3 Decision Trees . . . 10

2.1.4 K nearest neighbor (KNN) . . . 12

2.1.5 Recursive Density Estimation (RDE) . . . 12

2.1.6 Fuzzy rules . . . 13

2.2 Deep Learning . . . 14

2.2.1 Convolutional Neural Networks 14
2.2.1.1 Convolution 15
2.2.1.2 Pooling 16
2.2.2 Transformer 16
2.2.2.1 Attention mechanism 17
2.2.2.2 Transformer architecture 18
2.2.2.3 Vision Transformer 18
2.2.3 Foundation models 21
2.2.3.1 SatlasNet 21
2.2.3.2 Prithvi 22
2.2.4 Continual learning 22
2.3 Image segmentation 23
2.3.1 SLIC 24
2.3.2 U-Net 25
2.3.3 DeepLabv3+ 26
2.4 Remote sensing 26
2.4.1 Optical images 28
2.4.2 SAR images 28
2.4.3 Traditional flood detection methods 30
2.5 Deep Learning in Earth Observation 32
2.5.1 Semantic Segmentation on flood detection 32
2.5.2 Change detection 33
2.6 Explainable AI 34
2.6.1 Explainable AI taxonomies 34
2.6.1.1 Feature attribution 35
2.6.1.2 Distillation 37
2.6.1.3 Intrinsic explanations 38
2.6.1.4 Contrastive examples 40
2.6.2 Explainable AI in Earth Observation 41

3 An interpretable-by-design deep learning algorithm 44

3.1 Background . . . 45

3.2 Methodology . . . 48

3.2.1 Problem statement . . . 48

3.2.1.1 Offline learning . . . 48

3.2.1.2 Online (lifelong) learning . . . 49

3.2.2 Choice of functions d and h . . . 50

3.2.3 Difference from the other prototype-based frameworks . . . 50

3.2.3.1 xDNN . . . 51

3.2.3.2 DNC . . . 51

3.2.3.3 ProtoPNet . . . 51

3.2.4 Prototype selection through clustering . . . 52

3.3 Experiments . . . 52

3.3.1 Experimental setting . . . 54

3.3.1.1 Datasets . . . 54

3.3.1.2 Feature extractors . . . 54

3.3.1.3 Prototype selection techniques . . . 54

3.3.1.4 Baselines . . . 55

3.4 Empirical questions . . . 55

3.4.1 Offline classification . . . 57

3.4.2 Demonstration of overfitting in the finetuned feature spaces and the prototype selection impact . . . 58

3.4.3 Continual learning . . . 62

3.4.4 Study of Interpretability . . . 64

3.4.5 Impact of confounding on interpretations . . . 66

3.5 Additional Experimental Analyses and Results . . . 72

3.5.1 Experimental setup . . . 72

3.5.2 Complete experimental results . . . 73

3.5.3 Sensitivity analysis for the number of prototypes . . . 74

3.5.4 Additional results for the demonstration of overfitting 74
3.5.5 Linguistic interpretability of the proposed framework outputs 82
3.6 Broader Impact Statement 82
3.7 Conclusion 85

4 An Interpretable Deep Semantic Segmentation (IDSS) Method 87
4.1 IDSS architecture 87
4.2 Experiment setup 92
4.2.1 Worldfloods Dataset 92
4.2.2 Training 92
4.2.3 Testing 93
4.2.4 Performance Evaluation 93
4.3 Interpretability of the proposed IDSS methods 94
4.4 Results and Analysis 95
4.5 Conclusion 98

5 Interpretable flood detection and flood mapping methods 99
5.1 Introduction 99
5.2 IMAFD framework 101
5.2.1 Anomalous images detection 101
5.2.2 Binary Change detection 102
5.2.3 Semantic change detection 103
5.2.4 Decision making 107
5.3 Experiments 108
5.3.1 Datasets 108
5.3.2 Evaluation metrics 108
5.4 Results 109
5.4.1 Anomalous image detection 109
5.4.2 Semantic segmentation 110
5.4.3 Semantic change detection 113

5.5 Conclusion 113

6 An Interpretable Ensemble Geoscience Foundation Model for Flood Mapping 116

6.1 Background 116

6.2 Methodology 118

6.2.1 IDSS v2 module 118

6.2.2 Proposed IEGF framework 119

6.2.3 Model selection 122

6.3 Experiment setup 123

6.3.1 Dataset 123

6.3.2 GFM Fine Tuning Architecture 123

6.3.3 Fine Tuning strategy 124

6.3.4 Prototype generation 124

6.4 Results and Comparison 125

6.4.1 GFM generalization results 125

6.4.2 GFM comparison with the proposed method 126

6.4.3 Prototype visulization 127

6.4.4 Interpretability 127

6.5 Conclusion 130

7 Conclusions and Future Work 131

7.1 Main Contributions 132

7.2 Future Work 133

References 135

List of Tables

2.1 Spectral Bands of Sentinel-2 with their Descriptions and Resolutions (Drusch et al., 2012). 29

3.1 F1 score comparison for CUB dataset (Wah et al., 2011) 68

3.2 CIFAR-10 classification task comparison for the case of no finetuning of the feature extractor. 75

3.3 CIFAR-10 classification task comparison for the case of finetuned models 76

3.4 CIFAR-100 classification task comparison for the case of no finetuning of the feature extractor. 77

3.5 CIFAR-100 classification task comparison for the case of finetuned models 78

3.6 STL10 classification task comparison for the case of no finetuning . . 79

3.7 OxfordIIITPets classification task comparison for the case of no finetuning 79

3.8 EuroSAT classification task comparison for the case of no finetuning . 80

3.9 CalTech101 classification task comparison 80

4.1 Statistics of the Worldfloods dataset (Mateo-Garcia et al., 2021). . . . 92

4.2 Comparison of IoU and recall results. 95

5.1 Performance comparison of IMAFD anomalous image detection on the RavAEn dataset 109

5.2 Comparison of IoU results (%) among different models 111
5.3 Comparison of IoU results (%) for different numbers of prototypes per class. 111
5.4 Semantic segmentation change results comparison 115

6.1 Comparison between IDSS, IDSS+, and IDSS v2. 119
6.2 Geoscience foundation model summary. 123
6.3 Comparison of results for different bands on various foundation models 125
6.4 Performance comparison of various foundation models and the proposed method with IDSS v2 module. 126

List of Figures

2.1 A regression trees with two inputs (Murphy, 2022). 11
2.2 A classification tree for whether a customer buys a laptop in an electronics store (Gupta et al., 2017). 11
2.3 IF...THEN rules of the Caltech 101 dataset (P. P. Angelov and Xiaowei Gu, 2019). 14
2.4 2d convolution. 15
2.5 A CNN for image classification. 16
2.6 Scaled Dot-product Attention and Multi-Head Attention (Vaswani, 2017). 17
2.7 Transformer architecture (Vaswani, 2017). 19
2.8 Vision Transformer (ViT) architecture (Dosovitskiy et al., 2020). . . . 21
2.9 SatlasNet architecture (Bastani et al., 2023). 22
2.10 Prithvi architecture (Jakubik et al., 2023). 23
2.11 U-Net (Ronneberger, Fischer, and Brox, 2015). 26
2.12 DeepLabv3+ (L.-C. Chen et al., 2018). 27
2.13 (a) A radar pulse is transmitted from the satellite to a tree (b) The radar pulse is backscattered by the tree back to the satellite (P. Singh et al., 2021). 29
2.14 Grad-CAM (Selvaraju et al., 2017). 37
2.15 ProtoPNet reasoning process (C. Chen et al., 2019). 40
2.16 xDNN reasoning process (P. Angelov and Soares, 2020). 41

3.1 General prototype-based learning framework 44
3.2 Difference between (a) a standard deep-learning model, and (b) the proposed prototype-based approach, IDEAL 47
3.3 Experimental setup 53
3.4 Comparison of the proposed IDEAL framework vs the baseline DNN 57
3.5 Comparison of results on CIFAR-10 (ViT, k nearest neighbours). . . 58
3.6 Comparison of results with ViT (Dosovitskiy et al., 2020) as a feature extractor on a number of datasets 59
3.7 Comparison of training time expenditure on CIFAR-10 (left) and CIFAR-100 (right) with and without funetuning (ViT). 60
3.8 tSNE plots for original (top-left) vs finetuned (top-right) features of ViT 61
3.9 Comparison between the model performance on CIFAR-100 61
3.10 Results **without** finetuning for various problems (ViT). 62
3.11 Accuracy of IDEAL in class-incremental learning experiments for different backbones 63
3.12 Interpreting the predictions of the proposed model (k-means (nearest), CIFAR-10, ViT). 64
3.13 Interpreting the predictions (k-means (nearest), OxfordIIITPets/STL-10/Caltech101, ViT). 65
3.14 iCIFAR-10 class-incremental learning: evolution of prototype ranking. 67
3.15 Comparing the interpretations of the non-finetuned and finetuned model 70
3.16 Intuitive explanation behind better performance of non-finetuned model. 71
3.17 Accuracy sensitivity to the number of per-class prototypes 81
3.18 tSNE plots 81
3.19 An example of symbolic decision rules (OxfordIIITPets) 82
3.20 Interpreting the model predictions 83

4.1 Training process of IDSS. 88

4.2 Validation process of IDSS 91
4.3 Linguistic rules of IDSS 94
4.4 Comparison of segmentation results 97

5.1 IMAFD architecture 100
5.2 RGB visualization of Sentinel-2 time series multispectral images . . . 101
5.3 IDSS+ architecture. 104
5.4 Comparison of segmentation results 106
5.5 Comparison of segmentation results 107
5.6 Prototype visualization 112
5.7 Visualisation of the decision-making process with the prototype in the latent space using UMAP 112
5.8 Comparison of segmentation change detection results 114

6.1 Training process 120
6.2 Testing process 121
6.3 GFM fine tuning architecture. 124
6.4 Comparison of segmentation segmentation results on Worldfloods v2 dataset. 128
6.5 Visualization with Umap for Prithvi and Unet prototypes. 129
6.6 Visualization of the decision making process of the model. 129

Chapter 1

Introduction

People have been fighting floods since time immemorial. In the famous story of Noah's Ark, Noah had to build a large boat to escape the flood. The development of remote sensing technologies has enabled humans to carry out large-scale Earth Observation tasks, including flood detection, without the use of ground-based equipment. In the past decades, with the rapid development of machine learning, especially deep learning technologies, numerous deep learning neural network-based flood mapping and detection methods have emerged and been applied to reality. However, it poses a new challenge: the black-box nature of deep learning algorithms. Such algorithms are criticized because they have hundreds of millions of parameters that are not easily understood by humans, and their decision-making process is unexplainable. This dissertation aims to develop general interpretable deep learning methods and apply them to earth observation tasks, specifically for flood mapping and detection.

1.1 Motivation

Thanks to the development of remote sensing and deep learning technologies, Earth Observation (EO) has attracted increasing attention (Persello et al., 2022; Dimitrovski et al., 2023). In particular, the deep learning neural network has shown promising performance in flood mapping and flood detection (Portalés-Julià et al., 2023; B. Zhao, Sui, and J. Liu, 2023). However, the use of deep learning algorithms also poses a new challenge, as their decision-making process is unexplainable to humans. Yet, these characteristics are crucial for EO tasks, which require the model to be interpretable and process understanding (Reichstein et al., 2019; Höhl et al., 2024). Furthermore, governments and organizations require the AI algorithms to be more secure, transparent and interpretable (Ruschemeier, 2023). The motivation of this dissertation is to resolve such challenges by developing approaches that offer both high accuracy and interpretability, tailored to flood mapping and detection.

1.2 Objectives

The primary objective of this dissertation is to provide safe and trustworthy approaches that could benefit from deep learning and provide process understanding and interpretability for general image classification tasks. This is different from post-hoc explanation algorithms that can't accurately reflect how the model makes decisions (W. Yang et al., 2023). This dissertation provides algorithms that are interpretable-by-design, while their decision-making process is explainable and easily understood by human users.

The second objective is to apply such an architecture to EO tasks, specifically tailored to flood mapping and detection. Since the flood detection task itself is a high-risk task with high stakes in human life and property, the ability to provide high performance while allowing human users to audit and verify the algorithm's decision-making process is particularly important.

1.3 Contributions

This thesis focuses on providing a prototype-based interpretable deep learning framework for both general and specific domains, such as earth observation. The contributions are summarized below.

1.3.1 Interpretable-by-design deep learning algorithms

- In collaboration with my co-authors, we propose a framework called IDEAL (**I**nterpretable-by-design **DE**ep learning **AL**gorithms), which transforms a given non-interpretable latent space into an interpretable one based on prototypes, derived from the training set without fine-tuning and quantifies the performance gap between such a model, its fine-tuned counterpart and standard deep learning architectures.

- We demonstrate the benefits of the proposed framework on transfer and lifelong learning scenarios. Namely, in a fraction of the training time and without fine-tuning of latent features, the proposed models achieve performance competitive with standard deep learning techniques.

- We demonstrate the model's interpretability on classification and life-long tasks and show that without fine-tuning, the resulting models achieve better performance on confounded Caltech-UCSD Birds (CUB) data compared to fine-tuned counterparts (Wah et al., 2011; Bontempelli et al., 2022).

1.3.2 Interpretable deep semantic segmentation method

- I propose a framework called IDSS (**I**nterpretable **D**eep **S**emantic **S**egmentation), which provides a prototype-based interpretable method that offers both human interpretable decision-making and promising performance.

- The proposed methods were tested on the Worldfloods flood mapping dataset and outperform other methods, in particular the U-Net for IoU total water and Recall total water.

1.3.3 Interpretable multi-stage approach to Flood detection

- I propose a framework called IMAFD (**I**nterpretable **M**ulti-stage **A**pproach to **F**lood **D**etection), which, to the best of my knowledge, is the first to provide a comprehensive solution for flood detection. It treats flood detection as a multi-stage problem, where anomaly detection in the first stage efficiently reduces the number of images to be processed for dense prediction in the later stage.

- The first two stages: anomalous images detection and binary change detection are unsupervised and statistics based, allowing the framework to adapt seamlessly to diverse environmental conditions without relying on predefined labels or training data. Furthermore, the statistical feature enhances the interpretability, making the decision making process transparent and easy to understand by human users.

- The IDSS+ is introduced as a critically important component of the IMAFD framework. Compared to IDSS (Z. Zhang, P. Angelov, Soares, et al., 2022), IDSS+ improves the performance and provides an option for better interpretability by utilizing real pixels as prototypes at the expense of a slight reduction in performance. In addition, several techniques are introduced to help human users better understand the algorithm's decision-making process. These include UMAP plots, confidence maps, and the linguistic form of prototype-based explanation that IDSS already had.

- The IMAFD architecture including IDSS+ is validated on the change detection dataset RaVAEn (Růžička et al., 2022) and the semantic segmentation dataset WorldFloods (Mateo-Garcia et al., 2021), respectively. The proposed methods show promising performance in terms of numerical results and interpretability.

1.3.4 Interpretable ensemble geoscience foundation model for flood mapping

- I assess the ability of several geoscience foundation models to generalize across different bands.
- I propose a plug-and-play prototype-based module IDSS v2 that explains the decision-making process of the model using representative training samples. It enables human users to audit and understand the behaviour of the model.
- I propose an interpretable ensemble foundation model for flood mapping using earth observation data.
- I evaluate the proposed module and the interpretable ensemble foundation model on the WorldFloods v2 dataset, achieving state-of-the-art performance.

1.4 Thesis Outline

The structure of this thesis is as follows: Chapter 1 is an introduction, outlining the dissertation's motivation and objectives of the dissertation. Chapter 2 provides background and related work, mainly introducing the relevant methodologies and literature. Chapter 3 presents an Interpretable-by-design DEep learning ALgorithm(IDEAL) for classification and continual learning tasks. Chapter 4 introduces the Deep Interpretable Semantic Segmentation (IDSS) method. A novel method tailored to flood mapping has been proposed and developed. Chapter 5 presents an Interpretable Multi-stage Approach to Flood Detection from Time Series Multispectral Data (IMAFD). Chapter 6 introduces the proposed Interpretable Ensemble Geoscience Foundation Model (IEGF) for Flood Mapping. Finally, the conclusion and future work are presented in Chapter 7.

Chapter 2

Background and Related Work

This section describes the key concept included in this thesis, as well as the related work and the shortcomings of the existing work. Specifically, the first section introduces some basic machine learning algorithms relevant to this thesis. The second section introduces some deep learning algorithms, including the Convolutional Neural Network (CNN), the Transformer, and the foundation models. The third section introduces the image segmentation task, which is the main task of this thesis. The fourth section introduces the remote sensing concept related to the thesis. The fifth section introduces the deep learning applications in the Earth Observation tasks. Finally, the sixth section introduces Explainable AI, which mainly includes the explainable AI taxonomies and their applications in Earth Observation.

2.1 Machine learning

Machine learning technology has grown exponentially over the past 20 years, revolutionizing the field of computer science. Unlike traditional computer methods, which write a set of rules to fulfill a specific need, machine learning techniques make predictions or decisions about unknown data by automatically learning patterns from existing data (Murphy, 2012). There are three main types of machine learning approaches, supervised learning, unsupervised learning and reinforcement learning (Jordan and Mitchell, 2015). This section will briefly introduce some traditional machine learning algorithms related to this thesis.

2.1.1 K-means and its variants

K-means. K-means algorithm (MacQueen, 1967; Lloyd, 1982) is one of the most well-known unsupervised learning algorithms. The basic concept is to divide N data points in I dimensions into K clusters. It aims to minimize the with-cluster sum of squares, where each cluster is represented by a vector m_k, known as the cluster mean (MacKay, 2003). Specifically, given a set of data points x^n where $n = 1, 2, 3, \ldots, N$. Each vector x consists of I components x_i. Assuming that x lives in a real space, then a matrix can be defined to measure the distance between two data points, such as the Euclidean distance (MacKay, 2003),

$$d(x, y) = \frac{1}{2} \sum_i (x_i - y_i)^2 \tag{2.1}$$

Then, the K-means algorithm can be defined as Algorithm 1 (Sammut and Webb, 2017):

Algorithm 1: K-means

```
Input: A set of data points $\{\boldsymbol{x}^n\}_{n=1}^N$, number of clusters $K$
Output: A set of $K$ clusters with corresponding means $\{\boldsymbol{m}^k\}_{k=1}^K$
Initialize cluster means $\boldsymbol{m}^k$ to random values;
while convergence criteria are not satisfied do
    for each data point $\boldsymbol{x}^i$ do
        Find the nearest center and assign $\boldsymbol{x}^i$ to the corresponding cluster;
    end
    Update each cluster mean $\boldsymbol{m}^k$ by calculating the mean of all points
      assigned to it;
end
return final clusters;
```

K-medoids. K-medoids (Shewhart et al., 1986) is very similar to the K-means. The main difference is that it uses the real data point as the centre rather than the mean value. This makes it much more robust to noisy data. In such a case, it could provide a better explanation for decision making than K-means. It can be defined as Algorithm 2 (Shewhart et al., 1986):

Algorithm 2: K-medoids

```
Input: A set of data points $\{\boldsymbol{x}^n\}_{n=1}^{N}$, number of clusters $K$
Output: A set of $K$ clusters with corresponding medoids $\{\boldsymbol{m}^k\}_{k=1}^{K}$
Initialize medoids $\boldsymbol{m}^k$ to random data points;
while convergence criteria are not satisfied do
    for each data point $\boldsymbol{x}^i$ do
        Find the nearest medoid and assign $\boldsymbol{x}^i$ to the corresponding cluster;
    end
    Update each cluster medoid $\boldsymbol{m}^k$ by selecting the point that minimizes
      the total distance within the cluster;
end
return final clusters;
```

2.1.2 Support Vector Machines

Support Vector Machines (SVM) (Cortes and Vapnik, 1995) is one of the most famous algorithms for supervised learning. The key concept of SVM is to map the input vector to some high-dimensional feature space and then construct optimal hyperplanes to maximize the margin between positive and negative examples (Cortes and Vapnik, 1995). Let's start with a set of training data (Cortes and Vapnik, 1995):

$$(y_1, \mathbf{x}_1), \ldots, (y_\ell, \mathbf{x}_\ell), \quad y_i \in \{-1, 1\}. \tag{2.2}$$

Where x are the features of the training data and y are the labels. The training data is said to be linearly separable if there exists a hyperplane that perfectly separates such data points of two classes. For such a binary classification task, the optimal hyperplanes can be defined as a linear function (Cortes and Vapnik, 1995): ,

$$w^\top x + b_0 = 0 \tag{2.3}$$

Where x is the training samples, w is a vector that is perpendicular to the hyperplane and b_0 is the bias term. Linearly separability requires that all training data satisfy (Cortes and Vapnik, 1995):

$$\begin{aligned} \mathbf{w}\cdot\mathbf{x}_i + b &\geq 1, \qquad \text{if } y_i = 1,\\ \mathbf{w}\cdot\mathbf{x}_i + b &\leq -1, \qquad \text{if } y_i = -1. \end{aligned} \tag{2.4}$$

Both inequalities can be combined as (Cortes and Vapnik, 1995):

$$y_i\big(\mathbf{w}\cdot\mathbf{x}_i + b\big) \;\geq\; 1, \quad i = 1,\dots,\ell. \tag{2.5}$$

Since the goal is to maximize the distance between positive examples and negative examples, this distance $\rho(\mathbf{w}, b)$ can be given by (Cortes and Vapnik, 1995):

$$\rho(w,b) = \min_{x:y=1}\frac{x\cdot w}{\|w\|} \;-\; \max_{x:y=-1}\frac{x\cdot w}{\|w\|}. \tag{2.6}$$

Then, it can be deducted from 2.4 and 2.6 that (Cortes and Vapnik, 1995):

$$\rho(w_0,b_0) = \frac{2}{\|\mathbf{w}_0\|} = \frac{2}{\sqrt{\mathbf{w}_0\cdot\mathbf{w}_0}}. \tag{2.7}$$

To find the optimal hyperplane that maximizes the distance ρ, what needs to be done is to minimize the $\mathbf{w}\cdot\mathbf{w}$ under the constraint 2.5. The problem has thus been transformed into a quadratic programming problem.

A key innovation of SVM is the use of kernel tricks to enable non-linear classification. For example, the linear function used by the support vector machine can be rewritten as (Goodfellow et al., 2016):

$$w^\top x + b_0 \;=\; b_0 \;+\; \sum_{i=1}^{m}\alpha_i\, x^\top x^{(i)}. \tag{2.8}$$

Where $x_{(i)}$ is the training example, known as the support vector and α is a coefficient vector. In such a case, x can be replaced as the dot product of a feature function $\phi(x)$ with a kernel function $k(x, x^{(i)}) = \phi(x) \cdot \phi\big(x^{(i)}\big)$. Then, the prediction function can be formulated as (Goodfellow et al., 2016):

$$f(x) = b + \sum_i \alpha_i \, k\big(x, x^{(i)}\big). \tag{2.9}$$

Such a kernel-based function can be thought of as preprocessing the input data with $\phi(x)$ and then learning a linear model in the new transformed feature space (Goodfellow et al., 2016).

2.1.3 Decision Trees

Decision Trees is a supervised learning algorithm that can be easily understood by human users. It is defined as recursively partitioning the input space and fitting a simple prediction model within each partition (Loh, 2011). Graphically, such a partition can be represented as a decision tree. There are two different types of decision trees, depending on the nature of the input data. Regression trees are designed for continuous or ordered discrete input values, while classification trees are designed for a finite number or unordered input values (Loh, 2011).

For example, Figure 2.1 shows a regression tree. The tree has two inputs x_1 and x_2, at the first node, it asks if the input satisfies the given situation where x_1 is greater than t_1, if so, it goes through the right node and continues asking if x_2 is greater than t_3, if not, it ends with the left node R_1, which can be defined as (Loh, 2011):

$$R_1 = \{x : x_1 > t_1, x_2 \leq t_3\} \tag{2.10}$$

On the other hand, Figure 2.2 shows a classification tree about whether a customer will buy a laptop in an electronics store.

Compared to other traditional machine learning algorithms, decision trees are highly interpretable to human users. It can show the decision-making process to the users in a very intuitive and direct way by going through each of the conditional

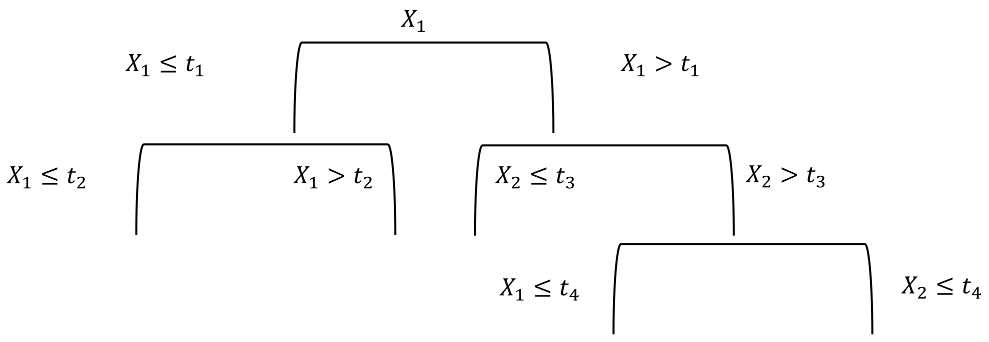


Figure 2.1: A regression trees with two inputs (Murphy, 2022).

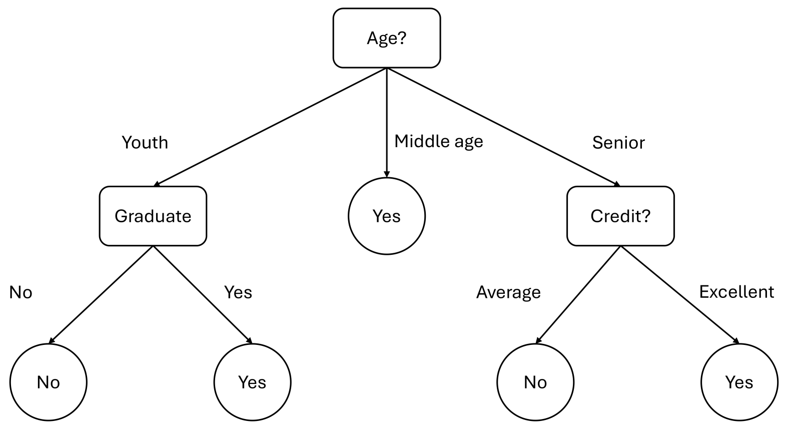


Figure 2.2: A classification tree for whether a customer buys a laptop in an electronics store (Gupta et al., 2017).

nodes in the tree. However, it can be difficult if the tree has a very large number of nodes.

2.1.4 K nearest neighbor (KNN)

K Nearest Neighbour (KNN) (Cover and Hart, 1967) is a famous prototype-based classification method. Given an input x, the goal is to find the K closest examples to x in the training set D, represented as $N_k(x, D)$. By analyzing their labels, it is possible to infer a distribution that helps to classify x based on its local neighborhood (Murphy, 2022). Specifically, this can be expressed as:

$$p(y = c|x, \mathcal{D}) = \frac{1}{K} \sum_{n \in N_K(x,\mathcal{D})} \mathbb{I}(y_n = c) \tag{2.11}$$

Where $\mathbb{I}$ is the indicator function, it equals 1 if the label y_n is equal to the target class c, and 0 otherwise.

Then, the majority of labels are returned.

An important parameter within the model is the distance metric, this thesis mainly uses the Euclidean distance:

$$d(x, \mu) = \|x - \mu\| \tag{2.12}$$

One limitation of KNN is that it needs to store all the training data, which requires a lot of memory and is computationally expensive.

2.1.5 Recursive Density Estimation (RDE)

Recursive Density Estimation (P. Angelov, 2012) is introduced to provide an online data density estimation method that is memory and computationally efficient. The recursive expression is as follows (P. Angelov, Sadeghi-Tehran, and Ramezani, 2011):

$$D(x_k) = \frac{1}{1 + \|x_k - \mu_k\| + \sum_k - \|\mu_k\|} \tag{2.13}$$

Where $D(x_k)$ is the data density of the current data sample x_k, the mean value μ_k and the scalar product $\sum_k$ can be recursively updated as follows (P. Angelov, Sadeghi-Tehran, and Ramezani, 2011):

$$\mu_k = \frac{k-1}{k}\mu_{k-1} + \frac{1}{k}x_k \quad \mu_1 = x_1 \tag{2.14}$$

$$\Sigma_k = \frac{k-1}{k}\Sigma_{k-1} + \frac{1}{k}\|x_k\|^2 \quad \Sigma_1 = \|x_1\|^2 \tag{2.15}$$

2.1.6 Fuzzy rules

Fuzzy rules can be formulated as linguistic expressions in the following way (P. Angelov, 2012):

$$\begin{aligned} \text{Rule}_i : \quad & \text{IF (antecedent)} \\ & \text{THEN (consequent)} \quad i = [1, R] \end{aligned}$$

Here, R represents the total number of fuzzy rules, antecedent depends on the type of fuzzy rules, some use linguistic terms while others use functional or mathematical expressions (P. Angelov, 2012).

IF... THEN rules rely on the linguistic representation that enables interpretable decision-making processes, making them a common choice in interpretable models. For example, in a deep rule-based (DRB) classifier, the decision-making process can be formulated as (P. P. Angelov and Xiaowei Gu, 2019):

$$\begin{aligned} R_i : \quad & \text{IF } \big(I \sim P_{i,1}\big) \text{ OR } \big(I \sim P_{i,2}\big) \text{ OR } \ldots \text{ OR } \big(I \sim P_{i,M_i}\big) \\ & \text{THEN (Class } i\text{)} \end{aligned} \tag{2.16}$$

Where $i = 1, 2, \ldots, C$ is the class label; $P_{i,j}$ denotes the jth prototype images of the ith class, $j = 1, 2, \ldots, M_i$, where M_i is the number of prototypes defined by the fuzzy rule (P. P. Angelov and Xiaowei Gu, 2019).

Such rules can also be represented visually. For example, for the Caltech 101 dataset, the IF...THEN rules of the DRB classifier can be shown in Figure 2.3.

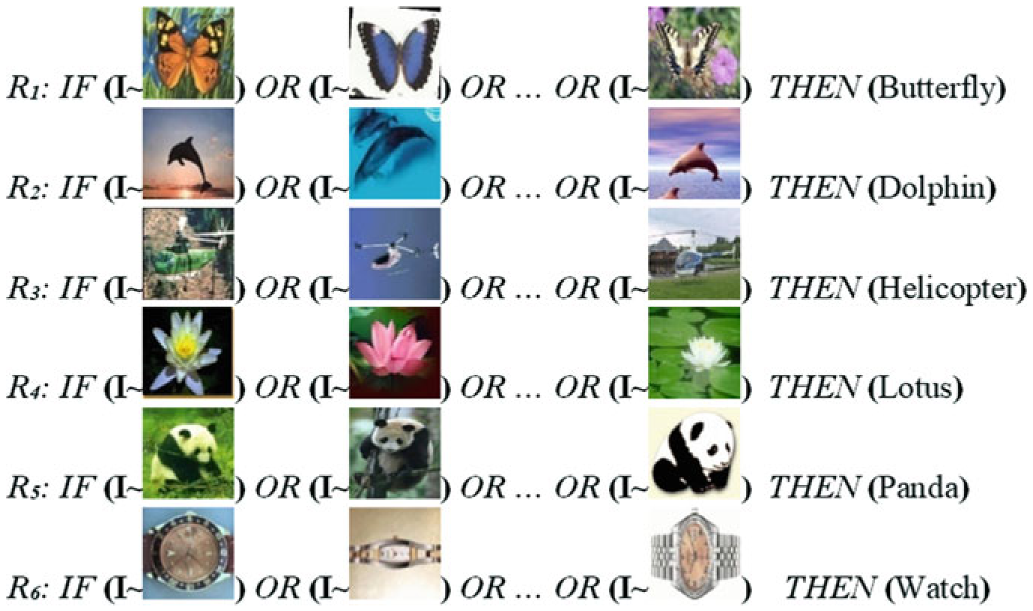

R₁: IF (I~) OR (I~) OR ... OR (I~) THEN (Butterfly)
R₂: IF (I~) OR (I~) OR ... OR (I~) THEN (Dolphin)
R₃: IF (I~) OR (I~) OR ... OR (I~) THEN (Helicopter)
R₄: IF (I~) OR (I~) OR ... OR (I~) THEN (Lotus)
R₅: IF (I~) OR (I~) OR ... OR (I~) THEN (Panda)
R₆: IF (I~) OR (I~) OR ... OR (I~) THEN (Watch)

Figure 2.3: IF...THEN rules of the Caltech 101 dataset (P. P. Angelov and Xiaowei Gu, 2019).

2.2 Deep Learning

Deep learning is an approach to machine learning. Thanks to the growth of large amounts of training data and the improvement of computing infrastructure, deep learning has been applied to almost every industry (Bengio, Goodfellow, and Courville, 2017). It is a very broad topic, and this section will only discuss some of the most fundamental, as well as some concepts and algorithms related to the work of this thesis.

2.2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) (LeCun et al., 1989) is the most fundamental concept within the field of deep learning. As its name suggests, it is mainly based on a mathematical operation called convolution.

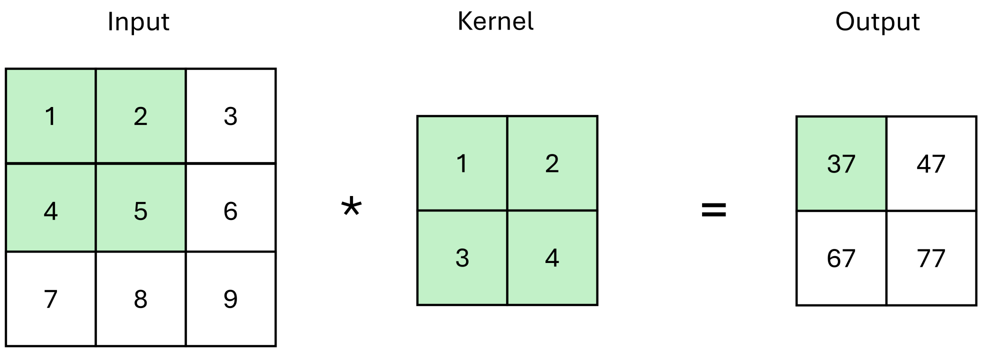


Figure 2.4: 2d convolution.

2.2.1.1 Convolution

The convolution operation in two-dimensional space can be defined as follows:

$$[\mathbf{W} \circledast \mathbf{X}](i, j) = \sum_{u=0}^{H-1} \sum_{v=0}^{W-1} w_{u,v}\, x_{i+u,j+v} \tag{2.17}$$

Here, the 2d filter $\mathbf{W}$ has size $H \times W$ and i, j indicate the respective dimensions of the output matrix (Murphy, 2022). For example, applying a 2×2 kernel to a 3×3 input $\mathbf{X}$ and produces a 2×2 output $\mathbf{Y}$ (Murphy, 2022):

$$\mathbf{Y} = \begin{pmatrix} w_1 & w_2 \\ w_3 & w_4 \end{pmatrix} \circledast \begin{pmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{pmatrix} \tag{2.18}$$

$$= \begin{pmatrix} (w_1x_1 + w_2x_2 + w_3x_3 + w_4x_5) & (w_1x_2 + w_2x_3 + w_3x_5 + w_4x_6) \\ (w_1x_4 + w_2x_5 + w_3x_7 + w_4x_8) & (w_1x_5 + w_2x_6 + w_3x_8 + w_4x9) \end{pmatrix} \tag{2.19}$$

The visualization process can also be found in Figure 2.4.

Convolution has three main advantages to improve performance over traditional machine learning algorithms, which are sparse interactions, parameter sharing and equivariant representations (Bengio, Goodfellow, and Courville, 2017).

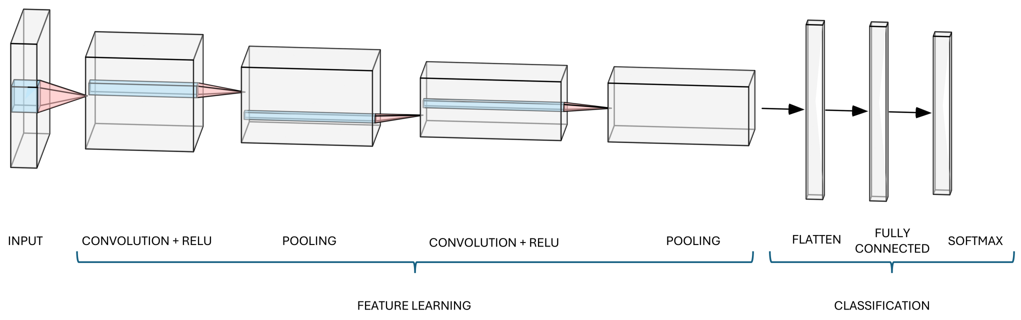


Figure 2.5: A CNN for image classification.

2.2.1.2 Pooling

Typically, there are three layers for a convolutional neural network: convolution, activation function and pooling (Bengio, Goodfellow, and Courville, 2017). A pooling layer mainly replaces the results from the above layer by summarising the statistics of the neighbouring outputs (Bengio, Goodfellow, and Courville, 2017).

There are several pooling techniques. For example, the max pooling returns the maximum value from the input while the average pooling returns the mean value (Murphy, 2022).

Figure 2.5 shows a very general CNN architecture. The input image is first passed through a series of convolution and pooling layers to perform the feature extraction, followed by a linear classification head to obtain the final prediction.

2.2.2 Transformer

The CNN architecture has been a standard paradigm in the field of Computer Vision (CV) since it was proposed. However, the rise of Transformer architecture (Vaswani, 2017) has completely changed the dominance of CNN. It was initially proposed for Natural language processing (NLP) tasks and was gradually adapted to other domains, such as computer vision (Dosovitskiy et al., 2020) and time-series analysis (H. Zhou et al., 2021).

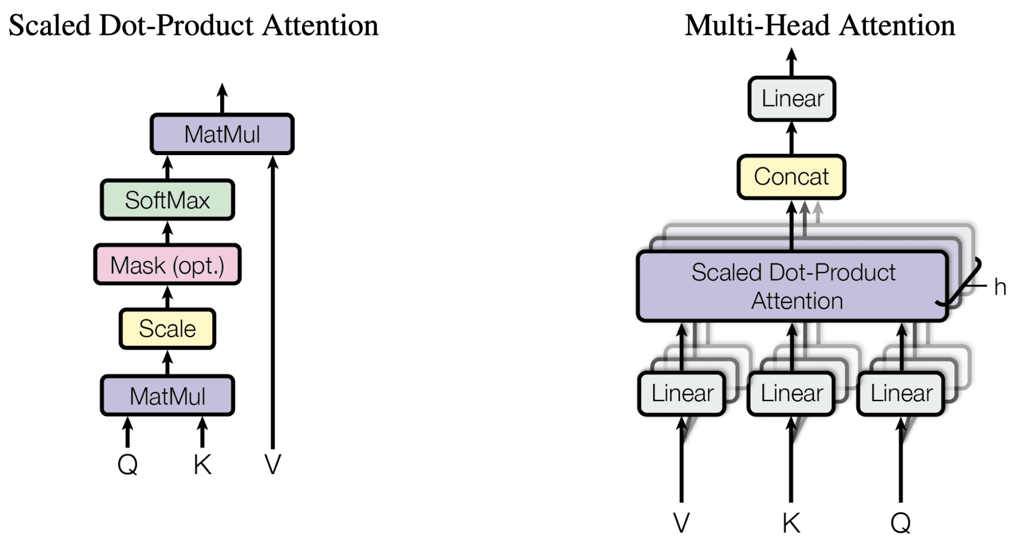


Figure 2.6: Scaled Dot-product Attention and Multi-Head Attention (Vaswani, 2017).

2.2.2.1 Attention mechanism

The attention mechanism is one of the core concepts of the Transformer architecture. The output features are computed as a weighted sum of the queries, keys and values. The main purpose of this operation is to enable the network to dynamically learn the feature representation to which it should pay more attention (Vaswani, 2017).

The one used in Transformer is called "Scaled Dot-Product Attention" as shown in Figure 2.6. The scaled dot-product attention formula is given as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{2.20}$$

Where Q, K and V represent the query, key, and value matrices, T is the sequence length and d_k, and d_v are the dimensionality of the keys and values. The dot products of query and keys were divided by $\sqrt{d_k}$, then the softmax function was applied to get the weights of the values. Here, $\sqrt{d_k}$ is a scaling factor, which is used to avoid pushing the softmax function into regions with extremely small gradients.

Instead of using only single attention functions, a multi-head attention mechanism can learn richer feature representations and take advantage of the parallel computing capability of GPUs. The formula is shown as (Vaswani, 2017):

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.21}$$

$$\text{where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \tag{2.22}$$

Where $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$

It can be observed that multi-head attention treats the input as a set of elements or tokens rather than sequences. It is therefore unable to learn information about the location of each token. Thus, “positional encodings” are introduced to mark and learn the positional information of the inputs (Vaswani, 2017).

2.2.2.2 Transformer architecture

The transformer has an encoder-decoder structure shown in Figure 2.7. The encoder consists of N identical blocks. First, positional encoding is added to the input embeddings, which then pass through a multi-head attention function. The output is combined with the original input using residual connections, followed by layer normalization. The result is then passed through a feed-forward network, where residual connection and layer normalization are applied again.

Similar to the encoder, the decoder also consists of N identical blocks, the main difference being that a new masked multi-head attention layer is added to the decoder. This is to prevent the model from cheating by observing the future token when making predictions about the current token.

2.2.2.3 Vision Transformer

Inspired by the success of Transformer in the application of NLP, Vision Transformer (ViT) (Dosovitskiy et al., 2020) was proposed to be applied to the image classification tasks.

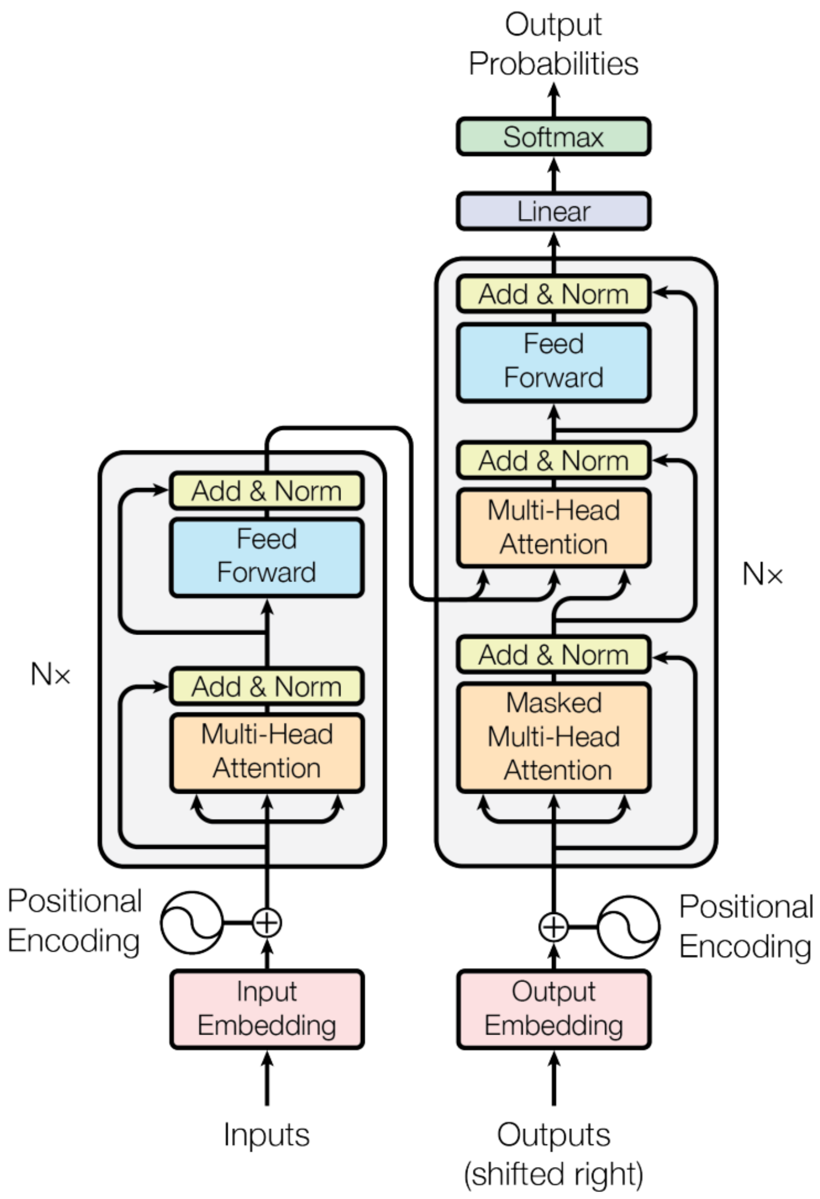


Figure 2.7: Transformer architecture (Vaswani, 2017).

The model architecture is shown in Figure 2.8. One challenge in applying the transformer to the image field is how to generate tokens. ViT shows a great example by splitting the image into patch tokens. The output is then linearly embedded and summed with positional embeddings and class embeddings (Eq. 2.23), and finally fed into a Transformer encoder. The Transformer encoder is very similar to the one proposed in (Vaswani, 2017), with only minor differences, where layer normalization is applied before the multi-head attention operation. Within the Transformer block, the embedded patches are first fed into a multi-head attention (Eq. 2.24), and then pass through a MultiLayer Perceptron (MLP) layer. After L such Transformer operations, the class token of the final Transformer block output Z_L^0 is considered as the final latent feature for the classification task.

$$\mathbf{z}_0 = \left[\mathbf{x}_{\text{class}}; \mathbf{x}_p^1\mathbf{E}; \mathbf{x}_p^2\mathbf{E}; \ldots; \mathbf{x}_p^N\mathbf{E}\right] + \mathbf{E}_{\text{pos}}, \quad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C)\times D}, \mathbf{E}_{\text{pos}} \in \mathbb{R}^{(N+1)\times D} \tag{2.23}$$

$$\mathbf{z}'_\ell = \text{MSA}(\text{LN}(\mathbf{z}_{\ell-1})) + \mathbf{z}_{\ell-1}, \quad \ell = 1 \ldots L \tag{2.24}$$

$$\mathbf{z}_\ell = \text{MLP}(\text{LN}(\mathbf{z}'_\ell)) + \mathbf{z}'_\ell, \quad \ell = 1 \ldots L \tag{2.25}$$

$$\mathbf{y} = \text{LN}(\mathbf{z}_L^0) \tag{2.26}$$

Where $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$ is the input images, $\mathbf{x}_p \in \mathbb{R}^{N\times(P^2 \cdot C)}$ is a sequence of flattened 2D patches, (H, W) is the resolution of the input image and C is the number of channels, the resolution of the patches is (P, P), and y is the output of the Transformer Encoder.

The main difference between CNNs and Transformers is that Transformers do not include as much inductive bias for images as CNNs, which makes them especially useful when dealing with large datasets (Dosovitskiy et al., 2020). Besides, the proposal of the ViT model has further advanced the development of multimodal

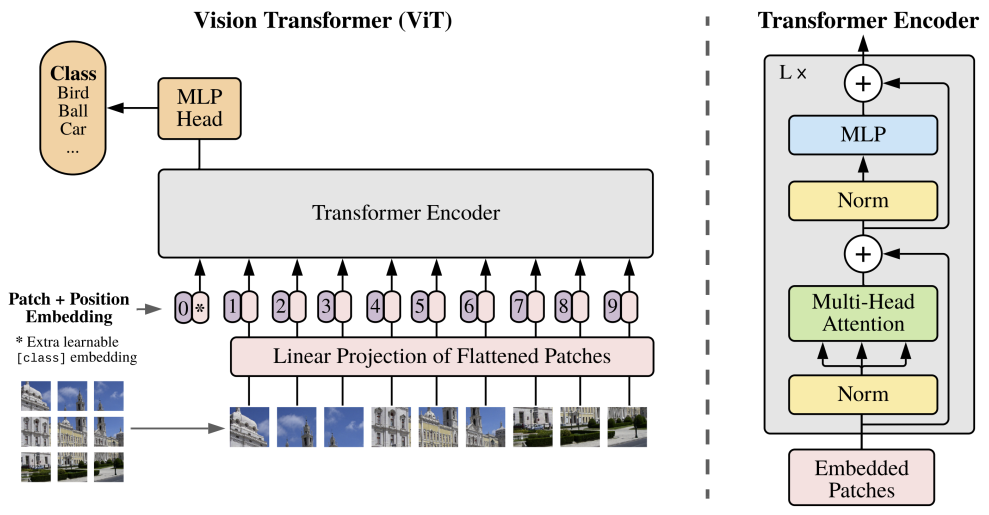


Figure 2.8: Vision Transformer (ViT) architecture (Dosovitskiy et al., 2020).

models thanks to the versatility of the Transformer model to handle diverse tasks across different domains.

2.2.3 Foundation models

In recent years, with the availability of large amounts of data and the advancement in computing infrastructure, a new paradigm has emerged known as the foundation model. Unlike traditional deep learning models, which are trained on specific datasets for particular tasks, foundation models are trained on broad data that can be adapted to a wide range of downstream tasks (Bommasani et al., 2021). Notable examples include CLIP(Radford et al., 2021), GPT-3(Brown et al., 2020), DINOv2(Oquab et al., 2023) and most recent DeepSeek-R1 (D. Guo et al., 2025).

The same paradigm has been applied to the field of remote sensing, especially Earth Observation (EO). Examples of such models are SatlasNet (Bastani et al., 2023) and Prithvi (Jakubik et al., 2023), which are described in detail below.

2.2.3.1 SatlasNet

SatlasNet (Bastani et al., 2023) is a foundation model designed for EO tasks. Unlike traditional models that are trained on individual label types, SatlasNet is jointly

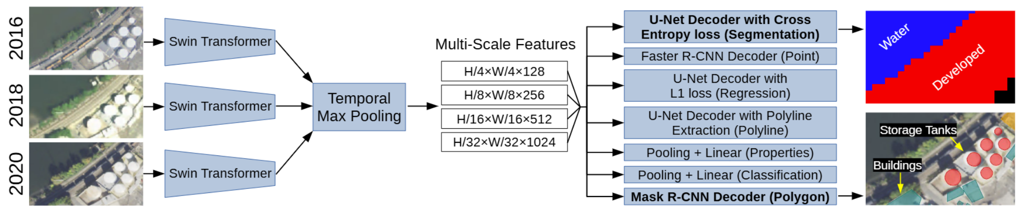

2016
2018
2020
Swin Transformer
Swin Transformer
Swin Transformer
Temporal Max Pooling
Multi-Scale Features
H/4×W/4×128
H/8×W/8×256
H/16×W/16×512
H/32×W/32×1024
U-Net Decoder with Cross Entropy loss (Segmentation)
Faster R-CNN Decoder (Point)
U-Net Decoder with L1 loss (Regression)
U-Net Decoder with Polyline Extraction (Polyline)
Pooling + Linear (Properties)
Pooling + Linear (Classification)
Mask R-CNN Decoder (Polygon)
Water
Developed
Storage Tanks
Buildings

Figure 2.9: SatlasNet architecture (Bastani et al., 2023).

trained on seven label types including 137 categories. It also uses the supervised training method rather than most other foundation models that use self-supervised training methods. The model architecture is shown in Figure 2.9.

It can be observed that SatlasNet uses three Swin Transformers to extract multi-scale features from time-series images. The features are then aggregated through temporal max pooling and then fed into seven different heads corresponding to different tasks, such as semantic segmentation, point detection, regression, polyline detection, property prediction, classification and polygon detection.

2.2.3.2 Prithvi

Prithvi (Jakubik et al., 2023) is a transformer-based geospatial foundation model designed for EO tasks. The model architecture is shown in Figure 2.10. The model is pretrained with the masked autoencoder (MAE), which is a commonly used self-supervised approach for most of the vision foundation models. The input images are first randomly masked, and only the unmasked image is fed into the encoder. Then a decoder is employed to receive the latent features from the encoder and to reconstruct the masked patches. To adapt multi-spectral remote sensing images to three-channel MAE, Prithvi also introduced 3D patch embedding and 3D position encoding.

2.2.4 Continual learning

Continual learning models solve a number of related problems (Ven, Tuytelaars, and Tolias, 2022). *Task-incremental learning* addresses the problem of incrementally

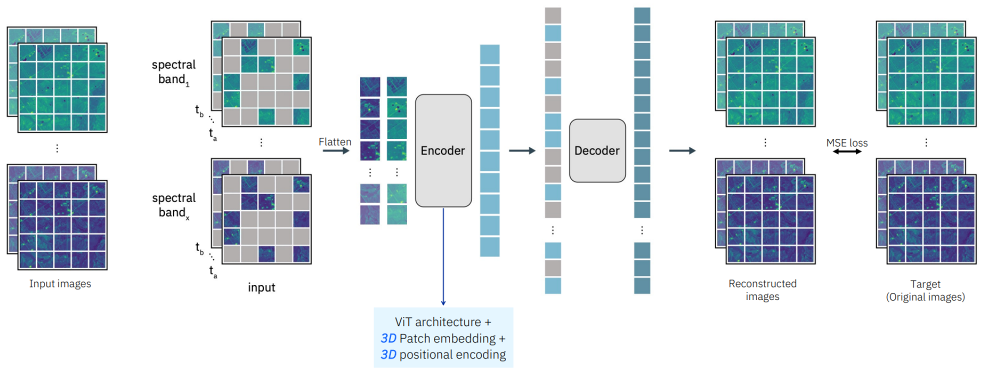


Figure 2.10: Prithvi architecture (Jakubik et al., 2023).

learning known tasks, with the intended task explicitly input into the algorithm (Ruvolo and Eaton, 2013; Z. Li and Hoiem, 2017; Kirkpatrick et al., 2017). *Domain-incremental learning* (Yabin Wang, Z. Huang, and Hong, 2022; Lamers et al., 2023) addresses the problem of learning when the domain is changing and the algorithm is not informed about these changes. This includes such issues as *concept drift* when the input data distribution is non-stationary (Widmer and Kubat, 1996). *Class-incremental learning* (S. Yan, J. Xie, and X. He, 2021; Z. Wang et al., 2022) is a problem of ever expanding number of classes of data. This thesis will only focus on this last problem. However, one can see how the prototype-based approaches could help solve the other two problems by circumventing catastrophic forgetting (French, 1999) through incremental update of the prototypes (Baruah and P. Angelov, 2012).

2.3 Image segmentation

Image segmentation has always been one of the most important tasks in the field of Computer Vision (CV), and it has been applied to diverse fields and industries, such as self-driving cars (Q. Zhou et al., 2020), medical image analysis (R. Wang et al., 2022), and remote sensing image analysis (Yuan, J. Shi, and L. Gu, 2021).

Image segmentation divides images into different segments and extracts the region of interest (ROIs) (Yu et al., 2023). It can be divided into 3 main types:

semantic segmentation, which assigns a semantic label to each pixel; instance segmentation, which assigns an instance-specific label to each pixel; and panoptic segmentation, which combines both approaches by assigning each pixel a semantic label and an object label (Minaee et al., 2021).

Image segmentation can be classified into traditional methods and deep learning-based methods. Traditional methods are usually based on hand-crafted features and often require domain experts to manually set up the feature extraction process. Such methods include the threshold-based Otsu method (Otsu et al., 1975), the clustering-based K-means algorithm (Lloyd, 1982; Sathya and Manavalan, 2011), and the simple linear iterative clustering (SLIC) method (Achanta et al., 2012). Since the K-means algorithm has been described in detail in the previous sections, the SLIC algorithm is described in detail below.

2.3.1 SLIC

SLIC (Simple Linear Iterative Clustering) is one of the superpixel algorithms that group pixels into meaningful regions instead of pixels to reduce the complexity of further processing of images (Achanta et al., 2012). It performs clustering in a 5D space defined by the 3D CIELAB colour space and the 2D x, y pixel coordinates.

It is worth noting that the distance metric used in the SLIC method should be chosen carefully because the CIELAB colour space and the pixel coordinate space are not consistent, the range of the pixel space will change according to the image size, while the CIELAB colour space is fixed. In such a case, simply applying the Euclidean distance will lead to different weighting of the colour and coordinate spaces according to the image size. Therefore, SLIC proposed a new distance matric D_s, shown as the Equation 2.27 (Achanta et al., 2012):

$$\begin{aligned} d_{lab} &= \sqrt{(l_k - l_i)^2 + (a_k - a_i)^2 + (b_k - b_i)^2}, \\ d_{xy} &= \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}, \\ D_s &= d_{lab} + \frac{m}{S} d_{xy}. \end{aligned} \tag{2.27}$$

The method can be summarized by Algorithm 3 (Achanta et al., 2012).

Algorithm 3: SLIC

```
Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at
  regular grid steps $S$;
Perturb cluster center within an $n \times n$ neighborhood to the lowest gradient
  position;
repeat
    for each cluster center $C_k$ do
        Assign the best-matching pixels within a $2S \times 2S$ square
          neighborhood around the cluster centre based on the distance
          metric (Eq. 2.27);
    end
    Recompute new cluster center and residual error $E$ (L1 distance
      between old and new centers);
until $E \leq$ threshold;
Enforce connectivity constraints ;
```

2.3.2 U-Net

The semantic Segmentation problem is very common in the remote sensing field. The target is to assign a class label $y_i \in 1, \ldots, C$ for each pixel, where the class might be tree, building, road, water, crop, etc. A very common architecture for semantic segmentation tasks is the encoder-decoder architecture. The Encoder typically use a convolutional neural network or Transformer to extract hierarchical features by progressively downsampling the input, while the decoder upsamples and fuses such features to obtain the final pixel-wise prediction (Zheng et al., 2021).

U-Net (Ronneberger, Fischer, and Brox, 2015) is a semantic segmentation network with an encoder-decoder architecture. It is widely used as it was introduced earlier and offers good performance. The architecture is shown in Figure 2.11. It can be observed that the input images were mapped to a 2D bottleneck with the convolution operation, then it was mapped back to a full-size output segmentation

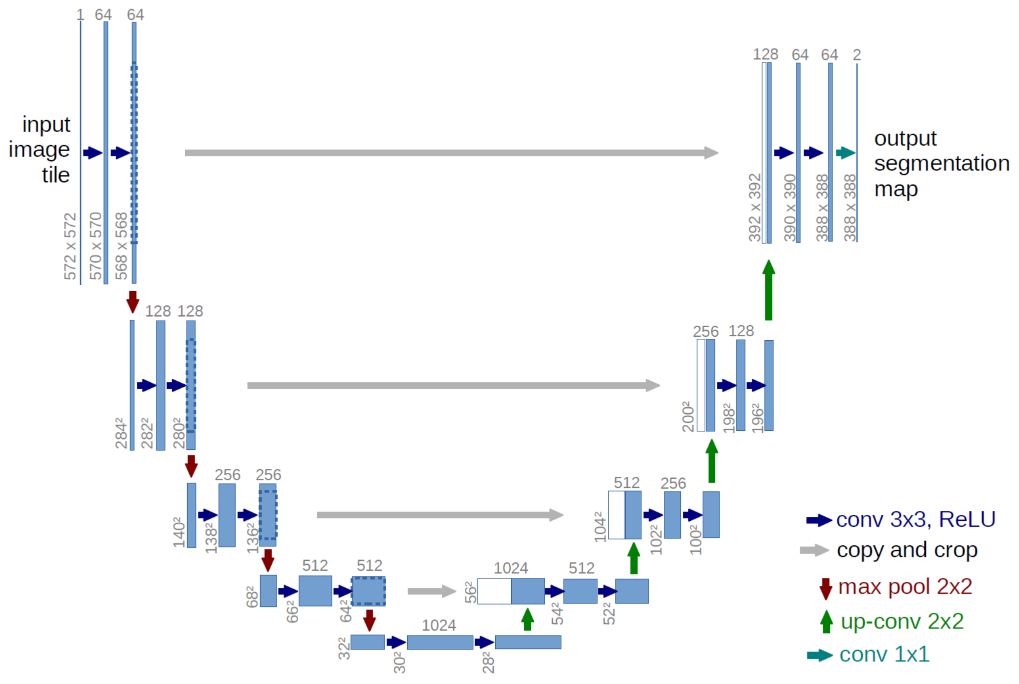


Figure 2.11: U-Net (Ronneberger, Fischer, and Brox, 2015).

map. A key feature of the U-Net is that it uses skip connections from the input to the output layer to preserve the information lost during the convolution process.

2.3.3 DeepLabv3+

DeepLabv3+ (L.-C. Chen et al., 2018) is another widely used semantic segmentation algorithm. The model architecture is shown in Figure 2.12.

DeepLabv3+ also employs the encoder-decoder architecture. Unlike the U-Net, it uses atrous convolution to control the resolution of the feature to obtain multi-scale information.

2.4 Remote sensing

Remote sensing (RS) technologies measure the properties of objects on the Earth's surface using data collected by aircraft or satellites (Schowengerdt, 2006). In such a concept, the measurement is made at a distance, which means that there is no

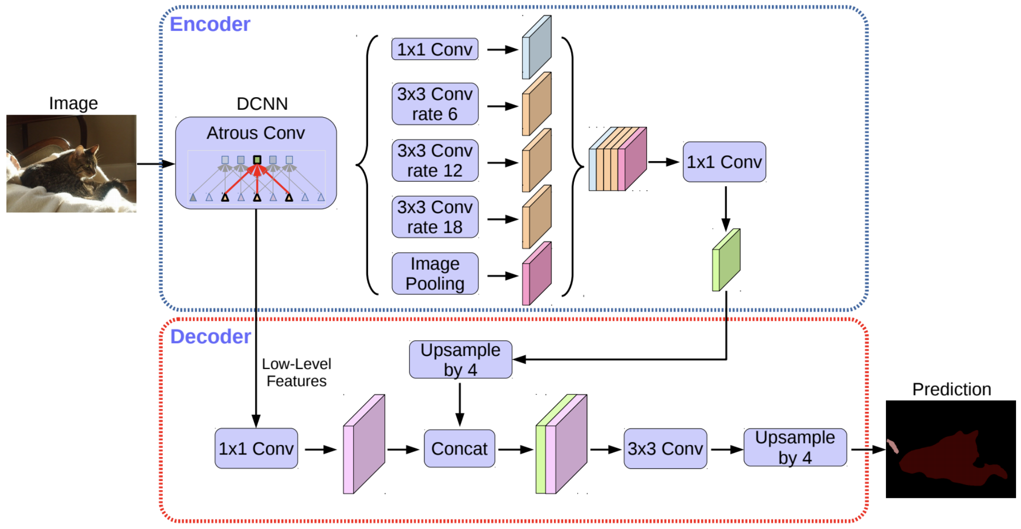


Figure 2.12: DeepLabv3+ (L.-C. Chen et al., 2018).

direct contact with the object. Therefore, people much rely on a series of propagated signals, such as optical, radio, or microwaves (Schowengerdt, 2006).

Landsat 1 was the first Earth Observation satellite, which was launched in July 1972 and specifically used to monitor and collect land images (D. L. Williams, Goward, and Arvidson, 2006; Schowengerdt, 2006). Landsat 1 uses a multispectral scanner (MSS), which provides four spectral bands (green, red, and two near-infrared bands) with 80 m resolution (D. L. Williams, Goward, and Arvidson, 2006). To date, with almost half a century of technological development, remote sensing technology has been revolutionized with a wide range of infrastructure, resolution and applications. It has now been widely used in diverse areas, such as disaster management (Shastry et al., 2023; Rashkovetsky et al., 2021), crop monitoring (Omia et al., 2023), urban changes (Fayshal et al., 2024) and even mammal migration (Z. Wu et al., 2024).

Two main types of remote sensing imaging instruments have been widely used in the past decades. Their difference primarily depends on the type of energy source used to collect the data. One is the passive optical sensor, such as multispectral and hyperspectral sensors, and the other is the active remote sensor, for example, Real Aperture Radar (RAR) or Synthetic Aperture Radar (SAR) (Camps-Valls et al., 2011).

2.4.1 Optical images

Multispectral optical images capture data across specific wavelength ranges of the electromagnetic spectrum, which could acquire additional information that cannot be captured by the human eye except for red, green, and blue (Shafique et al., 2022). It takes advantage of the fact that different materials reflect and absorb differently in different wavebands, and has been widely used in recent years for Earth Observation tasks (Vali, Comai, and Matteucci, 2020). One of the most widely used satellites providing multispectral optical images is the Sentinel-2 satellite. It is equipped with multispectral imaging instruments (MSI) that enable the acquisition of 13 spectral bands, including the visible (red, blue and green), near infra-red (NIR) and short wave infra-red (SWIR) bands with different resolutions from 10 to 60m (Drusch et al., 2012). The Sentinel 2 mission consists of two satellites, Sentinel 2A and Sentinel 2B, providing 5 days of revisits. The description and resolution of Sentinel 2 are shown in the Table 2.1.

2.4.2 SAR images

Unlike the passive optical sensor used by Sentinel 2, SAR images acquired by a satellite actively transmit the microwave pulse to the Earth's surface and produce the images using the latency of the reflected signals (P. Singh et al., 2021). The main advantage of SAR is that it can take images in different atmospheric conditions, no matter day or night, because it uses wavelengths from 1cm to 1m while multispectral imaging instruments make use of wavelengths much closer to visible light, such as 1 micron. Because of these characteristics, SAR could see through the clouds, rain, fog and snow where optical sensors cannot (P. Singh et al., 2021).

In SAR imaging, as the satellite or aircraft moves, the SAR antenna transmits high-frequency radar waves towards the ground and collects the backscattered signals. The process is shown in Figure 2.13. These signals contain pulse information

Name	Description	Resolution
B01	Coastal aerosol, 442.7 nm (S2A), 442.3 nm (S2B)	60m
B02	Blue, 492.4 nm (S2A), 492.1 nm (S2B)	10m
B03	Green, 559.8 nm (S2A), 559.0 nm (S2B)	10m
B04	Red, 664.6 nm (S2A), 665.0 nm (S2B)	10m
B05	Vegetation red edge, 704.1 nm (S2A), 703.8 nm (S2B)	20m
B06	Vegetation red edge, 740.5 nm (S2A), 739.1 nm (S2B)	20m
B07	Vegetation red edge, 782.8 nm (S2A), 779.7 nm (S2B)	20m
B08	NIR, 832.8 nm (S2A), 833.0 nm (S2B)	10m
B8A	Narrow NIR, 864.7 nm (S2A), 864.0 nm (S2B)	20m
B09	Water vapour, 945.1 nm (S2A), 943.2 nm (S2B)	60m
B11	SWIR, 1613.7 nm (S2A), 1610.4 nm (S2B)	20m
B12	SWIR, 2202.4 nm (S2A), 2185.7 nm (S2B)	20m

Table 2.1: Spectral Bands of Sentinel-2 with their Descriptions and Resolutions (Drusch et al., 2012).

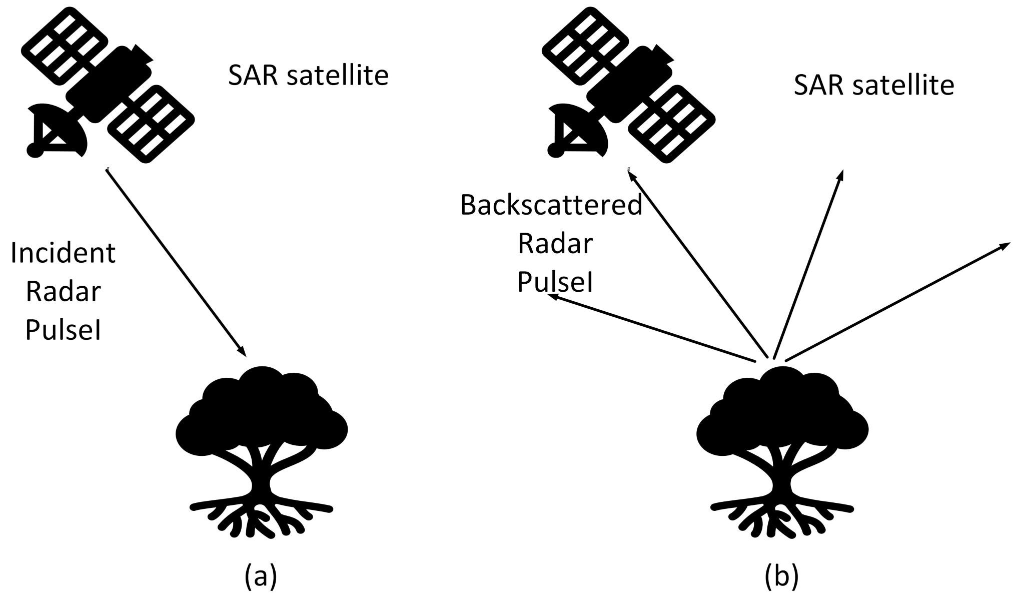


Figure 2.13: (a) A radar pulse is transmitted from the satellite to a tree (b) The radar pulse is backscattered by the tree back to the satellite (P. Singh et al., 2021).

with a distinct shape that allows the system to determine the exact location of the object and plot it on the image. Furthermore, SAR calculates the illumination of each point based on the intensity of the signal it receives. This is because objects absorb or reflect radar energy depending on their materials. For example, trees will appear grey because they absorb the radar energy, while metal will appear bright because of strong reflections (P. Singh et al., 2021).

Sentinel 1 is one of the most widely used SAR satellites. It consists of two satellites, Sentinel-1A and Sentinel-1B, it provides 6 days of revisit and four different types of data products including Interferometric Wide Swath (IW), Extra Wide Swath (EW), Stripmap (SM), and Wave (WV) (Torres et al., 2021).

2.4.3 Traditional flood detection methods

People have been fighting floods since time immemorial. In ancient mythology, Noah escaped the damage of flood by building Noah's Ark. More recently, with the advent of remote sensing technologies, People have been monitoring floods using satellite images to minimize the damage caused by floods. Flood mapping is an important part of flood detection, which can effectively help people to understand the specific area where the floods are occurring in order to take timely action. It can also help people with post-disaster reconstruction and urban planning.

Both optical and SAR images have been used for flood mapping. This thesis mainly focuses on flood mapping with optical images. Traditional flood detection methods mainly rely on rule-based methods, which distinguish between water bodies and other objects by analyzing the difference in absorption and reflection of water in different spectral bands (Bhaskaran et al., 2013). For example, water index-based methods are particularly popular, such as Normalized Difference Water Index (NDWI) (Gao, 1996), Modified Normalized Difference Water Index (MNDWI) (H. Xu, 2006), and Normalized Difference Vegetation Index (NDVI), etc.

One of the most well-known and widely used methods is the Normalized Difference Water Index (NDWI). It was proposed in (McFeeters, 1996) and uses near infra-red (NIR) and visible green light to delineate open water features in the optical images. The definition is as follows:

$$\text{NDWI} = \frac{(X_{\text{green}} - X_{\text{nir}})}{(X_{\text{green}} + X_{\text{nir}})} \tag{2.28}$$

The principle behind NDWI is to maximize the reflectance of water at green wavelengths and minimize the reflectance of water at NIR wavelengths. Meanwhile, it takes advantage of the high NIR reflectance of vegetation and soil features (McFeeters, 1996). The index value ranges from -1 to 1 and normally water bodies are greater than 0. However, NDWI is sensitive to build-up land noise (H. Xu, 2006). To address this challenge, Modification of Normalized Difference Water Index (MNDWI) (H. Xu, 2006) was proposed, the definition is shown as follows:

$$\text{MNDWI} = \frac{(X_{\text{green}} - X_{\text{mir}})}{(X_{\text{green}} + X_{\text{mir}})} \tag{2.29}$$

It can be seen that NIR is being replaced by Middle Infrared (MIR), which will help distinguish between built-up land and water. In this case, all build-up land, vegetation and soil will show a negative value while water bodies will show a positive value.

Although such algorithms are constantly evolving and iterating, the fact is that the performance of such methods is highly dependent on expert knowledge and constrained by various limitations. For example, NDWI may mix water and build-up land, while the optical threshold for MNDWI cannot remove the shadow noise in some regions. Most importantly, such methods require careful selection of the optimal threshold value, which must be adjusted according to different areas and times. All these drawbacks greatly hinder the application of such methods in global-scale flood mapping (Yan Zhou et al., 2017).

2.5 Deep Learning in Earth Observation

Recent years have witnessed rapid advancements in machine learning development, with a large number of algorithms being applied to EO. Among these, deep learning methodologies are particularly noteworthy due to their ability to achieve state-of-the-art results. Deep learning technologies have been used for several disaster management tasks, such as wildfire detection (Rashkovetsky et al., 2021), flood detection (Bonafilia et al., 2020), hurricane damage assessment (Berezina and D. Liu, 2022) and landslide detection (Ghorbanzadeh et al., 2022), etc., demonstrating its great potential for disaster management.

Within disaster management, flood detection has become a topic of great interest for EO deep learning applications. Floods, one of the most common natural disasters, cause thousands of deaths, millions of displaced people and hundreds of billions of dollars in damage every year (Rentschler, Salhab, and Jafino, 2022). Deep learning-based models have shown promising performance compared to traditional hydrological models as well as traditional machine learning models on the flood mapping task (Mateo-Garcia et al., 2021; Portalés-Julià et al., 2023). Thanks to the large-scale observation nature of remote sensing technology, these models are able to predict the flood extent on a global scale, which is important for flood warning and post-disaster reconstruction.

2.5.1 Semantic Segmentation on flood detection

In recent years, with the growth of visual data and the development of hardware devices, deep learning algorithms have been widely used in computer vision, including semantic segmentation. In particular, the introduction of the Fully Convolutional Network (Long, Shelhamer, and Darrell, 2015) is considered a milestone in semantic segmentation since it is the first framework that implements end-to-end semantic segmentation using deep learning techniques. Following this, several well-known semantic segmentation networks have been proposed, such as

U-Net (Ronneberger, Fischer, and Brox, 2015), SegNet (Badrinarayanan, Kendall, and Cipolla, 2017), and DeepLabV3+ (L.-C. Chen et al., 2018). More recently, with the success of the vision transformers thanks to their ability to provide state-of-the-art performance, which is facilitated by large-scale datasets and the model's global content modelling capabilities(Dosovitskiy et al., 2020). Several transformer-based semantic segmentation neural networks have emerged, such as SETR (Zheng et al., 2021), SegFormer (E. Xie et al., 2021), Segmenter (Strudel et al., 2021), etc. Such deep learning-based models usually employ an encoder-decoder architecture, where the encoder typically uses a convolutional neural network (CNN) or transformer to extract hierarchical features by progressively downsampling the input, while the decoder upsamples and fuses such features to get the final pixel-wise segmentation map (Zheng et al., 2021). Overall, these deep learning-based semantic segmentation models are broadly used in various domains due to their ability to provide encouraging performance.

For example, numerous deep learning-based algorithms were developed and applied to EO tasks, including flood extent mapping. For instance, (Mateo-Garcia et al., 2021) applied U-Net to Sentinel-2 images for flood mapping, while (Katiyar, Tamkuan, and Nagai, 2021) employed SegNet and U-Net for flood mapping with SAR images. (H. Wu et al., 2022) proposed a novel multi-scale DeepLab model based on MobileNetV2 backbone and DeepLabv3+ for flood detection with SAR images. (Yongsheng Zhou et al., 2022) introduced a CNN-transformer-based model applied to SAR images, demonstrating improved performance of fine water region detection.

2.5.2 Change detection

In remote sensing, most change detection methods are based on bi-temporal images, typically identifying changes that occur in the same region during a time interval. Most traditional bi-temporal change detection methods use image differences or ratios based on the raw features (Mall, Hariharan, and Bala, 2022). For example, Change Vector Analysis (CVA) (Malila, 1980) employed the difference between bi-temporal images to calculate the change vector, while (Gong, Cao, and Q. Wu, 2011) introduces a novel neighbourhood-based ratio operator to generate the change map. Recent work has mostly used deep learning-based methods (Bai et al., 2023; H. Chen, Qi, and Z. Shi, 2021) due to their powerful feature extraction capabilities and promising performance. However, bi-temporal change detection methods are usually difficult to deal with images taken from different seasons and tend to introduce seasonal errors in the results (S. Liu et al., 2015; Hao et al., 2023).

To address this problem, some researchers have proposed time-series change detection methods, which could effectively reduce the seasonal error and obtain a large amount of land use information (Z. Zhu, 2017). However, such methods usually require classification or segmentation for each image, which is time-consuming and inefficient (J. Yan et al., 2019).

2.6 Explainable AI

2.6.1 Explainable AI taxonomies

As deep learning techniques are widely used in various industries, more and more people are concerned about the limitations of these algorithms, with a particular focus on transparency and explainability (Gevaert, 2022). The necessity for interpretable algorithms is not only due to the need to meet the legal and regulatory requirements of the governments and organisations (Regulation, 2016), but also to help users understand and audit the decision-making process of the algorithms.

Despite the rapid development of deep learning, especially deep neural network techniques, the ever more complicated DL models (Krizhevsky, Sutskever, and G. E. Hinton, 2012; Dosovitskiy et al., 2020) do not keep pace with the demands for human understandable interpretability (Rudin, 2019), which have been criticised for their "black box" nature. This fact has motivated researchers to explore interpretable algorithms, and many representative works have been introduced.

Interpretability of deep neural networks is especially important in a number of applications: automotive (J. Kim and Canny, 2017), medical (Ahmad, Eckert, and Teredesai, 2018), Earth Observation (Z. Zhang, P. Angelov, Soares, et al., 2022) alongside others. Demand in such models is necessitated by the pursuit of safety (D. Wei et al., 2022), as well as ethical concerns (Peters, 2022). Some of the pioneering approaches to explaining deep neural networks involve *post hoc* methods; these include saliency models such as the saliency map visualisation method (Simonyan, Vedaldi, and Zisserman, 2014) as well as Grad-CAM (Selvaraju et al., 2017). However, saliency-based explanations may be misleading and not represent the causal relationship between the inputs and outputs (Atrey, Clary, and Jensen, 2019), representing instead the biases of the model (Adebayo et al., 2018). An alternative line of research includes game-theoretic analysis of feature importance, for instance, through the use of Shapley values(Shapley, 1953; Lundberg, 2017). However, such an approach has limited scalability and requires the use of simpler surrogate models or Shapley values' approximations to make it tractable for large-scale analysis (Covert, C. Kim, and S.-I. Lee, 2023).

The explainable AI (xAI) methods can be categorized into four main types: feature attribution, distillation, intrinsic explanations, and contrastive examples (Höhl et al., 2024). Feature attribution provides explanations by highlighting the importance of the input features that strongly influence the output. Distillation builds a separate, intrinsically explainable model to interpret the output of the deep neural network (DNN). Intrinsic explanations attempt to build an explainable model that is itself intrinsically interpretable. Contrastive examples are devoted to presenting a simulated or real example that provides an explanation by comparing them (Höhl et al., 2024; Ras et al., 2022).

2.6.1.1 Feature attribution

The feature attribution representation work is GradCam (Selvaraju et al., 2017), which provides an interpretation of a concept by generating a coarse localisation map in the final convolutional layer to highlight important regions associated with the concept. The localisation map is defined as follows (Selvaraju et al., 2017).

$$L^c_{\text{Grad-CAM}} \in \mathbb{R}^{u \times v} = ReLU\left(\sum_k \alpha^c_k A^k\right) \tag{2.30}$$

Where μ represents the width, v is the height and c denotes the class of interest.

The detailed progress can be summarized as follows (Selvaraju et al., 2017; Molnar, 2020):

1. Forward pass: Pass the input image through the convolutional neural network.

2. Class score extraction: Retrieve the raw score (before the softmax layer) for the target of interest.

3. Zero out other classes: Set the score of all other classes to zero.

4. Gradient Back-propagation: Compute the gradient of the target class score with respect to the features map of the last convolutional layer: $\frac{\delta y^c}{\delta A^k_{ij}}$.

5. Weight calculation: Compute the weight for each feature map using global average pooling:

$$\alpha^c_k = \frac{1}{Z}\sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \tag{2.31}$$

Where i and j are the weight and height indices, and Z is the total number of pixels in the feature map.

6. Weighted feature map: Calculate a weighted combination of the feature maps using the gradient.

7. Apply ReLU: Apply ReLU to only keep the positive values in the weighted feature map to focus on regions that contribute positively to the class score.

8. Visualization: Normalize the heatmap to the range [0, 1], upscale it to map the input image size and overlay it over the original image.

Figure 2.14 shows a comparison of the set of interpretable methods between the cat class and the dog classes. It can be seen that the interpretation provided by Grad-Cam is very clear and intuitive, allowing people to quickly identify important regions related to the class of interest.

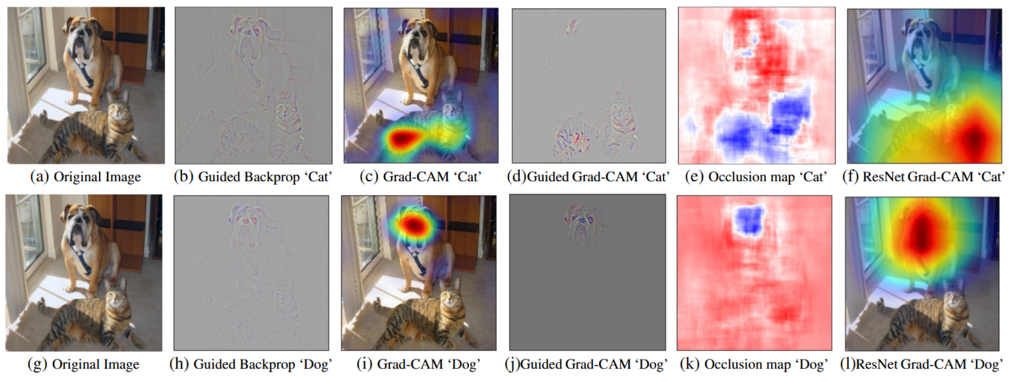


Figure 2.14: Grad-CAM (Selvaraju et al., 2017).

2.6.1.2 Distillation

Distillation attempts to build a separate interpretable model to explain the "black box" model. The representative works in this category are LIME (Ribeiro, S. Singh, and Guestrin, 2016) and SHAP (Lundberg, 2017).

LIME (Ribeiro, S. Singh, and Guestrin, 2016) provides an explanation by constructing a simple linear model trained on a set of generated perturbed instances to find the highest weighted superpixels. Specifically, for a "black box model" f, and an image example ξ. The LIME explanation for image classifiers can be summarized as follows (Ribeiro, S. Singh, and Guestrin, 2016; Garreau and Mardaoui, 2021):

1. Segment images into superpixels used as interpretable features of the image.

2. Generate a new set of images $x_1, x_2, \ldots, x_i$ by randomly grey out part of the superpixels.

3. Get the predictions y_i with this set of images by feeding into the original "black-box model".

$$y_i = f(x_i) \tag{2.32}$$

4. Build a local explainable surrogate model $\hat{\beta}_n$ to approximate the predictions of f locally around the original image ξ.

2.6.1.3 Intrinsic explanations

Unlike the other three post-hoc explanatory approaches, an arguably better approach is to construct interpretable-by-design *ante hoc* models (Rudin, 2019) that are inherently interpretable. These models could use different principles, such as (1) interpretable-by-design architectures (Böhle, Fritz, and Schiele, 2022), which are designed to provide interpretations at every step of the architecture, as well as (2) prototype-based models, which perform decision making as a function of (dis)similarity to existing prototypes (P. Angelov and Soares, 2020).

Several examples of interpretable-by-design architectures include Bayesian Neural Networks and B-cos networks. Bayesian Neural Networks (BNNs) provide explanations based on uncertainty predictions that could help human users understand how uncertain the results are (Clare et al., 2022; Mitros and Mac Namee, 2019). B-cos network is introduced to increase the interpretability of deep neural networks (DNNs) by promoting weight-input alignment during training (Böhle, Fritz, and Schiele, 2022; Böhle, N. Singh, et al., 2024).

In addition to the above, the prototype-based models have attracted attention for their natural proximity to human reasoning (Biehl, Hammer, and Villmann, 2016). Related works include xDNN (P. Angelov and Soares, 2020), ProtoPNet (C. Chen et al., 2019), DNC (W. Wang et al., 2022), IDEAL (P. Angelov, Kangin, and Z. Zhang, 2025) and IDSS (Z. Zhang, P. Angelov, Soares, et al., 2022). These models are trained either by adding a prototype layer to the neural network and training the model end-to-end or by performing a clustering algorithm on a trained/pre-trained feature space of the neural network to find the most representative prototypes. A main characteristic of this approach is the use of example or prototype-based explanations, which explain the model's decision-making by selecting representative instances directly from the dataset.

The idea of prototype-based machine learning is closely related to the symbolic methods (Newell, Shaw, and Simon, 1959), and draws upon the case based reasoning (B. Kim, Rudin, and Shah, 2014) and sparse learning machines (Poggio and Girosi, 1998), which are designed to learn a linear (with respect to parameters) model, which is (in general, nonlinearly) dependent on a subset of training data samples. At the centre of many such methods is the kernel trick (Schölkopf, Herbrich, and A. J. Smola, 2001), which involves mapping of training and inference data into a space with a different inner product within a reproducing Hilbert space (Aronszajn, 1950). Such models include support vector machines (SVMs) for classification (Boser, Guyon, and Vapnik, 1992) and support vector regression (SVR) models (A. Smola and Schölkopf, 2004) for regression, as well as relevance vector machines (RVMs), which have demonstrated improvements in sparsity (Tipping, 2001).

Example or prototype-based methods are usually model-agnostic. The reason is that such a model provides an explanation by selecting instances from the dataset rather than summarizing the importance of individual features (Molnar, 2020). These methods are particularly useful when the selected instances are human-understandable, such as images, as they allow users to directly view and interpret the examples.

A well-known example is the K-nearest neighbour (KNN) method. For the case where k is equal to 1. When making a decision, the algorithm compares the distance between the test data and the training instance. It will then return the nearest neighbour to the test data and assign the prediction based on the class of that neighbour. If the instance itself is meaningful to human users, the decision-making process becomes explainable, as the test data can be directly analogized to its closest training instance, providing intuitive reasoning for the prediction.

While traditional interpretable machine learning methods offer good interpretability and transparency, their performance is often not as expected. On the other hand, deep learning neural networks typically provide superior performance but suffer from the limitations of their "black box" nature. To address this gap, researchers have begun to explore methods that aim to combine the strengths of both approaches, offering both high accuracy and interpretability.

The representative works in this area are ProtoPNet (C. Chen et al., 2019)

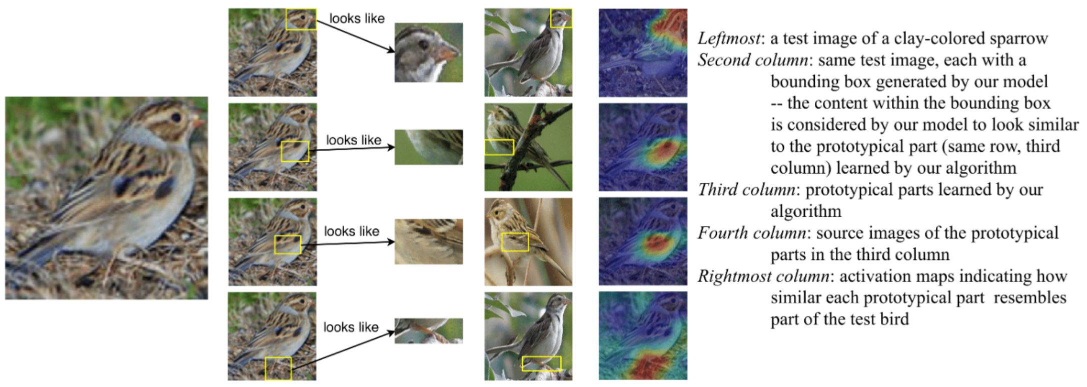

looks like
looks like
looks like
looks like
Leftmost: a test image of a clay-colored sparrow
Second column: same test image, each with a bounding box generated by our model -- the content within the bounding box is considered by our model to look similar to the prototypical part (same row, third column) learned by our algorithm
Third column: prototypical parts learned by our algorithm
Fourth column: source images of the prototypical parts in the third column
Rightmost column: activation maps indicating how similar each prototypical part resembles part of the test bird

Figure 2.15: ProtoPNet reasoning process (C. Chen et al., 2019).

and xDNN (P. Angelov and Soares, 2020). ProtoPnet introduces an additional prototype layer to a Convolutional neural network and explains its decision-making process by dissecting images into prototypical parts and comparing their latent features with the learned prototypes. As shown in Figure 2.15, ProtoPNet classifies an image as a clay-coloured sparrow because specific parts of the image look like prototypical parts from the training images. The ProtoPNet is trained end-to-end, which means for every new task, a separate model must be trained from scratch, which is computationally expensive.

In contrast, xDNN employs a different strategy. As shown in Figure 2.16, it utilises a pre-trained neural network for feature extraction and then learns a set of prototypes to provide explanations. Instead of requiring task-specific training, xDNN compares the similarity between the test image and the learned prototypes, making it computationally friendly while still offering interpretability.

2.6.1.4 Contrastive examples

(Höhl et al., 2024) shows two types of contrastive examples: counterfactual explanations and example-based explanations (Höhl et al., 2024). The counterfactual methods provide explanations mainly by showing a set of examples that are similar to the original input. The representative work is (Wachter, Mittelstadt, and Russell,

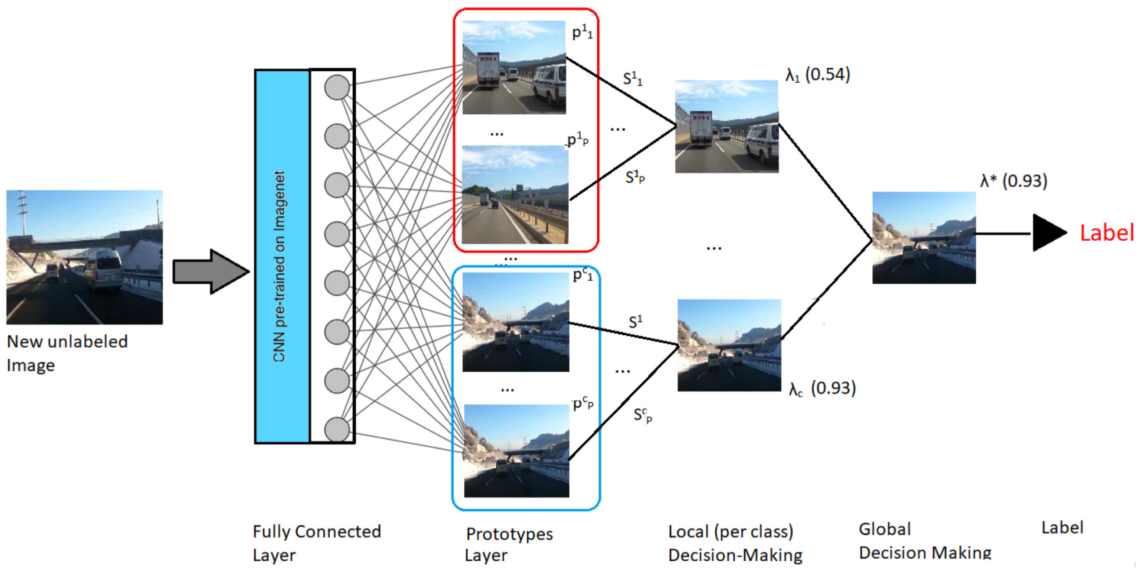


Figure 2.16: xDNN reasoning process (P. Angelov and Soares, 2020).

2017), which tries to find a counterfactual most close to the input. The example-based explanations provide explanations by comparing the input with a set of examples. However, I consider that the example-based explanations are more related to the prototype-based methods of intrinsic explanations, which are discussed in the previous section.

2.6.2 Explainable AI in Earth Observation

Recent years have witnessed rapid advancements in machine learning development, with a large number of algorithms being applied to EO. Among these, deep learning methodologies are particularly noteworthy due to their ability to achieve state-of-the-art results. However, these models are normally perceived as "black-box" models, due to their complexity, containing millions or billions of trainable parameters, which complicates the interpretability of the decision-making process for human users. This characteristic becomes a notable issue when applied to fields like EO, which require a clear and understandable decision-making process. It's crucial that human users can comprehend, inspect and audit the decisions made by such models (Gevaert, 2022). Consequently, addressing the "black box" nature of these algorithms is of paramount importance.

As deep learning techniques are widely used in various industries, more and more people are concerned about the limitations of these algorithms, with a particular focus on transparency and explainability (Gevaert, 2022). The necessity for interpretable algorithms is not only due to the need to meet the legal and regulatory requirements of the governments and organisations (Regulation, 2016), but also to help users understand and audit the decision-making process of the algorithms.

Despite the rapid development of deep learning, especially deep neural network techniques, they have been criticised for their "black box" nature. This fact has motivated researchers to explore interpretable algorithms, and many representative works have been introduced. Early work focused on post hoc explanations. For example, GradCam (Selvaraju et al., 2017) provides an interpretation of a concept by generating a coarse localisation map in the final convolutional layer to highlight important regions associated with the concept. LIME (Ribeiro, S. Singh, and Guestrin, 2016) provides an explanation by constructing a simple linear model trained on a set of generated perturbed instances to find the highest weighted superpixels. These interpretable methods have been widely used in Earth Observation. For example, (X. Guo et al., 2022) proposed a variant of GradCam, Prob-Cam, and successfully applied it to the remote sensing image classification task. Similarly, (Hung, H.-C. Wu, and Tseng, 2020) applied the LIME method to the remote sensing scene classification task. However, such classes of explanations may not faithfully reflect the intrinsic decision-making. More recent work addresses the question of post hoc explanations (Chan, Kong, and Liang, 2022). However, to address these problems entirely, one can consider interpretable-by-design methods.

Several interpretable-by-design models have been proposed. For example, Bayesian Neural Networks (BNNs) provide explanations based on uncertainty predictions that could help human users understand how uncertain the results are (Clare et al., 2022; Mitros and Mac Namee, 2019). B-cos network is introduced to increase the interpretability of deep neural networks (DNNs) by promoting weight-input alignment during training (Böhle, Fritz, and Schiele, 2022; Böhle, N. Singh, et al., 2024).

In addition to the above, the prototype-based models have attracted attention for their natural proximity to human reasoning (Biehl, Hammer, and Villmann, 2016). Related works include xDNN (P. Angelov and Soares, 2020), ProtoPNet (C. Chen et al., 2019), DNC (W. Wang et al., 2022), IDEAL (P. Angelov, Kangin, and Z. Zhang, 2025) and IDSS (Z. Zhang, P. Angelov, Soares, et al., 2022). These models are trained either by adding a prototype layer to the neural network and training the model end-to-end or by performing a clustering algorithm on a trained/pre-trained feature space of the neural network to find the most representative prototypes.

Chapter 3

An interpretable-by-design deep learning algorithm

Figure 3.1 illustrates the general prototype-based framework that underpins the method developed in this chapter and its extensions in subsequent chapters. This structure is applicable to various domains, including general image classification and remote sensing semantic segmentation. The model learns interpretable prototypes via clustering, and decisions are made based on the similarity of the input query to the prototypes.

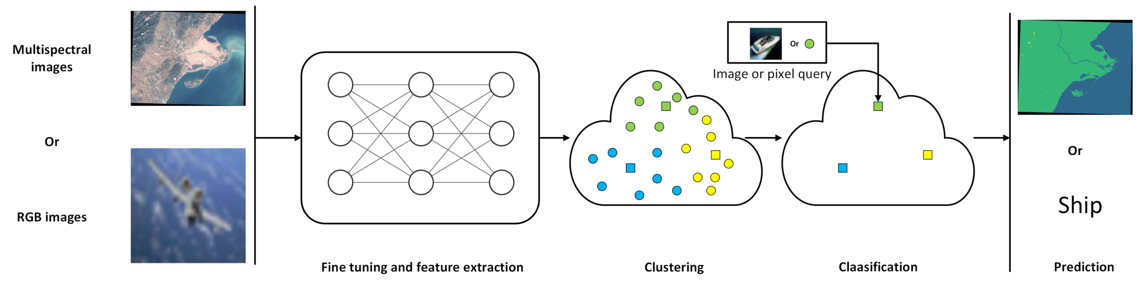


Figure 3.1: General prototype-based learning framework. This framework takes either multispectral or RGB images as input, performs feature extraction with deep learning neural networks, clusters the latent features to obtain prototypes, and uses similarity-based reasoning for classification and prediction.

3.1 Background

Deep-learning (DL) models can be formulated as deeply embedded functions of functions (P. P. Angelov and Xiaowei Gu, 2019; Rosenblatt et al., 1962), optimised through backpropagation (Rumelhart, G. E. Hinton, and R. J. Williams, 1986):

$$\hat{y}(\mathbf{x}) = f_n(\ldots(f_1(\mathbf{x};\boldsymbol{\theta}_1)\ldots);\boldsymbol{\theta}_n), \tag{3.1}$$

where $f_n(\ldots(f_1(\mathbf{x};\boldsymbol{\theta}_1)\ldots);\boldsymbol{\theta}_n)$ is a layered function of the input $\mathbf{x}$, which has a generic enough, fixed parameterisation $\boldsymbol{\theta}$. to predict desirable outputs $\hat{y}$.

However, this problem statement has the following limitations:

(1) transfer learning typically requires finetuning (Kornblith, Shlens, and Le, 2019) using error back-propagation (EBP) on the target problem and data of interest

(2) such formulation does not depend upon training data, so the contribution of these samples towards the output $\hat{y}$ is unclear, which hinders interpretability. For the interpretable architectures, such as ProtoPNet (C. Chen et al., 2019), finetuning leads to confounding interpretations (Bontempelli et al., 2022)

(3) finally, for lifelong learning problems, such finetuning creates obstacles such as catastrophic forgetting (Parisi et al., 2019)

Emergence of foundation models, aimed at better generalisation and facilitating transfer learning, allows for mitigating point (1). Studies such as DINOv2 (Oquab et al., 2023) demonstrate competitive results for ViT-based (Dosovitskiy et al., 2020) architectures on transfer learning tasks on a range of datasets with linear finetuning. However, these works do not address either interpretability or lifelong learning. In this work, to jointly address all three above-mentioned limitations, we propose a generic framework for prototypical transfer learning called IDEAL (Interpretable-by-design DEep learning ALgorithms). Through this framework, we extensively study the benefits and trade-offs of prototypical transfer learning without finetuning across different architectures and tasks.

Our solution for transfer learning, which generalises xDNN (P. Angelov and Soares, 2020) and ProtoPNet (C. Chen et al., 2019), can be summarised in the following form:

$$\hat{y} = g(\mathbf{x}; \boldsymbol{\theta}, \mathbb{P}), \tag{3.2}$$

where $\mathbb{P}$ is a set of prototypes. We consider a more restricted version of the function $g(\cdot)$:

$$\hat{y} = g(\mathbf{x}; \boldsymbol{\theta}_{\{d,h\}}, \mathbb{P}) = h(d(\mathbf{x}, \mathbf{p}; \boldsymbol{\theta}_d)|_{\mathbf{p}\in\mathbb{P}}; \boldsymbol{\theta}_h), \tag{3.3}$$

where d is some form of (dis)similarity function (which can include DL feature extractors), $\boldsymbol{\theta}_d$ and $\boldsymbol{\theta}_h$ are parameterisations of functions d and h, respectively.

The methods from this research draw from cognitive science and the way humans learn, namely, using examples of previous observations and experiences (Zeithamova, Maddox, and Schnyer, 2008). Prototype-based models have long been used in different learning systems: k nearest neighbours (Radovanovic, Nanopoulos, and Ivanovic, 2010); decision trees (Nauta, Van Bree, and Seifert, 2021); rule-based systems (P. Angelov, Lughofer, and X. Zhou, 2008); case-based reasoning (B. Kim, Rudin, and Shah, 2014); sparse kernel machines (Tipping, 1999). The advantages of prototype-based models have been advocated, for example, in (Bien and Tibshirani, 2011). The first prototypical architecture, learning both distances and prototypes, was proposed in (Snell, Swersky, and Zemel, 2017) and more recently developed in (C. Chen et al., 2019; P. Angelov and Soares, 2020) and (W. Wang et al., 2023).

In this thesis, we demonstrate the efficiency of the proposed framework: it is compact, easy to interpret by humans, fast to train and adapt in a lifelong learning setting and benefits from a latent data space learnt from a generic dataset transferred to a different, more specific domain.

Specifically, we make the following contributions:

- we define the framework called IDEAL, shown in Figure 3.2, which transforms a given non-interpretable latent space into an interpretable one based on prototypes, derived from the training set without finetuning, and quantify the performance gap between such a model, its finetuned counterpart and standard DL architectures.

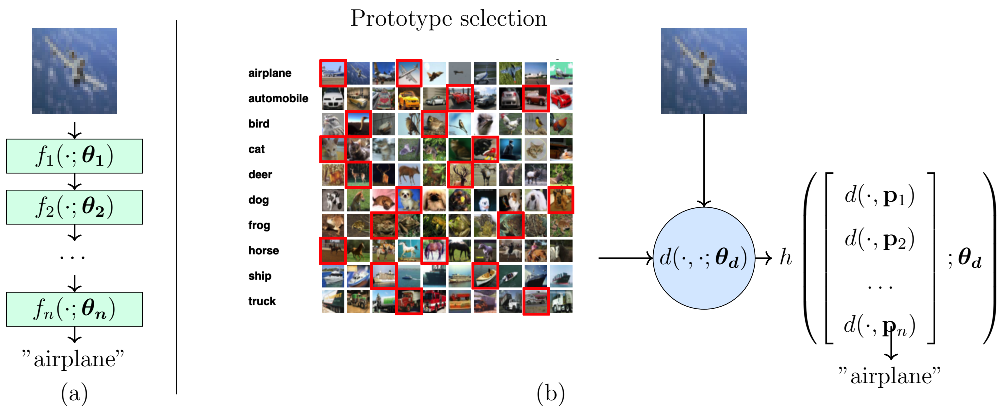


Figure 3.2: Difference between (a) a standard deep-learning model, and (b) the proposed prototype-based approach, IDEAL. Dataset credit: CIFAR-10 (Krizhevsky and G. Hinton, 2009).

- we demonstrate the benefits of the proposed framework on transfer and lifelong learning scenarios. Namely, in a fraction of training time and **without finetuning** of latent features, the proposed models achieve performance competitive with standard DL techniques.

- we demonstrate the model's interpretability on classification and lifelong learning tasks, and show that **without** finetuning, the resulting models achieve **better** performance on confounded CUB data compared to finetuned counterparts (Wah et al., 2011; Bontempelli et al., 2022)

We apply this generic IDEAL framework to a set of standard DL architectures such as ViT (Dosovitskiy et al., 2020; M. Singh et al., 2022), VGG (Simonyan and Zisserman, 2014), ResNet (K. He, X. Zhang, et al., 2016) and xDNN (P. Angelov and Soares, 2020) and evaluate the methodology on a range of well-known datasets such as CIFAR-10, CIFAR-100, CalTech101, EuroSAT, Oxford-IIIT Pet, and STL-10.

3.2 Methodology

3.2.1 Problem statement

Two different definitions of the problem statement are considered: offline and online (lifelong) learning.

3.2.1.1 Offline learning

Consider the following optimisation problem:

$$\arg\min_{\substack{\mathbb{P}=\mathbb{P}(\mathbb{X}),\\ \boldsymbol{\theta}_{\{d,h\}}}} \sum_{(\mathbf{x},y)\in(\mathbb{X},\mathbb{Y})} l(h(d(\mathbf{x},\mathbf{p};\boldsymbol{\theta}_d)|_{\mathbf{p}\in\mathbb{P}};\boldsymbol{\theta}_h),y), \tag{3.4}$$

where $(\mathbb{X},\mathbb{Y})$ are a tuple of inputs and labels, respectively, and $\mathbb{P}$ is a set of prototypes derived from data $\mathbb{X}$ (e.g., by selecting a set of representative examples or by clustering).

Brute force optimisation for the problem of selecting a set of representative examples is equivalent to finding a solution of the best-subset selection problem, which is an NP-hard problem (Natarajan, 1995). While there are methods for solving such subset selection problems in limited cases, such as sparse linear regression (Bertsimas, King, and Mazumder, 2016), it still remains computationally inefficient in a general case (polynomial complexity is claimed in (J. Zhu et al., 2020) and/or solving it only in a limited (i.e. linear) setting.

The common approach to dealing with such a selection problem is to replace the original optimisation problem (equation (3.4)) with a surrogate one, where the prototypes $\mathbb{P}$ are provided by a data distribution (P. Angelov and Soares, 2020) or a geometric, e.g. clustering (W. Wang et al., 2023) technique. Then, once the prototypes are selected, the optimisation problem becomes:

$$\arg\min_{\boldsymbol{\theta}_{\{d,h\}}} \sum_{(\mathbf{x},y)\in(\mathbb{X},\mathbb{Y})} l(h(d(\mathbf{x},\mathbf{p};\boldsymbol{\theta}_d)|_{p\in\mathbb{P}};\boldsymbol{\theta}_h),y), \tag{3.5}$$

where d is a (learnable) (dis)similarity function and h is an aggregation function, parameterised with $\boldsymbol{\theta}_d$ and $\boldsymbol{\theta}_h$ respectively. As we detail below, the example of function d could be a Euclidean distance, and h could be a winner-takes-all or k-nearest-neighbours (kNN) operator.

3.2.1.2 Online (lifelong) learning

Instead of solving a single objective for a fixed dataset, the problem is transformed into a series of optimisation problems for progressively growing set $\mathbb{X}$:

$$\{\arg\min_{\boldsymbol{\theta}_{\{d,h\}}} \sum_{(\mathbf{x},y)\in(\mathbb{X}_n,\mathbb{Y}_n)} l(h(d(\mathbf{x},\mathbf{p};\theta_d)|_{\mathbf{p}\in\mathbb{P}_n};\boldsymbol{\theta}_h),y)\}_{n=1}^{N}, \mathbb{X}_n = \mathbb{X}_{n-1} + \{\mathbf{x}_n\}, \mathbb{X}_1 = \{\mathbf{x}_1\}. \tag{3.6}$$

Once the prototypes are found, the problem would only require light-weight optimisation steps as described in Algorithms 4 and 5.

Algorithm 4: Training and testing (offline)

```
Data: Training data $\mathbb{X} = \{\mathbf{x}_1 \ldots \mathbf{x}_N\}$;
Result: Prototype-based classifier $c(\mathbf{x};\mathbb{P},\boldsymbol{\theta})$
$\mathbb{P} \leftarrow$ FindPrototypes($\{\mathbf{x}_1 \ldots \mathbf{x}_N\}$);   // Prototype selection function
 FindPrototypes : $\mathbb{X} \rightarrow \mathbb{P}$
$\theta \leftarrow$ SelectParameters($\mathbb{X},\mathbb{Y},\boldsymbol{\theta}$);   // SelectParameters is a solution
 of Eq. 3.5
$\hat{\mathbb{Y}}_T \leftarrow \{h(d(\mathbf{x},\mathbf{p};\boldsymbol{\theta}_d)|_{\mathbf{p}\in\mathbb{P}};\boldsymbol{\theta}_h)\}_{\mathbf{x}\in\mathbb{X}_T}$;
```

Algorithm 5: Training and testing (online)

```
Data: Training data $\mathbb{X} = \{\mathbf{x}_1 \ldots \mathbf{x}_N\}$;
Result: Prototype-based classifier $h(d(\mathbf{x},\mathbf{p};\boldsymbol{\theta}_1)|_{p\in\mathbb{P}};\boldsymbol{\theta}_2)$
$\mathbb{P} \leftarrow \{\}$;
for $\{\mathbf{x}, y\} \in \mathbb{X}$ do
    $\hat{y} = h(d(\mathbf{x},\mathbf{p};\boldsymbol{\theta}_d)|_{\mathbf{p}\in\mathbb{P}};\boldsymbol{\theta}_h)$;
    $\mathbb{P} \leftarrow$ UpdatePrototypes($\mathbb{P},\mathbf{x}$));   // Prototype update function
     UpdatePrototypes : $\mathbb{P} \times \mathbb{X} \rightarrow \mathbb{P}$
    $\theta \leftarrow$ UpdateParameters($\mathbb{X},\mathbb{Y},\boldsymbol{\theta}$);   // UpdateParameters is a
     solution of Eq. 3.6
end
```

3.2.2 Choice of functions d and h

While we define the framework in generic terms, we limit our analysis to a special case of Euclidean distance and winner-takes-all function. This helps focus on quantifying the trade-offs of accuracy, interpretability and generalisation between the model without finetuning, on one hand, and state-of-the-art, fully finetuned models. Although it may be possible to further improve the performance by finding better architectural choices, we decided to focus on the simple parameterisation of the framework with the Euclidean distance and a winner-takes-all decision making.

Throughout the experiments, we use the negative Euclidean distance between the feature vectors:

$$d(\mathbf{x}, \mathbf{p}; \theta_d) = -\ell^2(\phi(\mathbf{x}; \theta_d), \phi(\mathbf{p}; \theta_d)), \tag{3.7}$$

where ϕ is the feature extractor output. For the scenario without finetuning, θ_d is frozen: $\phi(\cdot) = \phi(\cdot; \theta_d), \theta_d = \text{const}$. The similarities bounded between $(0, 1]$ could be obtained by, for example, taking the exponential of the similarity function or normalising it. Except from the experiment in Figure 3.5, where h is implemented as k-NN, the function h is a winner-takes-all operator:

$$h(\cdot) = \text{CLASS}(\arg\min_{p \in \mathbb{P}} d(\cdot, \mathbf{p}; \theta_d)) \tag{3.8}$$

Note that the lack of finetuning makes the loss function trivial as the model does not have any free parameters $\theta_{\{d,h\}}$.

3.2.3 Difference from the other prototype-based frameworks

Existing prototype-based models, such as ProtoPNet (C. Chen et al., 2019), DNC (W. Wang et al., 2023) and xDNN (P. Angelov and Soares, 2020), focus on end-to-end training for the purpose of interpretability by design and not on transfer learning from the existing pretrained models. All of them can also be considered as specific cases of the presented framework. Neither of these models are aiming to address transfer learning, in contrast to this chapter's attention on the trade-offs between finetuned and non-finetuned models.

3.2.3.1 xDNN

(P. Angelov and Soares, 2020) is a special case of our IDEAL formulation with

$$d(\mathbf{x}, \mathbf{p}; \theta_d) = -\mathfrak{C}(\phi(\mathbf{x}; \theta_d), \phi(\mathbf{p}; \theta_d)), \tag{3.9}$$

where $\mathfrak{C}$ is a Cauchy similarity. It optimises the coefficients θ_d as a part of its finetuning procedure prior to the model training, and its decision making is defined according to the winner-takes-all procedure as per Equation 3.8.

3.2.3.2 DNC

(W. Wang et al., 2023) selects prototypes at every optimisation step using an online version of Sinkhorn-Knopp clustering algorithm (Cuturi, 2013) and defines l in equation 3.6 a softmax cross-entropy loss.

3.2.3.3 ProtoPNet

(C. Chen et al., 2019), in contrast to the former two methods, operates over patches and not the full images:

$$d(\mathbf{x}, \mathbf{p}; \theta_d) = \max_{\hat{\mathbf{x}} \in \text{patches}(\mathbf{x})} (\log(\ell^2(\hat{\mathbf{x}}, \mathbf{p}) + 1) - \log(\ell^2(\hat{\mathbf{x}}, \mathbf{p}) + \epsilon), \tag{3.10}$$

where ϵ is a parameter. ProtoPNet also defines a decision making function h as follows and optimises jointly the prototypes $\mathbb{P}$ and the parameters $\theta_{d,h}$ using cross-entropy loss:

$$h(\cdot) = \text{FC}\left(\begin{bmatrix} d(\cdot, p_1) \\ d(\cdot, p_2) \\ \cdots \\ d(\cdot, p_{|\mathbb{P}|}) \end{bmatrix}_{p_i \in \mathbb{P}, i \in [1, \ldots, |\mathbb{P}|]} \right) \tag{3.11}$$

3.2.4 Prototype selection through clustering

Selection of prototypes through many standard methods of clustering, such as k-means (Steinhaus et al., 1956), is used by methods such as (Z. Zhang, P. Angelov, Soares, et al., 2022), DNC (W. Wang et al., 2023). However, these methods have one serious limitation: they utilise the averaging of cluster values, so the prototypes $\mathbb{P}$ do not, in general, belong to the original training dataset $\mathbb{X}$. It is still possible, however, to attribute the prediction to the set of the cluster members.

The possible options for such prototype selection are summarised below. Standard *black-box* classifiers do not offer interpretability through prototypes. Prototypes, selected through k-means, are non-interpretable on their own account as discussed above; however, it is possible to attribute such similarity to the members of the clusters. Finally, one can select real prototypes as cluster centroids. This way it is possible to attribute the decision to a number of real image prototypes ranked by their similarity to the query image. Such a choice between averaged and real centroids can create, as we show in the experimental section, a trade-off between interpretability and performance (see Section 3.4.1).

3.3 Experiments

Throughout the experimental scenarios, we contrast three settings (see Figure 3.3):

A) Standard DL pipeline involving training on generic datasets as well as finetuning on target ("downstream") task or data — both with iterative error backpropagation.

B) IDEAL **without finetuning**: the proposed prototype-based IDEAL method involving clustering in the latent feature space with subsequent decision making process such as using winner-takes-all analysis or k nearest neighbours, as outlined in Algorithms 4 and 5.

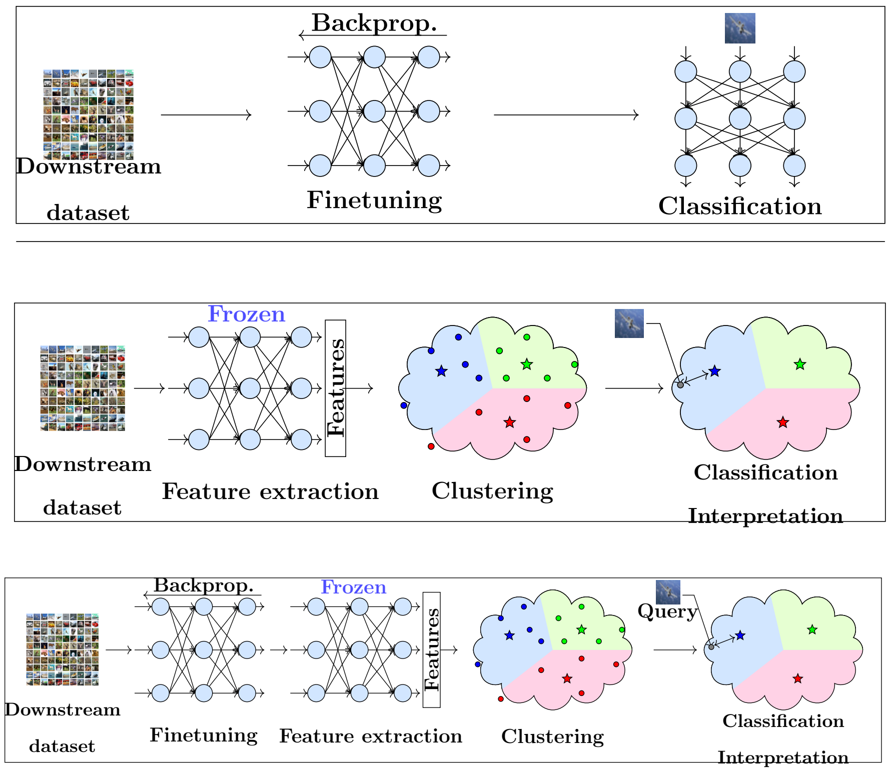


Figure 3.3: Experimental setup. Top: standard DL model; middle: proposed framework with **no** finetuning; bottom: proposed framework **with** finetuning.

C) IDEAL **with finetuning**: Same as B) with the only difference being that the clustering is performed in a latent feature space which is formed by finetuning on target data set (from the "downstream" task) using iterative error backpropagation. Unlike A), setting C) provides interpretable prototypes.

3.3.1 Experimental setting

3.3.1.1 Datasets

CIFAR-10 and CIFAR-100 (Krizhevsky and G. Hinton, 2009), STL-10 (Coates, Ng, and H. Lee, 2011), Oxford-IIIT Pet (Parkhi et al., 2012), EuroSAT (Helber et al., 2018; Helber et al., 2019), CalTech101 (F.-F. Li, Fergus, and Perona, 2006).

3.3.1.2 Feature extractors

We consider a number of feature extractor networks such as VGG-16 (Simonyan and Zisserman, 2014), ResNet50 (K. He, X. Zhang, et al., 2016), ResNet101 (K. He, X. Zhang, et al., 2016), ViT-B/16 (Dosovitskiy et al., 2020), henceforth referred to as ViT), ViT-L/16 (Dosovitskiy et al., 2020), henceforth referred to as ViT-L) with or **without** finetuning; the pre-trained latent spaces for ViT models were obtained using SWAG methodology (M. Singh et al., 2022); the computations for feature extractors have been conducted using a single V100 GPU.

3.3.1.3 Prototype selection techniques

We include the results for such clustering techniques as k-means, k-means with a nearest data point (referred to as k-means (nearest)), and two online clustering methods: xDNN (P. Angelov and Soares, 2020) and ELM (Baruah and P. Angelov, 2012). If not stated otherwise, we set the reported number of prototypes to 10% of the sample. In the Appendix, we present a detailed analysis with a varying number of prototypes.

3.3.1.4 Baselines

We explore trade-offs between standard deep neural networks, different architectural choices (averaged prototypes vs real-world examples) in Section 3.4.1

For reproducibility, the full parameterisation is described in 3.5.1.

3.4 Empirical questions

We group the results of our analysis in accordance with a number of empirical questions. Questions 1 and 2 confirm that the method delivers competitive results **even without finetuning**. Building upon this initial intuition, we develop the key Questions 3, 4 and 5, analysing the performance for lifelong learning scenarios and interpretations proposed by IDEAL, respectively.

Question 1. *How does the performance of the IDEAL framework* ***without*** *finetuning compare with the well-known deep learning frameworks?*

Section 3.4.1 and 3.5.2 show, with a concise summary in Figures 3.4 and 3.10, that the gaps between finetuned and non-finetuned IDEAL framework are consistently much smaller (tens of percent vs a few percentage points) for vision transformer backbones compared to ResNets and VGG. Furthermore, Figure 3.7 shows that the training time expenditure is more than an order of magnitude smaller compared to the finetuning time.

Question 2. *To what extent does finetuning of the feature space for the target problem lead to overfitting?*

In Section 3.4.2, Figures 3.8 and 3.9, we demonstrate the issue of overfitting on the target spaces by finetuning on CIFAR-10 and testing on CIFAR-100 in both performance and through visualising the feature space. Interestingly, we also show in Table 3.3 of the Appendix that, while the choice of prototypes greatly influences the performance of the IDEAL framework **without** finetuning of the backbone, it does not make any significant impact for the finetuned models (i.e., does not improve upon random selection).

Question 3 *How does the IDEAL framework* ***without*** *finetuning compare in the class-incremental learning setting?*

In Section 3.4.3 we build upon Questions 1 and 2 and demonstrate: the small gap between pretrained and finetuned ViT models ultimately enables us to solve class-incremental learning scenarios, improving upon well-known baseline methods. IDEAL framework **without** finetuning shows performance results on a number of class-incremental learning problems, comparable to task-level finetuning. Notably, in the CIFAR-100 benchmark, the proposed method provides 83.2% and 69.93% on ViT-L and ResNet-101, respectively, while the state-of-the-art method from (Z. Wang et al., 2022) only reports 65.86%.

Question 4 *How does the IDEAL framework provide insight and interpretation?*

In Section 3.4.4, we present the analysis of interpretations provided by the method. In Figures 3.12, 3.13 and 3.14, we demonstrate the qualitative experiments showing the human-readable interpretations provided by the model for both lifelong learning and offline scenarios.

Question 5. *Can models* ***without*** *finetuning bring advantage over the finetuned ones in terms of accuracy and help identify misclassifications due to confounding (i.e., spurious correlations in the input)?*

While admittedly, the model only approaches but does not reach the same level of accuracy for the same backbone without finetuning in the standard benchmarks such as CIFAR-10, it delivers a better performance in cases with confounded data (with spurious correlations in the input). In Section 3.4.5, Table 3.1, we demonstrate, building upon the intuition from Question 2, that finetuning leads to overfitting on confounded data, and leads to confounded predictions and interpretations. We also demonstrate that in this setting, IDEAL **without finetuning** improves upon F1 score against the finetuned baseline as well as provides interpretations for wrong predictions due to the confounding.

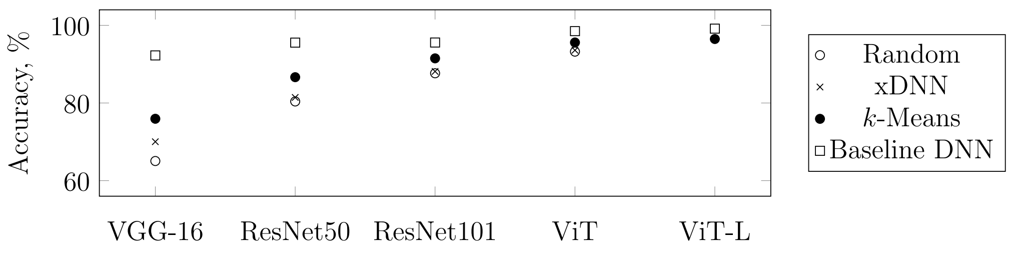


Figure 3.4: Comparison of the proposed IDEAL framework (**without** finetuning) on the CIFAR-10 data set with different prototype selection methods (random, the clustering used in xDNN (Soares, P. Angelov, and Z. Zhang, 2021) and k-means method) vs the baseline DNN.

3.4.1 Offline classification

We found that the gap between the finetuned and non-finetuned models on a range of tasks decreases for the modern, high performance, architectures, such as ViT (Dosovitskiy et al., 2020). For CIFAR-10, these findings are highlighted in Figure 3.4. While finetuned VGG-16's accuracy is close to the one of ViT and other recent models, different prototype selection techniques **without** finetuning (the one used in xDNN, k-means clustering, and random selection) all give accuracy between 60 and 80%. The picture is totally different for ViT, where k-means prototype selection **without finetuning** provides accuracy of 95.59% against finetuned ViT's own performance of 98.51%.

While the results above report on the performance of the k-means clustering used as a prototype selection technique, the experimental results in Figure 3.5 explore choosing the nearest prototype to k-means cluster centroid for interpretability reasons. Although it is clear (with further evidence presented in 3.5.2) that the performance when selecting the nearest to the k-means centroids prototypes is lagging slightly behind the direct use of the centroids (denoted simply as k-means), it is possible to bring this performance closer by replacing the winner-takes-all decision making approach (Equation (3.8)) with the k nearest neighbours method. For this purpose, we utilise the sklearn's `KNeighborsClassifier` function.

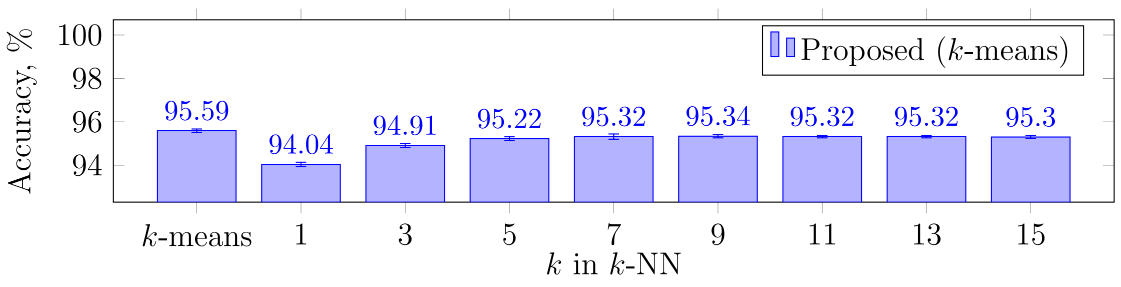


Figure 3.5: Comparison of results on CIFAR-10 (ViT, k nearest neighbours).

Below, we analyse closer just the results with using ViT as a feature extractor forming the latent data space. One can see in Figure 3.6 that: (1) **without finetuning**, on a number of tasks, the model shows competitive performance, and (2) for CIFAR-10 and CIFAR-100, with finetuning of the backbone, the difference between the standard backbone and the proposed model is insignificant within the confidence interval. Therefore, for the rest of the datasets, we focus on the experiments without finetuning. In Figure 3.7, one can see the comparison of the time expenditure between the finetuned and **non-finetuned** model.

We conducted (see 3.5.3) a sensitivity analysis experiment by varying the number of prototypes for CIFAR-10 on the ResNet101 backbone by changing the value k for the k-means method. In 3.5.2, we also show the results with the online clustering method ELM (Baruah and P. Angelov, 2012), which does not require the number of clusters to be pre-defined and instead uses a radius meta-parameter which affects granulation.

3.4.2 Demonstration of overfitting in the finetuned feature spaces and the prototype selection impact

One clear advantage of transfer learning without finetuning is dramatically lower computational cost reflected in the time expenditure. However, there is also another advantage. The evidence shows that the finetuned feature space shows

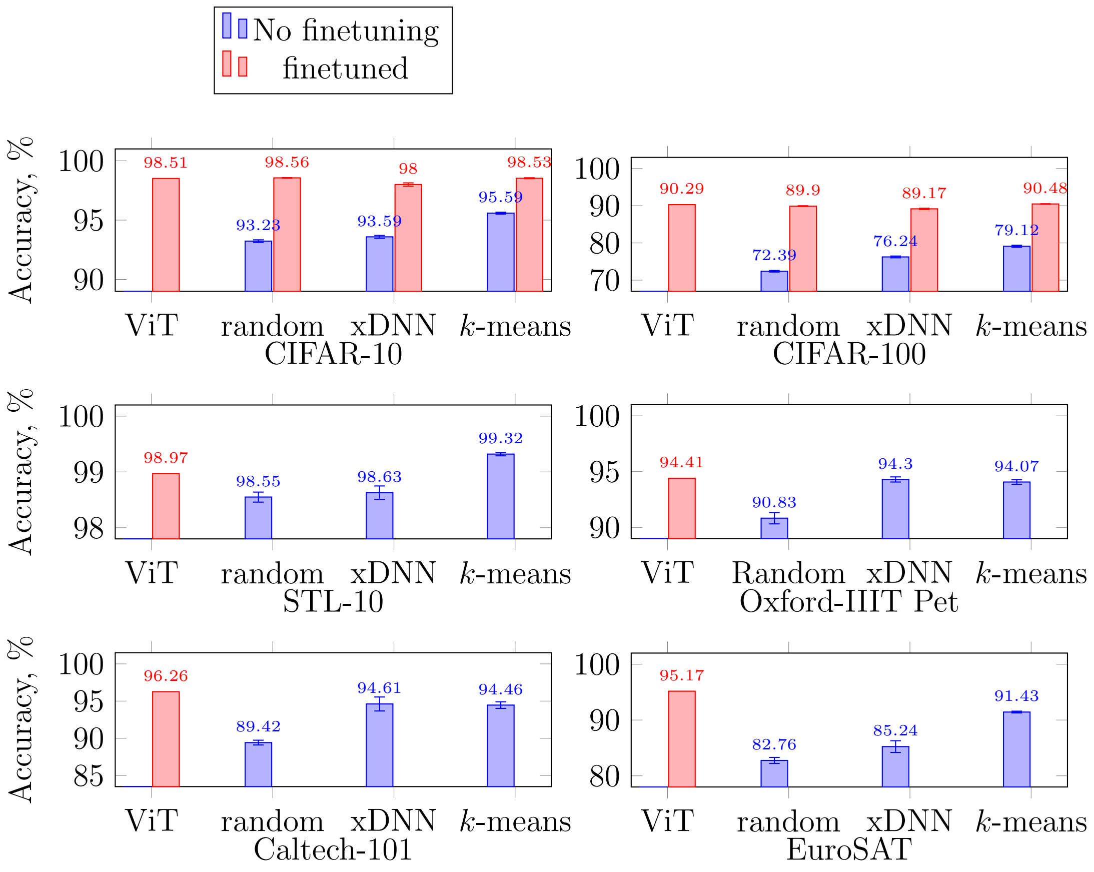


Figure 3.6: Comparison of results with ViT (Dosovitskiy et al., 2020) as a feature extractor on a number of datasets. Random, xDNN, k-means denote different prototype selection methods.

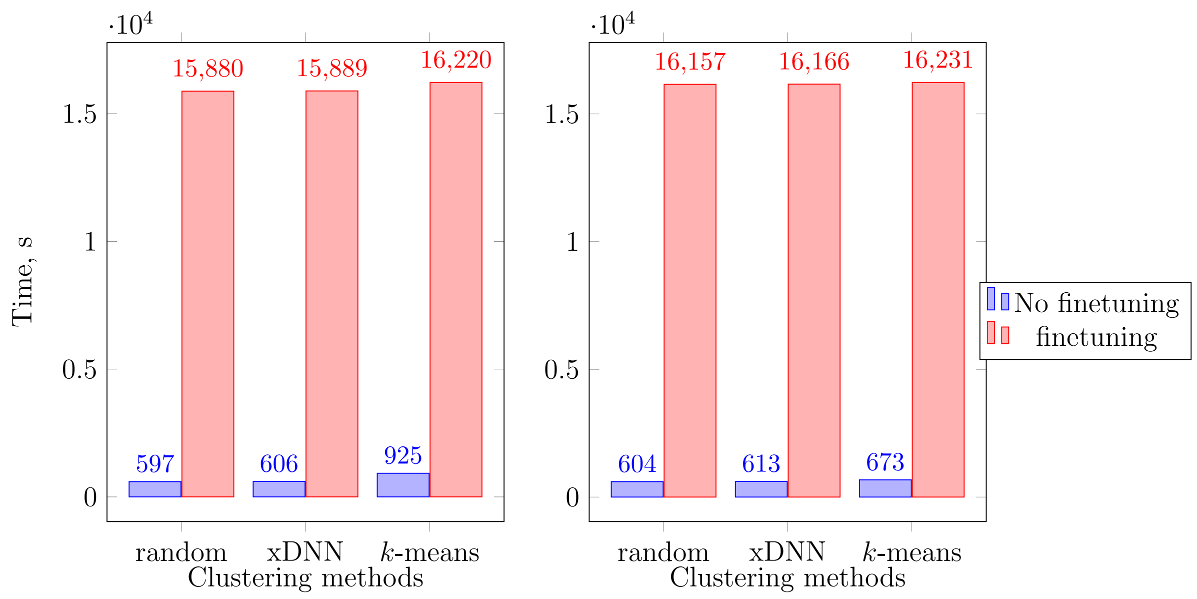


Figure 3.7: Comparison of training time expenditure on CIFAR-10 (left) and CIFAR-100 (right) with and without funetuning (ViT).

less generalisation. In Figure 3.8 (with extra comparison in 3.5.4, Figure 3.18 for ResNet-101), one can see the comparison of the tSNE plots between the finetuned and **non-finetuned** version of the method. While the finetuned method achieves clear separation on this task, using the same features to transfer to another task (from CIFAR-10 to CIFAR-100) leads to sharp decrease in performance (see Figure 3.9).

While for the finetuned backbone, predictably, the results are not far off the standard DL models, they also show no significant difference between different types of prototype selection, including random (in Figure 3.6 it has been demonstrated for CIFAR-10 and CIFAR-100). This can be explained by the previous discussion of Figures 3.18 and 3.8, which suggests that finetuning gives clear separation of features, so the features of the same class stay close. For the **non-finetuned** results, meanwhile, the difference in accuracy between random and non-random prototype selection is drastic, reaching around 24% for VGG16. This finding remains consistent for a number of vision benchmarks. In Figure 3.10 and 3.5.2, one can see that simple k-means prototype selection in the latent space can significantly improve the

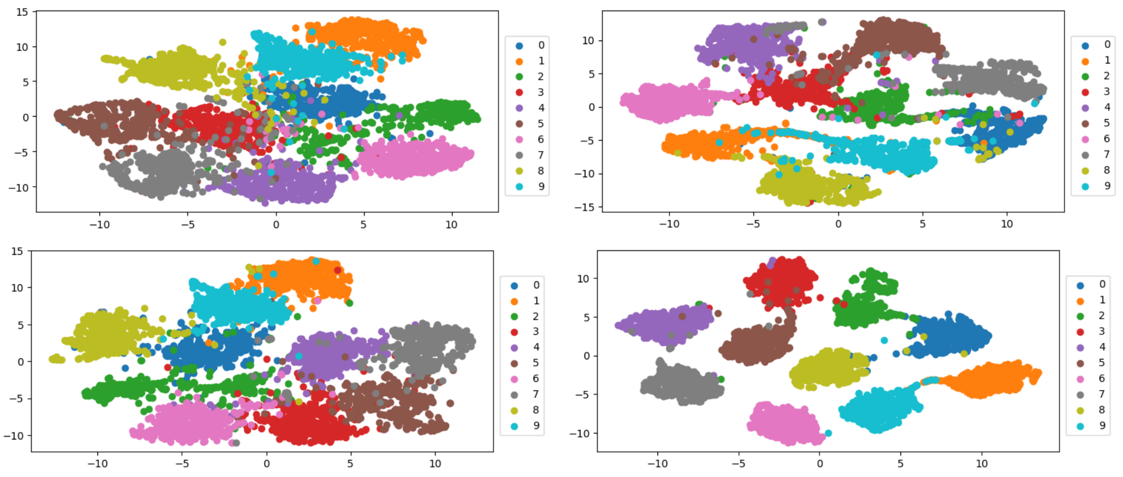


Figure 3.8: tSNE plots for original (top-left) vs finetuned (top-right) features of ViT, k-means prototypes; original (bottom left) vs finetuned (bottom right), ViT, random prototype selection, CIFAR-10.

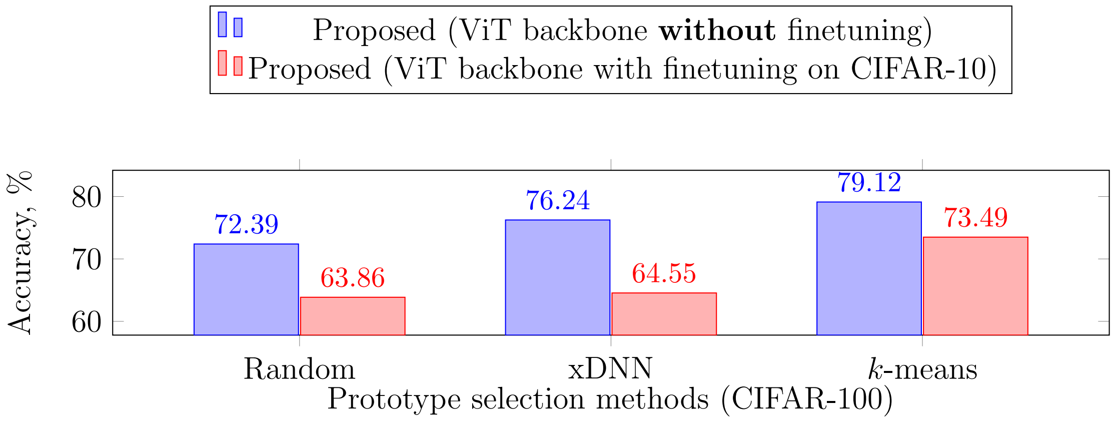


Figure 3.9: Comparison between the model performance on CIFAR-100 **without** finetuning and finetuning on CIFAR-10 for different prototype selection methods.

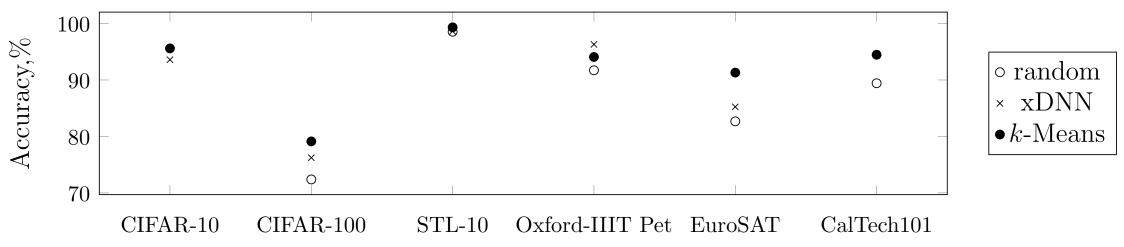
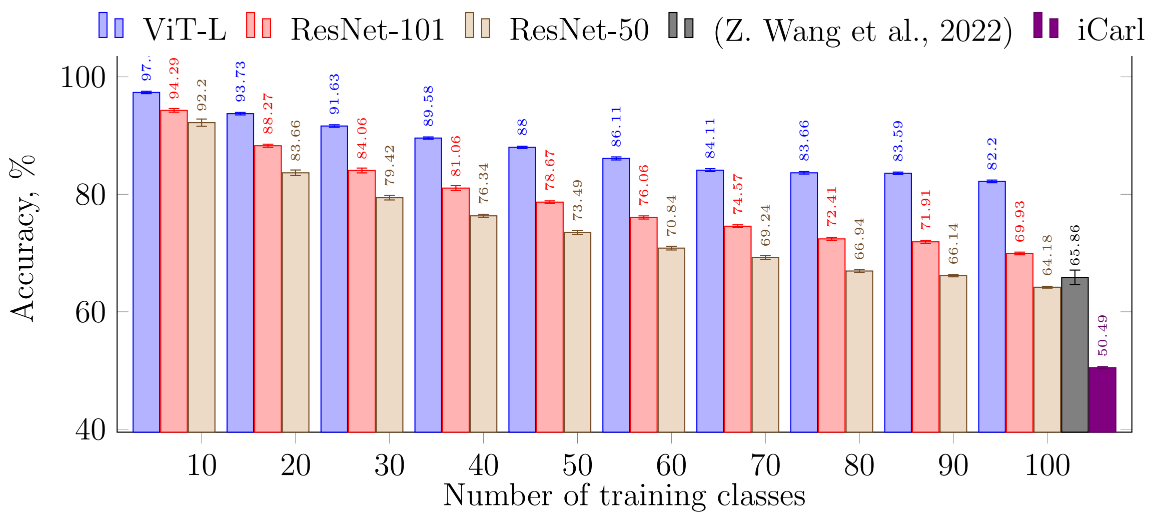


Figure 3.10: Results **without** finetuning for various problems (ViT).

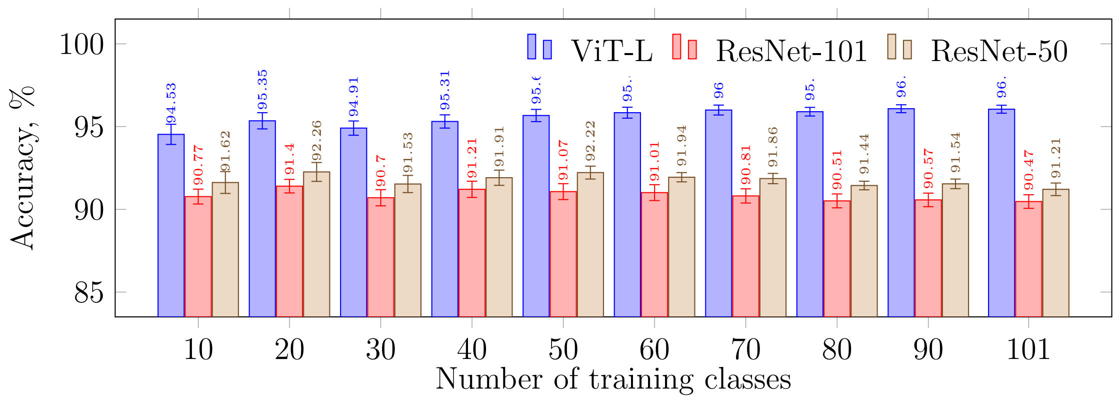
performance; with the increase of the number of prototypes this difference decreases, but is still present.

3.4.3 Continual learning

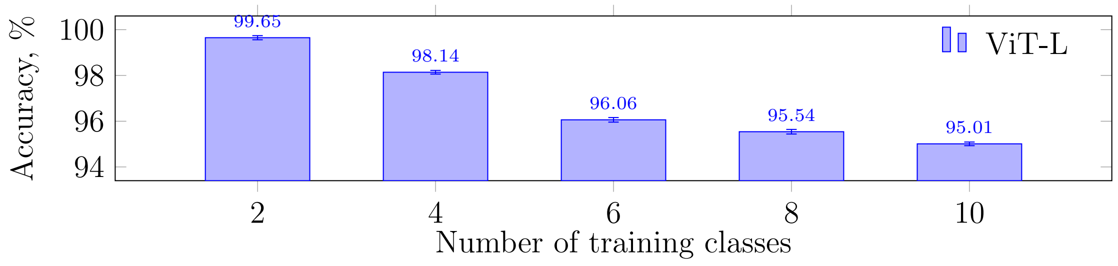
The evidence from the previous sections motivates us to extend the analysis to continual learning problems. Given a much smaller gap between the finetuned and non-finetuned ViT models, can the IDEAL framework **without** finetuning compete with the state-of-the-art class-incremental learning baselines? It turns out the answer is affirmative. We repeat the setting from (Rebuffi et al., 2017) (Section 4, iCIFAR-100 benchmark) using IDEAL without finetuning the latent space of the ViT-L model. The hyperparameters of the proposed methods are given in 3.5.1. This benchmark gradually adds the new classes with a class increment of 10, until it reaches 100 classes. The results, shown in Figure 3.11a, highlight excellent performance of the proposed method when the number of prototypes is set to 10% of data. As one can see in 3.5.3, even much lower number of prototypes, below 1000 or even just 10 per class on average can still lead to competitive results. While we observe $64.18 \pm 0, 0.16$, $69.93 \pm 0.23\%$, 82.20 ± 0.23 for ResNet-50, ResNet-101, and ViT-L respectively, (Z. Wang et al., 2022) reports in its Table 1 for the best performing method for class-incremental learning, based on ViT architecture and contrastive learning, accuracy of just $65.86 \pm 1.24\%$ (with the size of the buffer 1000), while the original benchmark model iCarl (Rebuffi et al., 2017) reaches, according



(a) iCIFAR-100



(b) iCaltech101



(c) iCIFAR10

Figure 3.11: Accuracy of IDEAL in class-incremental learning experiments for different backbones (ViT-L, ResNet-101 and 50); comparison with (Z. Wang et al., 2022) and (Rebuffi et al., 2017).

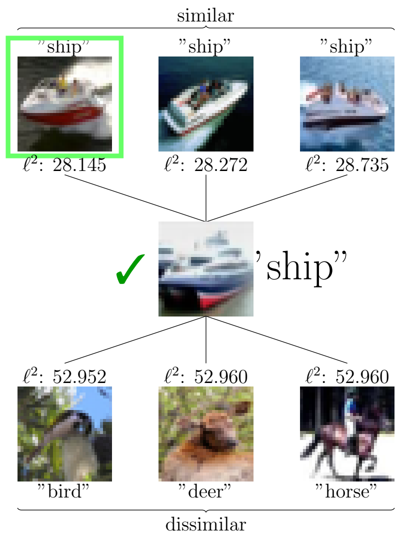


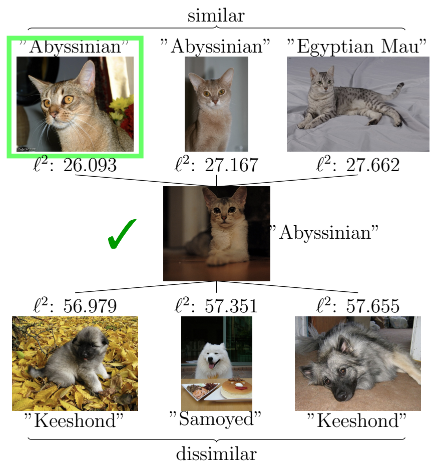
Figure 3.12: Interpreting the predictions of the proposed model (k-means (nearest), CIFAR-10, ViT).

to (Z. Wang et al., 2022), only $50.49 \pm 0.18\%$.

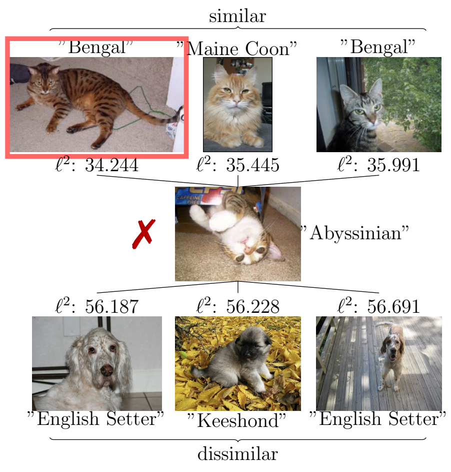
3.4.4 Study of Interpretability

To demonstrate the consistent performance, we expanded iCIFAR-100 protocol to other datasets, namely class-incremental versions of Caltech101 and CIFAR-10, which we refer to as iCaltech101 and iCIFAR-10. Figure 3.11 shows robust performance on iCaltech101 and iCIFAR-10. We use the class increment value of ten (eleven for the last step) and two for iCaltech101 and iCIFAR-10, respectively. We see that for iCaltech101, the model performance changes insignificantly when adding the new classes, and all three datasets demonstrate performance similar to offline classification (see Section 3.4.1).

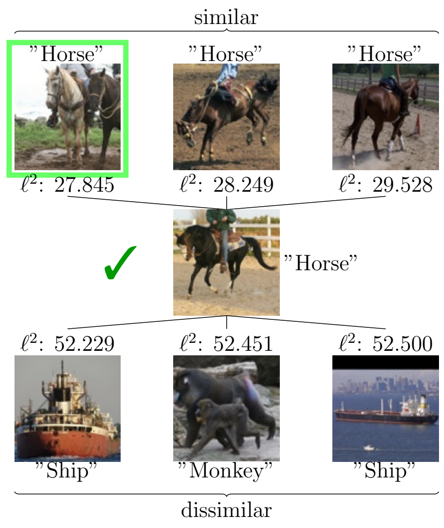
In Figures 3.12 and 3.13, we demonstrate the visual interpretability of the proposed model through both most similar and most dissimilar prototypes. In addition, the results could be interpreted linguistically (see 3.5.5). Figure 3.13 shows a number of quantitative examples for multiple datasets: Caltech101, STL-10, Oxford-IIIT Pets, all corresponding to the non-finetuned feature space scenario



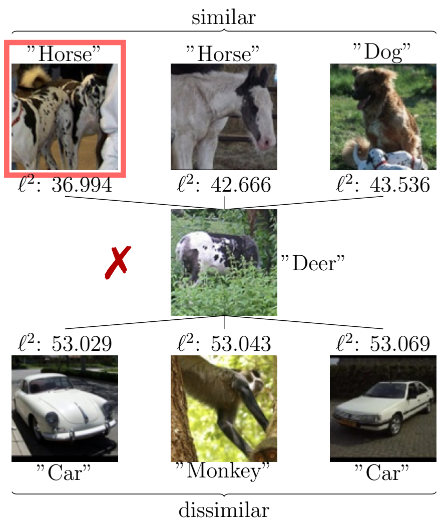
(a) Oxford-IIIT Pet (correct)



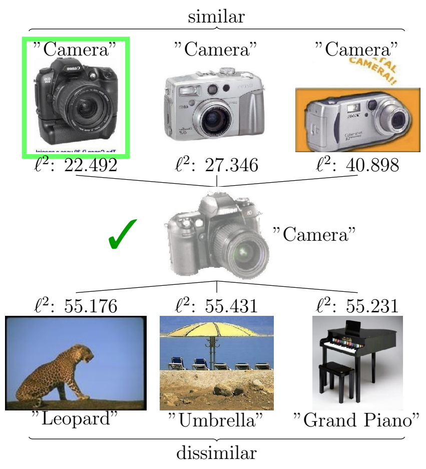
(b) Oxford-IIIT Pet (incorrect)



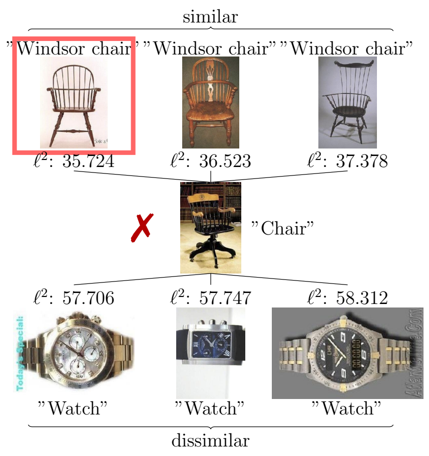
(c) STL-10 (correct)



(d) STL-10 (incorrect)



(e) Caltech101 (correct)



(f) Caltech101 (incorrect)

Figure 3.13: Interpreting the predictions (k-means (nearest), OxfordIIITPets/STL-10/Caltech101, ViT).

according to the experimental setup from 3.5.1. We see that on a range of datasets, without any finetuning, the proposed IDEAL approach provides semantically meaningful interpretations. Furthermore, as there has been no finetuning, the ℓ^2 distances are defined in exactly the same feature space and, hence, can be compared like-for-like between datasets (see Figures 3.13a-3.13f). The experiment in Figure 3.13 provides an additional reason to use our approach **without** finetuning as it demonstrates that the incorrectly classified data tend to have larger distance to the closest prototypes than the correctly classified ones. Finally, Figure 3.14 outlines the evolution of predictions for the class-incremental learning scenario. For the sake of demonstration, we used the same setting as the one for the class-incremental lifelong learning detailed in 3.5.1 and Section 3.4.3, taking CIFAR-10 for class-incremental learning using ViT model with the increment batch of two classes. We trace the best and the worst matching and select middle prototypes (according to the ℓ^2 metric) through the stages of class-incremental learning. For the successful predictions, while the best matching prototypes tend to be constant, the worst matching ones change over time when the class changes.

3.4.5 Impact of confounding on interpretations

The phenomenon of confounding takes its origin in causal modelling and is informally described, as per (Greenland, Pearl, and Robins, 1999), as *'a mixing of effects of extraneous factors (called confounders) with the effects of interest'*. In many real-world scenarios, images contain confounding features, such as watermarks or naturally occurring spurious correlations ('seagulls always appear with the sea on the background'). The challenge for the interpretable models is therefore multi-fold: (1) these models need to be resistant to such confounders, (2) should these confounders interfere with the performance of the model, the model should highlight them in the interpretations.

To model confounding, we use the experimental setup from (Bontempelli et al., 2022), which involves inpainting training images of three out of five selected classes

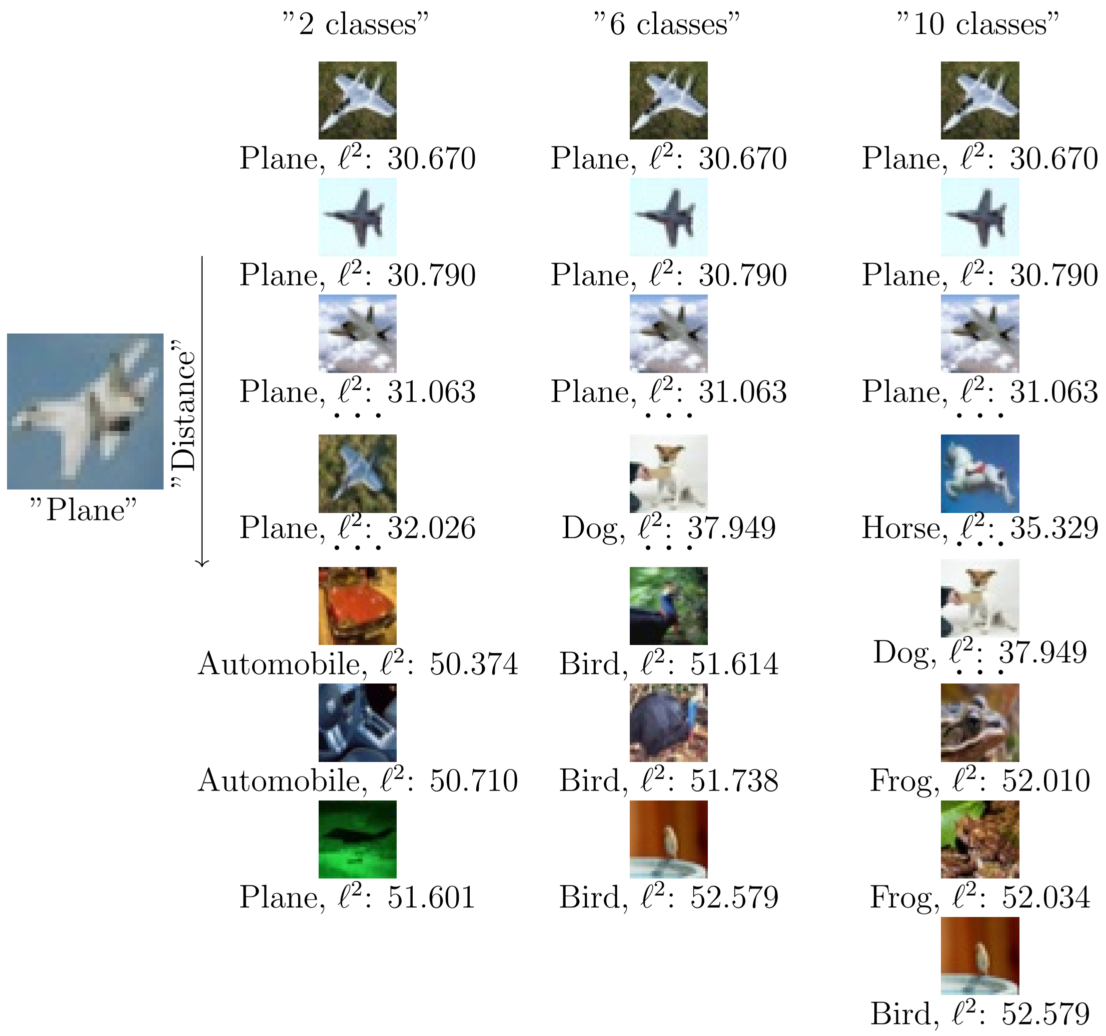


Figure 3.14: iCIFAR-10 class-incremental learning: evolution of prototype ranking.

Feature space	Prototype selection	VGG16	ResNet-50	ViT
Confounded data (Bontempelli et al., 2022)				
Finetuned	N/A, backbone network	73.99 ± 2.91	70.42 ± 2.68	69.06 ± 4.40
Non-finetuned	k-means	$\mathbf{78.52 \pm 1.31}$	$\mathbf{76.68 \pm 1.63}$	$\mathbf{80.70 \pm 2.26}$
Finetuned	k-means	73.19 ± 1.43	67.16 ± 2.25	66.58 ± 5.81
Non-finetuned	k-means (nearest)	64.13 ± 1.37	67.68 ± 0.90	$\mathbf{82.88 \pm 2.17}$
Finetuned	k-means (nearest)	$\mathbf{71.00 \pm 2.92}$	69.03 ± 1.19	73.99 ± 5.19
Original data				
Finetuned	N/A, backbone network	83.66 ± 1.16	83.49 ± 1.22	93.92 ± 1.31
Non-finetuned	k-means	80.01 ± 1.27	80.10 ± 1.66	90.67 ± 1.13
Finetuned	k-means	81.98 ± 1.53	79.38 ± 2.87	92.85 ± 1.70
Non-finetuned	k-means (nearest)	72.11 ± 1.62	72.64 ± 1.87	88.57 ± 0.96
Finetuned	k-means (nearest)	$\mathbf{78.90 \pm 2.77}$	$\mathbf{80.05 \pm 2.64}$	$\mathbf{92.80 \pm 1.77}$

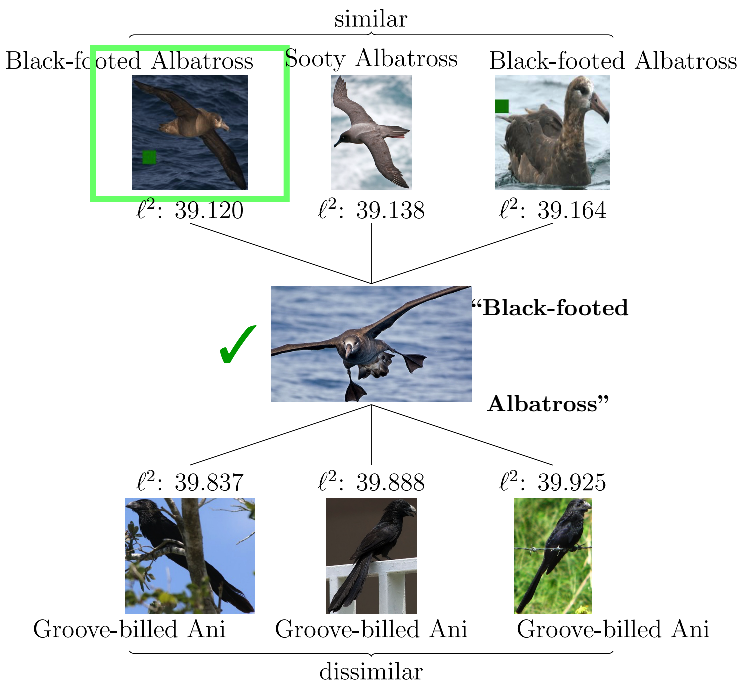
Table 3.1: F1 score comparison for CUB dataset (Wah et al., 2011), %, confidence interval calculated over five runs; all k-means runs are for 10% (15) clusters/prototypes; the better results within its category are highlighted in bold, taking into account the confidence interval. While for the original data finetuning has strong performance benefits, non-finetuned model has an edge over the finetuned one for all architectures; for k-means (nearest) the non-finetuned model still performs clearly better with ViT architecture than the finetuned counterpart.

of the CUB dataset with geometric figures (squares) which correlate with, but not caused by, the original data (e.g., every image of the `Crested Auklet` class is marked in the training data with a blue square). In Table 3.1, we compare the experimental results between the original (Wah et al., 2011) and confounded (Bontempelli et al., 2022) CUB dataset. We use the same original pre-trained feature spaces as stated in 3.5.1. The finetuned spaces are obtained through finetuning on confounded CUB data from (Bontempelli et al., 2022) for 15 epochs.

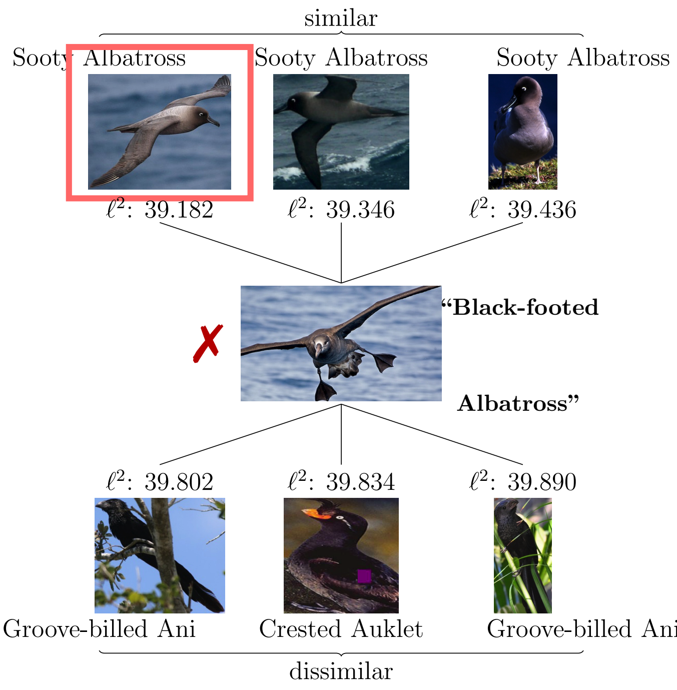
The results in Table 3.1 demonstrate a clear advantage of models **without finetuning** on the confounded dataset for both k-means and k-means (nearest), in the case of ViT. Such a gap, however, is much narrower for VGG-16 and ResNet-50. It is consistent with the results in Section 3.4.1 demonstrating the larger finetuning performance advantage for these models compared with the ViT. Furthermore, k-means (nearest) does not show improvements over finetuning in a k-means (nearest) scenario for VGG-16 and ResNet-50, in a stark contrast with the ViT results.

We demonstrate the interpretations for the confounding experiment for the ViT model in Figure 3.15. While the model **without** finetuning successfully predicts the correct confounded class, `Black-footed Albatross`, the finetuned model fails at this scenario and predicts a similar class `Sooty Albatross`, which does not contain the confounder mark. On the other hand, the finetuned model performs similarly or better on the original (not confounded) data. These results further build upon the hypothesis from Question 2 and demonstrate that the use of the proposed framework can help address the phenomenon of confounding.

Figure 3.16 gives an intuition behind improvements in performance of the non-finetuned model in a case of the ViT model. It shows that in the finetuned scenario, confounded training data stands further away from the testing data, which does not contain the confounder mark. In the scenario without finetuning, this does not happen and the training and testing data are matched closer, even in the presence of a confounder. The Sinkhorn approximation of Wasserstein-2 distance has been implemented using `SamplesLoss(loss='sinkhorn', p=2, blur=1e-5)` function from the `geomloss` python library.

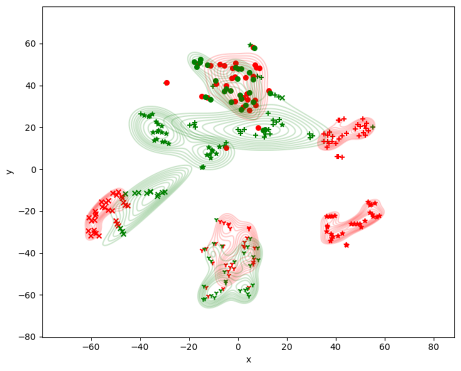


(a) Non-finetuned model interpretation

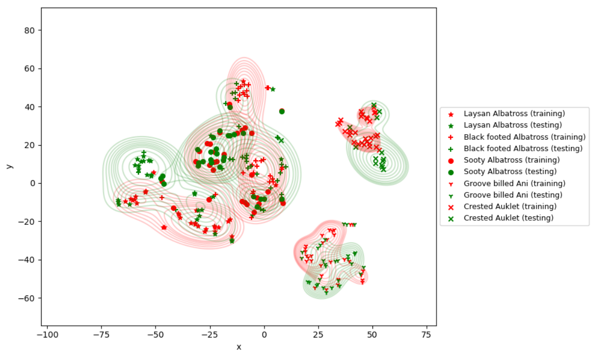


(b) Finetuned model interpretation

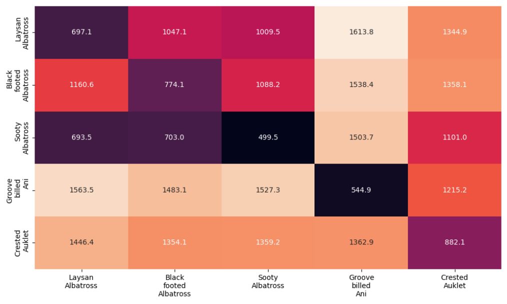
Figure 3.15: Comparing the interpretations of the non-finetuned and finetuned model with confounding on confounded CUB (Bontempelli et al., 2022) dataset.



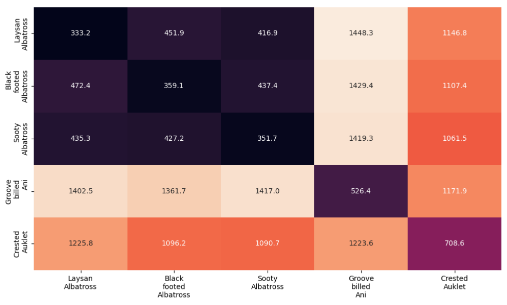
(a) tSNE plot: finetuned model. The clean testing data from confounded classes are better aligned with similar clean classes than with confounded ones



(b) tSNE plot: model without finetuning. The models show better distribution matching, including for similar classes such as different species of Albatross. In both tSNE plots, the density estimation is shown for tSNE embedded points



(c) Wasserstein-2 distance heatmap (Sinkhorn approximation): finetuned model, training (vertical) to testing (horizontal) distance. *Black-footed albatross* (testing distribution) is closer to a non-confounded *Sooty albatross* training distribution



(d) Wasserstein-2 distance heatmap (Sinkhorn approximation): model without finetuning. In contrast to the finetuned model, the similar classes' distributions are close yet closely match between training and testing classes.

Figure 3.16: Intuitive explanation behind better performance of non-finetuned model.

3.5 Additional Experimental Analyses and Results

3.5.1 Experimental setup

In this chapter, all the experiments were conducted in PyTorch 2.0.0. The pre-trained models used in these experiments were obtained from TorchVision [1] while the finetuned models have been obtained from three different sources:

1. *Models that come from MMPreTrain* [2]. Specifically, ResNet50 and ResNet101 finetuned on the CIFAR-10, and ResNet 50 finetuned on CIFAR-100.

2. *finetuned TorchVision models.* finetuning was conducted by continuing the EBP across all network layers. Such models include VGG-16 and Vision Transformer (ViT) finetuned on CIFAR-10, as well as ResNet101, VGG-16, and ViT finetuned on CIFAR-100. For ResNet101 and VGG-16 models, we ran the training for 200 epochs, while the Vision Transformer models were trained for 10 epochs. The Stochastic Gradient Descent (SGD) optimizer was employed for all models, with a learning rate of 0.0005 and a momentum value of 0.9.

3. *Linearly finetuned TorchVision models.* In such case, only the linear classifier was trained and all the remaining layers of the network were fixed. For these models, we conducted training for 200 epochs for ResNet50, ResNet101, and VGG16, and 25 epochs for the ViT models. We adopted the Stochastic Gradient Descent (SGD) optimizer, with a learning rate of 0.001 and a momentum parameter set at 0.9.

We utilized k-means clustering and random selection methods, setting the number of prototypes for each class at 10% of the training data for the corresponding

[1]https://pytorch.org/vision/main/models.html
[2]https://github.com/open-mmlab/mmpretrain

classes. Besides, we also set it to 12 per class and conducted experiments for ResNet50, ResNet101, and VGG-16 on CIFAR-10 and CIFAR-100 datasets, enabling us to evaluate the impact of varying the number of prototypes.

For ELM online clustering method, we experimented with varying radius values for each specific dataset and backbone network. We selected a radius value that would maintain the number of prototypes within the range of 0-20% of the training data. In the experiments without finetuning on the CIFAR-10 dataset, we set the radius to 8, 10, 19, and 12 for ResNet50, ResNet101, VGG-16, and Vision Transformer (ViT) models respectively. The radius was adjusted to 8, 11, 19, and 12 for these models when conducting the same tasks without finetuning on CIFAR-100. For STL10, Oxford-IIIT Pets, EuroSAT, and CalTech101 datasets, the radius was set to 13 across all ELM experiments. In contrast, the xDNN model did not require hyper-parameter settings as it is inherently a parameter-free model.

We performed all experiments for Sections 3.4.1 and 3.4.3 5 times and report mean values and standard deviations for our results, with the exception of the finetuned backbone models where we just performed finetuning once (or sourced finetuned models as detailed above).

For the class-incremental learning experiments in Section 3.4.3, we use the k-means clustering method for prototype selection and set the number of prototypes to 10% of the training data. Each time we add the incremental classes, the existing prototypes are unchanged, and the algorithm adds the prototypes for the new classes to the existing prototypes. All the experiments were executed 10 times to allow a robust comparison with the benchmark results.

To ensure a consistent and stable training environment, for every experiment we used a single NVIDIA V100 GPU from a cluster.

3.5.2 Complete experimental results

Tables 3.2-3.9 contain extended experimental results for multiple benchmarks and feature extractors. These results further support the findings of the chapter 3.

Table 3.2 demonstrates the data behind Figure 3.4 of the chapter 3. It also highlights the performance of the k-means model on ViT-L latent space, when the nearest real training data point to the k-means cluster centre is selected (labelled as k-means (nearest)). One can also see that even with the small number of selected prototypes, the algorithm delivers competitive performance without finetuning.

Table 3.3 compares different latent spaces and gives the number of free (optimised) parameters for the scenario of finetuning of the models. With a small additional number of parameters, which is the number of possible prototypes, one can transform the opaque architectures into ones interpretable through proximity and similarity to prototypes within the latent space (this is highlighted in the interpretability column).

Tables 3.4-3.9 repeat the same analysis, expanded from Figure 5 of the chapter 3 for different datasets. The results show remarkable consistency with the previous conclusions and further back up the claims of generalisation to different classification tasks.

3.5.3 Sensitivity analysis for the number of prototypes

Figure 3.17 further backs up the previous evidence that even with a small number of prototypes, the accuracy is still high. It shows, however, that there is a trade-off between the number of prototypes and accuracy. It also shows, that after a few hundred prototypes per class on CIFAR-10 and CIFAR-100 tasks, the performance does not increase and may even slightly decrease, indicating saturation.

3.5.4 Additional results for the demonstration of overfitting

In Figure 3.18, we show that the evidence of overfitting presented in Section 3.4.2 for the ViT backbone also extends to the other models such as ResNet-101.

FE	method	accuracy (%)	#prototypes	time, s
ResNet50	random	65.55 ± 1.93	120(0.24%)	85
	random	80.40 ± 0.37	5,000(10%)	85
	ELM	81.17 ± 0.04	5,500(11%)	365
	xDNN	81.44 ± 0.33	115(0.23%)	103
	k-means	84.12 ± 0.19	120(0.24%)	201
	k-means	$\mathbf{86.65 \pm 0.15}$	5,000(10%)	1,138
Resnet101	random	78.08 ± 1.38	120(0.24%)	129
	random	87.66 ± 0.25	5,000(10%)	129
	ELM	88.22 ± 0.09	7,154(14.31%)	524
	xDNN	88.13 ± 0.42	118(0.24%)	145
	k-means	90.19 ± 0.15	120(0.24%)	245
	k-means	$\mathbf{91.50 \pm 0.07}$	5,000(10%)	1,194
VGG-16	random	50.13 ± 2.37	120(0.24%)	95
	random	65.06 ± 0.32	5,000(10%)	95
	ELM	72.31 ± 0.08	1,762(3.52%)	215
	xDNN	70.03 ± 0.96	103(0.21%)	132
	k-means	74.48 ± 0.16	120(0.24%	346
	k-means	$\mathbf{75.94 \pm 0.15}$	5,000(10%)	2,362
ViT	random	93.23 ± 0.11	5,000(10%)	597
	ELM	90.61 ± 0.14	6,685(13.37%)	889
	xDNN	93.59 ± 0.12	112(0.2%)	606
	k-means	$\mathbf{95.59 \pm 0.08}$	5,000(10%)	925
ViT-L	k-means	$\mathbf{96.48 \pm 0.05}$	5,000(10%)	4,375
	k-means (nearest)	95.62 ± 0.07	5,000(10%)	4,352

Table 3.2: CIFAR-10 classification task comparison for the case of no finetuning of the feature extractor.

FE	method	accuracy (%)	#parameters	#prototypes	time, s	interp.
ResNet50	ResNet50	95.55 (80.71*)	$\sim 25M$ ($20K$)		$36,360$ ($13,122^*$)	✗
	random	94.92 ± 0.02	$\sim 25M + 50K$	120(0.24%)	$36,360 + 24$	✓
	random	95.32 ± 0.09	$\sim 25M + 50K$	5,000(10%)	$36,360 + 24$	✓
	xDNN	95.32 ± 0.12	$\sim 25M + 50K$	111(0.22%)	$36,360 + 43$	✓
	k-means	94.91 ± 0.14	$\sim 25M + 50K$	120(0.24%)	$36,360 + 208$	✓
	k-means	95.50 ± 0.06	$\sim 25M + 50K$	5,000(10%)	$36,360 + 1,288$	✓
ResNet101	Resnet101	95.58 (84.44*)	$\sim 44M$ ($20K$)		$36,360$	✗
	random	95.47 ± 0.06	$\sim 44M + 50K$	120(0.24%)	$36,360 + 37$	✓
	random	95.51 ± 0.01	$\sim 44M + 50K$	5,000(10%)	$36,360 + 37$	✓
	xDNN	95.50 ± 0.10	$\sim 44M + 50K$	107(0.21%)	$36,360 + 54$	✓
	k-means	95.55 ± 0.03	$\sim 44M + 50K$	120(0.24%)	$36,360 + 231$	✓
	k-means	95.51 ± 0.04	$\sim 44M + 50K$	5,000(10%)	$36,360 + 1,357$	✓
VGG-16	VGG-16	92.26 (83.71*)	$\sim 138M$ ($41K$)		$40,810$	✗
	random	87.48 ± 0.72	$\sim 138M + 50K$	120(0.24%)	$40,810 + 94$	✓
	random	90.86 ± 0.19	$\sim 138M + 50K$	5,000(10%)	$40,810 + 94$	✓
	xDNN	91.42 ± 0.25	$\sim 138M + 50K$	102(0.20%)	$40,810 + 123$	✓
	k-means	92.24 ± 0.10	$\sim 138M + 50K$	120(0.24%)	$40,810 + 369$	✓
	k-means	92.55 ± 0.16	$\sim 138M + 50K$	5,000(10%)	$40,810 + 2,408$	✓
ViT	ViT	98.51 (96.08*)	$\sim 86M$ ($8K$)		$15,282$ ($15,565^*$)	✗
	random	98.56 ± 0.02	$\sim 86M + 50K$	5,000(10%)	$15,282 + 598$	✓
	xDNN	98.00 ± 0.14	$\sim 86M + 50K$	117(0.23%)	$15,282 + 607$	✓
	k-means	98.53 ± 0.04	$\sim 86M + 50K$	5,000(10%)	$15,282 + 938$	✓

Table 3.3: CIFAR-10 classification task comparison for the case of finetuned models ($*$ denotes linear finetuning of the DL model).

FE	method	accuracy (%)	#prototypes	time, s
ResNet50	random	41.66 ± 0.74	$1,200(2.4\%)$	82
	random	54.37 ± 0.43	$10,000(20\%)$	82
	ELM	57.94 ± 0.11	$7,524(15.05\%)$	129
	xDNN	58.25 ± 0.64	$884(1.77\%)$	98
	k-means	62.67 ± 0.26	$1,200(2.4\%)$	124
	k-means	64.07 ± 0.37	$10,000(20\%)$	258
ResNet101	random	50.25 ± 0.71	$1,200(2.4\%)$	128
	random	61.90 ± 0.41	$10,000(20\%)$	128
	ELM	64.42 ± 0.12	$4,685(9.37\%)$	161
	xDNN	64.60 ± 0.39	$878(1.76\%)$	143
	k-means	68.59 ± 0.40	$1,200(2.4\%)$	170
	k-means	70.04 ± 0.12	$10,000(20\%)$	310
VGG16	random	26.16 ± 0.24	$1,200(2.4\%)$	94
	random	37.74 ± 0.48	$10,000(20\%)$	94
	ELM	48.53 ± 0.05	$2,878(5.76\%)$	122
	xDNN	47.78 ± 0.41	871 (1.74%)	119
	k-means	51.99 ± 0.24	$1,200(2.4\%)$	175
	k-means	52.55 ± 0.27	$1,200(2.4\%)$	437
ViT	random	72.39 ± 0.21	$10,000(20\%)$	604
	ELM	69.94 ± 0.06	$8,828(17.66\%)$	642
	xDNN	76.24 ± 0.24	$830(1.66\%)$	613
	k-means	79.12 ± 0.28	$10,000(20\%)$	673
ViT-L	k-means	$\mathbf{82.18 \pm 0.14}$	$10,000(20\%)$	$3,905$
	k-means (nearest)	78.75 ± 0.29	$10,000(20\%)$	$3,909$

Table 3.4: CIFAR-100 classification task comparison for the case of no finetuning of the feature extractor.

FE	method	accuracy (%)	#parameters	#prototypes	time, s	interp.
ResNet50	ResNet50	79.70 (56.39*)	$\sim 25M$ $(205K)$		$36,360(13,003^*)$	✗
	random	78.94 ± 0.17	$\sim 25M + 50K$	$1,200(2.4\%)$	$36,360 + 28$	✓
	random	79.52 ± 0.17	$\sim 25M + 50K$	$10,000(20\%)$	$36,360 + 28$	✓
	xDNN	79.75 ± 0.12	$\sim 25M + 50K$	$859(1.72\%)$	$36,360 + 45$	✓
	k-means	79.84 ± 0.07	$\sim 25M + 50K$	$1,200(2.4\%)$	36,360+82	✓
	k-means	79.77 ± 0.07	$\sim 25M + 50K$	$10,000(20\%)$	36,360+219	✓
ResNet101	ResNet50	84.38 (63.18*)	$\sim 44M$ $(205K)$		$45,619(18,955^*)$	✗
	random	82.26 ± 0.15	$\sim 44M + 50K$	$1,200(2.4\%)$	$45,619 + 175$	✓
	random	80.75 ± 0.19	$\sim 44M + 50K$	$10,000(20\%)$	$45,619 + 175$	✓
	xDNN	81.13 ± 0.16	$\sim 44M + 50K$	$831(1.66\%)$	$45,619 + 191$	✓
	k-means	83.03 ± 0.06	$\sim 44M + 50K$	$1,200(2.4\%)$	$45,619 + 220$	✓
	k-means	83.14 ± 0.19	$\sim 44M + 50K$	$10,000(20\%)$	$45,619 + 439$	✓
VGG-16	VGG-16	75.08 (62.74*)	$\sim 138M$ $(410K)$		$41,038(17,098^*)$	✗
	random	53.83 ± 0.91	$\sim 138M + 50K$	$1,200(2.4\%)$	$41,038 + 92$	✓
	random	64.17 ± 0.36	$\sim 138M + 50K$	$10,000(20\%)$	$41,038 + 92$	✓
	xDNN	72.63 ± 0.11	$\sim 138M + 50K$	$907(1.81\%)$	$41,038 + 120$	✓
	k-means	73.83 ± 0.16	$\sim 138M + 50K$	$1,200(2.4\%)$	$41,038 + 199$	✓
	k-means	73.73 ± 0.23	$\sim 138M + 50K$	$10,000(20\%)$	$41,038 + 460$	✓
ViT	ViT	90.29(82.79*)	$\sim 86M$ $(77K)$		$15,536(15,423^*)$	✗
	random	89.90 ± 0.10	$\sim 86M + 50K$	$10,000(20\%)$	$15,536 + 621$	✓
	xDNN	89.17 ± 0.18	$\sim 86M + 50K$	$809(1.61\%)$	$15,536 + 630$	✓
	k-means	90.48 ± 0.05	$\sim 86M + 50K$	$10,000(20\%)$	$15,536 + 695$	✓

Table 3.5: CIFAR-100 classification task comparison for the case of finetuned models ($*$ denotes linear finetuning of the DL model).

FE	method	accuracy (%)	#prototypes	time, s
ViT	random	98.55 ± 0.09	500(10%)	61
	ELM	95.27 ± 0.03	271(5.42%)	63
	xDNN	98.63 ± 0.12	84(1.68%)	62
	k-means	99.32 ± 0.03	500(10%)	65
ViT-L	k-means	99.71 ± 0.02	500(10%)	377
	k-means(nearest)	99.56 ± 0.05	500(10%)	377

Table 3.6: STL10 classification task comparison for the case of no finetuning (linear finetuning of the ViT gives 98.97%).

FE	method	accuracy (%)	#prototypes	time, s
ViT	random	90.82 ± 0.53	365(9.92%)	48
	ELM	90.85 ± 0.03	122(3.32%)	49
	xDNN	96.30 ± 0.23	239(6.49%)	49
	k-means	94.07 ± 0.20	365(9.92%)	50
ViT-L	k-means	95.78 ± 0.19	365(9.92%)	279
	k-means (nearest)	94.76 ± 0.30	740(9.92%)	279

Table 3.7: OxfordIIITPets classification task comparison for the case of no finetuning (linear finetuning of ViT gives 94.41%).

FE	method	accuracy (%)	#prototypes	time, s
VIT	random	82.67 ± 0.54	$2,154(9.97\%)$	266
	ELM	83.69 ± 0.01	$528(2.44\%)$	277
	xDNN	85.24 ± 1.05	$102(0.47\%)$	269
	k-means	91.30 ± 0.16	$2,154(9.97\%)$	330
VIT-L	k-means	88.93 ± 0.22	$2,154(9.97\%)$	1685
	k-means(nearest)	83.97 ± 0.16	$2,154(9.97\%)$	1685

Table 3.8: EuroSAT classification task comparison for the case of no finetuning (linear finetuning gives 95.17%).

FE	method	accuracy (%)	#prototypes	time, s
VIT	random	89.42 ± 0.32	$649(9.35\%)$	96
	ELM	91.12 ± 0.07	$516(7.43\%)$	97
	xDNN	94.61 ± 0.94	$579(8.34\%)$	97
	k-means	94.46 ± 0.44	$649(9.35\%)$	99
VIT-L	k-means	96.08 ± 0.34	$649(9.35\%)$	515
	k-means (nearest)	93.74 ± 0.42	$649(9.35\%)$	517

Table 3.9: CalTech101 classification task comparison (linear finetuning gives 96.26%).

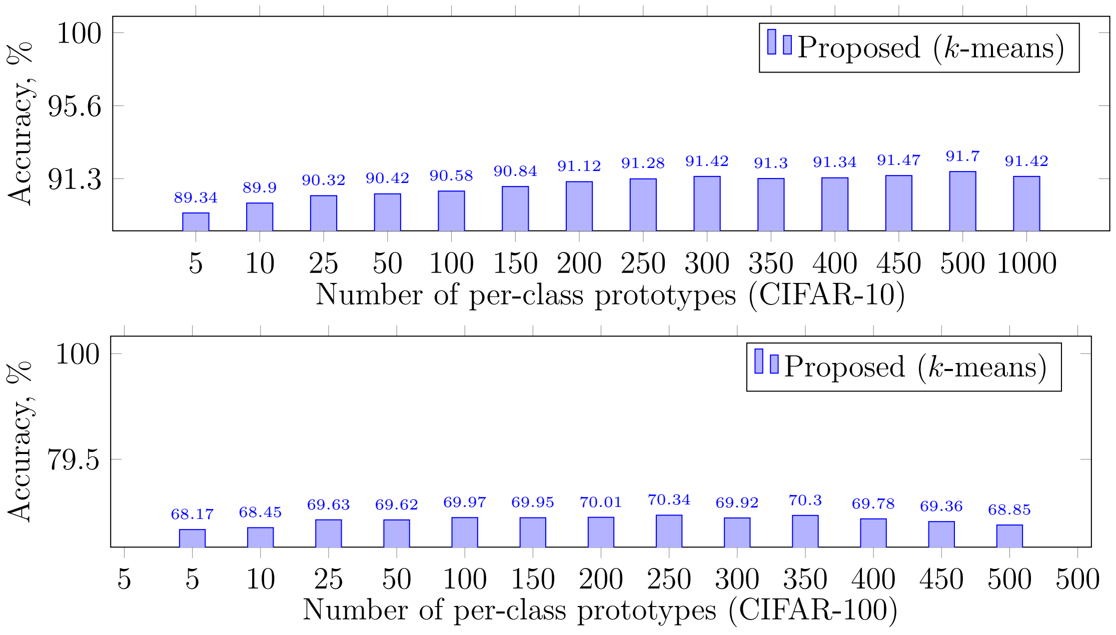


Figure 3.17: Accuracy sensitivity to the number of per-class prototypes (k-means, ResNet101, no finetuning).

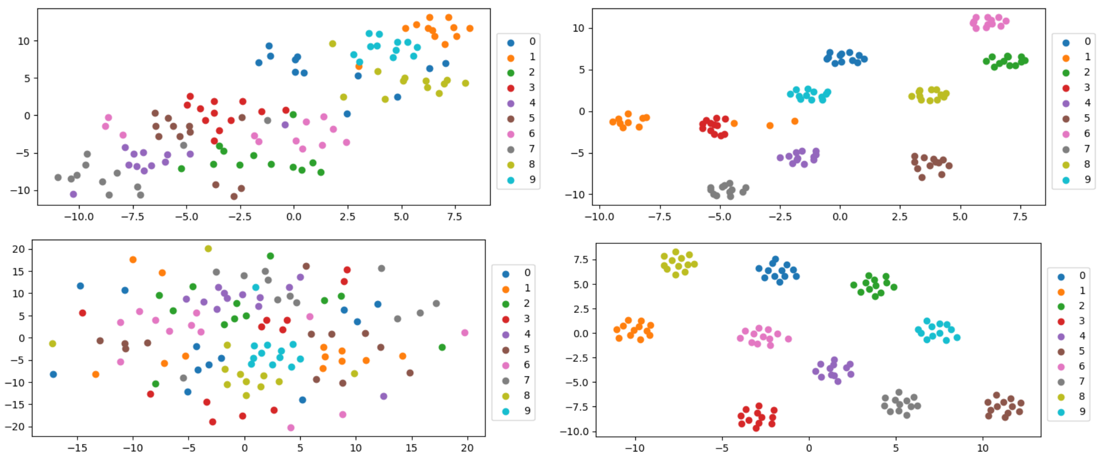


Figure 3.18: tSNE plots for original (top-left) vs finetuned (top-right) features of ResNet101, k-means prototypes; original (bottom left) vs finetuned (bottom right), ResNet101, random prototype selection, CIFAR-10.

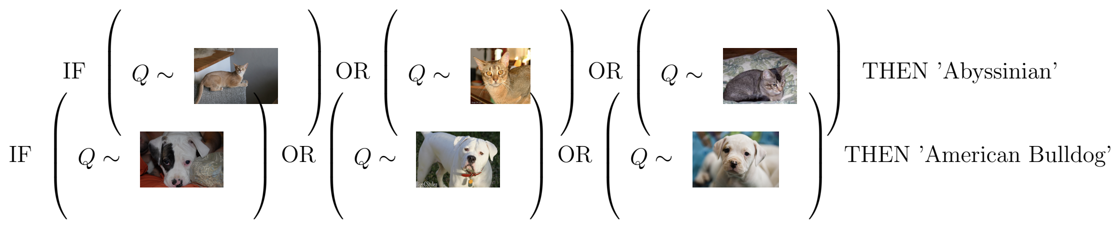


Figure 3.19: An example of symbolic decision rules (OxfordIIITPets), Q denotes the query image.

3.5.5 Linguistic interpretability of the proposed framework outputs

To back up interpretability claim, we present two additional interpretability scenarios complementing the one in Section 3.4.4.

First, we show the symbolic decision rules in Figure 3.19. These symbolic rules are created using ViT-L backbone, with the prototypes selected using the nearest real image to k-means cluster centroids, in a no-finetuning scenario for OxfordIIITPets dataset.

Second, in Figure 3.20 we show how the overall pipeline of the proposed method can be summarised in interpretable-through-prototypes fashion. We show the normalised distance obtained through dividing by the sum of distances to all prototypes. This is to improve the perception and give relative, bound between 0 and 1, numbers for the prototype images.

3.6 Broader Impact Statement

The IDEAL framework, proposed in this chapter, goes beyond the paradigm of first training and then finetuning complex models to the new tasks, which is standard for the field, where both these stages of the approach use expensive GPU compute to improve the model performance. We show that contemporary architectures, trained

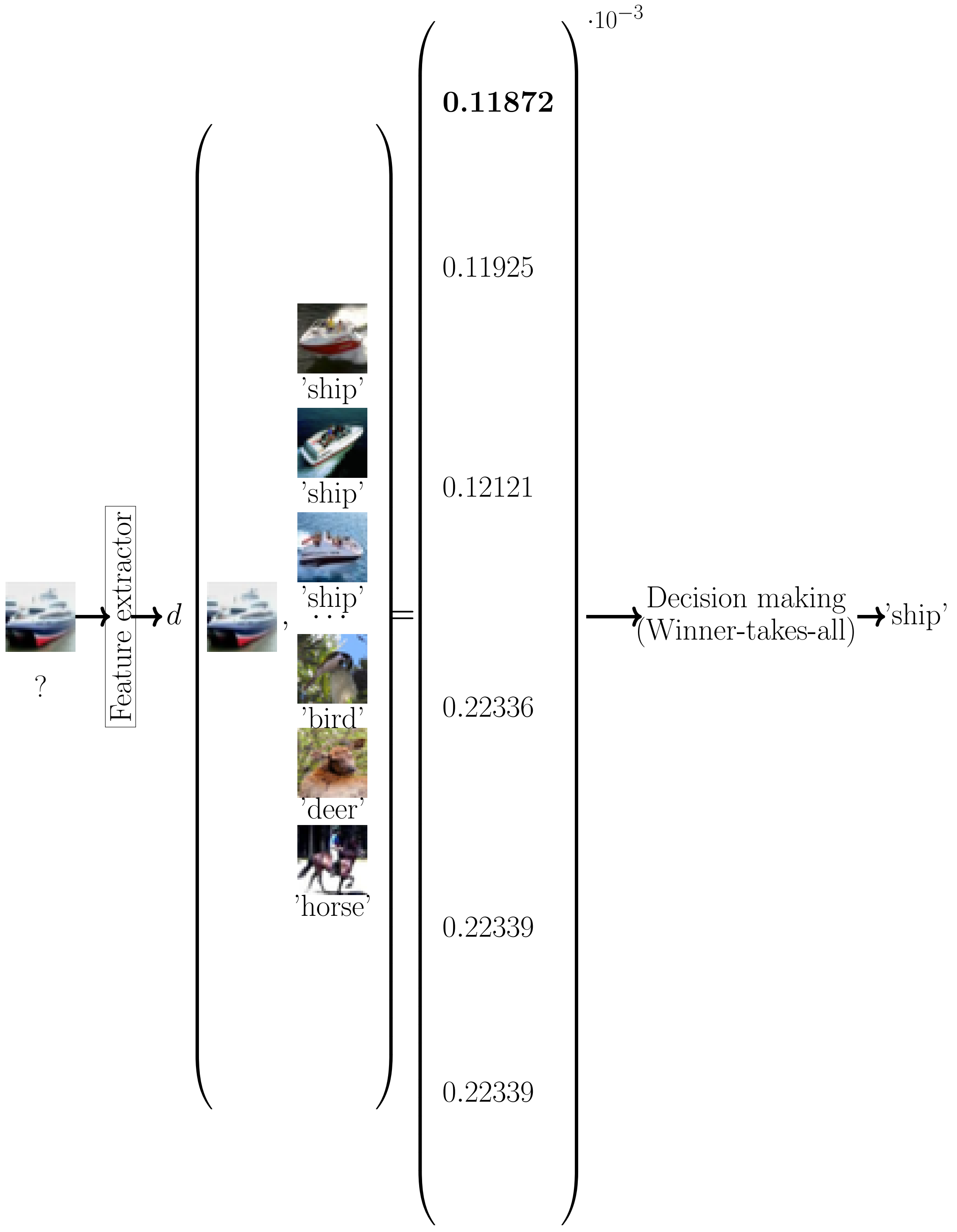


Figure 3.20: Interpreting the model predictions (k-means (nearest), 500 clusters per class, CIFAR-10, ViT).

with extensive datasets, can deliver performance comparable to task-level finetuning, in a class-incremental learning setting. This can deliver a profound impact on democratisation of high-performance machine learning models and implementation on Edge devices, on board of autonomous vehicles, as well as address important problems of environmental sustainability by avoiding using much energy to train and finetune new latent representations, providing instead a way to re-use existing models. Furthermore, the proposed framework can help define a benchmark on how deep-learning latent representations generalise to new tasks.

This approach also naturally extends to task- and potentially, domain-incremental learning, enabling learning new concepts. It demonstrates that with large and complex enough latent spaces, relatively simple strategies of prototype selection, such as clustering, can deliver results comparable with the state-of-the-art in a fraction of the time and computational efforts. Importantly, unlike most of the state-of-the-art approaches, as described in the Background and Related Work section of this thesis, the proposed framework directly provides interpretability in a linguistic and visual form and provides improved resistance to spurious correlations (confounding bias) in input features.

However, while this study can help advance transparency and trustworthiness of the machine learning models, one needs to duly take into account considerations of privacy and security risks pertinent to deep-learning feature spaces and prototype-based learning. In many cases, such as, notably, for medical applications, there may be a need in preserving the privacy of the prototypes and the training data, as exposing prototypes to the users may be unethical or illegal (Lucieri, Dengel, and Ahmed, 2023). Deep-learning models' latent spaces themselves, as well as the data they are trained upon, may be biased or unfair (Birhane et al., 2023). Another risk is a potential for adversarial attacks (Biggio et al., 2013), affecting either feature space representation or the distances between prototypes.

3.7 Conclusion

Our work shows that interpretable, prototype-based models over the latent spaces of ViT models **without** finetuning, learnt on large generic datasets, work surprisingly well in a number of scenarios. In an extensive set of experiments, we find that:

- *Contemporary ViT models drastically narrow the gap between the finetuned and non-finetuned models,* making it possible to avoid finetuning altogether and still have competitive results on a number of benchmarks. To give an example, for the VGG-16 backbone, the accuracy difference between the best-performing finetuned and non-finetuned scenarios on CIFAR-10 is 16.61%. The situation is drastically different for the ViT backbone, where this difference is just 2.94%.

- *The findings in the previous paragraph indicate that without finetuning, we can circumvent the problem of catastrophic forgetting in class-incremental learning.* If the models can achieve competitive performance even **without** finetuning, one can use this advantage to solve a number of problems of lifelong learning without iterative updates and, hence, catastrophic forgetting. The experimental results show the strong empirical advantage of such an approach, allowing to achieve, using a ViT-L backbone, a lead of 16.34% on a well-known iCIFAR-100 benchmark.

- *The IDEAL framework, proposed in this chapter, allows for interpretations* through similarities in the latent feature space, which is not only comparable within one dataset but also between the datasets. We find that the closest prototypes in case of misclassification tend to be further away from the input, and using qualitative analysis, we demonstrate how the IDEAL framework allows us to interpret the decision making process in both offline and class-incremental learning scenarios.

- *Finetuning results in consistently inferior performance when compared to non-*

finetuned models in the face of confounding bias. Our initial findings quantify the margin of feature space overfitting in a simple experiment, showing that the ViT model **without** finetuning has 5.63% advantage on CIFAR-100 over the model finetuned on CIFAR-10. We then build upon this observation to show, quantitatively and qualitatively, how the models **without** finetuning outperform the purpose-finetuned counterparts on confounded data. Notably, the model with ViT backbone **without** finetuning achieves 14.1% lead over the finetuned model on the confounded CUB dataset with prototypes selected using k-means clustering.

Despite these promising results, our approach has limitations. In particular, this study focuses on the final latent representations and does not analyse the intermediate layers of the common neural network architectures. Future work should consider addressing this issue to improve our understanding of models' inference at a granular level.

Chapter 4

An Interpretable Deep Semantic Segmentation (IDSS) Method

4.1 IDSS architecture

This research focused on developing a new interpretable deep semantic segmentation (IDSS) method, which is based on prototypes. It considers that each class can be divided into different sub-classes within the latent feature space. The mini-batch K-means clustering method is utilized to find the centroid of each subclass, thus generating the prototypes within the latent feature space. Meanwhile, the averaging operations were performed within the raw feature space, obtaining the corresponding mean value (prototypes). As a result, all the decisions about the re-allocations of points to clusters are made based on the latent feature representation; however, each center is exactly represented in both the raw and latent feature space. The raw feature space representation also enables the transformation of prototypes/means into an interpretable, linguistic *IF...Then* format. This semantic interpretation facilitates human comprehension of algorithmic decisions, thereby promoting transparency and understandability of the method.

The training architecture of the proposed IDSS method is shown in Figure 4.1. It is described in more detail below:

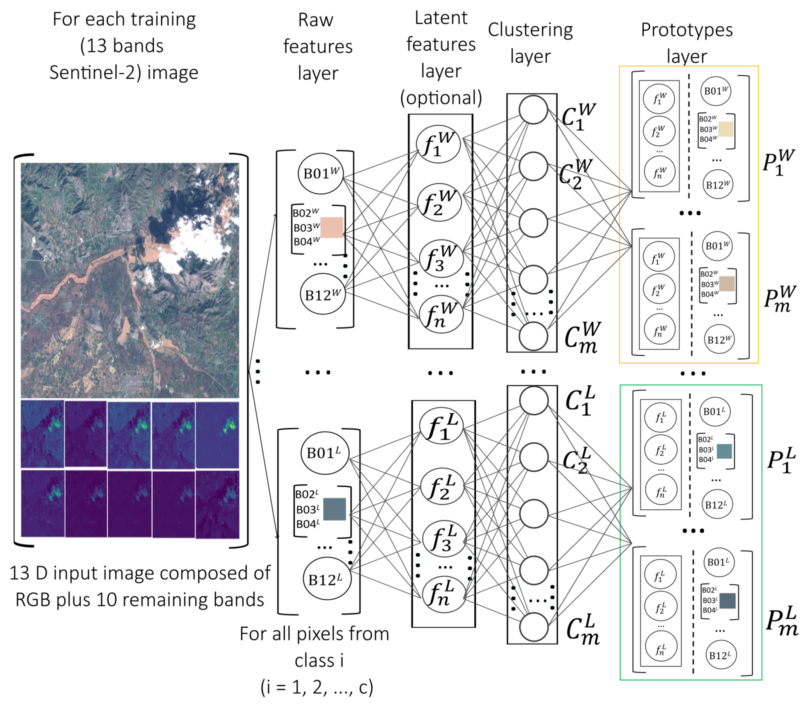

For each training (13 bands Sentinel-2) image
Raw features layer
Latent features layer (optional)
Clustering layer
Prototypes layer
13 D input image composed of RGB plus 10 remaining bands
For all pixels from class i (i = 1, 2, ..., c)
C_1^W
C_2^W
C_m^W
C_1^L
C_2^L
C_m^L
P_1^W
P_m^W
P_1^L
P_m^L

Figure 4.1: Training process of IDSS. The Raw features layer extracts the 13-dimensional spectral vector for every pixel and groups pixels by class. Then, these features are optionally passed through a Latent features layer to obtain latent vectors. The Clustering layer groups the latent features of each class into m clusters, and the Prototypes layer builds prototypes by pairing the clustering centres in the latent space with their corresponding ones in the raw feature space. These prototypes are later used to provide interpretable explanations during inference.

1) **Raw features layer**: The Raw features layer is responsible for extracting each pixel, x from the image and grouping it into different classes using corresponding labels since the algorithm is trained per class. In this layer, the raw features are extracted per pixel and per class. This vector of raw features is further passed to the next layer, which may be clustering or (optionally) - feature transformation.

2) **Latent features layer (optional)**: This layer is optional. For a better performance, the raw features can be transformed, e.g. by orthogonalization and optimization using PCA (Jolliffe and Cadima, 2016). Alternatively, deep feature representations using pre-trained convolutional neural networks, such as Segnet

(Badrinarayanan, Kendall, and Cipolla, 2017) or U-Net (Ronneberger, Fischer, and Brox, 2015) can be used. It is also possible to directly feed raw features into the next clustering layer. Currently, the U-Net deep neural network that is pre-trained on the Worldfloods dataset (Mateo-Garcia et al., 2021) is used to perform feature extraction. The input to the U-Net is formed by the raw 13 bands hyper-spectral Sentinel-2 pixel values from the training images, and the penultimate layer of the U-Net is used for feature extraction. The output is, thus, a 64-dimensional feature vector.

For all pixels, x_j^i (j = *1, 2,..., N*; where N denotes the number of pixels of a given image) for each class i (where i = *1, 2, ..., c*), the U-Net is used to map the raw features vector x_j^i into a n-dimensional latent feature space ϕ, where each feature is represented as F_j^i. Where $F_j^i \in R^{64}$, i.e. $n = 64$, since the output of the penultimate layer of the U-Net is a 64-dimensional vector. Then, L2 normalization is performed on the (latent) features as shown below:

$$F_j^i = \frac{F_j^i}{||F_j^i||} \tag{4.1}$$

For i = *1, 2, ..., c*, j = *1, 2,..., N*.

3) **Clustering layer**: The Clustering layer is responsible for two main tasks: applying clustering to the latent features and finding the cluster centers, C_m^i where m represents the number of cluster centers per class. Each cluster center does have a corresponding representation in the raw features (x_j^i) domain which can serve as a prototype providing a clear interpretability.

The mini-batch K-means (Sculley, 2010) clustering method is used without lose of generality. This particular version of the K-means approach is reported to be more suitable for handling large data sets and can be updated online. The main output of the K-means algorithm is the set of cluster centers, C_m^i, which minimize the distance between clusters in terms of the latent features used, $F_j^i \in F$.

The choice of the number of centers m is important for the performance of the mini-batch K-means algorithm. After several experiments, I chose m as 500, which gives the best results. It means that each of the land, water and clouds classes has 500 corresponding prototypes.

4) **Prototypes layer**: The prototypes are considered to correspond to the identified centres, C_m^i; however, they are being expressed in terms of raw features. Since the use of latent features is optional (in particular, one of the results from this study, shown in Table 4.2 is using raw features and no latent ones), prototypes may directly correspond to the centers. In cases when latent features are being used, a transformation is needed, which is a result of finding the means at each step of the K-means algorithm in terms of both raw and latent features. It has to be stressed, however, that the decision on how to reallocate data samples between clusters, which is a key part of the K-means approach, is based on latent features (if they are being used), whereas the raw features are primarily used to communicate the results to human users.

The validation architecture of IDSS is responsible for making predictions on new unlabeled Sentinel-2 test images, as shown in Fig. 4.2, described in detail below:

1) **Raw features layer**: This layer is responsible for extracting each pixel x from the unlabelled Sentinel-2 test image quite similarly to the similar layer of the Training architecture.

2) **Latent features layer (optional)**: This layer is the same as the training architecture, a 64-dimensional feature value is extracted from the penultimate layer of the U-Net. then, L2 normalization is performed same way as in the training architecture.

3) **Prototypes layer**: This layer is responsible for calculating the similarity to each prototype from each class. Similarity, S is calculated in the latent F_j (or raw, x_j) feature space using exponential kernel:

$$S(x^*, P_j^i) = exp(-||F(x^*) - F(P_j^i)||^2) \tag{4.2}$$

where x^* denotes a pixel from the test (unlabeled, new) image and P denotes the j^{th} prototypes of the i^{th} class.

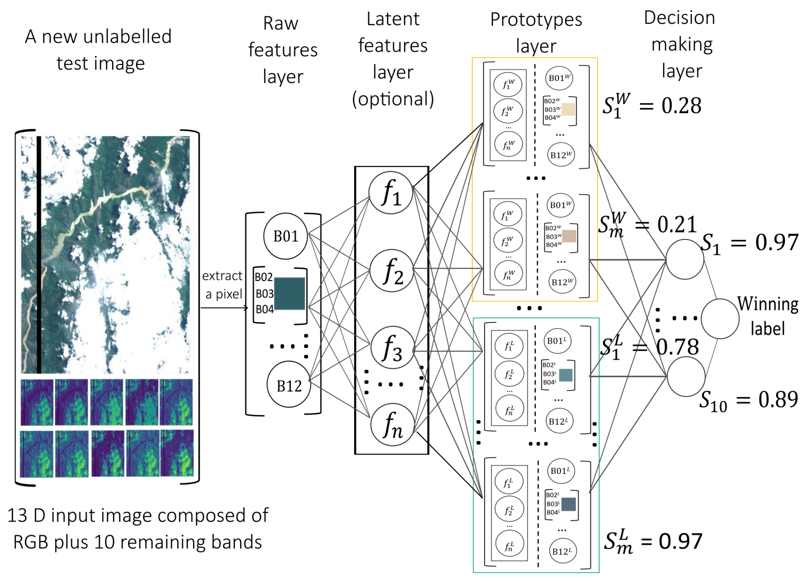


Figure 4.2: Validation process of IDSS. A new unlabeled Sentinel-2 test image is processed pixel-wise. Each 13-dimensional spectral vector is extracted in the Raw features layer and optionally transformed in the Latent features layer. In the Prototypes layer, the pixel's latent features are compared against all prototypes to compute similarity scores. Then, the Decision-making layer takes the "few winners take all" method to make the final decision.

4) **Decision-making layer**: The decision-making layer is responsible for assigning labels to unknown pixels. In this study, K nearest neighbours, also called the "few winners take all" method, is used to make the final decision. In this study, K is set to 10, which means that the labels corresponding to the ten prototypes with the highest similarity to the unlabeled pixels are used to perform decision-making. The label can be obtained by the following formula, where K is the number of neighbors, which is set to 10 in the experiments.

Table 4.1: Statistics of the Worldfloods dataset (Mateo-Garcia et al., 2021).

Dataset	***Flood events***	***Flood maps***	***Water pixels (%)***		***Land pixels (%)***	***Cloud pixels (%)***	***Invalid pixels (%)***
			Flood	**Permanent**			
Training	108	407	1.45	1.25	43.24	50.25	3.81
Validation	6	6	3.14	5.19	76.72	13.27	1.68
Test	5	11	20.23	1.16	59.05	16.21	3.34

4.2 Experiment setup

4.2.1 Worldfloods Dataset

In this chapter to test the newly proposed IDSS method the rich and recent Worldfloods dataset (Mateo-Garcia et al., 2021) is used. It describes 119 real flood events and has 424 flood maps. All flood maps are based on Sentinel-2 images containing 13 bands which form the raw features vector. The statistical analysis of the data is shown in Table 4.1 (Mateo-Garcia et al., 2021). The bands with a resolution larger than 10m were downsampled to 10m using nearest neighbours interpolation. From Table 4.1 it can be observed that the proportion of cloud pixels in the training data set is much higher than that in the validation and test data sets. Such an anomaly also takes place for the land pixel from the validation data set and the flood water pixels from the test data set. The Worldfloods data set is highly imbalanced (with a very small proportion of water pixels), which is a serious challenge for the classification task.

4.2.2 Training

The implementation of the proposed model is based on the ML4Floods pipeline (Mateo-Garcia et al., 2021) and scikit-learn (Pedregosa et al., 2011). The U-Net has been pre-trained on the Worldfloods dataset and used for the extraction of the latent features. The Sentinel-2 images from the training data set were cropped to size 256 × 256 for feature extraction via U-Net.

4.2.3 Testing

The test images are cut into 256 × 256 for feature extraction. It should be noted that Worldfloods images vary significantly in size. Most of the images cannot be cut entirely to the size of 256 × 256. The patches that are smaller than 256 × 256 are discarded in training, but this can not be done during testing. In this chapter, padding is performed on the test images so that they can be perfectly cut into 256× 256 patches. After that, the prediction masks are stuck together, and the padding part is simply cut off. All experiments in this chapter were conducted on the High-End Computing Cluster with Intel(R) Xeon(R) Gold 6248 CPU @ 2.50 GHz processor.

4.2.4 Performance Evaluation

The following metrics were used to evaluate the proposed algorithm:

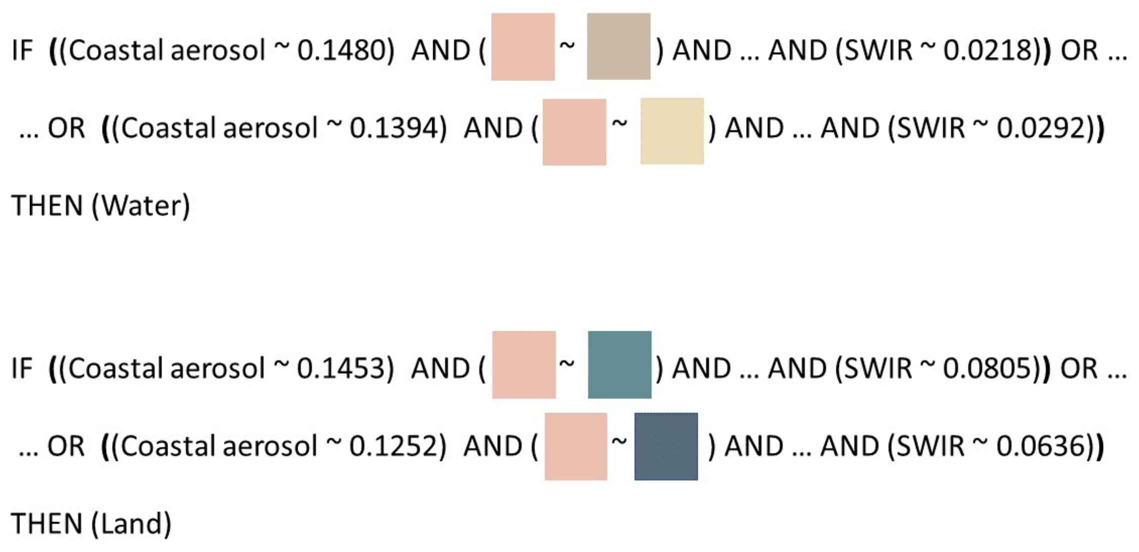
Intersection over Union (IoU):

$$IoU = \frac{TP}{FP + TP + FN} \tag{4.3}$$

Recall:

$$Recall = \frac{TP}{TP + FN} \tag{4.4}$$

Here, TP, FP and FN represent True Positive, False Positive and False Negative, respectively. IoU is the widely used measure of success for this problem because it maximizes the TP while minimizing FP as well as TP and FN. Recall is considered to be the more important measure than Precision since missing flood areas are more dangerous and risky than over-predicting flood areas.


IF **(**(Coastal aerosol ~ 0.1480) AND (▨ ~ ▨) AND ... AND (SWIR ~ 0.0218)**)** OR ...

... OR **(**(Coastal aerosol ~ 0.1394) AND (▨ ~ ▨) AND ... AND (SWIR ~ 0.0292)**)**

THEN (Water)

IF **(**(Coastal aerosol ~ 0.1453) AND (▨ ~ ▨) AND ... AND (SWIR ~ 0.0805)**)** OR ...

... OR **(**(Coastal aerosol ~ 0.1252) AND (▨ ~ ▨) AND ... AND (SWIR ~ 0.0636)**)**

THEN (Land)

Figure 4.3: Linguistic rules of IDSS. $\sim$ denotes "is similar to" and "Coastal", "SWIR", etc., represent the raw features of a single pixel.

4.3 Interpretability of the proposed IDSS methods

The prototypes within IDSS can generate linguistic *IF ... THEN* rules, as shown in Figure 4.3, where $\sim$ denotes "is similar to" and "Coastal", "SWIR", etc., represent the raw features (e.g. the 13 bands of the Sentinel-2 signal) for a single pixel. Such linguistic rules are easy to understand, explain and use by humans because they are intuitive. They can be applied to unlabeled test images as a tool to understand the decision making process for deciding which class a particular pixel is more likely to belong to.

One such linguistic rule can be formed per prototype connecting individual terms such as (SWIR ...), (Coastal ...) etc. with logical conjunction (AND). In particular, each term concerns an individual raw feature (SWIR, Coastal,...). Brought together with the logical conjunction, AND operator they form an *IF...THEN* rule where the THEN condition defines the class label (e.g. Water). 500 clusters are considered per class and, therefore, 500 such *IF...THEN* rules are being generated per class.

All these 500 rules can be combined together using logical disjunction, OR operator resulting in a single (but large) linguistic rule per class. Although such rule is large, it is quite clear and easy to interpret, understand and explain to a human. In summary, the outcome can be either one large linguistic rule per class or multiple smaller rules per class.

Unlike traditional deep learning models that provide a "black box" decision making process, the proposed IDSS framework offers explicit, human-understandable rules that directly connect raw spectral features to model decisions. this transparency allows domain experts, such as hydrologists or remote-sensing analysts, to trace the prediction back to specific spectral characteristics. As a result, the decision-making process becomes more transparent and easier to understand, allowing users to detect errors, validate predictions and build confidence in the system.

4.4 Results and Analysis

The results of the proposed IDSS method are presented in Table 4.2 and summarized in this section.

Table 4.2 shows the metric comparison between the proposed algorithms and a

Table 4.2: Comparison of IoU and recall results.

Model	*IoU* total water %	*Recall* total water %	# Parameters (x1000)	Interpretability
$IDSS^1$ (ours)	<u>**73.10**</u>	93.35	96	✓
$IDSS^2$ (ours)	70.03	<u>**96.11**</u>	96	✓
xDNN (P. Angelov and Soares, 2020)	71.50	90.27	20	✓
U-Net (Mateo-Garcia et al., 2021)	72.42	95.42	7790	✗
SCNN (Mateo-Garcia et al., 2021)	71.12	94.09	260	✗
$NDWI^1$ (McFeeters, 1996)	65.12	95.75	-	✓
Linear (Mateo-Garcia et al., 2021)	64.87	95.55	-	✓
$NDWI^2$ (McFeeters, 1996)	39.99	44.84	-	✓

number of state-of-the-art methods, including both threshold-based and deep neural network algorithms. It can be observed that the proposed algorithm achieves very competitive results, going beyond the state-of-the-art.

Two variants of the proposed IDSS model are evaluated, denoted as $IDSS^1$ and $IDSS^2$. The difference between these variants lies in the number of images used to generate the prototypes. Specifically, $IDSS^1$ uses prototypes generated from six validation images, while $IDSS^2$ uses prototypes generated from 206 training images of the WorldFloods dataset. In both cases, the U-Net was first fine-tuned using all the training and validation data, after which the mini-batch K-means algorithm was applied to extract representative prototypes in both the latent and raw feature spaces.

While generated only on six images, the proposed method $IDSS^1$ has 0.68% IoU total water improvement compared to U-Net. At the same time, the proposed *IDSS* method also offers a linguistic form of explainability (Figure 4.3) and a clear network architecture (Figures 4.1 and 4.2) and uses in orders of magnitude less model parameters. $IDSS^2$ generated on 206 training images has 0.36% Recall total water improvement compared to the second highest method $NDWI^1$. The thresholds used for $NDWI^1$ and $NDWI^2$ were -0.22 and 0, respectively.

xDNN is an interesting recently proposed explainable form of Deep Neural Network method, which in this chapter was trained on the raw Sentinel-2 images without feature extraction and, again, only trained on six images. In this chapter, before applying this method, the Worldfloods model was used to remove the clouds. The result using xDNN after cloud removal is also competitive with the IoU total water of 71.50%. More importantly, however, the xDNN algorithm provides direct interpretability through the linguistic *IF ... THEN* rules, and the decision making can be understood and analyzed by humans, which can play an important role in practical applications.

A visual comparison of the segmentation results is shown in Figure 4.4. It can be appreciated visually that the segmentation results obtained by the proposed method

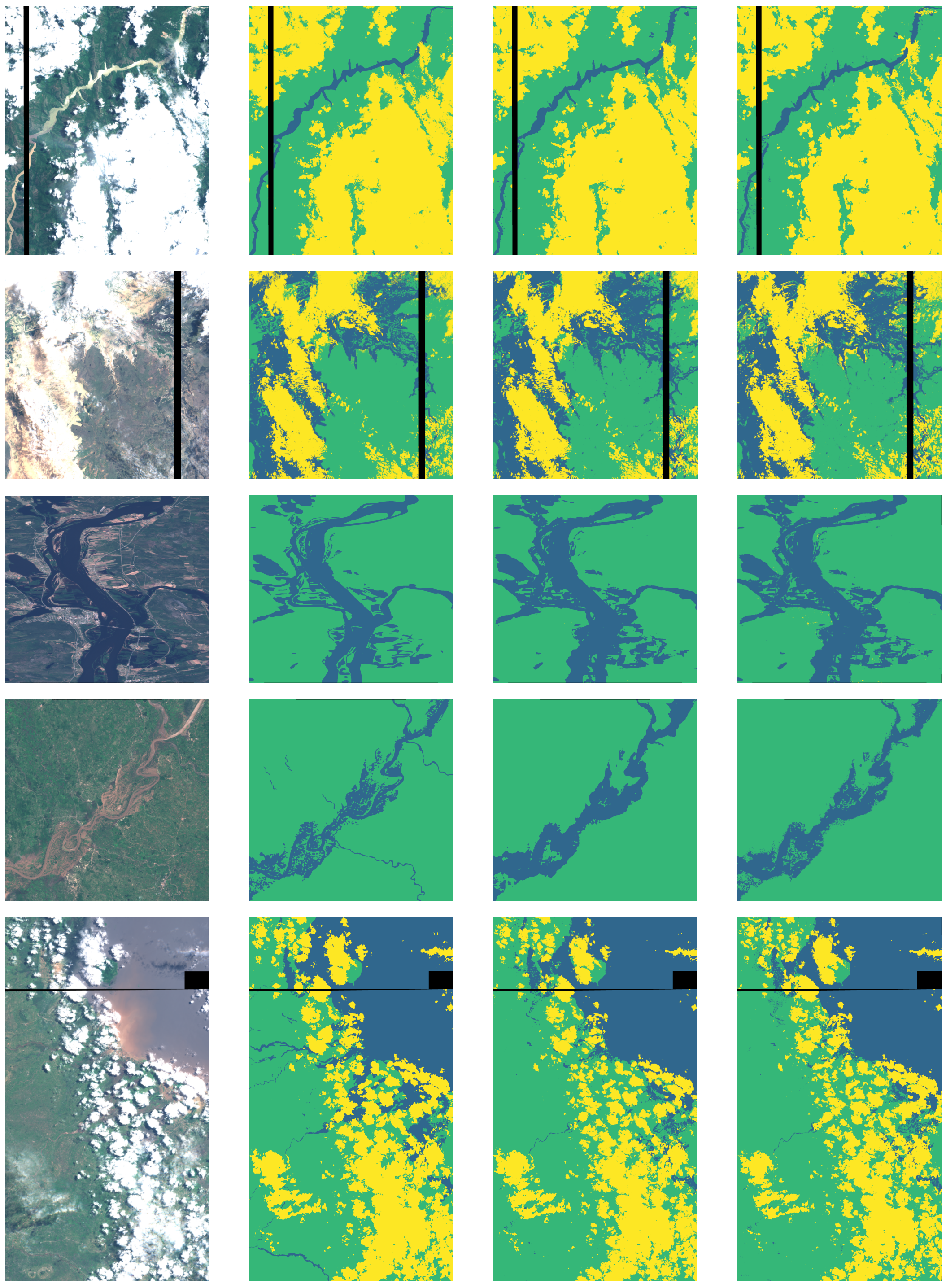


Figure 4.4: Comparison of segmentation results. The meaning of the colours is shown below: green - land, yellow - cloud, blue - water.

are very similar to those obtained by the U-Net model and to the reference labels. The U-Net model tends to over-predict water areas if comparing the images in the fourth row.

4.5 Conclusion

In this chapter, an interpretable deep semantic segmentation (IDSS) method is proposed for Earth Observation. It provides human interpretable results while ensuring a performance on par or surpassing that of the mainstream and state-of-the-art convolutional neural network. The proposed IDSS is a prototype-based method. Prototypes are the local focal points that are most representative in terms of the features. Working in a latent feature space, a projection to the raw feature space allows a human interpretable linguistic *IF...THEN* rules to be used to explain the decision making involved in determining the winning class label. Semantic rules are interpretable and easy for humans to understand, allowing users could inspect and audit the decision making process. The Worldfloods dataset was used, and the results show that IDSS outperforms the other methods in terms of IoU total water and Recall total water, requiring in orders of magnitude less parameters and training data. The next chapter will focus on the investigation of the application of the model to flood detection.

Chapter 5

Interpretable flood detection and flood mapping methods

5.1 Introduction

In time series image analysis, flood events are often considered to be a type of low-frequency anomalous event, and thus time series flood detection can be categorised as a time series change detection task. Currently, most time series change detection algorithms need to generate pixel-level change maps for each image frame, which is not only time-consuming but also inefficient. In addition, due to the widespread use of deep learning methods, the decision-making process of such methods is often hard to explain and incomprehensible to human users.

To address these challenges, an Interpretable Multi-stage Approach to Flood Detection from time series Multispectral Data (IMAFD) is proposed and introduced. IMAFD divides flood detection into four steps. First, IMAFD identifies anomalous images from a set of time series images. Recursive Similarity Estimation (RSE) based on the Recursive Density Estimation (RDE) (P. Angelov, 2012) was used. It is specifically adapted to the flooding task and is detailed in the Appendix. Next, if the image is identified as an anomaly, a similarity-based binary change detection method is used to locate the anomalous region within that image. Furthermore,

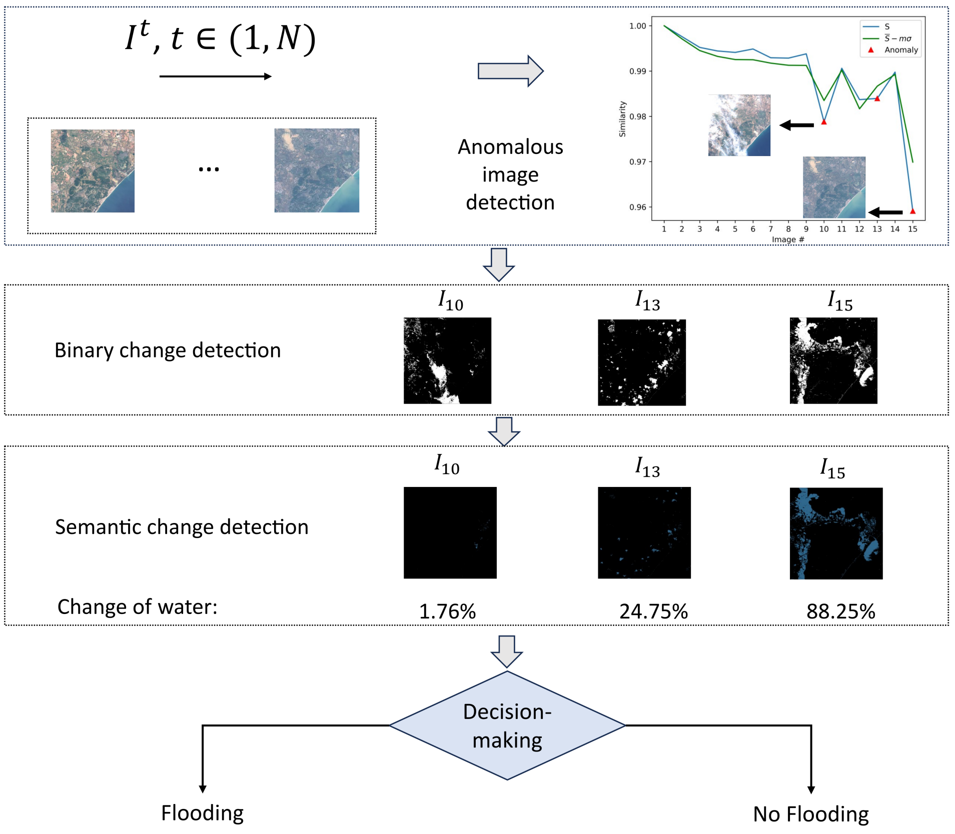


Figure 5.1: IMAFD architecture. The proposed IMAFD architecture consists of the following four stages, (1) anomalous image detection: identifying suspected images from time series multispectral optical images, (2) binary change detection: detecting changes within suspected images, (3) semantic change detection: semantic segmentation of changes into Land, Water or Cloud class and (4) decision making.

to classify such regions, the Interpretable Deep Semantic Segmentation + (IDSS+) method is employed to distinguish between Land, Water, and Clouds. Finally, a decision-making process combines the above information to determine where the flooding occurred and the specific flooded areas within that image.

The overall architecture of the IMAFD framework is shown in Figure 5.1. The details of the proposed architecture and its function are introduced as follows.

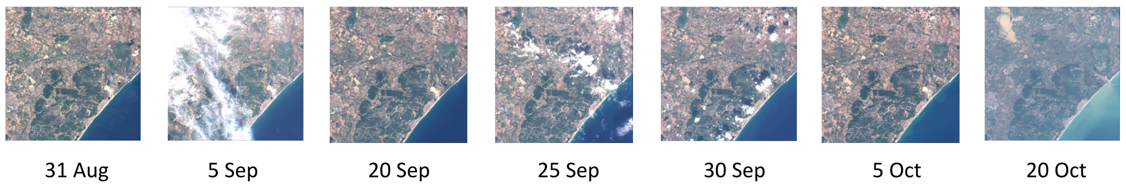


Figure 5.2: RGB visualization of Sentinel-2 time series multispectral images (All of the above images were taken in 2018).

5.2 IMAFD framework

5.2.1 Anomalous images detection

Figure 5.2 shows a series of Sentinel-2 time series multispectral images that are visualized by the RGB channels, which tend to show anomalies such as cloud cover, natural disasters, urban development, and agricultural harvesting. Among these anomalies, this chapter mainly focuses on flooding. In this context, a flood usually occurs when land is covered by floodwater, such as when cities or farmland are flooded, or when existing rivers or lakes begin to expand. Another situation that needs to be considered is that cloud cover is a common phenomenon in such time series multispectral images. In contrast, flooding is a low-probability event.

To detect anomalous images, including clouds and floods, I use the statistics for outlier estimation by detecting drops in the overall image similarity degree, S, according to the following criterion:

$$\begin{aligned} IF \quad & S^k < \bar{S}^k - m\sigma^k \\ THEN \quad & (possible \quad novelty) \\ ELSE \quad & (normal \quad image) \end{aligned} \tag{5.1}$$

For $k = 1, ..., N$ images of the data stream, m is a positive number, $S \in (0; 1]$ is the overall image similarity degree and $\bar{S} \in (0; 1]$ is the mean similarity degree.

5.2.2 Binary Change detection

Since the pixel-wise similarity, $s(x^k_{h,v})$ itself measures the distance between the current pixel value and all previous pixel values at (h, v)th position, where h denotes horizontal index within the images and $h = 1, 2, \ldots, H$ means pixel index that varies between 1 and H, v denotes vertical index within the images and $v = 1, 2, \ldots, V$ means pixel index that varies between 1 and V. The pixel-wise similarity, s, can naturally be used to perform binary change detection using a threshold, ϵ:

$$s(x^k_{h,v}) < \epsilon \tag{5.2}$$

The derivation of the similarity, s, which follows the density expression from (P. Angelov, 2012) can be found as follows:

$$s(x^k_{h,v}) = \frac{1}{(1 + ||x^k_{h,v} - \mu^k_{h,v}||^2 + \Sigma^k_{h,v} - ||\mu^k_{h,v}||^2)} \tag{5.3}$$

Here, the mean value $\mu_{h,v}$ and the scalar product $\Sigma^k_{h,v}$ can be updated recursively as follows (P. Angelov, 2012):

$$\mu^k_{h,v} = \frac{1}{L}\sum_{l=1}^{L} x^L_{h,v} \tag{5.4}$$

$$\Sigma^K_{h,v} = \frac{k-1}{k}\Sigma^{k-1}_{h,v} + \frac{1}{k}||x^k_{h,v}||^2 \tag{5.5}$$

It is important to note that L represents the number of images that were not identified as novel images. In such a case, anomalous pixels like clouds and floods will not affect the mean value μ and the mean similarity $\bar{(S)}$.

Furthermore, the overall image similarity, S^k, is calculated by taking the mean value of the pixel-wise similarity of the k-th image using Equation (4) (P. Angelov, 2012):

$$S^k = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} s(x^k_{h,v}) \tag{5.6}$$

Subsequently, the mean similarity $\bar{S}$ and standard deviation σ are updated recursively as follows (P. Angelov, 2012):

$$\bar{S}^k = \frac{k-1}{k}\bar{S}^k - 1 + \frac{1}{k}S^k, \bar{S}^1 = S^1 \tag{5.7}$$

$$(\sigma^k)^2 = \frac{k-1}{k}(\sigma^{k-1})^2 + \frac{1}{k}(S^k - \bar{S}^k)^2, (\sigma^1)^2 = 0 \tag{5.8}$$

5.2.3 Semantic change detection

I proposed the IDSS+ method and used it to label the binary change map obtained in the previous step. Both IDSS and IDSS+ use per-pixel prototypes. However, IDSS uses (averaged) centroid vectors, hence, it does not allow for associating the predictions with the real pixels from the training data. IDSS+ offers another option to make the decision-making process easier to understand for the users, with a small loss of accuracy, which uses the nearest real pixels to each centroid as prototypes. This allows the segmentation problem to be expressed in a dual form that combines latent and raw feature space vectors.

The training and testing architecture of the IDSS+ is shown in Figure 5.3. During the training phase, each 13-band Sentinel-2 multispectral image $I \in \mathbb{R}^{H\times V\times n}$ is, first, fed into a feature extractor to obtain a latent feature vector $F \in \mathbb{R}^{H\times V\times D}$, where $n = 13$ are the 13 bands and D represents the dimensionality of the latent feature space, e.g. for U-Net used in this study, $D = 64$. The feature extractor could be any semantic segmentation neural network. In this study, the U-Net is used due to its excellent performance at water segmentation (Mateo-Garcia et al., 2021). It is important to note that the U-Net has been pretrained on the Worldlfoods training dataset to achieve a better capability in feature extraction.

After feature extraction, for each pixel at position (h, v), latent features $f_c^{h,v}$ and corresponding raw features $x_c^{h,v}$ were obtained within each class, $c \in 1, 2, \ldots C$. For the WordldFloods dataset, $C = 3$ since there are three classes in total, Water, Land, and Cloud. I propose to then apply clustering algorithm to the latent features, obtaining the cluster centroids y_c^l, where y_c^l is the lth cluster center of class c, $l = 1, \ldots, L$ where L is the number of clusters per class. After clustering,

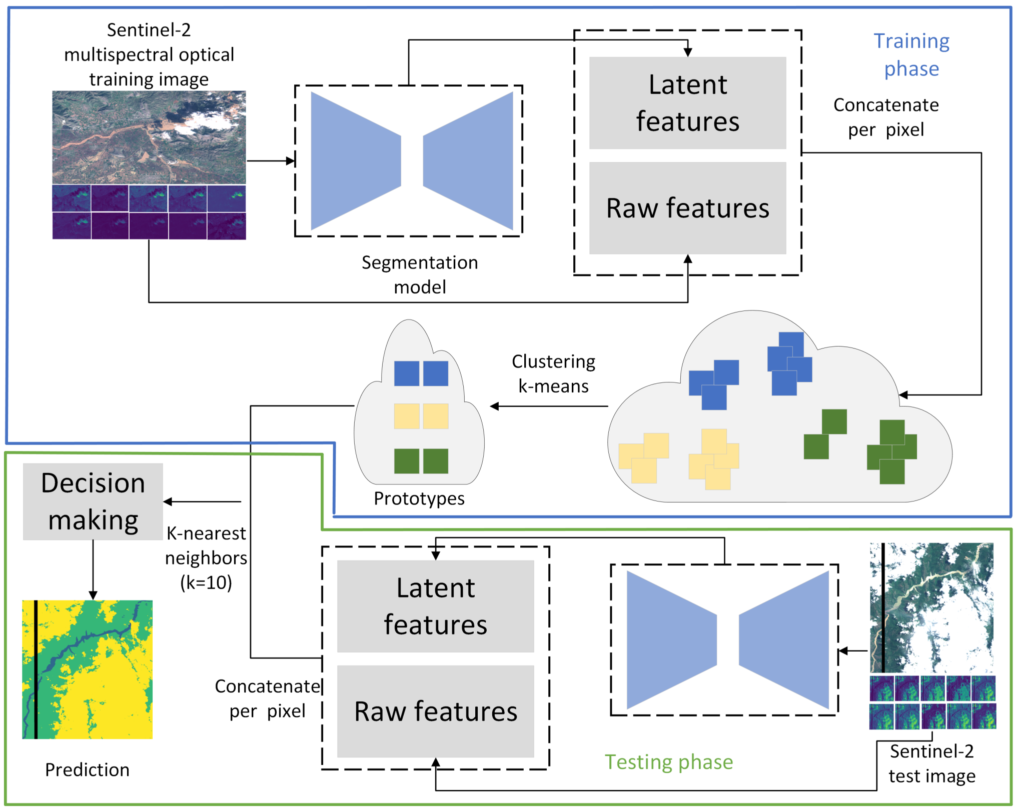


Figure 5.3: IDSS+ architecture.

prototypes P_c^l are selected for which both the latent features are closest to the cluster centroids and their corresponding raw features. For clustering, the mini-batch k-means (MacQueen, 1967; Sculley, 2010) or k-medoids (Kaufman, 1990) algorithms are used and implemented. From Table 5.2, it can be observed that mini-batch k-means has better performance compared to k-medoids, but the k-medoids method could provide better interpretability because it uses real data observations as prototypes. An ablation study considering different numbers of prototypes for mini-batch k-means is shown in Table 5.3.

The testing phase is also shown in Figure 5.3. The same feature extractor is used to obtain the latent features, $f^{h,v}$ from raw features, $x^{h,v}$. Then, the Euclidean distance is calculated between each latent feature $f^{h,v}$ corresponding to a given raw feature $x^{h,v}$ and the latent features y_c^l from the prototypes P_c^l. Then, the k-nearest neighbour algorithm is used to make the final prediction.

Furthermore, a confidence map is introduced within the IDSS+ method to provide more insight into the decision-making process of the algorithm, as shown in Figure 5.4. Since the k-nearest neighbour algorithm is used to make the final decision, additional insight into the rationale and the strength or "confidence" of the decision can be obtained. If the "confidence" is plotted per pixel, the heatmap shown can be obtained as shown in Figure 5.4. A threshold is utilized to distinguish between high and low confidence, which can be adjusted by human users. In the context of Figure 5.4, the threshold is set to 0.8, which means that if more than 8 out of 10 neighbouring prototypes have the same class label, the decision is considered to be with high confidence. Predictions with lower confidence are indicated by lighter colours, while darker colours indicate high confidence.

The threshold value of 0.8 was chosen empirically to balance clarity and accuracy. A higher threshold would make the visualization too restrictive, showing very few high-confidence regions, while a lower threshold would make most areas appear overly confident. Setting the threshold at 0.8 provides a clear and informative contrast between high- and low-confidence predictions, making the results easier to interpret while still reflecting meaningful differences in model certainty.

An advantage of IDSS+ method is its ability to significantly reduce noise pixels compared to the U-Net, as shown in Figure 5.5. It could be observed that the prediction of the U-Net has more noisy pixels. On the contrary, IDSS+ provides a much cleaner and more accurate prediction. This is considered the main reason that IDSS+ outperforms U-Net in terms of IoU water and Mean IoU, see Table 5.2.

In addition, IDSS+ provides more insight for human users by analyzing the prototypes, which not only contain latent features, but also raw features. Furthermore, when the k-medoids method was used to select the prototypes, the prototypes are real training data. In this case, the human user can realistically observe the geographic location of the training data that is closest to the test data in the latent feature space and the spectral information contained in the corresponding raw features. For example, the new test pixel might come from the NorthWest of the

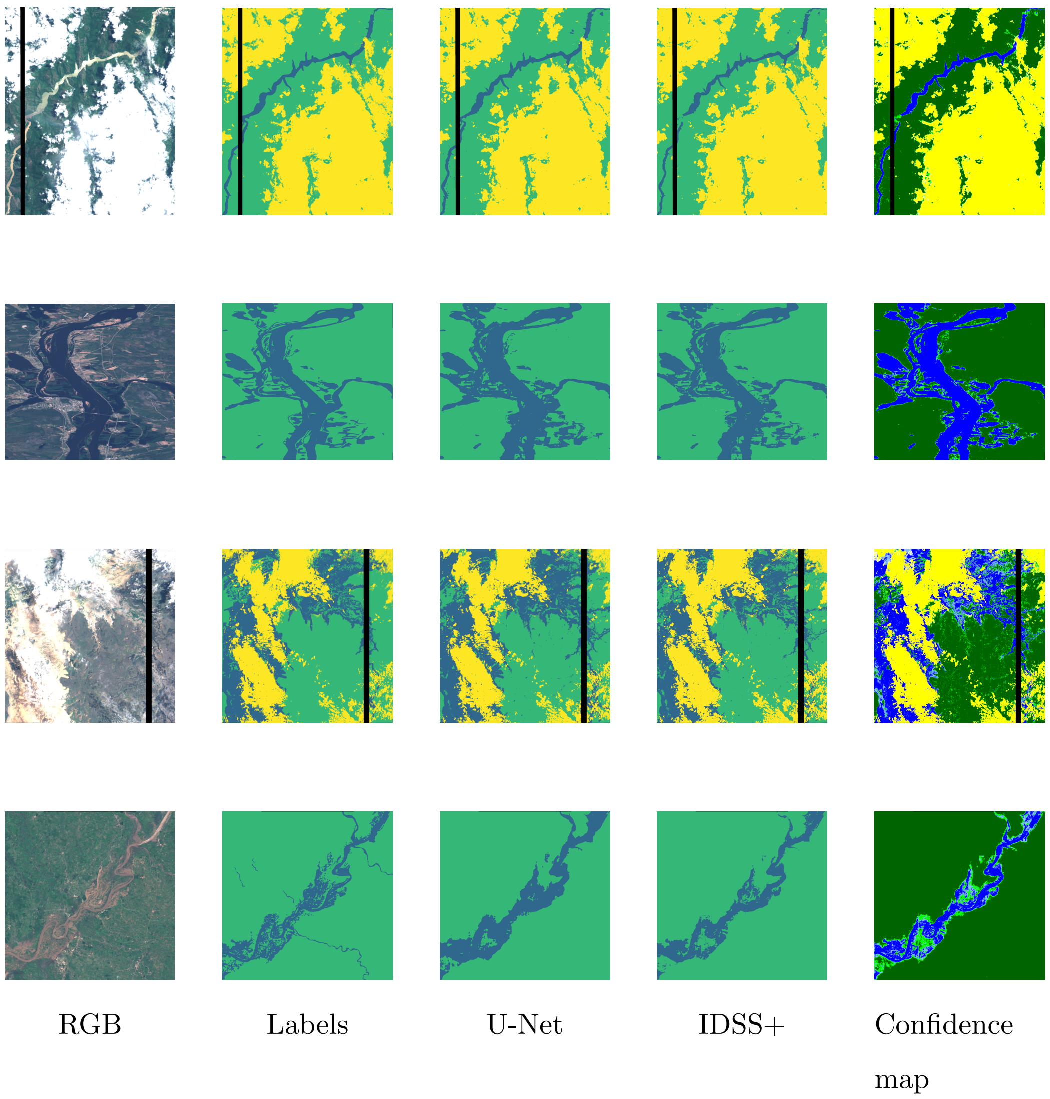


Figure 5.4: Comparison of segmentation results. The meaning of the colours is: green - Land, yellow - Cloud, blue - Water. For the confidence map, lighter colours indicate lower confidence, while darker colours indicate high confidence.

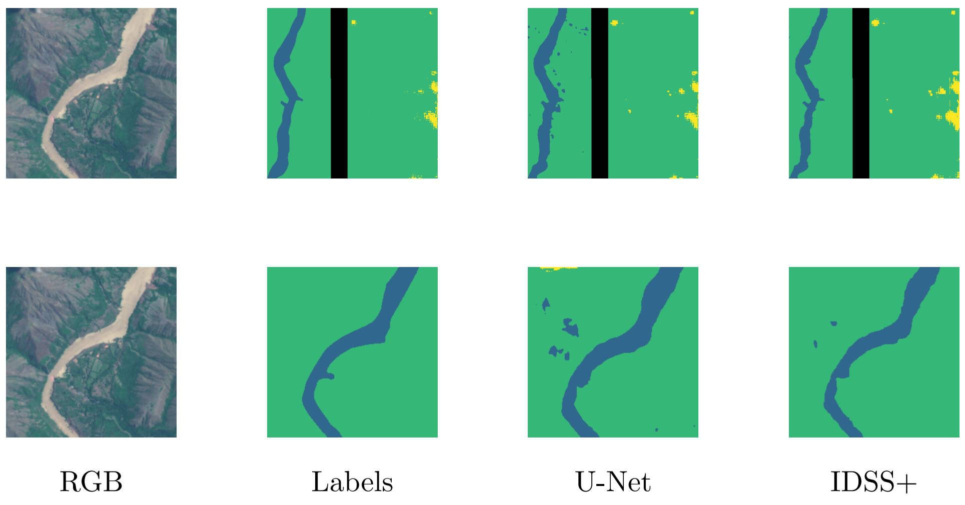


Figure 5.5: Comparison of segmentation results. The meaning of the colours is: green - Land, yellow - Cloud, blue - Water.

UK and be predicted as a Water pixel, while the closest prototype is a Water pixel coming from the SouthEast of the UK. Such information could help the expert to further analyze the relationship between the current flood and historical information.

5.2.4 Decision making

This stage is in charge of combining the information obtained from the previous steps to perform the decision-making. Specifically, it provides the time when the flooding starts based on the percentage of change in the water pixels. After obtaining the semantic change map, the percentage of the change of the water pixels is calculated. Furthermore, the decision about “Flooding ” or “No Flooding” is made by setting a threshold that could be adjusted by human users. In addition, the previously obtained semantic change map can also provide human users with the exact location of the flooding.

5.3 Experiments

5.3.1 Datasets

1) WorldFloods: WorldFloods (Mateo-Garcia et al., 2021) is a semantic segmentation dataset tailored for flood detection and is used to validate the proposed IDSS+ method. All flood maps were acquired from Sentinel-2 satellites. Specifically, there are three classes: Water, Land and Cloud. It contains 119 real flood events and has 424 flood maps with 407 images used for training, 6 for validation and 11 for testing.

2) RaVAEn: The RaVAEn (Růžička et al., 2022) dataset is used for the evaluation of the proposed IMAFD method, which contains four different natural disasters: hurricanes, fire burn scars, landslides and floods. This study focuses only on the flood-related subset of the dataset. Originally, there were four flood events, each containing five time series images captured by Sentinel-2 satellites. To better validate the proposed method, each flood event was extended to include fifteen images. The first fourteen images were taken before the flood, and the fifteenth was taken after the flood. The extended dataset is Sentinel-2A images and was downloaded using the Google Earth Engine, ensuring consistency with the original dataset. Each event provides a ground truth label of changes and clouds, indicating the differences between the last two time series images. To eliminate the effect of clouds, the ground truth label of clouds was discarded.

5.3.2 Evaluation metrics

The evaluation metrics can be divided into three categories. The first one mainly evaluates the performance of the first stage of the IMAFD for anomaly detection of Sentinel-2 time series images. Precision, Recall and F1 scores were calculated and compared. The second one is to evaluate the semantic segmentation performance of the IDSS+ on the WorldFloods dataset, Intersection over Union (IoU) is mainly used, including IoU water, IoU land, IoU cloud, and Mean IoU (mIoU). The latter is used to evaluate the semantic change map obtained after the third stage of semantic

Table 5.1: Performance comparison of IMAFD anomalous image detection on the RavAEn dataset. The better results within each catogory are highlighted in bold.

	Raw features (13 bands)			DINO-ViT-S/16 (Yi Wang et al., 2022)		
Scenario	Precision	Recall	F1	Precision	Recall	F1
1	**0.11**	**1.00**	**0.20**	0	0	0
2	**0.50**	**1.00**	**0.67**	0.12	**1.00**	0.22
3	**0.33**	**1.00**	**0.50**	0.20	**1.00**	0.33
4	0.17	**1.00**	0.29	0.17	**1.00**	0.29
Average	**0.28**	**1.00**	**0.42**	0.12	0.75	0.21

change detection. Precision, Recall, F1 score and IoU water are utilized.

5.4 Results

5.4.1 Anomalous image detection

To evaluate the performance of the first stage of IMAFD (anomalous image detection), experiments were conducted on the RavAEn datasets. The results are presented in Table 5.1. The RSE method used in the anomalous image detection stage is sensitive to the input features. Thus, I performed an ablation study to compare its performance under different features on the RavAEn dataset, as shown in Table 5.1. It can be observed that when using the raw features, the performance is better than when using latent features derived from the DINO-ViT-S/16 model pre-trained on EO datasets (Yi Wang et al., 2022). Specifically, all the recall values obtained by raw features are 1, meaning that no flood event was missed. High recall values were prioritized, given the reason that failing to predict an actual flood is more severe than a false alarm. Besides, even if a false alarm is made, the details will be further checked in the following stage of the IMAFD methods.

In addition, it can be observed that for scenario 1, the performance of the pre-trained DINO-ViT-S/16 model drops to zero across all metrics. This occurs because the visible change between the pre flood and after flood images is extremely small, which limited to a narrow river region. While this subtle change is easily noticed in the raw multispectral features, it is difficult for a pre-trained neural network to capture, as such models are generally less sensitive to fine-grained spectral variations.

5.4.2 Semantic segmentation

To evaluate the effectiveness of the IDSS+ semantic segmentation method, I perform experiments on the WorldFloods dataset and compare it with other methods, as shown in Table 5.2. It can be observed that the proposed IDSS+ provides the highest IoU water and mIoU, outperforming the benchmark U-Net and other methods. Specifically, the IoU water is exceeded by 1.57% and the mIoU is exceeded by 0.19% compared to U-Net. In particular, it can be seen from Figure 5.5 that the prediction map of IDSS+ has less noise compared to U-Net. Besides, it can be seen from Table 5.2 that mini-batch k-means has a better performance compared to k-medoids, but the k-medoids method could provide better interpretability because it uses real pixels from the dataset as prototypes. An ablation study considering different numbers of prototypes for mini-batch k-means is shown in Table 5.3.

Furthermore, the prototype in the latent space was visualized with the number of prototypes for each class equal to 10 and 100, as shown in Figure 5.6. It can be observed that 100 prototypes for each class have a better separation compared to 10 prototypes for each class. This also explains the better performance of the former as shown in the Table 5.3.

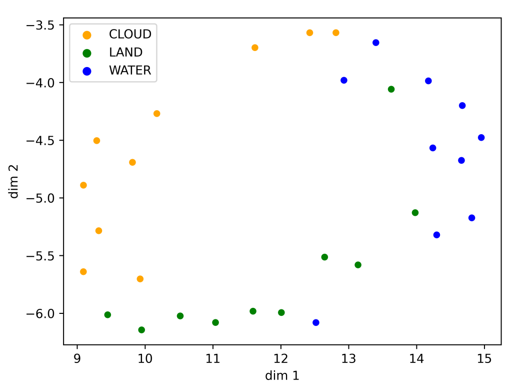
In addition, to better visualize the decision-making process of the algorithm, the UMAP plot could be drawn based on the visulization of the prototypes in the latent features to detail the prediction procedure of each pixel and the reason for the misclassification. As shown in Figure 5.7a, the prototypes in the latent feature space are marked as three different coloured circles. It can be seen that the unknown red triangular pixel is successfully predicted as Water, as it is surrounded by blue circles

Table 5.2: Comparison of IoU results (%) among different models. $NDWI^1$ represents when the threshold was set to -0.22 while $NDWI^2$ indicates when the threshold was set to 0.

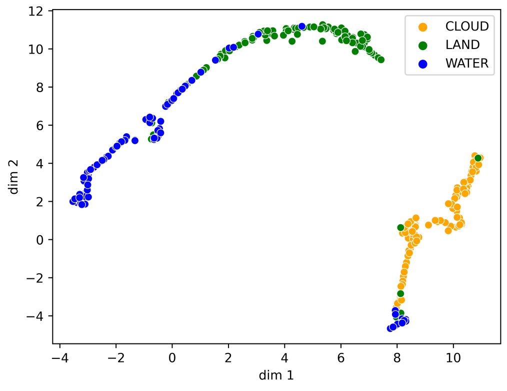
Model	**IoU (%)**			**mIoU (%)**
	Water	**Land**	**Cloud**	
$NDWI^1$ (Gao, 1996)	65.12	–	–	–
$NDWI^2$ (Gao, 1996)	39.99	–	–	–
Linear (Mateo-Garcia et al., 2021)	64.87	–	–	–
SCNN (Mateo-Garcia et al., 2021)	71.12	–	–	–
U-Net (Mateo-Garcia et al., 2021)	72.95	82.56	**89.67**	81.71
IDSS+ (k-means)	**74.52**	**82.72**	88.46	**81.90**
IDSS+ (k-medoids)	74.50	82.54	88.10	81.71

Table 5.3: Comparison of IoU results (%) for different numbers of prototypes per class.

# Prototypes	**IoU water**	**IoU land**	**IoU cloud**	**mIoU**
10	71.80	69.06	60.59	67.15
20	73.23	**83.03**	88.05	81.44
50	74.03	82.08	86.61	80.91
100	**74.52**	82.72	**88.46**	**81.90**
1000	74.00	81.78	88.01	81.26
2000	71.67	82.71	88.21	80.86
3000	70.77	82.34	88.00	80.37

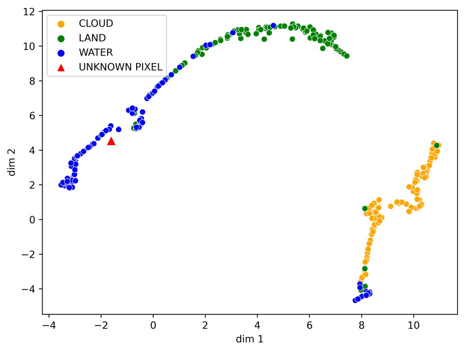


(a) #Prototypes for each class: 10

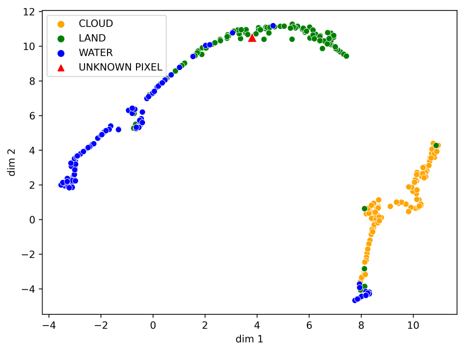


(b) #Prototypes for each class: 100

Figure 5.6: Prototype visualization in the latent space with different numbers of prototypes using Uniform Manifold Approximation and Projection (UMAP).



(a) UMAP plot for a correct prediction (prediction: Water, label: Water)



(b) UMAP plot for a wrong prediction (prediction: Land, label: Water)

Figure 5.7: Visualisation of the decision-making process with the prototype in the latent space using UMAP. Plot (a) shows an example where the unknown red triangular pixel is successfully predicted as Water because it is much closer to the blue Water prototypes, while plot (b) shows an example where the unknown red triangular pixel is misclassified as Land because it is surrounded by green Land prototypes, although its ground truth label is Water.

representing Water prototypes. Comparatively, in figure 5.7b, the unknown red triangular pixel is predicted as Land as it is surrounded by green Land prototypes, even though its ground truth label is Water. This explains why the unknown pixel was incorrectly predicted as Land.

Another advantage of IDSS+ is that it could provide more insight for human users by analyzing the prototypes, which contain both latent and raw features. Specifically, when employing the k-medoid method, the selected prototypes were real pixels from the training data. In this case, the human user can realistically observe the geographic location of the training data that is closest to the test data in the latent feature space and the spectral information contained in the corresponding raw features. For example, the new test pixel might come from the NorthWest of the UK and be predicted as a Water pixel, while the closest prototype is a Water pixel coming from the SouthEast of the UK. Such information could help the expert to further analyze the relationship between the current flood and historical information.

5.4.3 Semantic change detection

The ablation study of the semantic change detection was assessed on the RavAEn dataset. The results, shown in Table 5.4, compare the performance of IDSS+ and U-Net when labelling the binary change maps obtained from the second stage of IMAFD. It can be observed that IDSS+ achieves competitive results compared to U-Net. Specifically, in most scenarios, IDSS+ surpasses U-Net in terms of Precision, F1, and IoU water, while U-Net achieves better results in terms of Recall. This shows that the U-Net is more prone to overpredicting flood extent. The visualized results are shown in Figure 5.8.

5.5 Conclusion

In this chapter, I proposed a holistic, efficient, and interpretable flood detection framework, IMAFD. This approach divides flood detection into four steps and

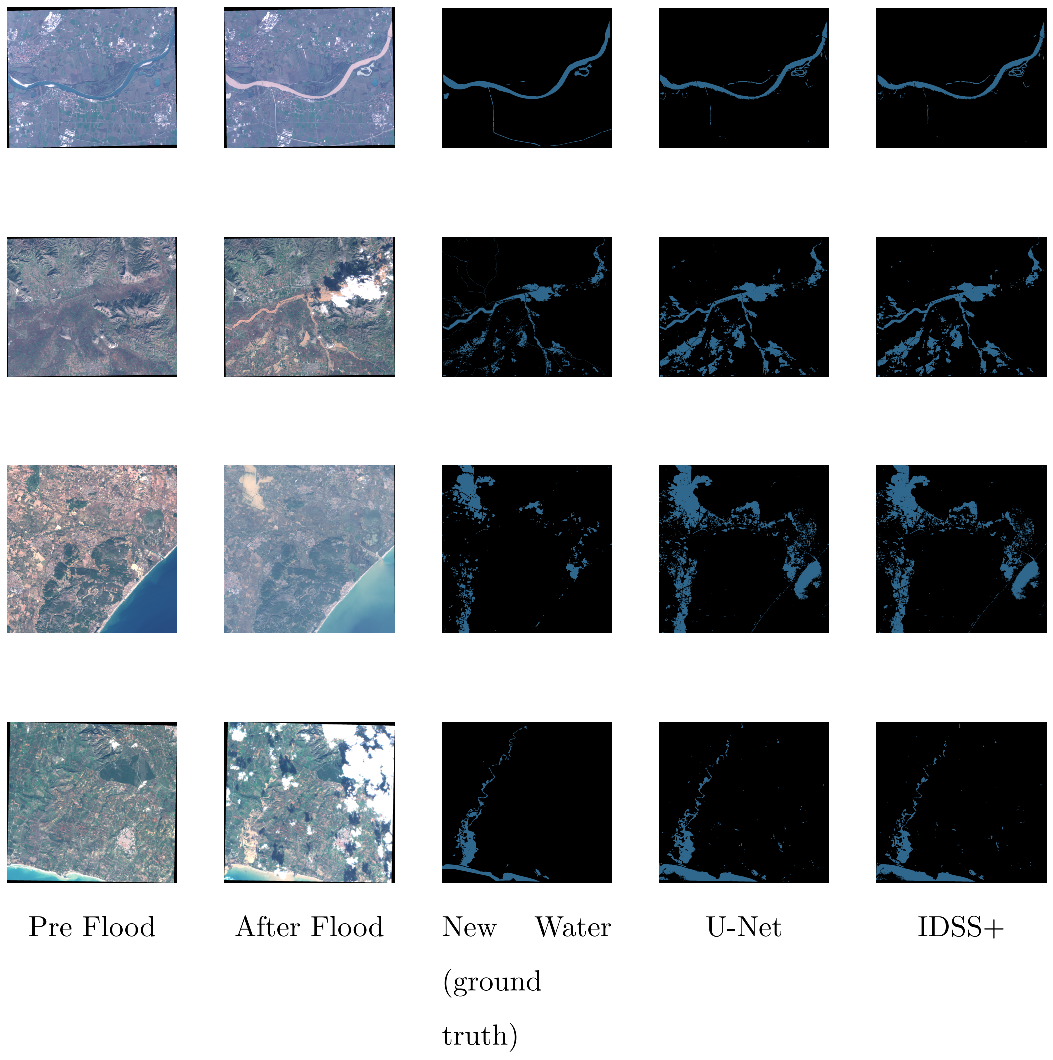


Figure 5.8: Comparison of segmentation change detection results. The blue colour in Ground Truth, IMAFD (U-Net) and IMAFD (IDSS+) indicates changed water/flood.

Table 5.4: Semantic segmentation change results comparison (all values in %). Here, P stands for Precision and R for Recall.

	IMAFD (IDSS+)				**IMAFD (U-Net)**			
Scenario	**P**	**R**	**F1**	**IoU**	**P**	**R**	**F1**	**IoU**
1	**83.64**	79.22	**81.37**	68.59	81.93	**79.39**	80.64	67.56
2	**60.87**	89.51	**72.46**	**56.82**	57.79	**90.71**	70.60	54.56
3	**43.51**	91.13	**58.90**	**41.74**	41.90	**91.66**	57.50	40.35
4	**65.63**	80.24	72.20	56.50	64.16	**86.75**	**73.77**	**58.44**
Average	**63.41**	85.03	**71.23**	**55.91**	61.45	**87.13**	70.63	55.23

progressively narrows down the flood detection problem, starting with detecting suspected images from the time series and moving to detect the semantic change at the pixel level. In addition, the semantic change detection (stage 3 of IMAFD) method IDSS+ provides additional insight based on UMAP plots and confidence maps to help the users further understand the decision-making process of the algorithm. Finally, the comparative experiments and ablation study were performed on WorldFloods and RavAEn datasets, demonstrating the effectiveness and efficiency of the proposed method.

Despite the above advantages, IMAFD has some limitations. One of the limitations of the proposed IMAFD method is that it is based on the Recursive Density Estimation (RDE) (P. Angelov, 2012) method, which is a recursive method. Thus, choosing the first image in a time series event is crucial and must be guaranteed to be free of anomalies such as clouds and floods. In addition, the choice of thresholds can significantly affect the performance of the algorithm. Addressing these issues should be critical for future work.

Chapter 6

An Interpretable Ensemble Geoscience Foundation Model for Flood Mapping

6.1 Background

Flooding is one of the most dangerous and widespread natural disasters, displacing people and causing huge economic losses (Bentivoglio et al., 2022). It not only destroys cities, farmlands, and infrastructure, but also contaminates food and water, causing disease outbreaks and loss of life (C. Li et al., 2022; Munasinghe et al., 2023). It is estimated that between 1995 and 2015, floods caused \$7.5 billion in damage and affected at least 109 million people worldwide (Janizadeh et al., 2021; Alfieri et al., 2017). Moreover, it is estimated that the number of people at risk of flooding will continue to grow (Tellman et al., 2021).

With the development of deep learning and the increasing availability of high-resolution satellite imagery, deep learning-based models have gradually become popular in flood mapping. Especially, recent advances in Foundation Models for texts, images, and videos have led to significant resources being spent on developing foundation models for Earth Observation Data Analysis. Several Geoscience

Foundation Models have been proposed recently, including PhilEO (Saux et al., 2024; Fibaek et al., 2024), Prithvi (Jakubik et al., 2023) and SatlasNet (Bastani et al., 2023).

Different from traditional computer vision, which mainly relies on RGB images, geoscience foundation models are mostly pre-trained on data from different sensors, varying resolutions, and different numbers of bands. Thus, applying such models to downstream tasks often needs to meet the same data requirements as the pre-trained ones, which greatly hinders the application of such geoscience foundation models. In addition, despite numerous geoscience foundation models that have been proposed in the past two years, very little work has explored the ensemble learning of such models. Besides, deep neural networks, particularly foundation models, typically have hundreds of millions of parameters and are often regarded as "black boxes", making their decision-making process difficult to interpret for human users.

Thus, this chapter investigates the ability of different geoscience foundation models to generalize across different bands. It further introduces an interpretable plug-and-play prototype-based module called Interpretable Deep Semantic Segmentation v2 (IDSS v2) that could explain the decision-making process with the help of representative training samples. Finally, an ensemble foundation model incorporating this module is proposed and evaluated on the WorldFloods v2 flood mapping dataset. The results show that the proposed module can effectively improve the performance of the geoscience foundation models and provide a transparent decision-making process for human users. Besides, the interpretable ensemble foundation model with the proposed module outperforms all other models on the WorldFloods v2 dataset.

The contribution can be summarized as follows:

- The generalization ability of several geoscience foundation models across different bands was evaluated.
- A plug-and-play prototype-based module IDSS v2 was proposed to explain the decision-making process of the model using representative training samples. It

enables human users to audit and understand the behavior of the model.

- An Interpretable Ensemble Geoscience Foundation Model (IEGF) for flood mapping using earth observation data was introduced.

- The proposed module and interpretable ensemble foundation model were evaluated on the WorldFloods v2 dataset and achieved state-of-the-art performance.

6.2 Methodology

This section first introduces the proposed interpretable decision-making module IDSS v2. It then details the training and testing process of the proposed architecture. Finally, the model selection and experiment setup are described.

6.2.1 IDSS v2 module

The proposed Interpretable Deep Semantic Segmentation v2 (IDSS v2) module is derived from the IDSS + module proposed in (Z. Zhang, P. Angelov, Kangin, et al., 2024). However, it is different from the IDSS + module, which uses the latent features and raw features of only one pixel to generate the prototypes, the IDSS v2 module provides not only the prototypes but also the surrounding context of the prototypes in an RGB visualization format. This can help human users to better understand the decision-making process of the model in a much more intuitive way. Besides, Besides, the IDSS v2 module was evaluated on more geoscience foundation models to test its generalization ability. For clarity, Table 6.1 summarizes the main differences among the three versions of the IDSS framework.

Next, the training and testing process of the proposed Interpretable Ensemble Geoscience Foundation Model (IEGF) that has integrated the IDSS v2 module is described.

Table 6.1: Comparison between IDSS, IDSS+, and IDSS v2.

Model	Prototype Type	Key Innovation	Interpretability
IDSS	Cluster centers (latent and raw feature space).	Introduced a prototype-based interpretable deep semantic segmentation approach with linguistic IF–THEN rules.	High — based on prototype similarity and rule-based reasoning.
IDSS+	Real pixels from training data (latent and raw feature space).	Extended IDSS by using real training samples as prototypes for more realistic explanations.	High — retains rule-based transparency with improved interpretability.
IDSS v2	Real pixels from training data (latent and raw feature space) with surrounding RGB context.	Adds prototype context visualization and integrates interpretability into ensemble foundation models.	Very high — combines prototypes with visual context for better human understanding.

6.2.2 Proposed IEGF framework

The training process of the proposed IEGF architecture is shown in Fig. 6.1.

Consider a semantic segmentation problem where each multispectral training image is denoted by $I \in \mathbb{R}^{h \times w \times n}$, with n representing the number of the multispectral bands. After fine-tuning with different Geoscience Foundation models, the IDSS v2 module is used to generate prototypes for each class. Specifically, for each foundation model, the latent features $F \in \mathbb{R}^{h \times w \times d}$, where d is the dimensionality of the learned latent feature space, are extracted from the last layer of the model. In this chapter, the validation set is used to generate prototypes. For each class $c \in \{1, \dots, C\}$, all pixel-wise features are then clustered to find K centroids in the feature space. The latent features closest to each centroid together with their corresponding ones in the raw feature space are considered K prototypes.

It can be seen from Fig. 6.1, that for the Worldfloods v2 dataset used in this chapter, there are only three classes: land, water and cloud. In the figure, different shapes and colours are used to represent each class, with blue rectangles denoting water, green triangles representing land and yellow circles indicating clouds.

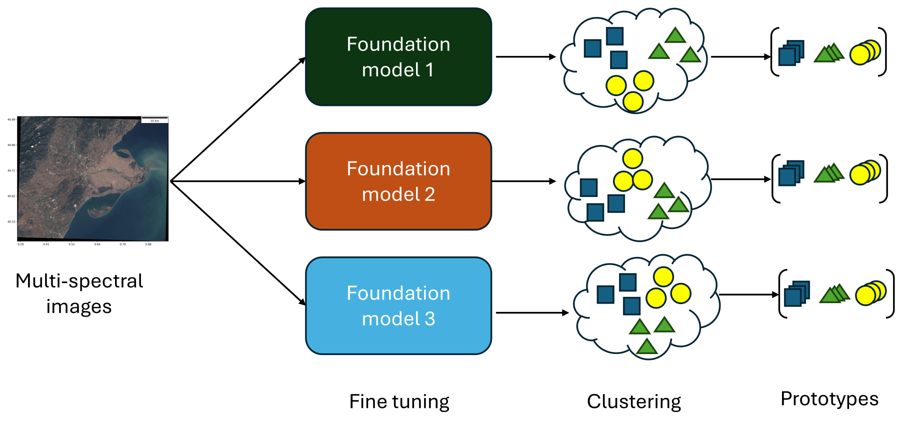


Figure 6.1: Training process. Multispectral images are passed through several fine-tuned geoscience foundation models to extract latent features. These features are clustered to obtain the prototypes for each class. In the WorldFloods v2 dataset shown here, water, land, and cloud classes are represented by blue squares, green triangles, and yellow circles, respectively.

The testing process of the proposed architecture is shown in Fig. 6.2.

Each test image was first passed through each different foundation model separately to obtain the latent features, as shown by the white pentagons in the figures. Formally, for a test image $I_{\text{test}} \in \mathbb{R}^{h \times w \times n}$, each foundation model outputs a feature map $F_i \in \mathbb{R}^{h \times w \times d}$ for $i = 1, 2, 3.$. Next, a distance metric S is computed between the feature vectors at each pixel level and the corresponding prototypes $P_i = \{p_{i,k}\}_{k=1}^{K}$, where $p_{i,k}$ is the k-th prototypes associated with foundation model i. Specifically, for each pixel x and foundation model i, the computation is given by:

$$s_i(x, k) = S\big(F_i(x),\, p_{i,k}\big) \tag{6.1}$$

Then, the class label with the lowest distance score was assigned to the pixel x. Finally, the predictions from all models are ensembled through majority voting to

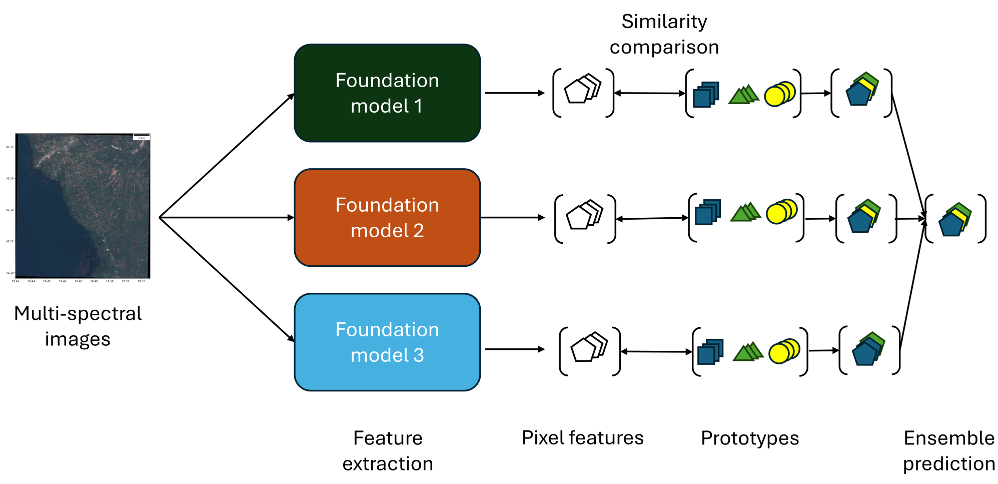


Figure 6.2: Testing process. Each multispectral test image is passed through the three foundation models to extract pixel-wise latent features. These features are compared with the prototypes of each model to compute similarity scores and assign a class label per pixel. The final prediction is obtained by majority voting across the three model outputs.

get the final prediction:

$$\hat{y}(x) = \mathrm{mode}\{y_1(x),\, y_2(x),\, y_3(x)\}. \tag{6.2}$$

In the current implementation of the proposed ensemble framework, all base models are assigned equal weights during the aggregation stage. This design choice was made to ensure a fair contribution from each model and to clearly demonstrate the impact of the interpretability mechanism itself, rather than confound it with additional optimization effects introduced by model weighting. Although weighting strategies were not considered in this work, future studies could explore their potential to further improve ensemble accuracy.

6.2.3 Model selection

Four Foundation Models were selected, including PhilEO, Satlas, DOFA and Prithvi. PhilEO (Le Saux et al., 2024) was pre-trained in a self-supervised manner on the ESA Major TOM (Francis and Czerkawski, 2024)dataset, which includes 11 bands of Sentinel-2 images at 10m resolution. SatlasNet was pre-trained on the NAIP and Sentinel-2 images in a supervised manner using the Swin Transformer (Z. Liu et al., 2021) backbone. DOFA was pre-trained on the Sentinel-1, Sentinel-2, Gaofen, NAIP, and EnMAP datasets in a self-supervised manner, while Prithvi (Jakubik et al., 2023) was pre-trained on the Harmonized Landsat and Sentinel-2 (HLS) data in a self-supervised manner with the Masked Autoencoder (MAE) structure (K. He, X. Chen, et al., 2022). Besides, I also use a U-Net (Ronneberger, Fischer, and Brox, 2015) and a DeepLab v3+ (L.-C. Chen et al., 2018) as benchmarks for comparison. The model details are shown in Table 6.2. It can be seen that the Phieo, SatlasNet, and Prithvi models have been pre-trained on the Sentinel-2 images, while the DOFA model has been pre-trained on datasets from multiple sensors. The U-Net and DeepLab v3+ models were pre-trained on the ImageNet dataset. The DOFA model has the most parameters with 151 million, while the U-Net model has the least with 7.8 million.

Table 6.2: Geoscience foundation model summary.

Model	Pretraining dataset	Pretraing image type	# Parameters (M)
U-Net	ImageNet	RGB	7.8
Deeplab v3+	ImageNet	RGB	45.7
Phieo	Major TOM	Sentinel-2	45.9
SatlasNet	SatlasPretrain	Sentinel 2	89.8
Prithvi	Harmonized Landsat Sentinel-2	Landsat Sentinel 2	127.0
DOFA	DOFA	Sentinel 1, Sentinel 2, Gaofen NAIP, EnMAP	151.0

6.3 Experiment setup

6.3.1 Dataset

The experiments were mainly conducted on the WorldFloods v2 dataset (Portalés-Julià et al., 2023), which is a flood segmentation dataset. It contains 114 flood events and 73,809 256 × 256 Sentinel-2 images. The dataset is split into 65,582 / 3,524 / 4,703 for train / val / test. There were mainly 3 classes: land, water and cloud.

6.3.2 GFM Fine Tuning Architecture

For the choice of model architecture, I adopted the same strategy as (Marsocci et al., 2024). Specifically, I followed the implementation as (Marsocci et al., 2024), which involves plugging in a decoder for each pre-trained Geoscience Foundation Model (GFM) encoder to produce dense predictions. Similar to (Marsocci et al., 2024), I utilized the UperNet (Xiao et al., 2018) decoder to fine-tune the GFM. The architecture is shown as Figure 6.3. However, I retained the original architecture of the Phieo model as its U-Net like architecture. Additionally, I trained two convolutional neural networks, U-Net (Ronneberger, Fischer, and Brox, 2015) and

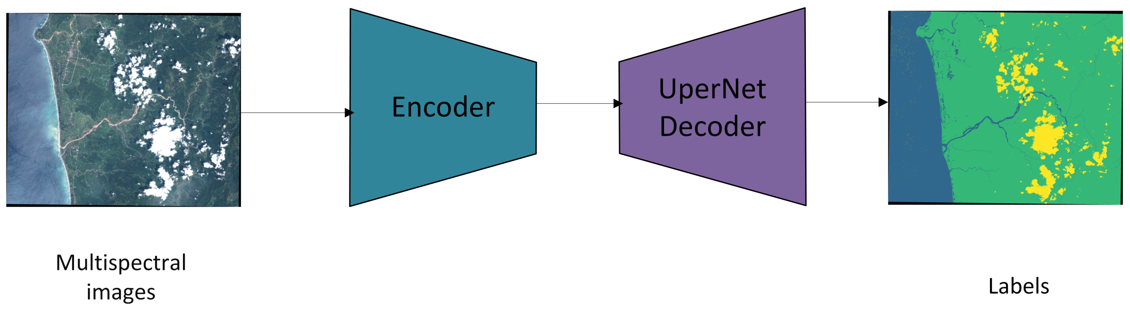


Figure 6.3: GFM fine tuning architecture.

Deeplabv3+ (L.-C. Chen et al., 2018), which were pre-trained on ImageNet for comparison. Both models use the ResNet101 encoder, and the implementations are from (Iakubovskii, 2019).

6.3.3 Fine Tuning strategy

To assess the ability of the GFM to generalize across different bands, I adopted two fine-tuning strategies. The first follows the pre-training setup of the GFM. Specifically, I use the same pre-processing and band selection as the pre-trained GFM. For example, the DOFA uses 9 bands for fine-tuning, and the preprocessing is to standardize the input band values by a given mean and standard deviation value. I then use the same pre-processing and 9 bands for fine-tuning with the Worldfloods v2 dataset. For the second strategy, I used all 13 bands from the Worldfloods v2 dataset at 10m resolution and normalized all input band values by dividing them by 10000 for fine-tuning, since these values range from 1 to 10000 (Gascon et al., 2017).

6.3.4 Prototype generation

I used the mini-batch K-means algorithm to generate the prototypes since it could handle larger datasets. Then, I used K-nearest neighbors to perform the decision-making process. I set the number of prototypes to 100 for each class and K

Table 6.3: Comparison of results for different bands on various foundation models. The table reports IoU scores for land, water, and cloud classes on the WorldFloods v2 dataset. For each GFM, two fine-tuning strategies are compared: (1) using the same band selection and preprocessing as in the model's original pre-training (e.g., Prithvi with 6 bands, DOFA with 9 bands), and (2) using all 13 multispectral bands.

Model	# Bands	IoU land (%)	IoU water (%)	IoU cloud (%)	Mean IoU (%)
Prithvi	6	86.55	72.00	67.17	75.05
Prithvi	13	86.92	73.76	66.71	**75.80 (↑0.75)**
DODA	9	82.01	68.58	53.93	68.17
DOFA	13	83.98	69.90	69.10	**74.33 (↑6.16)**
Phieo	10	86.39	73.51	67.31	75.74
Phieo	13	85.58	74.09	77.61	**79.09 (↑3.35)**
SatlasNet	9	87.75	74.39	68.38	76.84
SatlasNet	13	87.28	75.28	76.78	**79.78 (↑2.94)**
Unet	13	87.39	77.21	80.02	81.54
DeepLabv3+	13	**88.10**	**77.45**	**81.21**	**82.25**

to 20 for the K-nearest neighbors algorithm after multiple trials to find the best hyperparameters.

6.4 Results and Comparison

6.4.1 GFM generalization results

Table 6.3 shows the comparison of results for different bands on different GFMs. It can be seen that the performance of the GFMs generally improves with more bands. Specifically, the mean IoU increases by 0.75% for Prithvi, 6.16% for DOFA, 3.35% for Phieo, and 2.94% for SatlasNet. The results show that the GFMs can give better performance with more bands on the WorldFloods v2 dataset. However, despite the improvement, the performance of the GFMs is still lower than the U-Net and

Table 6.4: Performance comparison of various foundation models and the proposed method with IDSS v2 module.

Model	IoU land (%)	IoU water (%)	IoU cloud (%)	Mean IoU (%)
Prithvi (Jakubik et al., 2023)	86.92	73.76	66.71	75.80
Prithvi + IDSS v2 (ours)	87.76	73.32	70.08	**77.05 (↑1.25)**
DOFA (Xiong et al., 2024)	83.98	69.90	69.10	74.33
DOFA + IDSS v2 (ours)	87.26	73.49	76.27	**79.00 (↑4.67)**
Phieo (Saux et al., 2024)	85.58	74.09	77.61	79.09
Phieo + IDSS v2 (ours)	86.10	75.98	77.91	**80.00 (↑0.91)**
SatlasNet (Bastani et al., 2023)	87.28	75.28	76.78	79.78
SatlasNet + IDSS v2 (ours)	89.70	78.21	81.08	**83.00 (↑3.22)**
Unet (Ronneberger, Fischer, and Brox, 2015)	87.39	77.21	80.02	81.54
Unet + IDSS v2 (ours)	88.09	78.06	85.17	**83.77 (↑2.23)**
DeepLabv3+ (L.-C. Chen et al., 2018)	88.10	77.45	81.21	82.25
DeepLabv3+ + IDSS v2 (ours)	86.59	77.97	80.34	81.63(↓ 0.62)
EGFM	88.61	78.05	81.37	82.67
EGFM + IDSS v2 (ours)	**89.23**	**79.30**	**85.72**	**84.75 (↑2.08)**

DeepLabv3+ models. It can be observed that the DeepLabv3+ model outperforms all other models not only with 82.25% mean IoU but also with the highest IoU for each class.

6.4.2 GFM comparison with the proposed method

Table 6.4 shows the performance comparison of various GFMs and the proposed method. It can be seen that the proposed IDSS v2 module improves the performance of all the GFMs except for the DeepLabv3+ model. Specifically, the mean IoU increased by 1.25% for Prithvi, 4.67% for DOFA, 0.91% for Phieo, 3.22% for SatlasNet, and 2.23% for U-Net. I also proposed the Ensemble Foundation Model (EFM) which ensembles the three best-performing models, SatlasNet, U-Net, and DeepLabv3+. The results show that the EFM with the IDSS v2 module not only improves the performance of the EFM by 2.08% mean IoU but also outperforms all other models with the highest IoU for each class and the highest mean IoU. The results show that the proposed plug-and-play prototype-based module can effectively improve the performance of the GFMs on the WorldFloods v2 dataset.

Figure 6.4 shows the comparison of segmentation results on the Worldfloods v2 dataset. It can be observed from the last row that the cloud prediction of the SatlasNet + IDSS v2 is much consistent with the ground truth than the SatlasNet model. This indicates that the proposed IDSS v2 module effectively improves the segmentation performance of the SatlasNet model.

6.4.3 Prototype visulization

Figure 6.5 shows the visulization of the prototypes generated by the Prithvi and Unet models. It can be seen that the prototypes from the Prithvi model are more mixed with each other, while the prototypes from Unet are more clustered and separated. This indicates that the Prithvi model has more difficulty in separating the classes than the Unet model, which is consistent with the results in Table 6.4.

6.4.4 Interpretability

Unlike many other deep neural networks, the proposed architecture offers a human-understandable decision-making process that enables users to audit and understand the behaviour of the model. Figure 6.6 shows how a single foundation model makes decisions.

It can be observed that for a pixel in the test image, the proposed architecture can easily find the closest prototype in the latent feature space. In addition, it can further locate the exact position and show its surroundings in an RGB visualization format to human users. This can be very intuitive when explaining why a particular decision was made. As shown in the figure, the test pixel is highlighted in red circles, and the closest prototype pixel is highlighted in blue circles (The highlighted pixels have been bolded for better visualization). It can be observed that the model correctly predicts the pixel as water because the latent feature is closest to the water prototypes and the l_2 distance is 0.0567.It can be further observed that the

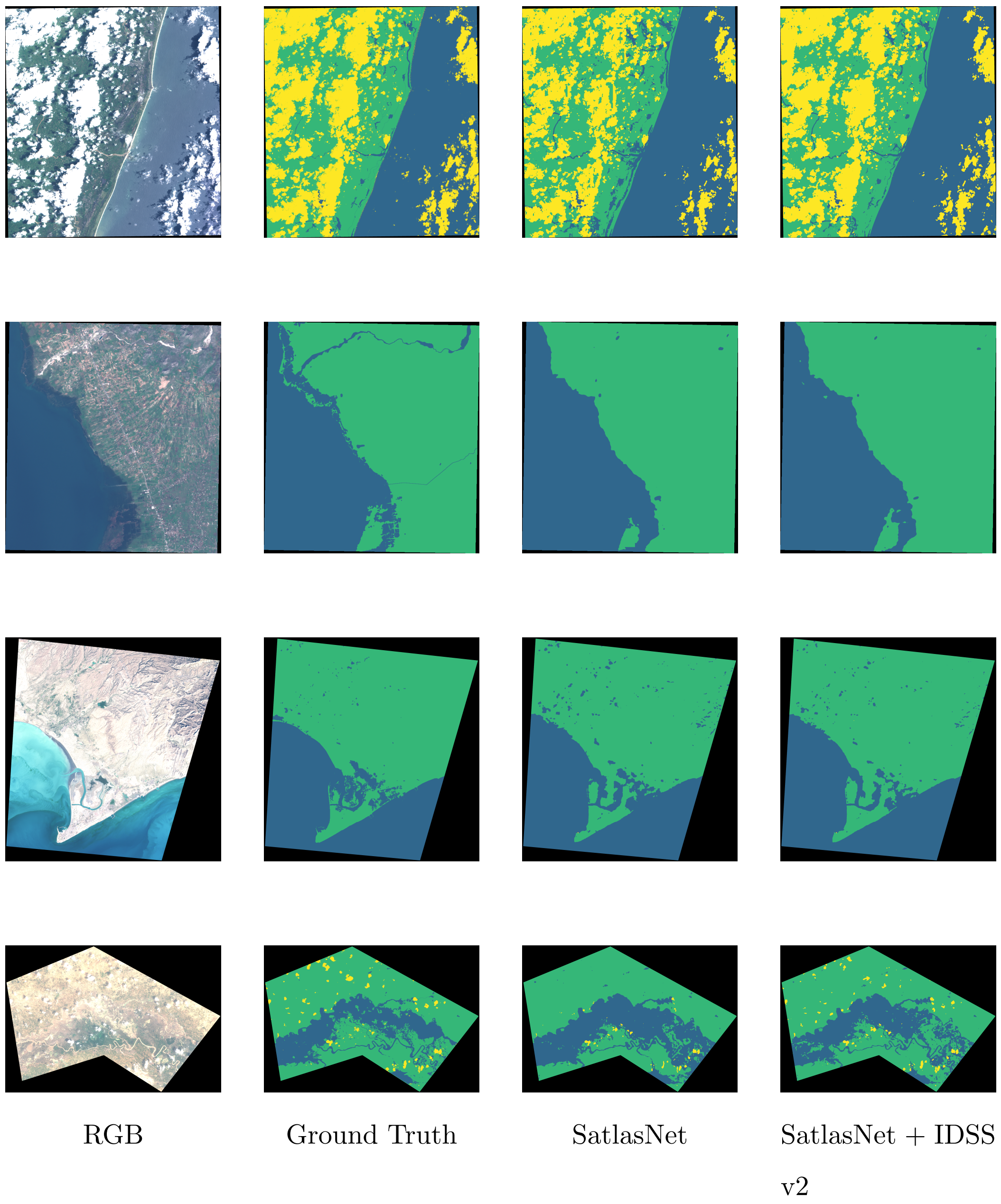
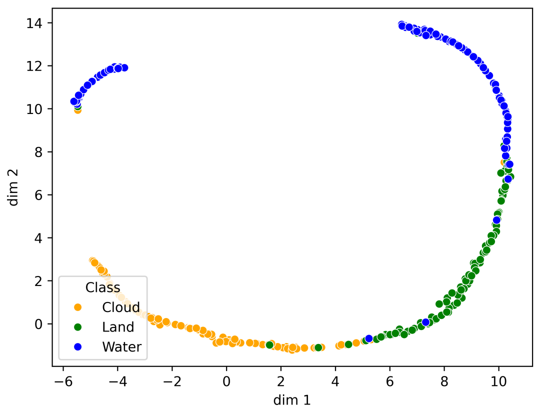
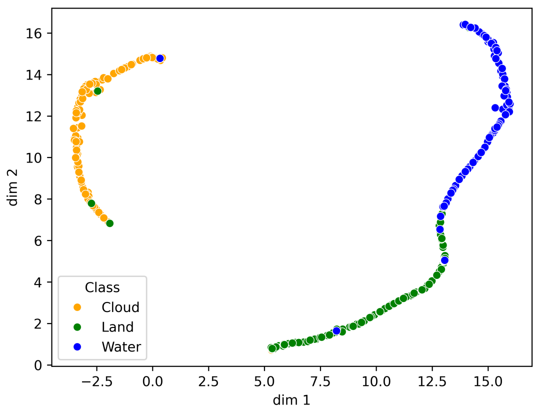


Figure 6.4: Comparison of segmentation segmentation results on Worldfloods v2 dataset.



(a) Prithvi



(b) Unet

Figure 6.5: Visualization with Umap for Prithvi and Unet prototypes.

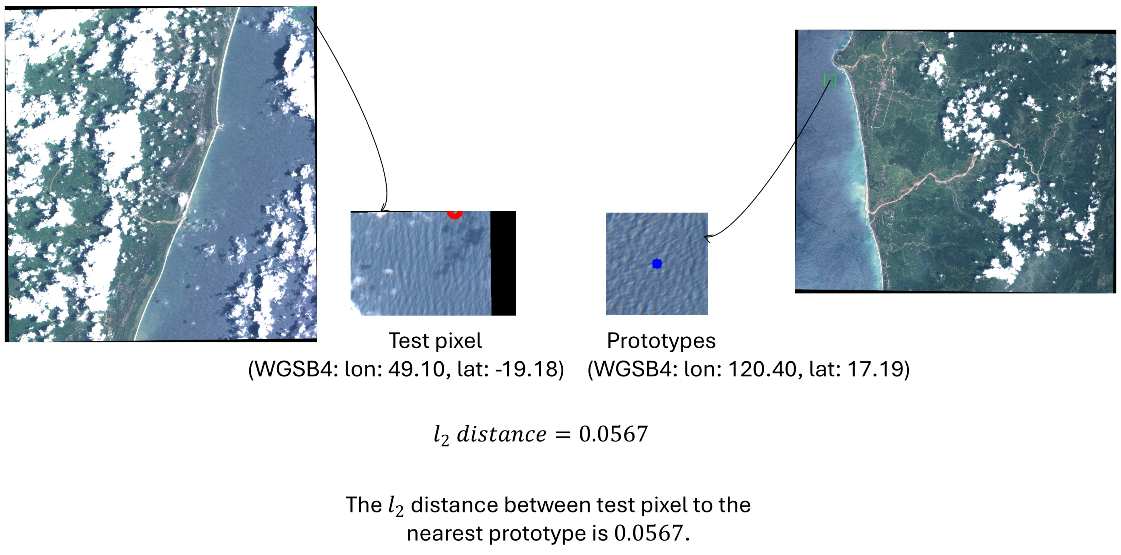

Test pixel
(WGSB4: lon: 49.10, lat: -19.18)
Prototypes
(WGSB4: lon: 120.40, lat: 17.19)
l_2 distance = 0.0567
The l_2 distance between test pixel to the nearest prototype is 0.0567.

Figure 6.6: Visualization of the decision making process of the model.

prototype pixel is surrounded by water pixels, which further confirms the decision made by the model. This effectively addresses the black-box nature of deep neural networks, which only provide a prediction without any explanation.

6.5 Conclusion

In this chapter, I evaluated the ability of several geoscience foundation models to generalize across different bands. The results show that the performance of the geoscience foundation models generally improves with more bands. I also proposed an interpretable plug-and-play prototype-based module IDSS v2, which can effectively improve the performance of the baseline geoscience foundation models and the ensemble foundation model. More importantly, the proposed module can provide a transparent decision-making process for human users. Finally, I evaluated the proposed interpretable ensemble foundation model on the WorldFloods v2 dataset and achieved state-of-the-art performance.

Chapter 7

Conclusions and Future Work

This thesis focuses on the research of interpretable deep learning methods and their application in the field of Earth Observation. The motivation and objectives of this thesis are outlined in Chapter 1. In Chapter 2, I reviewed the background knowledge related to this thesis, including the basic concept of machine learning, deep learning, image segmentation, remote sensing, deep learning in Earth Observation and Explainable AI. In Chapter 3, in collaboration with my co-authors, we introduced a general framework for interpretable deep learning called IDEAL and demonstrated it on the general image classification task. In Chapter 4, I proposed the IDSS method for the semantic segmentation task in remote sensing. It used the same prototype-based method as IDEAL but with a different task and scenario. In Chapter 5, I further extended the IDSS method and proposed a framework called IMAFD to provide a comprehensive solution for the flood detection task, which integrated with the updated prototype-based method IDSS +. In Chapter 6, I evaluated the ability of several geoscience foundation models to generalize across different bands. Furthermore, I proposed an interpretable ensemble foundation model, which integrated with the updated prototype-based method IDSS v2.

In the proceedings of this chapter, I will summarize the main contributions of this thesis and discuss future work. Section 7.1 will discuss the main contributions of this thesis. Section 7.2 will discuss the future work of this thesis.

7.1 Main Contributions

In this section, I reformulate the main contributions in the form of a list of questions and answers.

- Q1: How can an interpretable deep learning approach be developed to provide high performance while maintaining human-understandable explanations?
- A1: In collaboration with my co-authors, we proposed a general framework for interpretable deep learning called IDEAL. It utilized the feature space from the deep learning model and generated prototypes to provide human-understandable explanations. We demonstrated the IDEAL method on the general image classification task and achieved competitive performance while providing interpretability.
- Q2: How can an interpretable method be developed for semantic segmentation in remote sensing?
- A2: I proposed the IDSS method, which is an interpretable solution for the semantic segmentation task in remote sensing. Before this work, most works in interpretable deep learning focused on the image classification task. I extended the prototype-based method to the semantic segmentation task and demonstrated it on the remote sensing dataset.
- Q3: How can a comprehensive framework be developed for flood detection in remote sensing?
- A3: I proposed the IMAFD framework, which integrated with the IDSS + method to provide a comprehensive solution for the flood detection task. Different from other flood detection methods, the IMAFD framework solved the flood detection task step by step which starts from the time-series sequence level and ends with the pixel level. Besides, the IMAFD framework also provided human-understandable explanations for human users.

- Q4: How can a method be designed that benefits from geoscience foundation models and provides human-understandable explanations?

- A4: I evaluated the ability of several geoscience foundation models to generalize across different bands and proposed an interpretable ensemble foundation model. I further extended the IDSS and IDSS + methods and proposed the IDSS v2 method. Rather than providing pixel-level explanations, the IDSS v2 method provided more semantic-level explanations around the single pixel.

7.2 Future Work

With the rapid development of deep learning methods, such methods have become widely used across various industries, including Earth observation. Unlike traditional computer vision tasks, Earth observation problems are typically more data-intensive, complex, and often require models whose decision-making processes are transparent and trustworthy. However, traditional deep learning algorithms remain black boxes, and their lack of interpretability limits their applicability in high-stakes geoscience domains.

This thesis has explored prototype-based interpretability for both image classification and semantic segmentation, helping to bridge a gap in the literature where interpretable semantic segmentation has received relatively little attention. Nevertheless, several open challenges remain.

First, the distribution of satellite data can vary significantly across different regions. Most existing algorithms are designed for offline learning, where models lack the ability to continue learning. Such an approach is not ideal for real-world Earth observation systems, where new data arrive continuously and models must adapt to a changing environment. Developing online and continual learning approaches that preserve interpretability while maintaining performance is therefore a promising direction for future work.

Second, although this thesis has proposed several prototype-based interpretable deep learning methods, the underlying prototypes are still derived from the latent feature space of deep neural networks. Since latent features themselves remain difficult to interpret, how to make the latent space more interpretable is an important direction for future research.

Third, recent Earth observation missions typically involve multimodality data, such as multispectral, SAR, LiDAR and textual data. Exploring ways to leverage such data to improve the interpretability and performance of prototype-based models is a promising area of research. For example, prototypes from different modalities could be extracted in different latent spaces and then combined to create multimodality prototypes. This approach enables prototypes to capture cross-modality relationships, thereby enriching model explanations and performance.

Finally, while this paper proposes several interpretable approaches to classification and semantic segmentation, there are still areas that could be explored further in future studies. These could include alternative prototype selection methods, applying prototypes within large language models and investigating novel interpretability techniques.

Overall, the author wishes this thesis could provide some inspiration for future research in the field of interpretable deep learning and earth observation.

References

Achanta, Radhakrishna et al. (2012). “SLIC superpixels compared to state-of-the-art superpixel methods”. In: *IEEE transactions on pattern analysis and machine intelligence* 34.11, pp. 2274–2282.

Adebayo, Julius et al. (2018). “Sanity checks for saliency maps”. In: *Advances in neural information processing systems* 31.

Ahmad, Muhammad Aurangzeb, Carly Eckert, and Ankur Teredesai (2018). “Interpretable machine learning in healthcare”. In: *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, pp. 559–560.

Alfieri, Lorenzo et al. (2017). “Global projections of river flood risk in a warmer world”. In: *Earth’s Future* 5.2, pp. 171–182.

Angelov, Plamen (2012). *Autonomous learning systems: from data streams to knowledge in real-time*. John Wiley & Sons.

Angelov, Plamen, Dmitry Kangin, and **Ziyang Zhang** (2025). “IDEAL: Interpretable-by-Design ALgorithms for learning from foundation feature spaces”. In: *Neurocomputing*, p. 129464.

Angelov, Plamen, Edwin Lughofer, and Xiaowei Zhou (2008). “Evolving fuzzy classifiers using different model architectures”. In: *Fuzzy sets and systems* 159.23, pp. 3160–3182.

Angelov, Plamen, Pouria Sadeghi-Tehran, and Ramin Ramezani (2011). “An approach to automatic real-time novelty detection, object identification, and tracking in video streams based on recursive density estimation and evolving

Takagi–Sugeno fuzzy systems". In: *International Journal of Intelligent Systems* 26.3, pp. 189–205.

Angelov, Plamen and Eduardo Soares (2020). "Towards explainable deep neural networks (xDNN)". In: *Neural Networks* 130, pp. 185–194.

Angelov, Plamen P and Xiaowei Gu (2019). *Empirical approach to machine learning.* Springer.

Aronszajn, Nachman (1950). "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3, pp. 337–404.

Atrey, Akanksha, Kaleigh Clary, and David Jensen (2019). "Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning". In: *arXiv preprint arXiv:1912.05743.*

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12, pp. 2481–2495.

Bai, Ting et al. (2023). "Deep learning for change detection in remote sensing: a review". In: *Geo-spatial Information Science* 26.3, pp. 262–288.

Baruah, Rashmi Dutta and Plamen Angelov (2012). "Evolving local means method for clustering of streaming data". In: *2012 IEEE international conference on fuzzy systems.* IEEE, pp. 1–8.

Bastani, Favyen et al. (2023). "Satlaspretrain: A large-scale dataset for remote sensing image understanding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 16772–16782.

Bengio, Yoshua, Ian Goodfellow, and Aaron Courville (2017). *Deep learning.* Vol. 1. MIT press Cambridge, MA, USA.

Bentivoglio, Roberto et al. (2022). "Deep learning methods for flood mapping: a review of existing applications and future research directions". In: *Hydrology and Earth System Sciences* 26.16, pp. 4345–4378.

Berezina, Polina and Desheng Liu (2022). "Hurricane damage assessment using coupled convolutional neural networks: a case study of hurricane Michael". In: *Geomatics, Natural Hazards and Risk* 13.1, pp. 414–431.

Bertsimas, Dimitris, Angela King, and Rahul Mazumder (2016). "Best subset selection via a modern optimisation lens". In: *The Annals of Statistics* 44.2, pp. 813–852.

Bhaskaran, Sunil et al. (2013). "Rule-based classification of high-resolution imagery over urban areas in New York City". In: *Geocarto International* 28.6, pp. 527–545.

Biehl, Michael, Barbara Hammer, and Thomas Villmann (2016). "Prototype-based models in machine learning". In: *Wiley Interdisciplinary Reviews: Cognitive Science* 7.2, pp. 92–111.

Bien, Jacob and Robert Tibshirani (2011). "Prototype selection for interpretable classification". In: *The Annals of Applied Statistics*, pp. 2403–2424.

Biggio, Battista et al. (2013). "Evasion attacks against machine learning at test time". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part III 13*. Springer, pp. 387–402.

Birhane, Abeba et al. (2023). "Into the LAIONs Den: Investigating Hate in Multimodal Datasets". In: *arXiv preprint arXiv:2311.03449*.

Böhle, Moritz, Mario Fritz, and Bernt Schiele (2022). "B-cos Networks: Alignment is All We Need for Interpretability". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10329–10338.

Böhle, Moritz, Navdeeppal Singh, et al. (2024). "B-cos Alignment for Inherently Interpretable CNNs and Vision Transformers". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Bommasani, Rishi et al. (2021). "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258*.

Bonafilia, Derrick et al. (2020). “Sen1Floods11: A georeferenced dataset to train and test deep learning flood algorithms for sentinel-1”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 210–211.

Bontempelli, Andrea et al. (2022). “Concept-level debugging of part-prototype networks”. In: *arXiv preprint arXiv:2205.15769*.

Boser, Bernhard, Isabelle Guyon, and Vladimir Vapnik (1992). “A training algorithm for optimal margin classifiers”. In: *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152.

Brown, Tom et al. (2020). “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33, pp. 1877–1901.

Camps-Valls, Gustavo et al. (2011). “Remote sensing image processing”. In.

Chan, Chun Sik, Huanqi Kong, and Guanqing Liang (2022). “A comparative study of faithfulness metrics for model interpretability methods”. In: *arXiv preprint arXiv:2204.05514*.

Chen, Chaofan et al. (2019). “This looks like that: deep learning for interpretable image recognition”. In: *Advances in neural information processing systems* 32.

Chen, Hao, Zipeng Qi, and Zhenwei Shi (2021). “Remote sensing image change detection with transformers”. In: *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–14.

Chen, Liang-Chieh et al. (2018). “Encoder-decoder with atrous separable convolution for semantic image segmentation”. In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818.

Clare, Mariana CA et al. (2022). “Explainable artificial intelligence for bayesian neural networks: toward trustworthy predictions of ocean dynamics”. In: *Journal of Advances in Modeling Earth Systems* 14.11, e2022MS003162.

Coates, Adam, Andrew Ng, and Honglak Lee (2011). “An analysis of single-layer networks in unsupervised feature learning”. In: *Proceedings of the fourteenth* *international conference on artificial intelligence and statistics.* JMLR Workshop and Conference Proceedings, pp. 215–223.

Cortes, Corinna and Vladimir Vapnik (1995). “Support-vector networks”. In: *Machine learning* 20.3, pp. 273–297.

Cover, Thomas and Peter Hart (1967). “Nearest neighbor pattern classification”. In: *IEEE transactions on information theory* 13.1, pp. 21–27.

Covert, Ian Connick, Chanwoo Kim, and Su-In Lee (2023). “Learning to Estimate Shapley Values with Vision Transformers”. In: *The Eleventh International Conference on Learning Representations, 2023*.

Cuturi, Marco (2013). “Sinkhorn distances: Lightspeed computation of optimal transport”. In: *Advances in neural information processing systems* 26.

Dimitrovski, Ivica et al. (2023). “Current trends in deep learning for Earth Observation: An open-source benchmark arena for image classification”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 197, pp. 18–35.

Dosovitskiy, Alexey et al. (2020). “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929*.

Drusch, Matthias et al. (2012). “Sentinel-2: ESA’s optical high-resolution mission for GMES operational services”. In: *Remote sensing of Environment* 120, pp. 25–36.

Fayshal, Md Atik et al. (2024). “Unveiling the impact of rapid urbanization on human comfort: a remote sensing-based study in Rajshahi Division, Bangladesh”. In: *Environment, Development and Sustainability*, pp. 1–35.

Fibaek, Casper et al. (2024). “PhilEO Bench: Evaluating Geo-Spatial Foundation Models”. In: *arXiv:2401.04464*.

Francis, Alistair and Mikolaj Czerkawski (2024). “Major TOM: Expandable Datasets for Earth Observation”. In: *arXiv preprint arXiv:2402.12095*.

French, Robert (1999). “Catastrophic forgetting in connectionist networks”. In: *Trends in cognitive sciences* 3.4, pp. 128–135.

Gao, Bo-Cai (1996). “NDWI—A normalized difference water index for remote sensing of vegetation liquid water from space”. In: *Remote sensing of environment* 58.3, pp. 257–266.

Garreau, Damien and Dina Mardaoui (2021). “What does LIME really see in images?” In: *International conference on machine learning*. PMLR, pp. 3620–3629.

Gascon, Ferran et al. (2017). “Copernicus Sentinel-2A calibration and products validation status”. In: *Remote Sensing* 9.6, p. 584.

Gevaert, Caroline M (2022). “Explainable AI for earth observation: A review including societal and regulatory perspectives”. In: *International Journal of Applied Earth Observation and Geoinformation* 112, p. 102869.

Ghorbanzadeh, Omid et al. (2022). “Landslide detection using deep learning and object-based image analysis”. In: *Landslides* 19.4, pp. 929–939.

Gong, Maoguo, Yu Cao, and Qiaodi Wu (2011). “A neighborhood-based ratio approach for change detection in SAR images”. In: *IEEE Geoscience and Remote Sensing Letters* 9.2, pp. 307–311.

Goodfellow, Ian et al. (2016). *Deep learning*. Vol. 1. 2. MIT press Cambridge.

Greenland, Sander, Judea Pearl, and James M Robins (1999). “Confounding and collapsibility in causal inference”. In: *Statistical science* 14.1, pp. 29–46.

Guo, Daya et al. (2025). “DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning”. In: *arXiv preprint arXiv:2501.12948*.

Guo, Xianpeng et al. (2022). “Prob-POS: A Framework for Improving Visual Explanations from Convolutional Neural Networks for Remote Sensing Image Classification”. In: *Remote Sensing* 14.13, p. 3042.

Gupta, Bhumika et al. (2017). “Analysis of various decision tree algorithms for classification in data mining”. In: *International Journal of Computer Applications* 163.8, pp. 15–19.

Hao, Ming et al. (2023). “Bi-Temporal change detection of high-resolution images by referencing time series medium-resolution images”. In: *International Journal of Remote Sensing* 44.11, pp. 3333–3357.

He, Kaiming, Xinlei Chen, et al. (2022). “Masked autoencoders are scalable vision learners”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009.

He, Kaiming, Xiangyu Zhang, et al. (2016). “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Helber, Patrick et al. (2018). “Introducing EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification”. In: *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, pp. 204–207.

— (2019). “Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.

Höhl, Adrian et al. (2024). “Opening the Black Box: A systematic review on explainable artificial intelligence in remote sensing”. In: *IEEE Geoscience and Remote Sensing Magazine*.

Hung, Sheng-Chieh, Hui-Ching Wu, and Ming-Hseng Tseng (2020). “Remote sensing scene classification and explanation using RSSCNet and LIME”. In: *Applied Sciences* 10.18, p. 6151.

Iakubovskii, Pavel (2019). *Segmentation Models Pytorch*. `https://github.com/qubvel/segmentation_models.pytorch`.

Jakubik, Johannes et al. (2023). “Foundation models for generalist geospatial artificial intelligence”. In: *arXiv preprint arXiv:2310.18660*.

Janizadeh, Saeid et al. (2021). “Mapping the spatial and temporal variability of flood hazard affected by climate and land-use changes in the future”. In: *Journal of Environmental Management* 298, p. 113551.

Jolliffe, Ian T and Jorge Cadima (2016). “Principal component analysis: a review and recent developments”. In: *Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences* 374.2065, p. 20150202.

Jordan, Michael I and Tom M Mitchell (2015). “Machine learning: Trends, perspectives, and prospects”. In: *Science* 349.6245, pp. 255–260.

Katiyar, Vaibhav, Nopphawan Tamkuan, and Masahiko Nagai (2021). “Near-real-time flood mapping using off-the-shelf models with SAR imagery and deep learning”. In: *Remote Sensing* 13.12, p. 2334.

Kaufman, Leonard (1990). “Partitioning around medoids (program pam)”. In: *Finding groups in data* 344, pp. 68–125.

Kim, Been, Cynthia Rudin, and Julie A Shah (2014). “The bayesian case model: A generative approach for case-based reasoning and prototype classification”. In: *Advances in neural information processing systems* 27.

Kim, Jinkyu and John Canny (2017). “Interpretable learning for self-driving cars by visualizing causal attention”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 2942–2950.

Kirkpatrick, James et al. (2017). “Overcoming catastrophic forgetting in neural networks”. In: *Proceedings of the national academy of sciences* 114.13, pp. 3521–3526.

Kornblith, Simon, Jonathon Shlens, and Quoc V Le (2019). “Do better imagenet models transfer better?” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2661–2671.

Krizhevsky, Alex and Geoffrey Hinton (2009). *Learning multiple layers of features from tiny images.* Tech. rep. 0. Toronto, Ontario: University of Toronto.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105.

Lamers, Christiaan et al. (2023). “Clustering-Based Domain-Incremental Learning”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3384–3392.

Le Saux, Bertrand et al. (2024). “The PhilEO geospatial foundation model suite”. In: *EGU General Assembly Conference Abstracts*, p. 17934.

LeCun, Yann et al. (1989). “Backpropagation applied to handwritten zip code recognition”. In: *Neural computation* 1.4, pp. 541–551.

Li, Chengxiu et al. (2022). “Increased flooded area and exposure in the White Volta river basin in Western Africa, identified from multi-source remote sensing data”. In: *Scientific Reports* 12.1, p. 3701.

Li, Fei-Fei, Rob Fergus, and Pietro Perona (2006). “One-shot learning of object categories”. In: *IEEE transactions on pattern analysis and machine intelligence* 28.4, pp. 594–611.

Li, Zhizhong and Derek Hoiem (2017). “Learning without forgetting”. In: *IEEE transactions on pattern analysis and machine intelligence* 40.12, pp. 2935–2947.

Liu, Sicong et al. (2015). “Sequential spectral change vector analysis for iteratively discovering and detecting multiple changes in hyperspectral images”. In: *IEEE transactions on geoscience and remote sensing* 53.8, pp. 4363–4378.

Liu, Ze et al. (2021). “Swin transformer: Hierarchical vision transformer using shifted windows”. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022.

Lloyd, Stuart (1982). “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2, pp. 129–137.

Loh, Wei-Yin (2011). “Classification and regression trees”. In: *Wiley interdisciplinary reviews: data mining and knowledge discovery* 1.1, pp. 14–23.

Long, Jonathan, Evan Shelhamer, and Trevor Darrell (2015). “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.

Lucieri, Adriano, Andreas Dengel, and Sheraz Ahmed (2023). “Translating theory into practice: assessing the privacy implications of concept-based explanations for biomedical AI”. In: *Frontiers in Bioinformatics* 3.

Lundberg, Scott (2017). “A unified approach to interpreting model predictions”. In: *arXiv preprint arXiv:1705.07874*.

MacKay, David JC (2003). *Information theory, inference and learning algorithms.* Cambridge university press.

MacQueen, J (1967). “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press.*

Malila, William A (1980). “Change vector analysis: An approach for detecting forest changes with Landsat”. In: *LARS symposia*, p. 385.

Mall, Utkarsh, Bharath Hariharan, and Kavita Bala (2022). “Change event dataset for discovery from spatio-temporal remote sensing imagery”. In: *Advances in Neural Information Processing Systems* 35, pp. 27484–27496.

Marsocci, Valerio et al. (2024). “PANGAEA: A global and inclusive benchmark for geospatial foundation models”. In: *arXiv preprint arXiv:2412.04204.*

Mateo-Garcia, Gonzalo et al. (2021). “Towards global flood mapping onboard low cost satellites with machine learning”. In: *Scientific reports* 11.1, pp. 1–12.

McFeeters, Stuart K (1996). “The use of the Normalized Difference Water Index (NDWI) in the delineation of open water features”. In: *International journal of remote sensing* 17.7, pp. 1425–1432.

Minaee, Shervin et al. (2021). “Image segmentation using deep learning: A survey”. In: *IEEE transactions on pattern analysis and machine intelligence* 44.7, pp. 3523–3542.

Mitros, John and Brian Mac Namee (2019). “On the validity of Bayesian neural networks for uncertainty estimation”. In: *arXiv preprint arXiv:1912.01530.*

Molnar, Christoph (2020). *Interpretable machine learning.* Lulu. com.

Munasinghe, Dinuke et al. (2023). “A multi-sensor approach for increased measurements of floods and their societal impacts from space”. In: *Communications Earth & Environment* 4.1, p. 462.

Murphy, Kevin P (2012). *Machine learning: a probabilistic perspective.* MIT press.

— (2022). *Probabilistic machine learning: an introduction.* MIT press.

Natarajan, Balas Kausik (1995). “Sparse approximate solutions to linear systems”. In: *SIAM journal on computing* 24.2, pp. 227–234.

Nauta, Meike, Ron Van Bree, and Christin Seifert (2021). “Neural prototype trees for interpretable fine-grained image recognition”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14933–14943.

Newell, Allen, John C Shaw, and Herbert A Simon (1959). “Report on a general problem solving program”. In: *IFIP congress*. Vol. 256. Pittsburgh, PA, p. 64.

Omia, Emmanuel et al. (2023). “Remote sensing in field crop monitoring: A comprehensive review of sensor systems, data analyses and recent advances”. In: *Remote Sensing* 15.2, p. 354.

Oquab, Maxime et al. (2023). “Dinov2: Learning robust visual features without supervision”. In: *arXiv preprint arXiv:2304.07193*.

Otsu, Nobuyuki et al. (1975). “A threshold selection method from gray-level histograms”. In: *Automatica* 11.285-296, pp. 23–27.

Parisi, German et al. (2019). “Continual lifelong learning with neural networks: A review”. In: *Neural networks* 113, pp. 54–71.

Parkhi, Omkar et al. (2012). “Cats and dogs”. In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE, pp. 3498–3505.

Pedregosa, Fabian et al. (2011). “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12, pp. 2825–2830.

Persello, Claudio et al. (2022). “Deep learning and earth observation to support the sustainable development goals: Current approaches, open challenges, and future opportunities”. In: *IEEE Geoscience and Remote Sensing Magazine* 10.2, pp. 172–200.

Peters, Uwe (2022). “Explainable AI lacks regulative reasons: why AI and human decision-making are not equally opaque”. In: *AI and Ethics*, pp. 1–12.

Poggio, Tomaso and Federico Girosi (1998). “A sparse representation for function approximation”. In: *Neural computation* 10.6, pp. 1445–1454.

Portalés-Julià, Enrique et al. (2023). “Global flood extent segmentation in optical satellite images”. In: *Scientific Reports* 13.1, p. 20316.

Radford, Alec et al. (2021). “Learning transferable visual models from natural language supervision”. In: *International conference on machine learning*. PMLR, pp. 8748–8763.

Radovanovic, Milos, Alexandros Nanopoulos, and Mirjana Ivanovic (2010). “Hubs in space: Popular nearest neighbors in high-dimensional data”. In: *Journal of Machine Learning Research* 11.sept, pp. 2487–2531.

Ras, Gabrielle et al. (2022). “Explainable deep learning: A field guide for the uninitiated”. In: *Journal of Artificial Intelligence Research* 73, pp. 329–396.

Rashkovetsky, Dmitry et al. (2021). “Wildfire detection from multisensor satellite imagery using deep semantic segmentation”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14, pp. 7001–7016.

Rebuffi, Sylvestre-Alvise et al. (2017). “iCaRL: Incremental Classifier and Representation Learning”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 5533–5542.

Regulation, Protection (2016). “Regulation (EU) 2016/679 of the European Parliament and of the Council”. In: *Regulation (eu)* 679, p. 2016.

Reichstein, Markus et al. (2019). “Deep learning and process understanding for data-driven Earth system science”. In: *Nature* 566.7743, pp. 195–204.

Rentschler, Jun, Melda Salhab, and Bramka Arga Jafino (2022). “Flood exposure and poverty in 188 countries”. In: *Nature communications* 13.1, p. 3527.

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin (2016). “” Why should i trust you?” Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). “U-net: Convolutional networks for biomedical image segmentation”. In: *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international confer-*

ence, Munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer, pp. 234–241.

Rosenblatt, Frank et al. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Vol. 55. Spartan books Washington, DC.

Rudin, Cynthia (2019). "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5, pp. 206–215.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536.

Ruschemeier, Hannah (2023). "AI as a challenge for legal regulation–the scope of application of the artificial intelligence act proposal". In: *Era Forum*. Vol. 23. 3. Springer, pp. 361–376.

Ruvolo, Paul and Eric Eaton (2013). "ELLA: An efficient lifelong learning algorithm". In: *International conference on machine learning*. PMLR, pp. 507–515.

Růžička, Vít et al. (2022). "RaVÆn: unsupervised change detection of extreme events using ML on-board satellites". In: *Scientific reports* 12.1, p. 16939.

Sammut, Claude and Geoffrey I Webb (2017). *Encyclopedia of machine learning and data mining*. Springer Publishing Company, Incorporated.

Sathya, Bharath and R Manavalan (2011). "Image segmentation by clustering methods: performance analysis". In: *International Journal of Computer Applications* 29.11, pp. 27–32.

Saux, Bertrand Le et al. (2024). "The PhilEO Geospatial Foundation Model Suite". In: *EGU*.

Schölkopf, Bernhard, Ralf Herbrich, and Alex J Smola (2001). "A generalized representer theorem". In: *Computational Learning Theory: 14th Annual Conference on Computational Learning Theory, COLT 2001 and 5th European Conference on Computational Learning Theory, EuroCOLT 2001 Amsterdam, The Netherlands, July 16–19, 2001 Proceedings 14*. Springer, pp. 416–426.

Schowengerdt, Robert A (2006). *Remote sensing: models and methods for image processing*. elsevier.

Sculley, David (2010). "Web-scale k-means clustering". In: *Proceedings of the 19th international conference on World wide web*, pp. 1177–1178.

Selvaraju, Ramprasaath R et al. (2017). "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.

Shafique, Ayesha et al. (2022). "Deep learning-based change detection in remote sensing images: A review". In: *Remote Sensing* 14.4, p. 871.

Shapley, Lloyd S (1953). "A value for n-person games". In: *Contribution to the Theory of Games* 2.

Shastry, Apoorva et al. (2023). "Mapping floods from remote sensing data and quantifying the effects of surface obstruction by clouds and vegetation". In: *Remote Sensing of Environment* 291, p. 113556.

Shewhart, Walter Andrew et al. (1986). *Wiley series in probability and statistics*. Wiley Online Library.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman (2014). "Deep inside convolutional networks: Visualising image classification models and saliency maps". In.

Simonyan, Karen and Andrew Zisserman (2014). "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556*.

Singh, Mannat et al. (2022). "Revisiting Weakly Supervised Pre-Training of Visual Perception Models". In: *CVPR*.

Singh, Prabhishek et al. (2021). "A Review on SAR Image and its Despeckling". In: *Archives of Computational Methods in Engineering* 28, pp. 4633–4653.

Smola, Alex and Bernhard Schölkopf (2004). "A tutorial on support vector regression". In: *Statistics and computing* 14, pp. 199–222.

Snell, Jake, Kevin Swersky, and Richard Zemel (2017). "Prototypical networks for few-shot learning". In: *Advances in neural information processing systems* 30.

Soares, Eduardo, Plamen Angelov, and **Ziyang Zhang** (2021). “An explainable approach to deep learning from CT-scans for covid identification”. In.

Steinhaus, Hugo et al. (1956). “Sur la division des corps matériels en parties”. In: *Bull. Acad. Polon. Sci* 1.804, p. 801.

Strudel, Robin et al. (2021). “Segmenter: Transformer for semantic segmentation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 7262–7272.

Tellman, B et al. (2021). “Satellite imaging reveals increased proportion of population exposed to floods”. In: *Nature* 596.7870, pp. 80–86.

Tipping, Michael (1999). “The relevance vector machine”. In: *Advances in neural information processing systems* 12.

— (2001). “Sparse Bayesian learning and the relevance vector machine”. In: *Journal of machine learning research* 1.Jun, pp. 211–244.

Torres, Ramon et al. (2021). “The Sentinel-1C/-1D Development and Deployment Plan”. In: *EUSAR 2021; 13th European Conference on Synthetic Aperture Radar*. VDE, pp. 1–4.

Vali, Ava, Sara Comai, and Matteo Matteucci (2020). “Deep learning for land use and land cover classification based on hyperspectral and multispectral earth observation data: A review”. In: *Remote Sensing* 12.15, p. 2495.

Vaswani, A (2017). “Attention is all you need”. In: *Advances in Neural Information Processing Systems*.

Ven, Gido van de, Tinne Tuytelaars, and Andreas S Tolias (2022). “Three types of incremental learning”. In: *Nature Machine Intelligence*, pp. 1–13.

Wachter, Sandra, Brent Mittelstadt, and Chris Russell (2017). “Counterfactual explanations without opening the black box: Automated decisions and the GDPR”. In: *Harv. JL & Tech.* 31, p. 841.

Wah, Catherine et al. (2011). “The caltech-ucsd birds-200-2011 dataset”. In.

Wang, Risheng et al. (2022). “Medical image segmentation using deep learning: A survey”. In: *IET image processing* 16.5, pp. 1243–1267.

Wang, Wenguan et al. (2022). “Visual recognition with deep nearest centroids”. In: *arXiv preprint arXiv:2209.07383*.

— (2023). “Visual recognition with deep nearest centroids”. In: *International Conference on Learning Representations (ICLR)*.

Wang, Yabin, Zhiwu Huang, and Xiaopeng Hong (2022). “S-prompts learning with pre-trained transformers: An occam’s razor for domain incremental learning”. In: *Advances in Neural Information Processing Systems* 35, pp. 5682–5695.

Wang, Yi et al. (2022). “SSL4EO-S12: A Large-Scale Multi-Modal, Multi-Temporal Dataset for Self-Supervised Learning in Earth Observation”. In: *arXiv preprint arXiv:2211.07044*.

Wang, Zhen et al. (2022). “Online Continual Learning with Contrastive Vision Transformer”. In: *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XX*. Springer, pp. 631–650.

Wei, Dennis et al. (2022). “On the Safety of Interpretable Machine Learning: A Maximum Deviation Approach”. In: *Advances in Neural Information Processing Systems* 35, pp. 9866–9880.

Widmer, Gerhard and Miroslav Kubat (1996). “Learning in the presence of concept drift and hidden contexts”. In: *Machine learning* 23, pp. 69–101.

Williams, Darrel L, Samuel Goward, and Terry Arvidson (2006). “Landsat”. In: *Photogrammetric Engineering & Remote Sensing* 72.10, pp. 1171–1178.

Wu, Han et al. (2022). “Flood Detection in Dual-Polarization SAR Images Based on Multi-Scale Deeplab Model”. In: *Remote Sensing* 14.20, p. 5181.

Wu, Zijing et al. (2024). “Satellite-based monitoring of the world’s largest terrestrial mammal migration using deep learning”. In: *EGU General Assembly 2024*.

Xiao, Tete et al. (2018). “Unified perceptual parsing for scene understanding”. In: *Proceedings of the European conference on computer vision (ECCV)*, pp. 418–434.

Xie, Enze et al. (2021). “SegFormer: Simple and efficient design for semantic segmentation with transformers”. In: *Advances in Neural Information Processing Systems* 34, pp. 12077–12090.

Xiong, Zhitong et al. (2024). “Neural plasticity-inspired multimodal foundation model for Earth observation”. In: *arXiv preprint arXiv:2403.15356*.

Xu, Hanqiu (2006). “Modification of normalised difference water index (NDWI) to enhance open water features in remotely sensed imagery”. In: *International journal of remote sensing* 27.14, pp. 3025–3033.

Yan, Jining et al. (2019). “A time-series classification approach based on change detection for rapid land cover mapping”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 158, pp. 249–262.

Yan, Shipeng, Jiangwei Xie, and Xuming He (2021). “Der: Dynamically expandable representation for class incremental learning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3014–3023.

Yang, Wenli et al. (2023). “Survey on explainable AI: From approaches, limitations and applications aspects”. In: *Human-Centric Intelligent Systems* 3.3, pp. 161–188.

Yu, Ying et al. (2023). “Techniques and challenges of image segmentation: A review”. In: *Electronics* 12.5, p. 1199.

Yuan, Xiaohui, Jianfang Shi, and Lichuan Gu (2021). “A review of deep learning methods for semantic segmentation of remote sensing imagery”. In: *Expert Systems with Applications* 169, p. 114417.

Zeithamova, Dagmar, W Todd Maddox, and David M Schnyer (2008). “Dissociable prototype learning systems: evidence from brain imaging and behavior”. In: *Journal of Neuroscience* 28.49, pp. 13194–13201.

Zhang, Ziyang, Plamen Angelov, Dmitry Kangin, et al. (2024). “IMAFD: An Interpretable Multi-stage Approach to Flood Detection from time series Multispectral Data”. In: *arXiv preprint arXiv:2405.07916*. Major revision with Applied Soft Computing journal.

Zhang, Ziyang, Plamen Angelov, Eduardo Soares, et al. (2022). “An interpretable deep semantic segmentation method for earth observation”. In: *2022 IEEE 11th International Conference on Intelligent Systems (IS)*. IEEE, pp. 1–8.

Zhao, Bofei, Haigang Sui, and Junyi Liu (2023). “Siam-DWENet: Flood inundation detection for SAR imagery using a cross-task transfer siamese network”. In: *International Journal of Applied Earth Observation and Geoinformation* 116, p. 103132.

Zheng, Sixiao et al. (2021). “Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6881–6890.

Zhou, Haoyi et al. (2021). “Informer: Beyond efficient transformer for long sequence time-series forecasting”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 12, pp. 11106–11115.

Zhou, Quan et al. (2020). “AGLNet: Towards real-time semantic segmentation of self-driving images via attention-guided lightweight network”. In: *applied soft computing* 96, p. 106682.

Zhou, Yan et al. (2017). “Open surface water mapping algorithms: A comparison of water-related spectral indices and sensors”. In: *Water* 9.4, p. 256.

Zhou, Yongsheng et al. (2022). “Water–Land Segmentation via Structure-Aware CNN–Transformer Network on Large-Scale SAR Data”. In: *IEEE Sensors Journal* 23.2, pp. 1408–1422.

Zhu, Junxian et al. (2020). “A polynomial algorithm for best-subset selection problem”. In: *Proceedings of the National Academy of Sciences* 117.52, pp. 33117–33123.

Zhu, Zhe (2017). “Change detection using landsat time series: A review of frequencies, preprocessing, algorithms, and applications”. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 130, pp. 370–384.