Politecnico di Torino

ScuDo
Scuola di Dottorato ~ Doctoral School
WHAT YOU ARE, TAKES YOU FAR

Doctoral Dissertation
Doctoral Program in Computer Engineering (35^{th}cycle)

Enhancing Robustness and Interpretability in Computer Vision AI

By

Fabio Garcea
******

Supervisor:
Prof. Fabrizio Lamberti, Supervisor

Doctoral Examination Committee:
Prof. Maria Gloria Bueno Garcia, Referee, Universidad de Castilla – La Mancha
Prof. Paolo Garza, Politecnico di Torino
Dott. Claudio Gennaro, Referee, Istituto di Scienza e Tecnologie dell'Informazione
Prof. Marina Paolanti, Università degli Studi di Macerata
Prof. Antonio Santangelo, Università di Torino

Politecnico di Torino
2023

Declaration

I hereby declare that, the contents and organization of this dissertation constitute my own original work and does not compromise in any way the rights of third parties, including those relating to the security of personal data.

Fabio Garcea
2023

* This dissertation is presented in partial fulfillment of the requirements for **Ph.D. degree** in the Graduate School of Politecnico di Torino (ScuDo).

I would like to dedicate this thesis to Elisa, who never lost faith in me even when I doubted myself. Her unwavering support and encouragement kept me going through the most challenging moments of my academic journey.

To my loving parents, who always believed in me and provided me with the resources and opportunities to pursue my dreams. Their unwavering support and sacrifices have made this achievement possible.

To my friends, who were always there to listen, offer words of encouragement, and share their expertise. Their support and camaraderie made the ups and downs of this academic journey much more bearable.

Thank you.

Acknowledgements

I would like to acknowledge my tutor, Prof. Fabrizio Lamberti, for his guidance, feedback, and expertise throughout my research. I would also like to express my gratitude to Prof. Lia Morra for her input and insights. Their dedication to my academic success has been instrumental in shaping this thesis.

I would also like to extend my gratitude to my colleagues and those who I have had the privilege of working with. Their collaboration, support, and encouragement have been an essential source of motivation and inspiration for me.

Finally, I am grateful to the academic institution of Politecnico di Torino, where I have studied for almost 10 years, for supporting my research providing me with the necessary resources and opportunities to pursue my academic goals.

Abstract

Artificial intelligence (AI) and deep learning (DL) have significantly advanced in computer vision (CV), particularly in image classification and object recognition. However, ensuring the trustworthiness of AI systems, especially in safety-critical applications remains a major challenge. Enhancing model robustness and interpretability is crucial for making AI systems reliable and explainable, which in turn improves user trust and facilitates widespread adoption. In this context, the primary goal of the research during the Ph.D. program was to address the challenges of improving the robustness and interpretability of AI systems. Specifically, innovative techniques were leveraged to ensure data-quality and provide new insights into the opaque and complex nature of DL models.

On one hand, the recent discipline of data-centric AI emphasized the importance of high-quality data for building effective AI systems, moving the focus towards improving data treatment steps such as data augmentation and monitoring after deployment to ensure fairness and robustness. On the other hand, continuous advances are being made in the field of Explainable AI (XAI), where techniques such as attention mechanisms and saliency maps were developed to provide insights into the decision-making process of these models. Expanding upon these domains, my research endeavors to address several unresolved challenges pertaining to the areas of robustness and interpretability. These include the exigency of studying novel systematic approaches to data preprocessing in data-centric AI, which arises from the need to structure datasets in a way that avoids or reduces biases in data, leading to more reliable and trustworthy AI systems. The lack of publicly available visual collections of XAI methods is another challenge that hinders a widespread adoption of interpretability in AI. Additionally, the need for more research in efficient unsupervised drift detection and for a systematic approach to create challenging real-world benchmark datasets for OOD detection arises due to the complexity and diversity of real-world data. Addressing these challenges is crucial for the development of more

reliable and robust data-centric AI models that can be trusted to make decisions in real-world scenarios.

The research described in this thesis aims to provide innovative solutions to the challenges mentioned earlier. By delving into each topic, the ultimate goal of this document is to propose solutions to address some of the most pressing issues in trustworthy AI development and provide directions for future research in related domains.

Contents

List of Figures **x**

List of Tables **xiii**

Glossary **xv**

1 Introduction **1**

2 Understanding the Data Domain **6**

2.1 Introduction 6

2.2 Causal Modeling in DL: Understanding Complex Data Relationships 7

2.2.1 Related Work 9

2.2.2 Problem Definition 11

2.2.3 Methodology 12

2.2.4 Case Studies 15

2.3 Road Condition Estimation with CNNs and LSTMs 19

2.3.1 Dataset 22

2.3.2 Methodology 27

2.3.3 Experimental Settings 32

2.3.4 Results 35

2.4 Concluding Remarks 44

3 Interpreting DL Models **48**

3.1 Introduction . . . 48

3.2 iNNvestigate-GUI: a Scalable GUI for XAI Techniques . . . 50

3.2.1 Related Work . . . 50

3.2.2 Design Challenges . . . 53

3.2.3 iNNvestigate-GUI . . . 54

3.2.4 Usability Results . . . 59

3.3 HOLMES: Exaplaining CNNs Through HOLonym-MEronym Relationships . . . 60

3.3.1 Related Work . . . 62

3.3.2 Methodology . . . 64

3.3.3 Experimental Settings . . . 69

3.3.4 Results . . . 74

3.3.5 Discussion . . . 80

3.3.6 Limitations and Future Directions . . . 85

3.4 Concluding Remarks . . . 86

4 Monitoring Data Changes **89**

4.1 Introduction . . . 89

4.2 Drift Detection: Ensuring Model Robustness and Performance . . . 92

4.2.1 Related Work . . . 92

4.2.2 Problem Definition . . . 94

4.2.3 Simulating Data Drift in Scanned Document Segmentation . 96

4.2.4 Methodology . . . 98

4.2.5 Results . . . 100

4.3 OOD Detection in DL . . . 101

4.3.1 Related Work . . . 103

4.3.2 OOD Detection: Problem Definition 107
4.3.3 Proposed Benchmark 112
4.3.4 Preliminary Analysis 117
4.3.5 Experimental Settings 127
4.3.6 Results 131
4.3.7 Discussion 134
4.4 Concluding Remarks 135

5 Conclusions **138**

References **142**

List of Figures

2.1 Illustrations of causal models for a causal and anticausal tasks. . . . 12

2.2 Causal model template for medical imaging dataset analysis. 15

2.3 Causal model representation for classifying prostate cancer. 17

2.4 Causal model representation for predicting wet road conditions. . . 18

2.5 Geographical positions of the road-cameras involved in the analysis. 23

2.6 Sequence generation process. 28

2.7 Sequences generated by accounting for temporally coherent spatial augmentation and for a realistic simulation of outdoor brightness conditions. 29

2.8 Self-supervised pre-training and the Semi-supervised fine-tuning methodologies. 30

2.9 Unrolles representation of the Temporal (ConvLSTM) model. 33

2.10 ROC curves comparing the performance of the baseline model with self-supervised initialization (*Self-sup*) to models fine-tuned through SSL on the large Training-300K dataset. 37

2.11 ROC curves of the ConvLSTM model (a) and 2D CNN (b) trained on the complete Training-300K dataset. 38

2.12 Per-event recall vs. average number of FP events. 40

2.13 An explanation generated by LIME for a transition frame, as predicted by both models. 41

2.14 Comparison of the transition profile for a severe event, where the ConvLSTM model shows better detection of the two events during morning hours. 42

2.15 Explanation generated by LIME is shown for a transition frame, as predicted by both models. 42

3.1 Comparison of different model predictions for one of the images included in T1. 57

3.2 *Suggestion* panel of iNNvestigate-GUI. 58

3.3 Visual comparison of the predictions made by the three models in T1. 61

3.4 HOLMES pipeline. 65

3.5 HOLMES Explanation example for the *horse* class – PASCAL-Part and *sorrel* class – ImageNet. 70

3.6 Distribution (violin plot) of the average per-part calibrated F1-score. 75

3.7 Average per-part calibrated score as a function of the number of parts per holonym class. 76

3.8 Violin plots of the average score drop and maximum score drop per image on the PASCAL-Part and ImageNet validation sets. 77

3.9 Top-5 meronyms distribution. 77

3.10 HOLMES Global Explanation. 79

3.11 Insertion/Deletion Ratio distribution for the PASCAL-Part and ImageNet datasets. 80

3.12 HOLMES explanations example. 81

3.13 HOLMES explanations example. 82

3.14 Examples of end-user explanations comprising 1 meronym. 82

3.15 Examples of end-user explanations comprising 2 meronyms. 82

3.16 Examples of end-user explanations comprising 3 or more meronyms. 83

4.1 Structure of the DL model utilized for document segmentation. . . . 95

4.2 The structure of the artificial document generator. 97

4.3 Correlation plots for IoU and Hellinger distance. 99
4.4 ROC curves for drift types. 101
4.5 Average semantic similarity between the ground truth category and the predicted ID labels. 120
4.6 Average semantic similarity between the and the predicted ID labels, and the OOD ground truth classes. 121
4.7 Pruned version of the original graph. 123
4.8 Examples of isolated clusters consisting of few nodes. 124
4.9 Examples of strong edges between classes representing the same concepts with slightly different names. 124
4.10 Examples of strong edges between different concepts, linked by meronym-holonym and similar relationships. 125
4.11 Examples of network biases that affect classification results. 125
4.12 T-SNE visualization of the features extracted from the Baseline dataset at different network layers. 126
4.13 T-SNE visualization of the features extracted from the Inter-Dataset OOD Detection dataset at different network layers. 127
4.14 T-SNE visualization of the features extracted from the FACETS OOD Detection T1 dataset at different network layers. 128
4.15 ODIN validation procedure. 129
4.16 OODL validation AUROC for each candidate layer. 131

List of Tables

2.1 Period of acquisition for each road-camera involved in the analysis. 22

2.2 Distribution of the training, validation and testing sets. 27

2.3 Definition of similarity between samples. 31

2.4 AUCs on validation and test sets. 36

2.5 AUC values and their 95% CI for the validation and test sets. 39

2.6 Comparison between best performing CNN and ConvLSTM models. 40

2.7 Performance (AUC) for the ConvLSTM model with respect to the time of the day. 41

3.1 Summary of the main DL visualization tools and comparison with the proposed tool. 52

3.2 Summary of the available libraries of visualization algorithms. . . . 53

3.3 Per-pixel AUC results. 75

3.4 Deletion/Insertion/Preservation AUCs for HOLMES and Grad-CAM. 79

4.1 Usage statistics for the 10 most popular data sources used as both ID and OOD. 104

4.2 Composition of the Baseline dataset. 114

4.3 Composition of the splits used for the Inter-Dataset OOD Detection. 114

4.4 Composition of the WordNet ImageNet datasets 116

4.5 Composition of the FACETS OOD detection dataset 118

4.6 OODL Validation results 131
4.7 OOD detection results for each dataset. 132
4.8 Misclassification detection results on Places365-Standard (val) . . . 133
4.9 Comparison of AUROC scores for ODIN and OCSVM applied to the Places vs ImageNet task. 134

Glossary

Acronyms / Abbreviations

AI Artificial Intelligence

AUC Area Under the Curve

AV Activation Vector

CDT Change-Detector Test

CI Confidence Intervals

CNN Convolutional Neural Network

ConvLSTM Convolutional Long Short-Term Memory

CV Computer Vision

DAG Directed Acyclic Graph

DL Deep Learning

DNN Deep Neural Network

FACETS Face Aesthetics in Contemporary E-Technological Societies

FC Fully Connected

FRESCO Face Representations in E-Societies through Computational Observation

FROC Free-Response Operating Curve

GAN Generative Adversarial Network

GUI Graphical User Interface

HOLMES Holonym-Meronym Based Semantic Inspection

IoU Intersection over Union

KB Knowledge Base

LSTM Long Short-Term Memory

MAV MEAN Activation Vector

ML Machine Learning

MLOps Machine Learning Operations

MLV Maximum Logit Value

MMD Maximum Mean Discrepancy

MOS Minimum Others Score

MSP Max Softmax Probability

OCR Optical Character Recognition

OOD Out-Of-Distribution

PCA Principal Component Analysis

ROC Receiver Operating Curve

SSL Self-Supervised Learning

SUS System Usability Scale

SVM Support Vector Machine

TS Temperature Scaling

XAI Explainable Artificial Intelligence

Chapter 1

Introduction

Deep Learning (DL) is a subfield of Machine Learning (ML) that involves training neural networks to learn increasingly complex representations of data [1]. In recent years, DL has emerged as a powerful tool in computer vision (CV), which is the field concerned with enabling computers to interpret and understand visual information from the world around us [2]. This approach has led to breakthroughs in a range of applications, such as image and video recognition, object detection, face verification, etc. [3]. For instance, DL algorithms have been used to improve doctors' diagnoses by analyzing medical images [4], to let autonomous vehicles navigate complex environments by recognizing objects and obstacles [5], and to enhance security by identifying individuals from facial images [6].

DL and CV have a rich history. The origins of CV can be traced back to the 1960s, when researchers first began developing algorithms to analyze and interpret images [7]. However, it wasn't until the 1980s and 1990s that neural networks began to emerge as a powerful tool for image recognition and CV [8]. These early neural networks were relatively shallow and could only recognize simple patterns, but they paved the way for the development of more sophisticated DL models in the 2000s and 2010s [9].

The applications of DL and CV extend far beyond those already mentioned. One area where DL has made significant contributions is that of gaming, where it has been used to create sophisticated agents powered by artificial intelligence (AI) capable of beating human champions in games such as Go and Chess [10]). Another area of application that of cybersecurity, where DL algorithms can be used to detect

and prevent cyberattacks by analyzing network traffic [11]. Additionally, DL has been used in the entertainment industry to create realistic visual effects, such as computer-generated characters and environments in movies and video games [12]. The potential applications of DL and CV are vast, with potential to transform fields such as transportation, finance, and even agriculture. In addition to these applications, DL could have a significant impact in environmental monitoring and conservation. For example, DL algorithms could be used to analyze satellite imagery to detect deforestation, monitor wildlife populations, and track changes in the Earth's climate [13]. Other potential applications are in the field of robotics, where DL could be used to improve the perception and decision-making capabilities of autonomous systems [14]. For instance, DL algorithms could enable robots to recognize and respond to human gestures, navigate complex environments, and perform tasks such as object manipulation and assembly [15]. Additionally, DL has shown promising results in fields such as finance, where it can be used to analyze large datasets and detect fraud or predict market trends [16].

However, despite its remarkable achievements, the growing usage of DL algorithms in critical domains like healthcare and finance highlights the pressing need for ensuring *trustworthy* AI systems. Trustworthiness refers to the ability of AI to perform its intended function reliably, safely, and ethically. Trustworthiness can be achieved by ensuring the transparency and accountability of AI systems, guaranteeing their fairness and inclusivity, as well as their safety and security. By ensuring that AI systems are developed in a way that is transparent and accountable, stakeholders can have greater confidence in the fairness and reliability of these systems [17]. The guidelines for building trustworthy AI provided by bodies like, e.g., the European Commission, are intended to guide the development and deployment of AI systems. Guidelines stress the importance of a human-centric approach, with AI systems that need to be designed to respect human rights, cultural diversity, and the principles of the rule of law. Technical robustness and safety are critical considerations, thus AI systems must be auditable, interpretable, and reliable. Privacy and data governance are also important, with AI systems that shall be designed to ensure data protection and individual control over their data. Interpretability is emphasized, with AI systems that are requested to be open about their capabilities and decisions. Diversity, non-discrimination, and fairness are critical considerations, with AI systems that need to avoid bias and discrimination, and ensure that their use is fair and non-discriminatory. Societal and environmental well-being are also important,

with AI systems that shall be designed to enhance human well-being and sustainable development. Accountability is a fundamental principle, with AI systems that must be subject to human oversight, and their developers and operators be responsible for their actions. By following these guidelines, AI systems can be developed and deployed in a responsible and ethical manner. The present thesis will delve into the issues of robustness and interpretability in DL, to provide specific solutions that can help establish trust in AI systems.

Data is a significant factor in determining model robustness, and problems related to data quality can have a significant impact on the overall performance of the model. The lack of diversity in training data for DL models has been widely acknowledged as a challenge for the development of robust and fair models. Research has shown that biases in the training data can lead to biased models that perform poorly on certain groups of people. For instance, facial recognition systems have been found to be less accurate in identifying people with darker skin tones, which can result in discriminatory outcomes [18]. To mitigate this issue, researchers have proposed various approaches, including data augmentation and bias correction techniques. Data augmentation involves generating new data samples from existing data, which can help increase diversity in the training data [19]. Bias correction techniques aim to reduce bias in the training data by modifying the data or the learning algorithm [20]. Moreover, topics such as concept drift and anomaly detection become crucial in monitoring the performance of the model in real-world scenarios. Concept drift refers to the phenomenon where the statistical properties of the data distribution change over time, causing the model performance to degrade [21]. Anomaly detection, on the other hand, helps to identify instances where the model is making unexpected or erroneous predictions [22]. To ensure model robustness and reliability, it is important to continuously monitor the data being fed into the model and adapt the model accordingly [23]. This requires ongoing data collection, analysis, and maintenance, as well as the use of techniques such as transfer learning and active learning to adapt to changing data distributions [24]. Effective data preprocessing, maintenance and monitoring are also key aspects in the field of Machine Learning Operations (MLOps). MLOps is a promising field of ML engineering that offers many benefits for the efficient deployment and management of ML models in production environments. One of the major challenges related to the MLOps paradigm is the need for effective data management, including data storage and data labeling. Another challenge is the need for continuous model training and updating to keep up with changing data

and user needs, which requires effective model versioning and management [25, 26]. Additionally, ensuring the security and privacy of sensitive data used in ML models is another critical challenge that requires robust approaches. Solving these challenges requires approaches that can help automate and optimize the MLOps workflow by improving data processing and analysis.

Recent advances towards the resolution of these challenges eventually led to the birth of a new *discipline* named data-centric AI. According to Andrew Ng [27], *Data-centric AI is the discipline of systematically engineering the data used to build an AI system.* In this regard, Ng suggests that focusing on high-quality, consistently labeled data will unlock the full potential of AI in sectors such as health care, government technology, and manufacturing. However, to achieve this, organizations will need to treat data more systematically and frequently include the data preparation process more domain experts, who can provide input on how data should be labeled and what features are most relevant for AI systems in specific fields. By following these principles, data-centric AI can enable organizations to build custom AI systems that are trained on their data, rather than relying on multipurpose AI systems that are not tailored to specific industries. More in specific, the pioneers of this discipline focus their attention on improving data-treatment steps such as data-augmentation and data-monitoring after deployment. They highlight how crucial these steps are in guaranteeing both fairness in real-world scenarios characterized by imperfect datasets as well as on robustness in deployment scenarios where the quality of the data plays a fundamental role both in terms of performance and overall cost. In fact, according to the authors, 70% of the complexity of an AI based system is tied to data processing, handling, and monitoring. In this context, factors such as data preprocessing, dataset drift, and anomaly detection must be tackled to advance towards data-centric and trustworthy AI.

Building on the aforementioned introduction to open challenges, the aim of this thesis is to disseminate the key studies conducted during the three-year Ph.D. path that focus on understanding domains, monitoring data, and interpreting models within the field of CV research. Specifically, the thesis will address the issues of model robustness, reliability, domain understanding, and model interpretability in separate chapters. Various data-centric approaches will be presented as potential methods for addressing model robustness issues. One study, for instance, will explore the use of causal models in data preprocessing and management, demonstrating their effectiveness in avoiding or reducing biases in data and in the selection of an optimal

learning approach for real-world datasets. The thesis will also discuss multiple studies on data treatment and maintenance, including a novel and efficient technique for detecting and monitoring concept drift in CV datasets, as well as an analysis of out-of-distribution (OOD) detection techniques for large-scale and challenging real-world datasets. These studies will introduce innovative methodologies and avenues towards developing robust data-centric AI models, which may prove to be useful tools in addressing MLOps challenges. Finally, the thesis will report on various studies related to interpretability in AI, including a novel Graphical User Interface (GUI) enabling an easier usage of XAI techniques and an innovative methodology capable of providing part-based explanations for Convolutional Neural Networks (CNNs). Research in this area will undoubtedly contribute to the demand for interpretability in neural networks and overall AI trustworthiness.

The thesis has been organized according to the following structure. Chapter 2 explores the use of causal models for modelling the data domains in DL and ML. The potential benefits of causal analysis are demonstrated through a dedicated study. In Chapter 3, a novel XAI methodology for CNNs interpretation is presented, accompanied by an easy-to-use GUI that provides a comprehensive range of XAI techniques. Lastly, Chapter 4 delves into the complexities of data understanding, by illustrating a cutting-edge unsupervised methodology for detecting concept drift that was extensively investigated and successfully tested in a real-world context; additionally, a thorough analysis that was conducted on OOD detection and resulted in the creation of a challenging benchmark is reported.

Chapter 2

Understanding the Data Domain

Work described in this chapter was originally presented in [28, 29].

2.1 Introduction

DL has become an increasingly popular field of study in recent years, particularly in the area of CV [1]. The vast amounts of data that are now available have facilitated the development of powerful algorithms that are capable of recognizing and interpreting images, videos, and other types of visual data [2]. However, the effective use of DL techniques in CV tasks requires more than just access to large datasets; it also requires a deep understanding of the data domain.

The first section of this chapter will provide a brief introduction to causal modeling and its relevance to DL. As, previously highlighted, the diverse and imperfect nature of data in real-world scenarios have led to the urge for systematic approaches to data-treatments and data-maintenance. Causal modeling is a powerful tool for structuring complex datasets, and it has been used extensively in fields such as economics, epidemiology, and social science [30]. By explicitly modeling the causal relationships between variables, researchers can gain a deeper understanding of the underlying processes that generate the data. In the context of DL, causal models represent an effective tool to move towards more data-centric AI. They can help to identify which features are relevant for a given task and how they should be represented. To illustrate the use of causal models in DL, it will also be presented a real-world use case from industry. Specifically, a study will be discussed where

causal models were used to structure a dataset of camera surveillance videos. The goal of this study was to detect wet/dry roads from the videos, a task that is essential for ensuring road safety in adverse weather conditions [31]. It will be described how causal models were used to identify the relevant features in the videos and how the dataset was structured to facilitate the training of a DL model. The second section of this chapter will focus on the application of DL techniques in the wet/dry road detection task. The DL architecture that was used will be described, together with the preprocessing steps that were necessary to prepare the data for training. The challenges that were faced in this task will be also illustrated, including issues related to data quality and bias. Finally, the results of the experiments will be described, and the implications of the findings discussed.

Throughout this chapter, it will be emphasized the importance of taking a data-centric approach to DL. Rather than relying solely on the power of DL algorithms, it is argued that it is essential to have a deep understanding of the data domain, possibly gathering information directly from domain experts. By leveraging the insights gained from causal modeling and other techniques, researchers can develop more effective DL models and contribute to the continued advancement of the field.

2.2 Causal Modeling in DL: Understanding Complex Data Relationships

ML and DL have become the high performance solutions for a large number of classification tasks, from autonomous driving [32] to natural sciences [33] to disease diagnosis in the medical domain [34]. It is well-established that the quality of a trained ML model is closely tied to the underlying dataset, as famously stated by the adage "Garbage In, Garbage Out". While much research in the field of ML and DL focuses on improving models through novel training strategies or diagnosing models post-training, only recently has the research community begun to take a more data-centric approach [35].

Dataset bias can greatly impact the performance and the robustness of a ML model during its operational life, causing the model to exploit spurious correlations and preventing generalization to unseen data [36–40]. Furthermore, there is mounting evidence that biased ML models may perpetuate social and racial biases, exacerbating

discrimination and resulting in unfair outcomes [40, 41], or even have serious implications in domains where robustness and safety are critical [42, 43].

Deep Neural Networks (DNNs), due to their high nonlinearity and black-box nature, are particularly prone to dataset biases. DNNs tend to learn and rely on shortcuts to solve a specific task, a behavior which has been traced to several properties of DNNs [44]. A simple example of this is when a model that recognizes cats and dogs is exclusively trained on images of cats with an overlayed text, it may learn an association between the cat class and the presence of text, and even rely solely on the latter. This is not a far-fetched example, but it has occurred in real-world medical problems where hospital archival systems often superimpose textual information on X-ray scans [44].

Another important phenomenon to consider is concept drift, as it will be discussed in details in Chapter 4. It occurs when the statistical properties of the training data and deployment data diverge over time. For example, if new species of cats and dogs that were not originally included in the training data appear over time, would the cats vs dogs classifier still be able to generalize?

Given these considerations, it is clear that a more systematic and robust approach is needed when collecting, structuring, and characterizing the development datasets used to train a DL model. One of the most promising approaches is the use of causal models. Causal models provide a way to understand the underlying mechanisms that generate the data, by explicitly modeling the causal relationships between variables. This can aid in identifying and mitigating sources of bias, as well as better understanding the robustness and generalizability of the models trained on the data.

Causal inference methods, such as do-calculus and instrumental variables, allow to infer causality from observational data, which is often the only feasible option in practice. Additionally, recent developments in counterfactual thinking allow to infer the effect of interventions on a system, which is especially useful for decision making and understanding how ML models work.

With a data-centric approach that utilizes causal models, practitioners can select more effective training, collection, annotation, and data augmentation strategies, detect biases and other limitations in the current dataset, and possibly resolve or at least anticipate any issues in the resulting ML models. This approach would ultimately lead to models that are less prone to biases, have improved robustness,

and enhance their overall performance, especially when dealing with the challenge of concept drift.

Moreover, adopting a causal modeling approach would allow all stakeholders to be aware of the characteristics and potential pitfalls of a dataset, as well as document the underlying assumptions in the data collection and generation process in a clear and transparent fashion that can be easily validated or integrated by domain experts. It would also allow to better understand the nature of the problem, identify important variables and interactions among them, and estimate the effect of interventions, even in the presence of measurement errors and unobserved confounders.

This analysis aims to showcase how causal models provide a way to better understand the underlying mechanisms that generate the data and make informed decisions on how to collect, structure, and analyze the data to mitigate bias, improve robustness, and enhance performance. The remaining of the section will be organized according to the following structure. In Section 2.2.1 the related work on causal modeling in ML and DL is presented. Sections 2.2.2 and 2.2.3 respectively define the main principles of causal modeling and the methodology that can be followed to build a causal model for a specific dataset. In Section 2.2.4 two case studies involving the usage of causal modeling are presented, one from the medical literature domain and the other from an industrial real-world scenario tackled during the Ph.D. path.

2.2.1 Related Work

Several recent works have aimed to increase the reliability and reproducibility of the data collection and annotation phases in ML, as well as to raise awareness of the risks and perils of dataset biases. Initiatives such as Dataset Datasheets [38] and Data Nutrition Labels [45] have been developed to standardize the way datasets are collected and reported. A Dataset Datasheet includes discursive descriptions such as the motivation behind the creation of the dataset, its composition, and other methodological information such as any applied preprocessing and the recommended usage. The Data Nutrition Labels provide similar information but in a more concise format inspired by food nutritional labelling. Both initiatives enhance the transparency of the data collection process, and have had a mitigating effect on undesired or undetected dataset biases.

On the other hand, causal diagrams represent a formal tool that can be used during and after the creation of a dataset to reason about the relationships between different variables. A few authors have worked on quantifying biases in ML datasets by employing statistical techniques [37, 46, 47]. For example, Beretta et al. studied the intrinsic discriminatory risk by assessing the degree of dependency between a protected attribute (e.g., race or gender) and the target variable [46]. However, the majority of these techniques work on structured datasets and are not directly applicable to typical DL models for unstructured data, such as images or text.

Causal and counterfactual reasoning have also been increasingly used to quantify and/or mitigate biases in trained ML models [48, 49, 40]. Nevertheless, most existing causality-based algorithms require knowledge of the underlying causal graph. The examples presented in this paper show how causal diagrams can connect these two lines of research by linking unstructured data (e.g., images) to latent factors, whose distribution can be modelled and analyzed by either statistical methods or counterfactual reasoning.

An open question is how to evaluate the validity of the causal model when used in this fashion. First, the correctness and completeness of the causal model cannot be checked against the ground truth, as the latter is unknown. Recent work has tackled the problem of verifying causal models learned from observational data, but the method is not readily applicable to unstructured data [50]. Second, the benefit of integrating causal models into dataset construction, while strongly supported by intuition, has not been experimentally linked to an increase in performance and generalization of the trained ML models.

The nature, type, and presence of biases in CV datasets have also been extensively studied, starting from the seminal work by Torralba and colleagues. This is most commonly done by studying the cross-dataset generalization performance of trained DNNs [44, 36, 51, 39]. Torralba et al. suggested general strategies to avoid common biases such as selection bias, capture bias, and negative set bias [36]. Biases due to gender and ethnicity have been addressed by collecting larger, more diverse datasets [49]. These strategies are complementary to causal diagrams, which however are more easily applied to specialized datasets in which the domain is well defined (e.g., industry) and the data generation process can be precisely and accurately modelled.

In summary, recent works have focused on increasing the reliability and reproducibility of data collection and annotation in ML by standardizing the way datasets are collected and reported through initiatives such as Dataset Datasheets and Data Nutrition Labels. Statistical techniques and causal and counterfactual reasoning have also been used to quantify and mitigate biases in ML datasets and models. However, there are still open questions and challenges in evaluating the validity of causal models, and linking the integration of causal models into dataset construction to an increase in performance and generalization of trained ML models. Furthermore, the nature, type, and presence of biases in CV datasets have been extensively studied, and strategies such as collecting larger and more diverse datasets have been proposed to address them.

2.2.2 Problem Definition

In this section, a brief overview of the key concepts of causality theory is presented, adhering to the terminology of previous seminal works (e.g., [52–55]).

A causal model can be represented by a Directed Acyclic Graph (DAG) consisting of nodes (representing variables or factors) and arrows, also known as edges (representing causal relationships between variables). The notation $A \rightarrow B$ represents a direct causal relationship between A and B, meaning that an experimental manipulation of A would change the probability of B, assuming all other factors remain constant. One important principle in causal inference is that the probability distribution of a cause, $P(A)$, should not affect the conditional probability distribution $P(B|A)$, which is known as the independence of cause and mechanism [55]. An intervention, defined as any forced change to the value or probability distribution of a node, regardless of its direct cause, results in a modified DAG in which that node is disconnected from its parents. This can be achieved through techniques such as randomization, stratification and controlling for a variable when building a statistical model.

Causality theory also introduces three main canonical relationships between three (or more) variables: confounders, mediators, and colliders. In large graphs, it is necessary to reason in terms of paths [53]. A mediator B is a variable that connects an indirect cause A to the final effect C ($A \rightarrow B \rightarrow C$). Controlling for B, such as by conditioning a statistical model on the value of B, eliminates the link between A

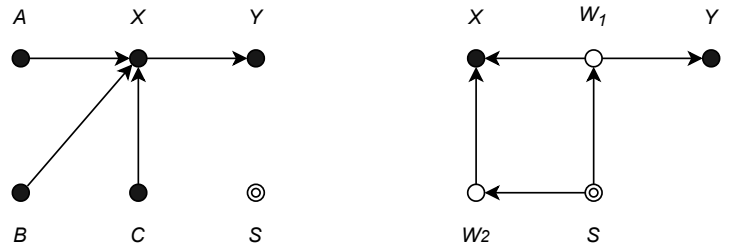


Fig. 2.1 Illustrations of causal models for a causal (left) and anticausal (right) task are shown. X and Y denote the input and output variables, respectively, while the selection variable is denoted by a double line circle. Empty circles indicate hidden (latent) variables.

and B, thereby completely blocking the effect of A. A confounder C is a common cause of two variables A and B ($A \leftarrow C \rightarrow B$). A variable A that is simultaneously an effect of multiple independent causes B and C is referred to as a collider of B and C ($B \rightarrow A \leftarrow C$). Another concept in causality is the idea of "adjustment" for confounding in observational study, which involves controlling for the effect of a confounder by including it as a covariate in the statistical model or by matching on it.

2.2.3 Methodology

When modeling a ML task, which is defined as a mapping function $f : X \mapsto Y$ between an input space X and an output space Y, a key question that must be answered is whether the task is causal or anticausal. A task is considered causal when the goal is to estimate $P(Y \mid X)$ when $X \rightarrow Y$, meaning that Table are trying to estimate the conditional distribution of an output Y which is an effect of the input X. Conversely, a task is defined as anticausal if the goal is to estimate $P(Y \mid X)$ when $Y \rightarrow X$. Anticausal problems are common in ML/DL. A specific case of anticausal task occurs when there is no direct relationship between X and Y, but both have a common unobserved common cause (confounded). The distinction between causal and anticausal tasks may not be straightforward, depending also on the level of information available about the data collection process. Examples of causal and anticausal tasks are shown in Fig. 2.1. Some practical examples will be provided in Section 2.2.4.

Causal diagrams allow practitioners to link the characteristics of their problem to properties established by ML and statistical theory, and to select the most appropriate training and statistical methods accordingly. For instance, Self-Supervised Learning (SSL) techniques should give little benefits for causal tasks, meaning that other training strategies should be prioritized. This is because, in SSL, it is only possible to access unlabelled data, hence to the distribution $P(X)$, which for the principle of independence of cause and mechanism should be uninformative with respect to $P(Y|X)$ if $X \rightarrow Y$. On the other hand, SSL could work for anticausal tasks.

A causal model should not be limited to the input and target variables, but should also include all factors (either observed or hidden) that may influence their distribution. With causal diagrams it is possible to define and assess the role of confounders and colliders in a robust way. Confounding influence can introduce what is often called a "spurious" correlation, i.e., when the considered variables are statistically correlated through another variable but have no causal relationship between each other. Controlling for a confounder blocks the corresponding path, effectively removing the spurious correlation. This may be achieved in several ways, e.g., by introducing the confounder as an additional variable to the ML model, or by stratified sampling. On the other hand, controlling for a collider (or its descendants) introduces an association between the two otherwise independent causes, and hence should be avoided. A vast body of literature in the epidemiological sciences exploits causal models for determining which variables should be controlled for.

Colliders play a fundamental role in the appearance of dataset bias. for instance, it is possible to consider the case of a dataset of patients and their medical records. The medical records include information about whether a patient has a disease, whether they have taken a certain medication, and whether they have a certain genetic variant. In this case, the genetic variant and the medication are colliders because their presence depends on the presence of the disease. Training a model on this dataset, the model may develop a bias towards the medication as a cause of the disease, even though it is actually a result of the disease. This is because the collider bias causes the model to overestimate the association between the medication and the disease, leading to inaccurate conclusions about causality. Therefore, it is important to be aware of colliders and take steps to control for them when modeling the data.

Another important aspect of causal modeling is the consideration of the data generation and selection processes. This includes understanding how the data was

collected and what factors may have influenced the selection of the data points. For example, if a dataset is collected through a self-selection process, such as a survey, the data may be biased towards certain groups of individuals who are more likely to respond to the survey. Similarly, if a dataset is collected through a convenience sampling process, the data may be biased towards certain groups of individuals who are more easily accessible. These biases can lead to inaccurate conclusions about the underlying population and must be taken into account when analyzing the data.

In conclusion, understanding the causal structure of a task and the data generation and selection processes is crucial for accurate modeling and interpreting the results of ML tasks. The use of causal diagrams and the consideration of confounders and colliders can help practitioners to link the characteristics of their problem to properties established by ML and statistical theory, and to select the most appropriate training and statistical methods accordingly. Additionally, understanding the data generation and selection processes can help to identify and control for sources of bias in the data.

In order to thoroughly represent the data generation, collection, and annotation process, a comprehensive causal model can be created using the methodology reported in the following (adapted from [53]).

- Collect information about the data collection, annotation, and selection processes to build a complete model, including relevant mediators, confounders, and colliders.
- Determine if the task is causal or anticausal.
- Identify potential discrepancies between the training and testing sets and implement strategies such as data augmentation, domain adaptation, or resampling to address them, depending on the nature of the domain shift.
- Evaluate if the data collection was biased with respect to the input X, the target Y or any other variable.
- Draw the causal model and include all factors, paying attention to the emergence of collider biases.
- Decide if and how further selection (randomization or stratification) should be conducted to control for confounders.

2.2.4 Case Studies

In this Section, two case studies are presented, one from the medical domain and one from an industrial CV application, to demonstrate the practical application of causal diagrams in characterizing the data collection, annotation, and selection processes for ML tasks.

During the Ph.D. path, the use of causal models in the medical imaging field has been explored, as proposed by Castro et al. in their seminal work [53]. Medical imaging analysis tasks often share a common workflow and involve a limited number of factors. The role of these factors in the construction, sampling, and annotation of datasets has been extensively studied in literature [42, 56].

Building on this research, Castro and colleagues propose a general causal model 2.2 structure that can be applied to the majority of medical imaging ML tasks. The model includes selection and domain variables, which allow for the modeling of different types of dataset biases and domain shifts. The introduction of a selection variable D is particularly noteworthy, as it models factors that may differ between the training and testing population. The resulting causal model and the type of task (anticausal vs. causal) determine the appropriate countermeasures to address domain shifts [53].

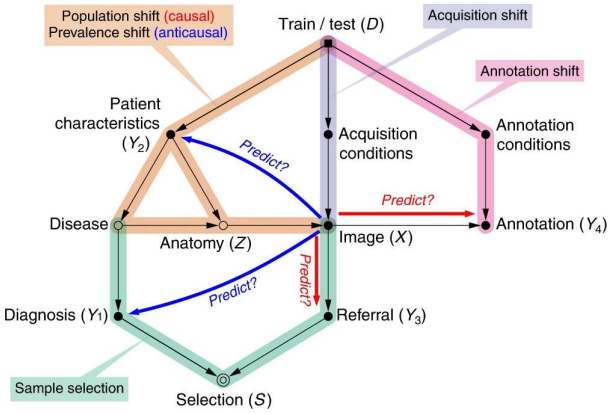


Fig. 2.2 A causal model template for medical imaging dataset analysis is presented. Measured values are denoted by filled circles, hidden variables by empty circles, and selection variables by double-line empty circles. Adapted from [53].

To illustrate the application of this model, a sample medical task is considered herewith: the diagnosis of cancer (e.g., prostate cancer) from an image (e.g., a magnetic resonance image). The goal is to predict the probability of the presence or absence of cancer (Y) from the input image X. Training sets for this model are typically based on datasets collected from one or multiple institutions.

The image X is the result of various causes, including the presence of the disease, patient anatomy, and acquisition conditions. Patient anatomy is an internal hidden variable that accounts for inter-patient variability, whereas acquisition conditions represent all factors (type of scanner, acquisition protocol, etc.) that may affect image appearance.

To determine whether the task is causal or anticausal, it is essential to understand how the reference standard is established. Whenever possible, the presence or absence of the disease should be defined by means of biopsy and/or follow-up for a suitable period of time [42]. In this case, the presence or absence of the disease is known, and manipulating or changing the image would not alter the label. This task would be anticausal, as is the case with many computer-aided diagnosis tasks.

In contrast, if labels are established solely on the radiologist report, then the task is considered causal (Fig. 2.3). The true disease status W_2 is considered a hidden variable, and the image X is a mediator between the disease W_2 and the output label Y. In this case, manipulating the image could alter the radiologist perception, and hence the resulting labels. However, practical situations can be more nuanced. For example, biopsy is typically performed only for cases that the radiologist deems suspicious. This factor can be represented by adding a selection variable *biopsy* which is an effect of the label Y.

Finally, the input image depends on the acquisition conditions, which may produce diverse sets of images, and the model should be robust to these variations. The general causal model structure proposed by Castro et al. is a useful tool for understanding and addressing these factors, ensuring robust and accurate performance in medical imaging tasks.

For the use case in the industrial field, it will be discussed the causal diagram for the task of estimating road conditions using surveillance Full-HD cameras. The goal is to identify the presence of wet road conditions from video frames captured by the cameras. The frames have been collected from areas with varying characteristics such as illumination, road morphology, and point of view. The causal diagram,

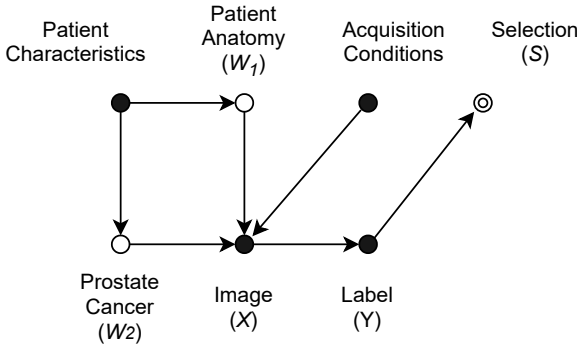


Fig. 2.3 A representation of a causal model is given for classifying prostate cancer. Hidden factors of interest are illustrated using empty circles.

as shown in Fig. 2.4, helps to understand the role of different factors in the data collection, sampling, and annotation process.

The main variables of interest are the image (X) and the manually determined label (Y), which is a binary value indicating whether the road appears to be dry or wet. The phenomenon of interest is the presence of water W, which is caused by weather conditions such as rain or snow, or other factors like floods. These variables are considered hidden because they are not directly measured in the adopted experimental setup. In causality terminology, the input image X acts as a mediator between the phenomenon of interest W and the label Y.

The camera site, time of day and month (or season) are also important factors to consider. They are indirect causes of both W and X (highlighted in light blue in Fig. 2.4). For example, the site variable embeds information such as geographical location, road morphology, the type of asphalt, and the frequency of car passing. Similar factors are crucial in determining the presence of water on the road [53]. Additionally, images taken at different sites have distinctive visual features due to the different road morphology, illumination, etc. This results in a domain or acquisition shift.

To address these issues, data resampling is a possible strategy to mitigate prevalence and population biases [53]. By considering the causal diagram and the factors that influence the data, it is possible to improve generalization during training and testing.

The task of classifying road conditions using surveillance Full-HD cameras can be modeled as a causal task. The images are manually labeled, and the assumption

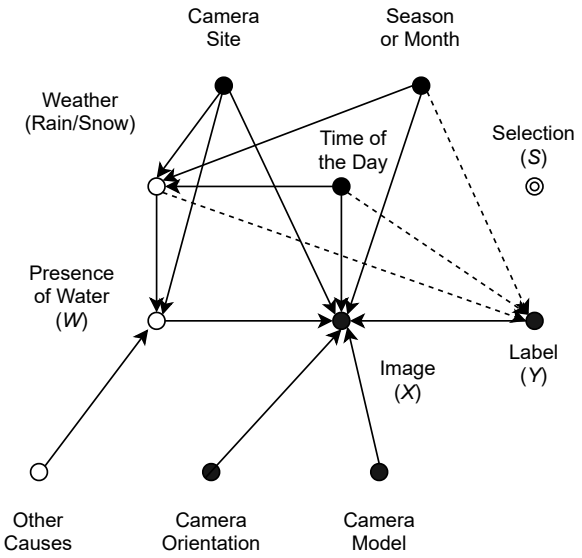


Fig. 2.4 A causal model is presented for predicting wet road conditions, where hidden variables are shown using empty circles. The selection method involves random sampling.

is that the label Y is caused by the image X, and not vice versa. This is because the labels are generated through manual annotation without explicit knowledge of the hidden variable W, and any substantial modification of the image may change the value of the assigned label. For example, if the presence of water is not visible due to occlusions or poor visibility, any rater would give a negative label, regardless of the actual road condition.

However, the distinction between causal or anticausal tasks is not entirely clear-cut. If the image-derived labels determined by an expert are nearly identical to the hidden variable, then the labels could serve as proxies for the ground truth, possibly configuring an anticausal relationship. In this case, the annotators were aware of the site, date and time of acquisition, and the weather could be inferred from the input image. From this viewpoint, the task can be seen as confounded, and hence anticausal. This perspective is supported by the observation that SSL was effective to improve performance. Further investigations are needed to clarify the role of SSL in ML tasks for which the causal or anticausal nature is not easily determined.

The domain $\mathscr{D}$ is determined by several variables such as the site, season, camera orientation, and camera model. Camera orientation and camera model have a direct influence on the image and are contributors to the domain or acquisition shift. From

a causal perspective, the image X is a collider of many variables, including W and camera parameters, thus care must be taken to prevent collider biases affecting the training process. The distribution of all known factors was studied and corrected, if needed, by sampling and data augmentation. For example, when data is acquired continuously during the day and for the whole year, the distribution may be balanced with respect to time of day and season, but it is unlikely to account for all possible sites, which is the most critical factor for generalization. Camera orientation has a profound impact on the presence of reflections, myrages, as well as under- and over-exposure. Data augmentation strategies that simulate different camera orientations and illumination conditions are useful in this case to prevent the appearance of spurious correlations as a result of imperfect data selection or small dataset sizes.

In conclusion, a predictive model can be influenced by confounders (such as the season) by incorporating them as additional inputs. This allows the model to condition its prediction on the value of the confounder. However, this approach is only feasible if the confounder is observed and measured during both training and inference, and if the training set includes examples from all possible values of the confounder distribution.

2.3 Road Condition Estimation with CNNs and LSTMs

In this Section, a real-world case study tackled with the usage of causal modeling is presented in detail. The study highlights how the configuration of a data domain through a data-centric approach eventually led to an unbiased and robust model capable of generalizing to unseen data. In particular, as previously mentioned in Section 2.2.4, the study was carried out in the context of road condition estimation using DL models trained on surveillance camera footage captured from cameras placed over the road.

The different methodologies in literature can be divided into categories based on factors such as the source of data acquisition (camera mounted on the vehicle or near the road), type of sensors, and methodology (analysis based on data or based on physics measurements). In this section, Table focus on the latter category, which relies on distributed fixed sensors for road infrastructure management [57–59]. Historically, RWIS stations have been largely adopted to gather utilized to collect measurements on atmospheric conditions, road status, and visual percep-

tibility. RWIS can have various sensors such as cameras and thermometers, but their installation and operational costs limit their availability and cause low spatial resolution. Outdoor cameras provide an economical alternative with low installation costs [57, 60, 61], but factors such as their distance from the road and varying illumination conditions can make the detection task challenging. Hence, semi-supervised techniques have been proposed to reduce manual labeling costs [57].

Different methods have been used to detect road conditions based on physics or data. Physics-based methods are intended for detecting weather conditions like fog, snow, or rain [62–65]. An example of a proposed method is a camera-based rain gauge by Allamano et al. [62], which measures rain intensity using a single camera. These methods offer more interpretability and stability compared to data-driven methods, but they cannot handle complex phenomena like water buildup on roads and are dependent on camera parameters [62]. On the other hand, DL-based techniques such as CNNs have been widely used in related applications [57, 66, 67, 59]. They can be trained to manage various illumination and acquisition conditions without the need for calibrated cameras, unlike physics-based methods. Frame-by-frame classification through fine-tuning of state-of-the-art CNNs is the approach taken by most existing works, without consideration of the temporal correlation between frames [67, 58, 59, 57, 66, 68].

In [57], the authors studied a system for real-time road condition detection in a similar configuration. They more than 1 million frames captured by cameras placed on highways and streets, and applied a semi-supervised strategy to label the large dataset before fine-tuning CNN models. However, their work has limitations compared to the study reported herewith. Their dataset was collected over a period of three months while the dataset developed for the work reported herewith spans two years and includes continuous sampling over several months. Additionally, this study demonstrates that a temporal model like a Convolutional Long Short-Term Memory (ConvLSTM) outperforms a frame-by-frame approach by reducing false alarms and capturing the temporal evolution of the phenomenon more effectively. The proposed pipeline involves self-supervised and semi-supervised training of the temporal model with spatially and temporally consistent data augmentation for improved generalization.

The utilization of self-supervised pre-training is employed to take advantage of the availability of unlabeled frames. Such methods learn feature representations that

are semantically meaningful through pretext tasks, which don't require semantic annotations but do require a higher-level semantic understanding to solve [69]. These representations can later be fine-tuned to the target downstream task with the help of a labeled subset of data. Self-supervised pre-training and pseudo-labelling has been previously shown to outperform pseudo-labelling alone in a semi-supervised setting [70]. The proposed methodology draws inspiration from contrastive SSL techniques [71, 72], which have proven effective in transferring to fine-grained image classification [69]. Contrastive self-supervised methods, which treat each image as a unique instance, utilize aggressive data augmentation to generate multiple views [71, 69, 73]. Meanwhile, spatiotemporal extensions for video sequences rely on the assumption that each sequence represents a different scene [74]. Both approaches are unsuitable for the proposed dataset, which contains a small number of scenes acquired by fixed cameras, but a large number of consecutive frames. SSL has also been applied to remote sensing [75, 76]. However, the proposed dataset, which contains multiple images of the same area acquired at regular intervals, is sampled sparsely along the spatial dimension and at high resolution along the temporal dimension, whereas satellite imagery is sampled continuously in the spatial dimension but sparsely in the temporal dimension.

Thus, drawing inspiration from SSL in the medical domain [72] - in which similarity between two images is based on discrete variables such as lesion type, location, and size - here an SSL approach is adopted relying on the discrete features of the dataset and using a ConvLSTM. The technique is validated on a real-life dataset containing continuous video streams spanning multiple months, providing a realistic picture of performance under varying illumination and weather conditions. The contributions are:

- The introduction of a semi-supervised training approach that leverages unlabeled data through self-supervised pre-training and established semi-supervised techniques, specifically, contrastive self-supervised pre-training for road-side and surveillance camera video streams;
- The design and training of a ConvLSTM for road condition classification, including temporally consistent data augmentation techniques to mimic outdoor illumination changes;

Camera site	A1	A2	A3	A4	A5	A6
Acquisition period	02/2020 - 05/2020	02/2020 - 05/2020	07/2018 - 07/2019	10/2018 - 07/2019	10/2018 - 05/2020	10/2018 - 07/2019

Camera site	A7	A8	A9	A10	B1 - B15
Acquisition period	07/2018 - 09/2018	10/2019 - 11/2019	08/2018 - 07/2019	02/2020 - 05/2020	12/2020 - 01/2021

Table 2.1 Period of acquisition for each road-camera involved in the analysis.

- The evaluation of by-frame (CNN) and temporal (ConvLSTM) models on a well-balanced dataset covering a wide range of acquisition, illumination, and weather conditions, using appropriate per-event metrics to measure the ability to precisely identify the start and end time of wet road events.

The remaining of the chapter is structured as follows. Section 2.3.1 provides an overview of the dataset used for training and testing. The training strategies adopted for the task are described in Section 2.3.2. In Section 2.3.3, the settings used to carry out the experiments are reported in detail, whereas in Section 2.3.4

2.3.1 Dataset

This study makes use of a real-life dataset collected from 25 roadside Full HD cameras positioned primarily in Europe and primarily in the Northern hemisphere. The cameras were primarily located in extra-urban areas along crucial segments of the road network, exposing them to a wide range of weather and traffic conditions [77]. The dataset consists of images captured from a combination of bullet and PTZ cameras, installed at an approximate height of 5m and angled towards the road at 10°-20° [77]. The camera locations are presented in Fig. 2.5 and their acquisition periods in Table 2.1. The original frame rate of 12 frames per minute was reduced to 1.3 frames per minute to lower computational costs. Only frames captured between sunset and sunrise were used, with light sensors utilized in some cameras to distinguish between day and night. In other cases, sunset and sunrise hours were calculated using ephemeris equations [78] and geographical information. This section will delve into the data acquisition and selection process through causal analysis (Section 2.3.1), annotation process (Section 2.3.1), and the properties and training/validation split of the dataset (Section 2.3.1).

Causality Analysis A range of factors impact road conditions, and therefore must be taken into account when constructing a training and validation set from collected

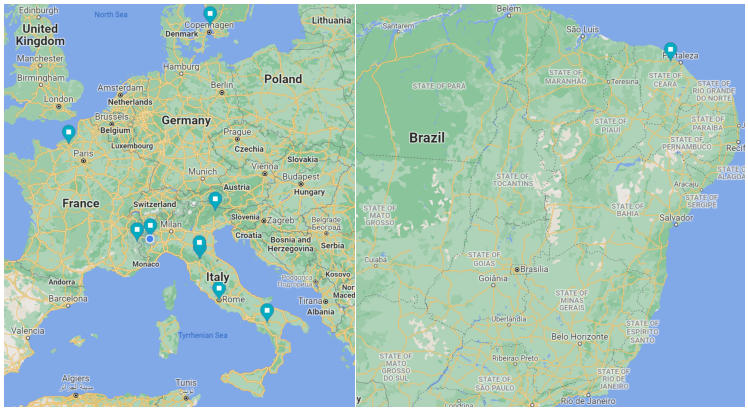


Fig. 2.5 Geographical positions of the road-cameras involved in the analysis.

data [28, 79, 80]. The aim of this analysis was to provide a solid foundation for the data to be split into a training and validation set and to characterize the properties of the domain for the classifier.

The causality diagram for the task of detecting the presence of water in video frames captured on a specific road stretch was described in details in Section 2.2.4. The resulting model is shown in Fig. 2.4, and here it is described in short to help the reader. The goal is to predict the likelihood Y of water being present on the road, given a frame or sequence of frames X, where Y can also represent different weather conditions such as *wet*, *dry*, or *snowy*. The actual presence of water is depicted by the latent variable W, as direct observation of the phenomenon was not possible, such as through on-field sensors.

There are several reasons for water buildup on the road. Weather conditions play a significant role, but the interplay between weather and the road's drainage system is also crucial. For example, light rain may not result in water buildup, while heavy rainfall could cause persistent water buildup. Weather conditions were treated as a latent variable in the presented analysis as it was not possible to access weather reports, but they can largely be inferred from the images. Variables such as camera site, time of day, and season act as confounders as they affect both illumination and water buildup frequency. The camera site variable encompasses multiple elements such as geographical location, road morphology, type of asphalt, road drainage, etc. All of these factors impact the frequency and characteristics of wet road events. The image X acts as a collider between the true phenomenon W and other independent

variables such as camera orientation and model. Hence, care must be taken as colliders can lead to biased datasets [81, 56].

Based on the diagram, the task is considered *causal*, as the label is directly inferred from the input frame, which acts as a mediator with respect to the underlying phenomenon [54]. On the other hand, the annotators were aware of the season, time of day, and weather, which were either available or could be easily inferred from the image, as indicated by the dashed arrows in the diagram. Although annotators were instructed not to rely on these variables for classification, it is possible that this information was useful for human annotators to perform the task. Hence, from this perspective, the task can also be considered *confounded* and *anticausal*, which is in line with theoretical conjecture that SSL should not improve performance on causal tasks [54].

Annotation Procedure To validate and test the performance of SSL strategies, a subset of the data was manually annotated to obtain a set of clean labels. Furthermore, this annotated subset was used to transfer the learned representation to the target task. In particular, all of sites B1-B15 were annotated due to their short acquisition period and challenging illumination conditions, which were not well represented in sites A1-A10. For sites A1-A10, a random number of days per site was selected and annotated.

The annotation process involved five annotators: two experts and three juniors. The expert annotators established the criteria for labeling and provided initial training, while consensus meetings were used to review the initial annotations and resolve ambiguous cases. A frame was labeled as wet if the water build-up or water pools were deemed significant enough to reduce vehicle traction and increase the risk of accidents. All other frames were labeled as dry, even if the road was moist or drying up. This choice, which is more challenging than previous works that defined the wet class as "a spectrum of conditions from moist roads to puddles to soaking wet" [57], was made to avoid an unacceptable false positive rate in real-world applications.

Initially, the problem was framed as a binary classification problem, but auxiliary classes were defined to account for cases in which labeling was not possible:

- *Poor visibility*: This class represents frames in which visibility is reduced by fog or other weather conditions to the point that labeling is not possible.

- *Dark*: This class represents frames in which the road is severely under-illuminated or under-exposed, typically at the start or end of the day.
- *Offline*: This class represents the static feed when the camera is offline due to network errors.
- *Over-exposed*: This class represents frames in which the image is too over-exposed to clearly assess the state of the road, typically around mid-day, depending on camera orientation.

To reduce the annotation time, only a small portion of the dataset (approximately 3,000 frames) was completely annotated by hand. For the remaining frames, a classifier was trained to produce tentative labels, which were then manually refined following the procedure outlined by Ramanna et al. [57]. Features were extracted by fine-tuning a ResNet50 CNN pre-trained on ImageNet and applying a k-Nearest Neighbor classifier ($k = 50$). The tentative labels were grouped by camera and denoised using a majority filter to exploit temporal consistency. Finally, all labels were manually refined and confirmed using the Microsoft CVAT tool [82] by loading each day of acquisition as a separate video. Critical or borderline cases were discussed in consensus meetings. Hence, this process is considered equivalent to manual annotation and reasonably noise-free.

Frames labeled as poor visibility, dark, offline, or over-exposed were removed from the final dataset. These special classes were used only during manual labeling, but were not included as additional classes due to their low prevalence, as discussed in Section 2.3.2. The only exception was over-exposed frames, which were automatically discarded by applying a threshold on the average intensity. This simple approach was found to be equivalent to manual labeling on the annotated subset.

Dataset Division and Training/Validation Split Strategy The overall dataset encompasses around 400,000 frames collected from 24 cameras. The annotated portion of the dataset was separated into a training set and a validation set, while the rest was kept for further training purposes.

The dataset was sampled with each day being treated as a separate entity. As night frames were disregarded in the analysis, each day represented a disjoint and non-overlapping set of frames. The frames captured by a single camera in a day are referred to as a *camera day* throughout the analysis. Thus, each camera day was

either allotted to the training set or the validation set, and was either annotated or not. This strategy ensures that the frames in the validation and testing sets are statistically uncorrelated and helps to evaluate the performance on an event-wise basis, instead of a frame-wise basis. Additionally, all the illumination conditions that occur during the day are equally represented, thereby eliminating biases.

The training/validation/testing split was done at the camera day level, taking into account the major confounding factors discussed in Section 2.3.1, specifically the camera location and season.

As depicted in Table 2.1, the number of camera days available varied across the different sites. Sites with more than 50 camera days (6) were randomly divided into a training and validation-test set to guarantee that all seasons were equally represented. The validation-test set was further split into a validation set (50%) and a test set (50%), stratifying by camera orientation. Sites with fewer than 10 camera days (15) were either assigned to the training or validation set, as it was not feasible to reserve part of the data for external testing in these cases. Sites were manually assigned to ensure that sites with similar characteristics and morphologies (e.g., the presence of elevated roads, bridges, or tunnels) were evenly divided between the two datasets. This strategy guarantees that the training set and the validation set include similar acquisition conditions, while also avoiding network memorization of each acquisition site. Furthermore, sites located outside of the Northern hemisphere were included in the training set, as the amount of data available would not be adequate to test the performance of the model in these climatic conditions.

The distribution of the training and validation sets is detailed in Table 2.2, in which "Training-50K" refers to the annotated portion of the training set, while "Training-300K" refers to the complete set, including both annotated and non-annotated frames.

Finally, to account for temporal continuity, an *event* is defined as a sequence of consecutive frames labelled as "wet". This definition will be further discussed in Section 2.3.3 in the context of performance assessment. As shown in Table 2.4, very short events (less than 15 minutes) and very long events (more than 6 hours) are rare, and many wet events start in the morning, prior to 9 am.

Table 2.2 Distribution of the training, validation and testing sets.

	Training-300K	**Training-50K**	**Validation Set**	**Test Set**
Total Frames	307,034	50,152	47,613	42,164
Dry Frames	245,087	43,038	41,837	38,786
Wet Frames	61,947	7,114	5,776	3,378
Camera Sites	14	14	16	9
Camera Days	1,458	263	239	208
Wet Events	/	172	118	78
Wet Events Longer Than 120 Frames (360 minutes)	/	12	13	12
Wet Events Shorter Than 5 Frames (15 minutes)	/	23	14	8
Wet Events Starting In The Morning (Before 9AM)	/	58	52	34

2.3.2 Methodology

This section outlines the method for wet road detection. Data preprocessing and augmentation techniques are presented in Section 2.3.2. The proposed approach trains models of increasing complexity, starting with a self-supervised and SSL trained CNN (Section 2.3.2) and followed by the integration of the feature extractor into a temporal ConvLSTM model that is fine-tuned on video sequences (Section 2.3.2).

Data Preprocessing and Augmentation The first step in the process involves the preprocessing of the data to remove any text overlays present in the surveillance camera feeds. This is achieved through simple thresholding and morphological operators as the text is typically black or white in color.

For the CNN model, a set of random transformations were applied independently to each frame to augment the data, including shear (in the range –45 to +45 degrees), horizontal flipping (20%), cropping (in the range –0.05% to 0.10%), perspective transformation (0.1 ratio), random rotation (in the range –45 to +45 degrees), additive Gaussian noise, coarse dropout (3–15% of the pixels), and brightness variation (in the range –30 to +30).

For the temporal ConvLSTM model, the data augmentation should generate spatially and temporally consistent sequences to capture subtle changes in the appearance of the road. The temporally consistent method from Qian et al.[74] was adopted for the spatial and geometric transformations, where a set of random transformations (shear, flipping, cropping, perspective, rotation, additive noise, and dropout) is sampled for each sequence and applied consistently to all frames. The same settings were used as in the CNN model.

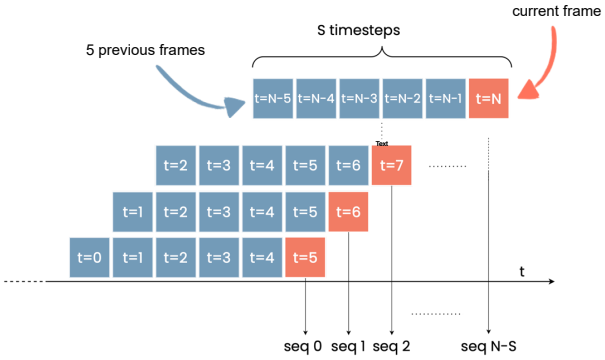


Fig. 2.6 Sequence generation process. Sequences generated for each frame included the $S-1$ previous frames.

A novel brightness transformation was introduced for the temporal ConvLSTM model to simulate changes in illumination conditions in outdoor scenes. This transformation multiplies the intensity of each frame by a factor $A\sin(k(i+b))$, where A, k, and b are randomly selected parameters. The parameters were set to have a maximum brightness increase of 2 and a maximum decrease of 0.5 to simulate a wide range of different illumination and climatic conditions. Examples of augmented video sequences are shown in Fig. 2.7.

CNN Model and Training Procedure The baseline model employed in this study is a Resnet50 network that has been pre-trained on ImageNet [83]. Given that incidents of wet roads are relatively infrequent and class imbalance could potentially hinder the convergence speed and overall generalization performance of the model on the test set [22], the focal loss function was selected for all experiments. While oversampling by repetition is a common technique to address data imbalance, it was discovered to be susceptible to overfitting.

To mitigate the effects of class imbalance, the multi-class formulation proposed by Cui et al. [84] was employed. This approach considers each sample along with a small neighboring region, rather than a single point. The effective number of samples E_{n_y} is calculated as $(1-\beta^{n_y})/(1-\beta)$, where n_y is the number of samples for class y and $\beta \in [0,1]$ is a hyper-parameter that regulates the rate of growth. This re-weighting strategy can be applied to any loss function $\mathscr{L}(\mathbf{p},y)$, resulting in a class-balanced loss $CB(\mathbf{p},y)$ expressed as:

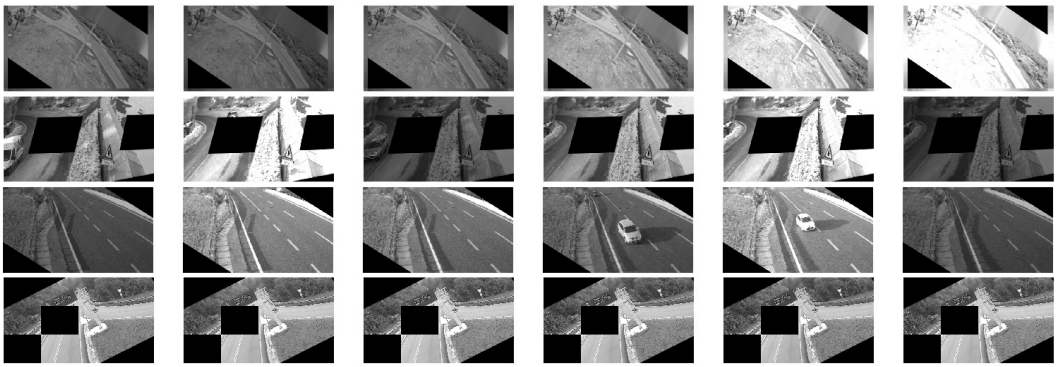


Fig. 2.7 Sequences generated by accounting for temporally coherent spatial augmentation and for a realistic simulation of outdoor brightness conditions.

$$CB(\mathbf{p},y) = \alpha_i \mathscr{L}(\mathbf{p},y) = \frac{1}{E_{n_y}} \mathscr{L}(\mathbf{p},y) = \frac{1-\beta}{1-\beta^{n_y}} \mathscr{L}(\mathbf{p},y) \tag{2.1}$$

where $y \in 1,2,...,C$ is the label and $\mathbf{p} = [p_1, p_2, ..., p_C]$ is the class probability vector estimated by the model. The class weight $\alpha_i \propto \frac{1}{E_{n_y}}$ is proportional to the inverse of the effective number of samples for class i. Setting $\beta = 0$ corresponds to no re-weighting, while choosing $\beta = 1$ leads to re-weighting by the inverse of class frequency.

Finally, the class-balanced focal loss is obtained by combining the focal loss with the above weighting scheme:

$$CB_{focal}(\mathbf{p},y) = -\frac{1-\beta}{1-\beta^{n_y}} \sum_{i=1}^{C} \left(1-p_i^t\right)^{\gamma} \log\left(p_i^t\right) \tag{2.2}$$

where $p_i^t = \text{sigmoid}(z_i^t)$ is the model estimated probability for class i and

$$z_i^t = \begin{cases} (z_i) & \text{if } i = y \\ (-z_i) & \text{otherwise.} \end{cases} \tag{2.3}$$

Parameter γ smoothly adjusts the rate at which easy examples are downweighted.

Self-Supervised Pre-training The neural network undergoes pre-training through a self-supervised contrastive method that leverages the inherent similarities present in the dataset, followed by fine-tuning to the target classification task, as depicted

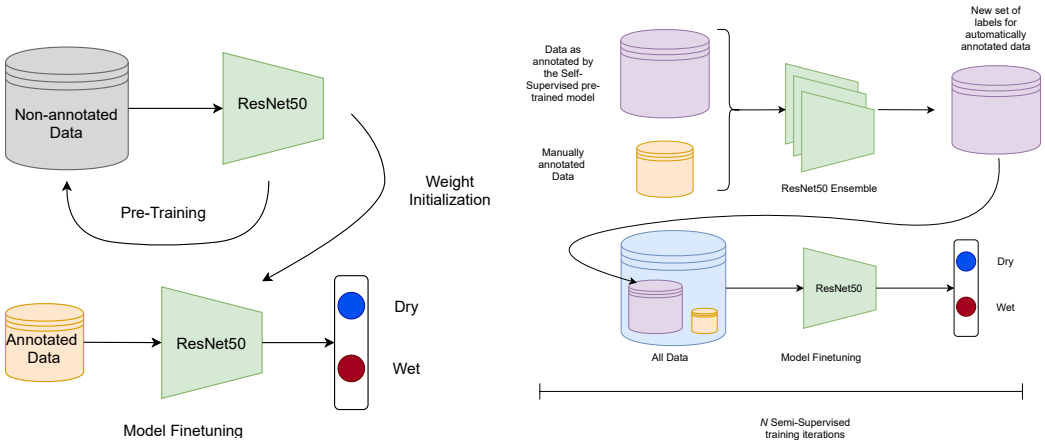


Fig. 2.8 Self-supervised pre-training (left) and the Semi-supervised fine-tuning (right) methodologies.

in Fig. 2.8. The network is pre-trained with the objective of learning to distinguish fine variations in spatiotemporal features by comparing frames taken at varying times and locations. In this work, time is treated as a continuous variable and the camera location is considered as a discrete variable. These variables are combined to generate sequences that contain samples of gradually increasing differences, as inspired by previous studies [72].

Starting from an anchor sample (A), sequences of five samples, A, B, C, D, and E were generated from positive samples with varying levels of similarity, and one negative sample, as shown in Table 2.3. The degree of similarity was determined based on three factors: the camera location, date, and time of day. Two images were considered similar if they were collected from the same camera location, within the same month (30 calendar days, excluding the year), or acquired less than two hours apart. This approach discretizes the concept of seasonality (month) and varying illumination conditions (time of day), and combines them with the camera location.

A five-stream Siamese network with a ResNet50 backbone was used for pre-training. The feature vectors were normalized to unit norm before computing the Euclidean distance, and a contrastive loss based on margins was defined as in Yan et al. [72]:

Table 2.3 The similarity between the anchor sample and other samples in the sequence is established by considering the camera site, date, and time of day in that order. A ✓ indicates similarity between the samples with respect to the considered factor, while an ✗ indicates difference.

	A	***B***	***C***	***D***	***E***
Camera site	Anchor	✓	✓	✓	✗
Date of the year	Anchor	✓	✓	✗	Don't care
Time of the day	Anchor	✓	✗	Don't care	Don't care

$$\begin{aligned} \mathscr{L}sequence = \max(0, d^2AB - d_{AC}^2 + m_1) + \\ \max(0, d_{AC}^2 - d_{AD}^2 + m_2) + \\ \max(0, d_{AD}^2 - d_{AE}^2 + m_3) \end{aligned} \tag{2.4}$$

where d_{ij}^2 is the squared Euclidean distance in the feature space between frames i and j, and m_i represents the margin distance applied to each sample feature representation, subject to the condition $m_3 > m_2 > m_1 > 0$. The loss encourages sample B to be close to anchor A in the feature space, while pushing samples C, D, and E further away, in that order. Given that each batch comprises S sequences, the final loss is calculated as $\mathscr{L} = \frac{1}{2S}\sum_{s=1}^{S}\mathscr{L}_s$. The self-supervised pre-trained backbone is then used to initialize the baseline instead of the standard ImageNet pre-training [85].

Semi-supervised Method The proposed approach is based on the widely-used semi-supervised learning technique referred to as pseudo-labeling [86]. The method involves generating pseudo-labels for unlabeled data utilizing an initial version of the classifier, which is then refined iteratively. To mitigate the effect of noisy pseudo-labels, an ensemble of three ResNet50 networks is employed, initialized through self-supervised pre-training or previous SSL iterations. Binarized predictions are obtained by selecting a threshold with a 10% false positive rate, and then combined using majority voting. Further refinement is done by applying majority voting along the temporal axis for temporal consistency.

The weight of pseudo-labeled samples is controlled by a hyper-parameter ε, which starts with a low value and gradually increases with the improvement of the classifier's performance. This allows for a trade-off between labeled and unlabeled data and is crucial in reducing the impact of noisy labels in early iterations [87, 88].

Temporal (ConvLSTM) Model and Training The proposed temporal model classifies each i^{th} frame by considering the preceding $N-1$ frames. The model is built using a ConvLSTM architecture, which comprises a ResNet50 backbone as a visual encoder, and a stack of three LSTM cells and two Fully Connected (FC) layers, as shown in Fig. 2.9.

Different variations of the LSTM model were experimented with. The first variation is the DenseLSTM [89], where the three LSTMs receive the visual features concatenated with the outputs from all previous cells, similar to the DenseNet architecture. The second variant substitutes the LSTM with a 1D CNN layer with kernel size equal to 2 and stride 1. The output from the 1D CNN is flattened and fed to the two FC layers. Dropout is applied after the first FC layer to reduce overfitting.

Temporal sequences were extracted from the continuous data stream by employing a sliding window approach, defined by the parameters: sequence length N, sampling rate s, and stride K. In the experiments, the length of the sequence was set to $N = 6$, with a sampling rate of $s = 1.3$ (1 frame every 3 minutes), resulting in a time window of 18 minutes. This time window was chosen based on domain knowledge and preliminary experiments and was deemed sufficient to capture the evolution of the phenomenon of interest. The stride K was set equal to the sampling rate for simplicity.

To reduce computational effort during training, the temporal model was initialized from the by-frame model through a two-step transfer learning approach. During the first phase, referred to as *warm-up*, only the LSTM was trained, with the feature extractor frozen. In the second phase, both the backbone and the LSTM were fine-tuned using the class-balanced focal loss defined in Section 2.3.2.

2.3.3 Experimental Settings

CNN model All experiments were carried out by resizing Full HD frames to 480×270 pixels, which slightly affects the classification accuracy but considerably accelerates training and testing processes. The hyper-parameters for the focal loss were set as $\beta = 0.9999$ and $\gamma = 2$ unless otherwise specified. Only the frames labeled as wet and dry were used, while others such as over-exposed, dark, offline, and poor visibility were removed as described in Section 2.3.1. The learning rate was determined on the training set via the learning rate finder methodology [90]

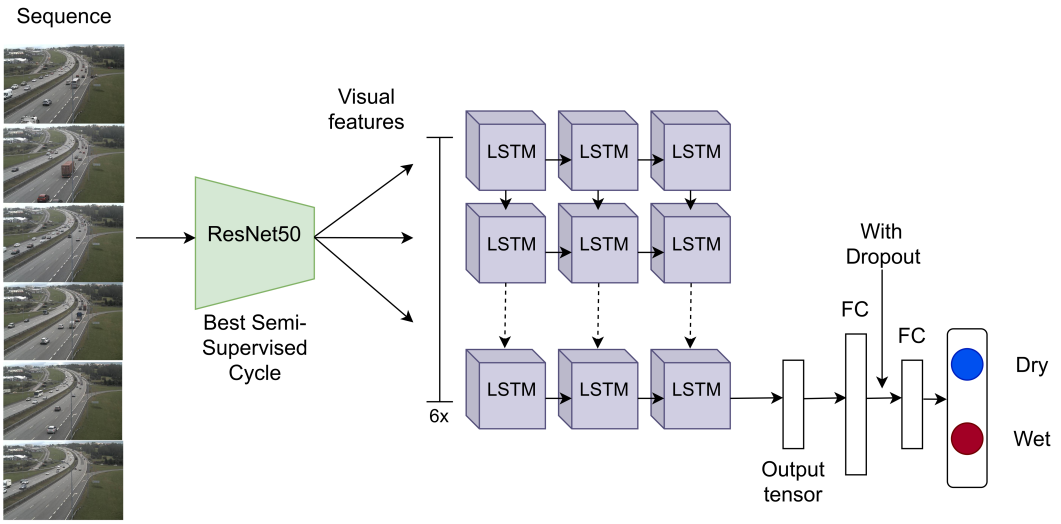


Fig. 2.9 Unrolles representation of the Temporal (ConvLSTM) model.

and set to $10^{-}6$. The Adam optimizer was applied with a batch size of 16 for all experiments.

The baseline was trained using SSL on 308,523 sequences for 20 epochs. The original training set was used to generate the sequence dataset where each sample acted as anchor A. Samples B, C, D and E were randomly selected to meet the requirements for each anchor. The generated sequences were split into 2,982 for validation and 305,541 for training. The hyper-parameters of the loss function were set to $m_1 = 0.2$, $m_2 = 0.3$ and $m_3 = 0.4$ (refer to Section 2.3.2 for more details on the loss function) with a batch size of $S = 3$. After the pre-training stage, the backbone was fine-tuned to the classification task. The learning rate was determined through the learning rate finder methodology [90] and set to 5×10^{-6}.

For SSL iterations, the pseudo-labels were downweighted by a factor ε, which started at 0.2 for the first iteration and increased to 0.9 for the fourth and final SSL iteration.

It should be noted that only the data from sites A1-A10 was used for the self-supervised pre-training and the first four SSL rounds as these sites have a broader temporal range than sites B1-B15. However, the latter sites include more diverse scenes and challenging illumination conditions, making them a suitable choice for the temporal model training and evaluation.

ConvLSTM model The training process for the temporal ConvLSTM model baseline followed a two-phase procedure. The optimizer utilized was the Adam optimizer and the loss function used was the focal loss with hyperparameters $\beta = 0.9999$ and $\gamma = 2$. During the warm-up phase, the network was trained for 4 epochs with a batch size of 64 and a learning rate of 9×10^{-5}. In the second phase, the network was fine-tuned for a further 6 epochs with a lower learning rate of 9×10^{-6} and a batch size of 16. In both SSL rounds, the value of ε was set to 0.9. The optimal learning rates, as determined using the learning rate finder algorithm, were 9×10^{-5} and 9×10^{-6} for the first and second SSL rounds, respectively. The frames were resized to 480×270 pixels, as was the case with the CNN model. The ConvLSTM model was trained on 256,882 sequences of $S = 6$ frames extracted from the Training-50K dataset through the methodology outlined in Section 2.3.2.

Experiments All the experiments were performed on a system equipped with a Intel® Core™ i9-9940X CPU with 32 GB RAM and a NVIDIA Titan RTX GPU (24 GB). The models were implemented in Keras v2.3 and Tensorflow v2 frameworks.

Evaluation of Model Performance In order to assess the performance of the devised models, two classes of performance metrics were used, at both the *frame* and *event* levels. The first set of metrics aimed to evaluate the ability of the models to classify each frame independently as either dry or wet. For this, the *Receiver Operating Curve (ROC)* was used, with the *Area under the ROC curve (AUC)* as the summary performance measure. The 95% *confidence intervals (CI)* for the AUCs were computed using bootstrap sampling with 1000 repetitions and replacement [91]. The p values were calculated using the paired non-parametric DeLong method for comparing correlated ROC curves [92], with correction for multiple tests using the Bonferroni method. The differences between the final two models, as well as between each semi-supervised iteration and the baseline model, were tested.

The second set of metrics evaluated the accuracy with which the models detected wet road events, the number of false alarms generated, and their distribution throughout the day or across different sites. An event was defined as a series of consecutive frames with the same label, and predictions were binarized to generate the predicted events. A threshold was selected at a fixed false positive rate of 10% to balance specificity and sensitivity and to make fair comparisons between models.

The ground truth and predicted events were matched along the temporal axis using the *Temporal Intersection over Union (IoU)* metric [93]. An event was considered to be detected if the IoU exceeded a threshold of 0.2. Any predicted event that was not associated with a ground truth event was considered a false positive or false alarm. However, if a predicted event was completely included in a ground truth event but did not reach the IoU threshold, it was not counted as either a false positive or true positive detection.

Finally, per-event performance was estimated through a variant of the ROC curve, known as the *Free-Response ROC curve (FROC)*. The FROC plots the recall versus the average false positive rate, which was computed by dividing the total number of false positives by the number of camera days and corresponds to the average number of false positives generated daily at each site. False positives with a duration of one frame were discarded. Each detected event was assigned a score equal to the maximum score assigned by the model to the frames included in the event. By construction, each event had a score equal to or higher than the threshold used for binarizing predictions.

2.3.4 Results

CNN Model: Self-Supervised and Semi-Supervised Pre-Training To evaluate the CNN model's ability to distinguish between dry and wet frames, a set of experiments that excluded sites B1-B15 due to the unavailability of unlabeled data and a relatively short acquisition period was conducted. The validation set consisted of 208 camera days from eight different sites, with four of them never seen during training. The CNN model was trained in consecutive steps as described in Section 2.3.2. The *Self-sup baseline* refers to the model that was pre-trained using the self-supervised methodology and then fine-tuned on the labeled portion of the dataset, while the *Semi-sup X* model refers to the X^{th} SSL round on the complete Training-300K dataset.

As shown in Fig. 2.10(a), the baseline AUC was $0.855(0.850-0.860)$, which increased to $0.910(0.906-0.913)$ in the 4^{th} round ($p < 0.0001$). However, only the first three iterations showed improvement over the previous one in ROC space, as evidenced by the intersecting curves in Fig. 2.10(a). The differences between each *Semi-sup X* model and the *Self-sup* baseline were highly significant ($p < 0.0001$)

Table 2.4 The validation and test set AUCs along with their corresponding 95% CI are presented. Statistical significance of differences between each SSL round and the *Self_sup* baseline were tested and indicated by asterisks (*) for p value < 0.001.

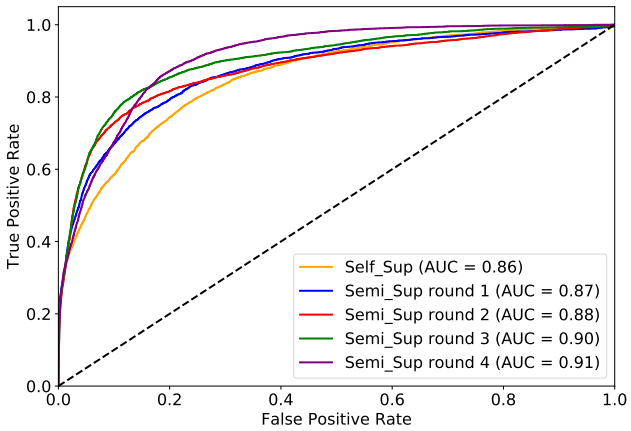
Model	Validation set (A1-A10)	Test set
Self_Sup	0.857 (0.852 - 0.861)	0.890 (0.885 - 0.895)*
Semi_Sup round 1	0.874 (0.869 - 0.879)*	0.912 (0.908 - 0.917)*
Semi_Sup round 2	0.878 (0.872 - 0.883)*	0.914 (0.909 - 0.918)*
Semi_Sup round 3	0.904 (0.889 - 0.908)*	0.923 (0.919 - 0.928)*
Semi_Sup round 4	0.911 (0.907 - 0.914)*	0.920 (0.915 - 0.923)*

on both the validation and test sets. The AUC values with 95% CI are presented in Table 2.4.

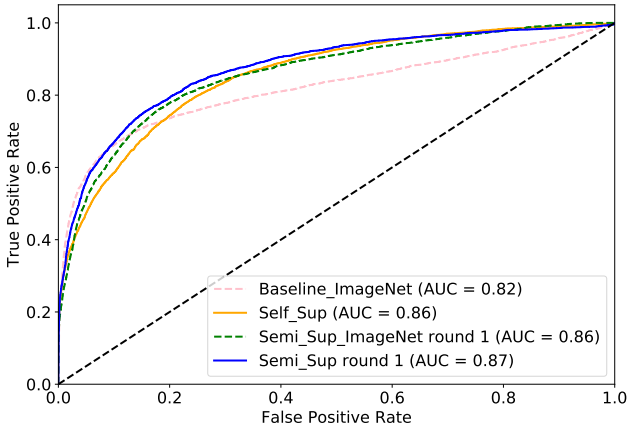
The performance varied greatly by site, for both the *Self-sup* baseline (per-site AUC: median 0.86, IQR 0.09, range $[0.58, 0.97]$) and the final *Semi-sup 4* model (median 0.895, IQR 0.06, range $[0.81, 0.98]$). The final performance for sites seen/unseen during training was 0.91 (range $[0.81, 0.98]$) and 0.875 (range $[0.84, 0.96]$), respectively, indicating that the model did not overfit to the specific sites in the training set. Performance was lower for sites with challenging weather or acquisition conditions, such as low prevalence of wet events or cameras located far from the road.

An ablation study was performed to compare the impact of self-supervised pre-training to the standard ImageNet pre-training. The *Baseline-ImageNet* refers to the model pre-trained on ImageNet and fine-tuned on the Training-50K dataset excluding sites B1-B15. The *Semi-sup-ImageNet* model refers to the 1^{st} SSL round on the complete Training-300K dataset, starting from the *Baseline-ImageNet*. The results of the ablation study are illustrated in Fig. 2.10(b) and the AUCs with 95% CI are compared in the same figure. The results showed that the *Self-sup* baseline outperformed the *ImageNet* baseline by a large margin ($p< 0.0001$) with AUC of 0.855 (0.850 - 0.860) compared to 0.819 (0.810 - 0.826). Despite a reduction in the performance gap, the *Self-sup* baseline still outperformed the *Semi-sup-ImageNet* model ($p< 0.0001$) with AUC 0.874 (0.869 - 0.879) compared to 0.859 (0.854 - 0.864).

Evaluation of ConvLSTM Next, the performance of the ConvLSTM models, which were introduced in Section 2.3.2 and fine-tuned from the final model described



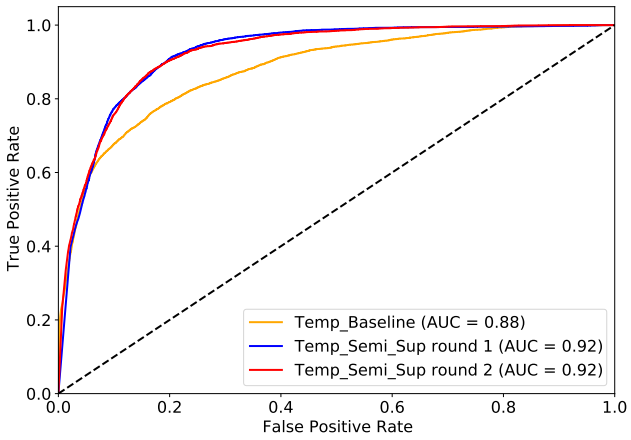
(a)



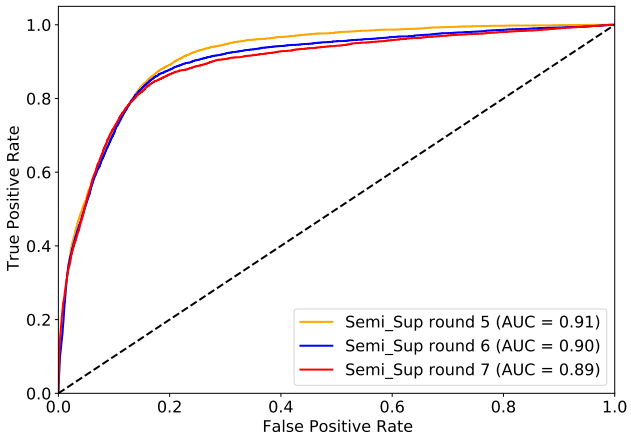
(b)

Fig. 2.10 ROC curves comparing the performance of the baseline model with self-supervised initialization (*Self-sup*) to models fine-tuned through SSL on the large Training-300K dataset. (a) Four rounds of SSL performance comparison. (b) Comparison of the baseline model with (continuous line) and without (dashed line) *Self-sup* initialization. Models initialized with self-supervised methods achieve superior performance, with or without fine-tuning via SSL.

in Section 2.3.4 was evaluated. To begin with, the performance of the LSTM was compared with those of DenseLSTM and 1D CNN models on a subset of the Training-50K dataset, excluding the B1-B15 sites. The LSTM model achieved the highest AUC, which was equal to 0.87, followed by the DenseLSTM with 0.88, and the 1D CNN with 0.85. Therefore, the LSTM model was selected, as it offered the best balance between accuracy and computational complexity.



(a)



(b)

Fig. 2.11 ROC curves of the ConvLSTM model (a) and 2D CNN (b) trained on the complete Training-300K dataset. Both models were fine-tuned from the *Semi-sup 4* model shown in Fig. 2.10(a). Three rounds of SSL were performed for both models.

All further experiments were performed on the complete dataset, including sites B1-B15. The models were initially trained on the labelled Training-50K dataset (*Temp-Baseline*) and fine-tuned for two more semi-supervised iterations (*Temp-Semi-Sup X*). To ensure a fair comparison between the ConvLSTM and CNN models, the CNN was also fine-tuned for three additional rounds on the complete dataset.

As shown in Fig. 2.11, the *Temp-Baseline* model obtained an initial AUC of $0.879(0.875-0.883)$, which is higher than the *Self-sup* baseline but lower than the best *Semi-sup X* model ($p< 0.0001$). With two SSL rounds, the performance increased to $0.923(0.920-0.926)$ ($p< 0.0001$), with each iteration outperform-

Table 2.5 The AUC values and their 95% CI for the validation and test sets are provided. Significant differences between each SSL round and the *Temp_baseline* were tested and are denoted with asterisks (*) for p value < 0.001.

Model	**Validation set**	**Test set**
Temp_Baseline	0.879 (0.875 - 0.883)*	0.890 (0.885 - 0.895)*
Temp_Semi_Sup round 1	0.923 (0.921 - 0.926)*	0.907 (0.902 - 0.912)*
Temp_Semi_Sup round 2	0.924 (0.921 - 0.927)*	0.923 (0.919 - 0.928)*
Semi_Sup round 5	0.914 (0.911 - 0.917)*	0.914 (0.909 - 0.918)*
Semi_Sup round 6	0.899 (0.895 - 0.902)*	0.883 (0.876 - 0.889)*
Semi_Sup round 7	0.892 (0.888 - 0.896)*	0.886 (0.880 - 0.892)*

ing the previous one. The performance of the CNN model reached a peak of $0.914(0.920-0.917)$ before reducing to 0.891 (0.887 - 0.895). The AUC values with 95% CI are reported in Table 2.5. The difference between each *Temp-Semi-sup X* model and the *Temp-Baseline* was highly significant ($p< 0.0001$) on both the validation and test sets.

It's worth mentioning that the ConvLSTM and CNN models have different behaviours at the event level, even though they achieved similar AUC values. For the ConvLSTM model, sensitivity increased from 50% (60/118) to 60% (71/118), while the FP rate dropped from 0.689 (181/263) to 0.37 (99/263), as shown in Table 2.6. Of the 47 false negatives, 33 were not detected at all, while the rest failed to meet the IoU threshold. Many of the false positive events occurred when the road was damp, but not wet. About 25% of the false positive events occurred in the tail of a wet event (13 within a 1-hour window and 10 within a 3-hour window), and another 25% occurred between 4 am and 9 am in the early morning. The 2D CNN model also exhibited a similar behaviour. The superior performance of the ConvLSTM model is further illustrated by the FROC curves in Fig. 2.12.

The average inference time, as determined through validation on a system equipped with an Intel® Core™ i9-9940X CPU with 32 GB of RAM and a NVIDIA Titan RTX GPU (24 GB VRAM), was found to be 0.024 seconds/frame for the CNN model and 0.034 seconds/frame for the ConvLSTM model. The ConvLSTM model's inference time is greater, as it requires processing of six frames via its convolutional backbone. However, the ConvLSTM model's time efficiency can be improved through optimization when applied to a continuous video stream, as most of the computations from adjacent sequences can be cached. It should be noted that

Table 2.6 Comparison between best performing CNN and ConvLSTM models.

	Recall	*FP Events*	*FN Events*	*TP Events*	*10% FP Rate Threshold*
Best CNN	0.50	181	58	60	0.90
Best ConvLSTM	0.60	99	47	71	0.96

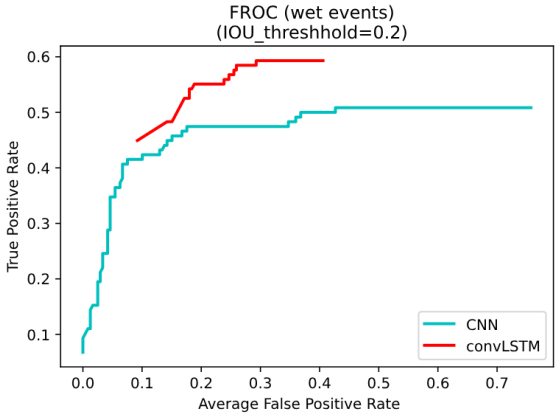


Fig. 2.12 Per-event recall vs. average number of FP events.

the inferred time does not account for the latency involved in transferring images to a cloud server with similar hardware capabilities. Further optimization or compression is necessary for the proposed models to be executed at the edge, for example by a smart camera.

Impact of Performance Several factors were observed to have an impact on the ConvLSTM model's performance. To start with, the performance is still site-dependent (median 0.915, IQR 0.11, range $[0.80, 0.99]$), similar to the previous *Semi-sup 4* model (median 0.895, IQR 0.06, range $[0.81, 0.98]$). The final performance for sites encountered or unseen during training was 0.91 (range $[0.80, 0.96]$) and 0.915 (range $[0.80, 0.99]$), respectively, indicating the model's ability to generalize to new sites.

Additionally, frames were divided based on the time of acquisition and AUCs were calculated for each hour of the day (Table 2.7). To account for variations in sunrise and sunset across seasons and sites, frames before 8am and after 5pm were grouped together. The performance decreases during early morning, when roads are often wet, and in late afternoon/evening, mostly due to challenging lighting conditions.

Table 2.7 Performance (AUC) for the ConvLSTM model with respect to the time of the day.

Time range	ROC-AUC
03:00 - 08:00	0.897
08:00 - 09:00	0.900
09:00 - 10:00	0.923
10:00 - 11:00	0.933
11:00 - 12:00	0.952
12:00 - 13:00	0.949
13:00 - 14:00	0.952
14:00 - 15:00	0.929
15:00 - 16:00	0.893
16:00 - 17:00	0.894
17:00 - 23:00	0.952

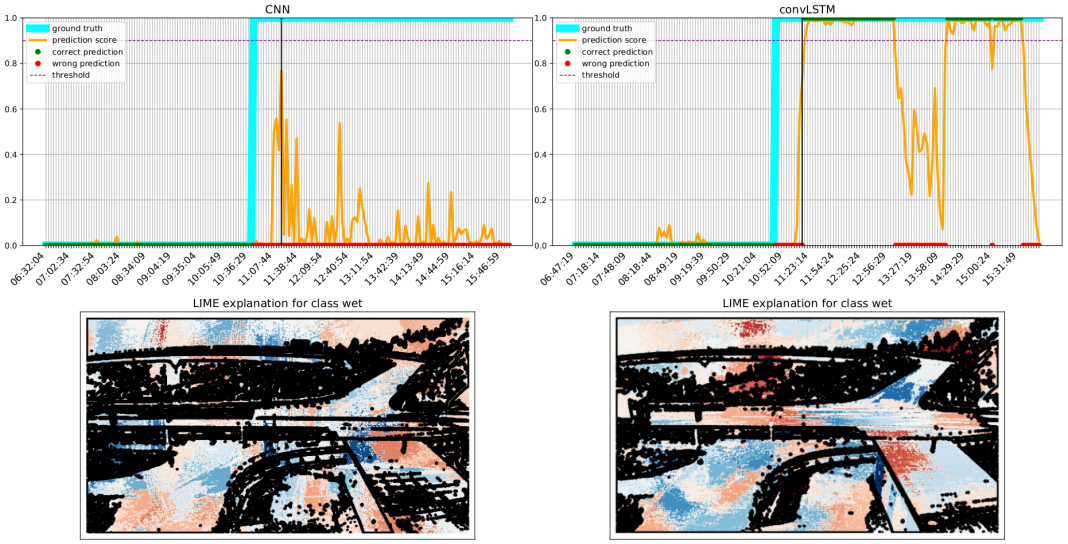


Fig. 2.13 An explanation generated by LIME is shown for a transition frame, as predicted by both models. The blue areas highlight features that are indicative of the wet class, while the red areas highlight features that are indicative of the dry class. Please refer to the color version for best viewing.

Transition plots and explainability A comparison of the effectiveness of the top CNN and ConvLSTM models was performed by utilizing transition plots. These plots display how the model's score assigned to each frame evolves during a rainy scenario, providing insight into the model's prediction confidence and consistency. Ground truth values and the trigger threshold for a wet frame prediction are also displayed

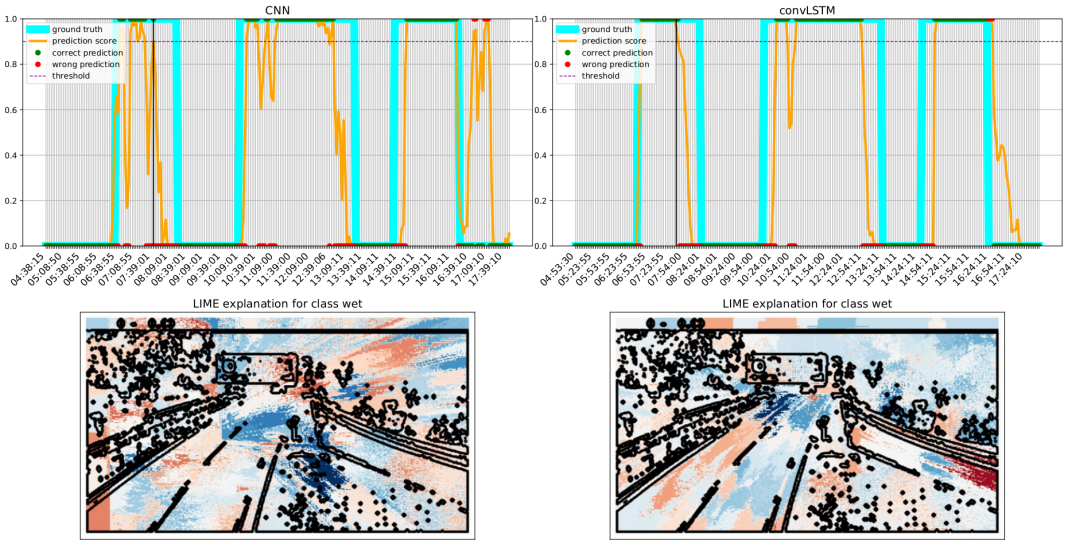


Fig. 2.14 Comparison of the transition profile for a severe event, where the ConvLSTM model shows better detection of the two events during morning hours. The model highlights blue areas indicating wet conditions and red areas indicating dry conditions. For better visualization, please refer to the colored image.

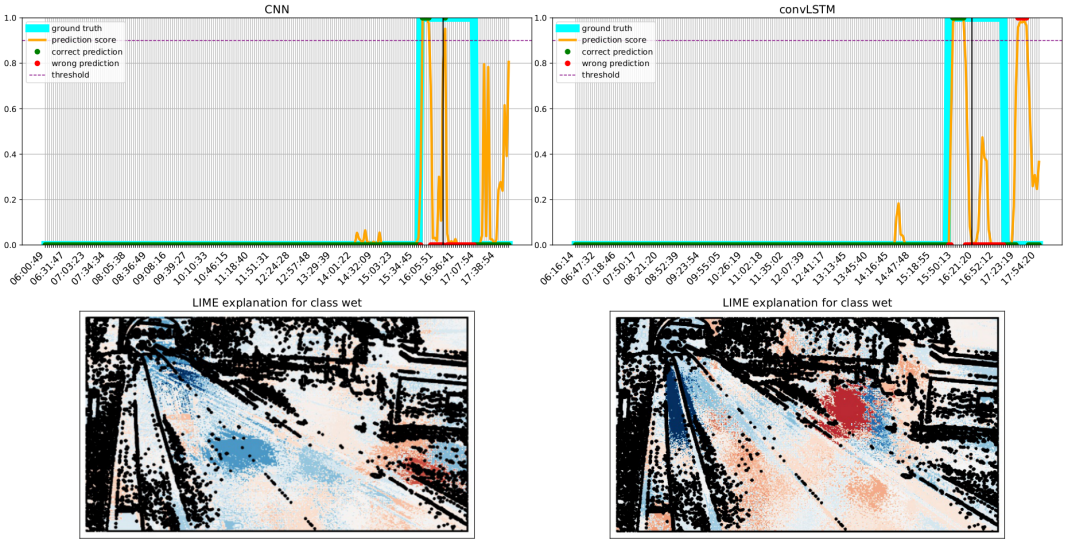


Fig. 2.15 An explanation generated by LIME is shown for a transition frame, as predicted by both models. The blue areas highlight features that are indicative of the wet class, while the red areas highlight features that are indicative of the dry class. Please refer to the color version for best viewing.

on the plots. Additionally, LIME [94] is employed on a selected frame (indicated by a black line) to demonstrate how different regions impact the predictions.

Fig. 2.13, Fig. 2.14 and Fig. 2.15 present transition plots for three extreme weather events at sites with varying topography. Although both models detect the event, the ConvLSTM predictions more accurately depict the underlying event. The LIME plots demonstrate that the ConvLSTM model is better equipped to identify which sections of the road are wet. In fact, the parts highlighted by the ConvLSTM model were found to align more closely with the manual annotations made by the raters. Both models consider contextual information, such as the presence of humidity and fog between trees in Fig. 2.13. It is worth noting that the model's performance decreased significantly when it was trained solely on segmented road portions (results omitted for brevity).

Discussion The results of this study indicate that the use of self-supervised semi-supervised learning can effectively enhance the performance of road condition estimation using partially labeled datasets, which aligns with previous findings in weather analysis [57] and CV [70]. However, a direct comparison with other studies is not feasible due to the absence of a public benchmark [57–59].

This study adopted a more stringent definition of "wet" road compared to previous literature [57], with the goal of reducing false alarms and accurately identifying road conditions that pose a risk to road safety. This definition affected both the training and evaluation processes, as many false positives came from instances where the road was damp but not truly "wet".

A visual inspection of the ground truth, demostrated how hand-crafted labels may have some degree of subjectivity, even if potentially mitigated through consensus meetings. However, accurately determining the start and end time of each event still remains a challenge, as the transition between "dry" and "wet" is often gradual. In future work, this uncertainty will be addressed through the use of temporal label smoothing [95].

This study is the first to consider intra-frame correlation in both the training and test phases, and to report performance at both the frame and event levels for road condition estimation and weather analysis. This study proposed novel strategies for self-supervised pre-training and temporally-aware data augmentation, and the ConvLSTM model achieved better results than CNNs, as confirmed by LIME explanations. However, the convolutional backbone was pre-trained using semi- and self-supervised techniques on individual frames, as the training time for the

ConvLSTM model was significantly higher and curriculum learning was found to stabilize the LSTM training process [89]. Future work may include exploring the impact of self-supervised pre-training on performance, possibly by extending the self-supervised technique to handle temporal sequences, or by comparing against other self-supervised pre-training methods like clustering-based ones [69].

It should be noted that the absolute performance of the models could be improved by increasing the temporal and spatial resolution of the data. To test this hypothesis, the final ConvLSTM model was fine-tuned for six additional epochs after doubling the image size from 480×270 to 960×560. The overall AUC increased from 0.92 to 0.93, with some sites showing an increase in AUC (average increase of 0.046) and others showing a decrease (average decrease of 0.017), depending on the camera-to-road distance. The models were trained on grayscale images due to the limited availability of RGB videos, which may have affected performance, but the conclusions of this study should not be impacted as the focus was on the relative performance of different models using the same data.

2.4 Concluding Remarks

Incorporating causality is a fundamental challenge in ML research. In the first part of the chapter, causal analysis was investigated as a highly effective technique to characterize the properties of ML datasets. Previous works have proven its effectiveness in the medical domain [53, 56], and here the methodology was extended to a real-life example from an industrial research project to prove its feasibility and potential benefits. In future works, a more in-depth evaluation of the impact of modelling the dataset based on causal diagrams on the performance and generalization ability of the trained models will be performed. Nonetheless, this work should encourage practitioners adopting this systematic approach to the analysis of data collections in other domains, as this may help to manage dataset biases and concept drift in the training and deployment of neural networks.

Nonetheless, this chapter offers compelling encouragement for practitioners across domains to adopt this systematic approach when analyzing data collections. Such an approach holds promise in managing dataset biases and deftly navigating the challenges associated with concept drift during the training and deployment of neural networks.

Beyond its conceptual significance, causal modeling profoundly shapes the modeling process, particularly in scenarios involving novel feature relationships or alterations to existing features. As new feature relationships come to the fore, causal modeling provides a structured framework for comprehending the interplay between these new elements and their established counterparts. Through the utilization of causal diagrams, the nature of these features as causes, effects, or confounders in relation to the existing ones can be discerned. This critical understanding not only guides the seamless integration of these features into the model but also informs strategies to effectively address their potential impacts.

Moreover, causal modeling enriches the understanding of underlying data relationships, thereby enhancing the model's robustness. When new features are introduced or changes are implemented, evaluating the model's resilience involves assessing the alignment of these modifications with the causal relationships identified within the model. By gauging the consistency between alterations and established causal relationships, vulnerabilities can be unearthed, facilitating adjustments that ensure the model's performance remains coherent and aligned with anticipated behavior.

In instances where causal features transition into hard variables—indicating a shift from inferred relationships to directly measured quantities (e.g., when a mediator or confounder is directly observed)—adaptations to the model's structure might be warranted. Causal models provide valuable insights into the consequences of such feature transformations on the overarching causal fabric of the data. This knowledge guides decisions surrounding the refinement of the model's architecture, encompassing potential structural amendments or the incorporation of supplementary variables to effectively manage confounding factors.

It is imperative to acknowledge that while causal modeling offers substantial advantages, it does not confer immunity against modeling challenges. Its efficacy hinges on factors such as data quality, accuracy of presumed causal relationships, and the intricacies of underlying mechanisms. In cases of data scarcity, causal modeling can still yield informative glimpses into potential relationships, although outcomes might be more susceptible to noise and uncertainties. Similarly, within data of subpar quality, the accuracy of inferred causal relationships might be compromised.

Moreover, the complexity introduced by non-linear relationships between variables and scenarios involving a multitude of interacting variables adds an additional

layer of challenge to causal modeling. When confronted with such intricacies, traditional causal modeling approaches might struggle to capture the intricate web of causality accurately. These complexities can lead to misinterpretations or oversimplifications, potentially undermining the reliability of causal insights. In these situations, advanced techniques that can handle non-linear relationships and complex interactions become essential to ensure the fidelity of the causal model.

Thus, while causal modeling remains a potent tool, its outputs necessitate judicious interpretation and validation against real-world outcomes whenever feasible. Embracing the complexity inherent in non-linear relationships and intricate variable interactions is essential to unlock the full potential of causal modeling in these challenging scenarios.

In the second part of the chapter, the complete case study - configured with causal modeling - is reported in details, along with a newly proposed technique for road condition assessment (in particular, the detection of wet road events in untrimmed video sequences). While prior works in the field typically use standard CNNs for frame-by-frame categorization, this study incorporates the temporal correlation among successive frames by applying a ConvLSTM model. The utilized training method integrates novel self-supervised pre-training procedures with a simple pseudo-labelling approach to make use of a vast pool of unlabeled data.

Experimental outcomes demonstrate the superiority of the ConvLSTM model over a standard CNN, not only in detecting more events with lower false positive rates, but also producing predictions that are more congruent with human observation. The temporal model appears to generalize effectively to unseen sites, and the differences seem to stem from the inherent difficulties, such as unique lighting or weather conditions related to a given location.

Future studies will aim to refine experimental results, for instance, by extending the self-supervised pre-training to handle sequences instead of individual frames. Further obstacles to be addressed include differentiating between borderline conditions and potential labeling noise due to inter-rater variability, as well as misclassification in the pseudo-labeling steps. To minimize the labeling expenses, more robust and systematic SSL methods could aid in separating challenging samples from noisy ones, and eliminate the need to adjust the extra hyperparameter ε. Improved results could also be obtained by evaluating alternative designs for the temporal model, including those based on 3D convolutions [96].

Finally, although the classification problem is binary, issues related to long-tailed distributions arise as certain weather and road conditions are intrinsically rare and only occur in limited locations. In such cases, fine-tuning the model to specific sites may enhance performance, although accumulating larger datasets could resolve this challenge in the future.

Chapter 3

Interpreting DL Models

Part of the work described in this chapter was originally presented in [97, 98].

3.1 Introduction

With the ability to recognize patterns in data, DL models can learn from large datasets and make accurate predictions. However, one of the primary challenges with these models is their opacity. The models' decision-making processes are often black-boxes, making it difficult to understand how the models arrived at their conclusions [94]. This lack of transparency creates a trust gap between the model and its users, which is why there is a growing demand for explainability techniques for these models.

Drawing from the most recent literature on the topic, explainability can be defined as the ability to *provide insights to a targeted audience to fulfill a need* [99]. According to this definition, XAI techniques can, for instance, help researchers and practitioners (the audience) get a better understanding of model behaviors (the insight) to perform a validation before deployment in safety-critical applications (the need). In other words, the field of XAI focuses on developing methods and techniques to explain the decision-making processes of complex AI models. XAI aims to bridge the gap between the user and the model by providing a transparent view of the model's decision-making process. Explainability is crucial in various fields where decisions made by AI systems can have significant consequences, like healthcare and finance, as it can help ensure that models are fair, transparent, and free

from bias [100]. In the medical field, explainability can help physicians understand why a model recommends a particular diagnosis or treatment, leading to better patient care [101].

In the case of CV tasks and CNN models, most XAI approaches rely on saliency and produce heatmaps [102] as a form of explanation, quantifying explanations in the form of *this image depicts a cat because of the highlighted region*. Although this approach can be useful for identifying wrong correlations, a reasonably placed heatmap is not enough for a human user to fully trust the algorithmic decision, nor for a developer to sufficiently debug a model to assess its learning progress. These approaches provide context-less label-level heatmaps, which pair deep models with shallow explanations. In contrast, when asked to justify an image classification task, humans typically produce part-based explanations, e.g. *this image depicts a cat because of the pointy ears up there and a tail there, etc.*

Given the need for an open-source easy-to-use GUI-based collection of XAI techniques, in the first part of the chapter a novel GUI is presented, which provides an intuitive toolset of XAI state-of-the-art methods for CNNs from literature. Accordingly, it is demonstrated how the GUI allows users to efficiently visualize the decision-making process of CNNs and explore how changes in the input data affect the model's output. Dwelling on the need for a richer XAI technique, a novel methodology is additionally proposed to explain CNNs predictions in terms of meronym-holonyms. Meronym-holonyms are linguistic concepts that describe the part-whole relationships between objects. The methodology uses these relationships to explain the decision-making process of CNNs. As such, in the second part of the chapter, a detailed explanation of the methodology, its implementation, and validation is provided. It is demonstrated how the methodology can be used to provide meaningful explanations of CNN predictions in various domains.

The rest of this chapter is organized as follows. In Section 3.2, the study that led to the design of a GUI-based collection of XAI techniques is described. In Section 3.3, a semantically-rich novel XAI methodology is described. Finally, in Section 3.4, conclusions are reported on both the studies along with future directions for their respective future directions.

3.2 iNNvestigate-GUI: a Scalable GUI for XAI Techniques

Researchers and, more in general, DL practitioners, require vital insights into the models they have trained. For example, vital insights may be needed to debug models that do not converge or perform poorly on target labels, or to identify samples that the model cannot correctly classify for the specified task. Visual interpretation techniques are particularly effective, particularly for DNNs that target image interpretation [103]. However, there is a lack of tools that can easily integrate such techniques into the development cycle. One of the most comprehensive GUIs available for this purpose is Activis [104], but it is not publicly available.

Arguably similar open-source and publicly available interfaces are essential to support the integration of XAI techniques in the development of DL models. In this section, iNNvestigate-GUI[1] is introduced as an open-source, user-friendly, visual analytics tool for DNNs, specifically designed for CV. It is based on the open-source iNNvestigate library [105], which provides a reference implementation of several visualization algorithms. The goal is to provide a GUI that simplifies model interpretation by providing easy, code-free access to a comprehensive set of visualization methods.

Creating such software has its challenges. First and foremost, visual comparison of multiple DNNs will be computationally expensive. Since the target users are diverse, with varying levels of ML knowledge and needs, visual interpretability is crucial. The tool must be easily integrated into the research and development workflow. Finally, as a critical step towards XAI, when comparing DNNs, the rationale behind the outputs must be considered, in addition to the final performance.

3.2.1 Related Work

In this section, the current state-of-the-art GUI-based tools and libraries for performing visual analysis in DL applications will be discussed. Previous research has classified the available tools based on their license, availability, and target audience, as shown in Table 3.1. **Open-source tools for advanced users.** The majority of

[1] code is available at https://gitlab.com/grains2/innvestigate-gui/

visual analytic tools in the literature fall into this category and include most of the popular applications used in DL. For instance, the TensorFlow Graph Visualizer [106], which is part of a widely adopted TensorFlow framework, is a well-known tool that allows a neural network to be visualized as a directed graph and important information, such as hyperparameters, to be embedded in a single scalable view. The Embedding Projector [107], another visualization tool developed by Google and included in the TensorFlow framework, enables the plotting of tensors in space through different dimensionality reduction techniques. A dynamic real-time visual system to monitor the 2D representation of the filters learned by different layers and an interactive approach to steer the model configuration during the training process were proposed by Chung et al. [108]. The Deep Visualization Toolbox [109] provides a matrix-like grid-view representation of the activations of the neurons in a given layer for a specific input image or video. Summit [110], a tool developed to let practitioners and experts visualize neuron activations, adopts a similar approach, thus enhancing the interpretability of the models. Several visualization tools targeting neural networks focus on the training and optimization phase. For instance, during the training phase, DeepEyes [111] supports the interpretation of the features learned by a CNN model. As shown in Table 3.1, most of these tools allow visualization of gradients and activations and are more suited for effectively monitoring the training process than for XAI.

Other recent visualization tools, such as LSTMvis [112] and GANviz [113], are focused on specific types of networks, such as LSTMs or Generative Adversarial Networks (GANs). On the other hand, in this work, a more general-purpose tool is aimed to be developed.

Proprietary tools for advanced users. ActiVis [104] is an example of a proprietary web application developed by Facebook that provides a comprehensive alternative to the TensorFlow Graph Visualizer. It enables the visualization of a neural network through a node-graph representation and allows behavioral analysis at different levels, from a subset of samples to a single instance and down to the activation of a single neuron. However, it is only available for internal researchers and practitioners, as it is deployed on FBLearner Flow, Facebook's ML platform. In [114], Shixia Liu et al. proposed a tool named CNNVis, which allows the visualization of a clustered representation of the features learned by neurons and the connections between neurons at different layers, with a minimal representation aimed at reducing the visual clutter caused by a high number of links between nodes. How-

ever, this tool has no public implementation, and only an online demo is currently available.

Visualization tools for Neural Network Education. Some visualization tools are specifically designed to aid in educating students on how neural networks function, and to serve educational purposes. For example, TensorFlow Playground is an online application that allows interactive manipulation of a simple neural network model, including its architecture, hyper-parameters, and data points. Through this tool, a better understanding can be gained on the impact of changes made on the decision boundaries learned by the network. However, these types of tools may not be sufficient for visualizing more complex networks and datasets commonly encountered in real-world projects, even for inexperienced researchers. [115, 116].

Table 3.1 Summary of the main DL visualization tools and comparison with the proposed tool.

	TF Graph Visualizer	**Embedding Projector**	**ActiVis**	**DeepVis**	**CNNVis**	**ReVACNN**	**DeepEyes**	**Summit**	**iNNvestigate-GUI**
Visualization									
Node-Link Graph	✓		✓						
Embeddings		✓	✓			✓	✓	✓	
Activations			✓	✓	✓	✓	✓	✓	✓
Gradients			✓	✓	✓	✓			
Hyperparameters	✓					✓		✓	
Attributions								✓	✓
Training History							✓		
Framework									
TensorFlow	✓	✓							✓
Keras	✓					✓			✓
FBlearner Flow			✓						
ConvNetJS						✓			
Caffe				✓			✓		
User Interface									
GUI (web-app)	✓	✓	✓			✓		✓	✓
GUI				✓			✓		
Command Line									
Availability									
Open Source	✓	✓		✓		✓	✓	✓	✓
Proprietary			✓						

Visualization Algorithm Collections The lack of interpretability of DNNs has led to the development of various visualization algorithms, providing insights into the behavior of neural networks [103]. These algorithms allow for the visualization of learned features, neuron activations, and gradients flowing through the layers. Perturbation-based techniques are also used to observe the effect of changes in the input on the model's output. These techniques employ visual paradigms ranging

Table 3.2 Summary of the available libraries of visualization algorithms.

	Keras Explain	**DeepExplain**	**iNNvestigate**
Visualization Technique			
Gradient/Saliency maps	✓	✓	✓
SmoothGrad			✓
DeconvNet			✓
Guided Backpropagation	✓		✓
PatternNet			
Grad-CAM	✓		
Guided Grad-CAM	✓		✓
Input * Gradient		✓	✓
LRP	✓	✓	✓
Integrated Gradients	✓	✓	✓
DeepTaylor			✓
DeepLIFT		✓	✓
Pattern Attribution			
Prediction Difference	✓		
Grey-box Occlusion	✓	✓	
LIME	✓		
Shapley Value Sampling		✓	

from pixel display grids to heatmaps. With the growing interest in XAI, many visualization techniques have been proposed, and a variety of surveys have been published [103, 117].

However, the lack of reference implementations limits the practicality of these techniques [105]. To address this issue, several libraries have been developed, including Keras Explain [118], DeepExplain [119], and iNNvestigate [105]. As shown in Table 3.2, these libraries offer a wide range of visualization techniques, including gradient-based methods such as DeepLIFT [120], model-independent methods such as LIME [94], and perturbation-based techniques. However, integrating these libraries into the model development process can be complex, and GUIs may help address some of the challenges, as discussed in Section 3.2.2.

3.2.2 Design Challenges

To design an XAI visual paradigm that meets the needs of both expert and non-expert users, several critical design challenges need to be addressed. A set of design challenges (C1-C5) have been identified through joint design sessions involving researchers and practitioners with different levels of expertise. These design challenges are focused on open-source solutions and take into account the analysis of existing tools, which is discussed in Section 3.2.1.

C1. Limited computation time for visualization. Generating visualizations for DNNs is computationally intensive, especially when working with images. Given the need to limit computation time to enhance user acceptability, a variety of computing setups should be provided to allow access to the proposed visualization techniques.

C2. User diversity. Designing a tool that is easy to use and accommodates both expert and non-expert users is challenging. The tool should follow a clear and simple workflow structure with different views that are self-explanatory. The visualizations designed should provide clarity and produce useful insights in non-trivial projects.

C3. Analysis on instance and dataset levels. Several XAI techniques explain predictions on a specific instance, but DNNs are trained and tested on large datasets, making it impractical to manually analyze the entire dataset. A suggestion system is required to identify data instances that are worth inspection.

C4. Easy integration in R&D. The lack of a readily available implementation and the need to design specific code for XAI techniques often hinder their practical adoption. A graphical tool should produce expected results significantly faster than coding from scratch and generate the required visualization through a limited number of clicks.

C5. Model complexity and variety. XAI tools should be able to evaluate multiple models that have different architectures, and not just focus on model performance but also take into account the quality of the prediction and the presence of systematic biases. [102]

Addressing these design challenges will be critical in the development of an effective XAI tool that can provide interpretable and actionable insights to a broad range of users.

3.2.3 iNNvestigate-GUI

This section provides a comprehensive overview of the functionalities available in the iNNvestigate-GUI visualization tool. It begins by describing the primary design objectives (G1 - G4) and the rationale behind them. It then outlines the methods

used to achieve these goals, including a simple workflow for visualizing DNNs and an effective method for identifying poorly classified and borderline data instances within a large dataset.

Design Objectives The iNNvestigate-GUI visualization tool aims to fulfill the following design objectives:

G1. Provide a code-free tool that enables efficient interpretation of models by researchers and practitioners. Current visualization libraries attempt to establish a universal reference implementation for models' explainability and interpretability [105]. However, using them requires working with an Application Programming Interface (API), which entails studying the documentation and writing custom code. A visual analytics tool offers a convenient, ready-to-use GUI that can support multiple visualization methods. It needs to enable in-depth examinations at various levels, from the entire model to individual layers and units.

G2. Enable graphical comparison of models quickly. During DL projects, numerous models are trained and evaluated with different architectures, hyperparameters, and configurations. Given that no consensus exists on which visualization techniques are the most effective [105], the visualization tool must provide an intuitive interface to compare various XAI techniques across multiple models.

G3. Allow navigation of extensive datasets and identification of misclassified and borderline instances. DL models are typically trained and validated on extensive datasets, whereas most visualization methods (excluding embeddings) operate on the instance level. Selecting relevant input instances for analysis is difficult. Random sampling can be time-consuming and may overlook errors. The proposed tool should provide an intuitive and efficient way to select samples for further analysis, such as instances that DNNs cannot classify correctly. This method could save the time required to conduct ad hoc analysis of the samples and highlight critical instances that may be overlooked, thus increasing the tool's ability to provide insights into the model's behavior.

G4. Implement a web-based solution to handle computationally demanding tasks. While visualization is less computationally intensive than training, some

visualization techniques require a dedicated training phase, and a large number of samples may need processing. To support users with varying computational resources and memory, disk space, and computational power requirements, the tool was developed as a web application. This approach allows computationally demanding tasks to be offloaded to a back-end, which may be equipped with GPUs, while providing users with a lightweight front-end that can be accessed from any location via a web browser. This framework is already employed by several popular tools (e.g., TensorBoard, Embedding Projector), accommodating both users with high-end workstations and those leveraging cloud computing services (e.g., Amazon Web Services).

In the following paragraphs, the methods used to achieve these design objectives will be described, including a workflow for easy visualization of DNNs and a method for navigating large datasets to select critical samples.

Explaining Custom Models Through Visualization Methods Following the analysis performed in Section 3.2.1, iNNvestigate [105] was chosen as the reference implementation for visualizations. However, the iNNvestigate package was extended to include Grad-CAM and Guided Grad-CAM [102] methods that are particularly suitable for novice and non-expert users due to their intuitive visualizations. Additionally, the ability to visualize the output activations of a specific neuron is provided.

The iNNvestigate-GUI workflow starts by uploading the dataset and pre-trained model(s) for analysis. Options are provided to analyze both user-trained models and the ImageNet-trained models available in the Keras library. To improve usability, all the visualization options and parameters are available as scroll-down lists. A single layer or neuron for activation visualization can also be specified by the user.

Once the configuration phase is complete, the chosen visualizations are generated and displayed to analyze the behavior of the DNNs. The visualization panel is divided into multiple boxes, one for each model loaded in the configuration phase. The visualization method is applied to the output of the selected layer (default being the last convolutional layer) or neuron for each data instance. An interactive panel allows the user to inspect the produced visualizations in a synchronized manner, enabling quick and intuitive comparisons between the behavior of multiple networks.

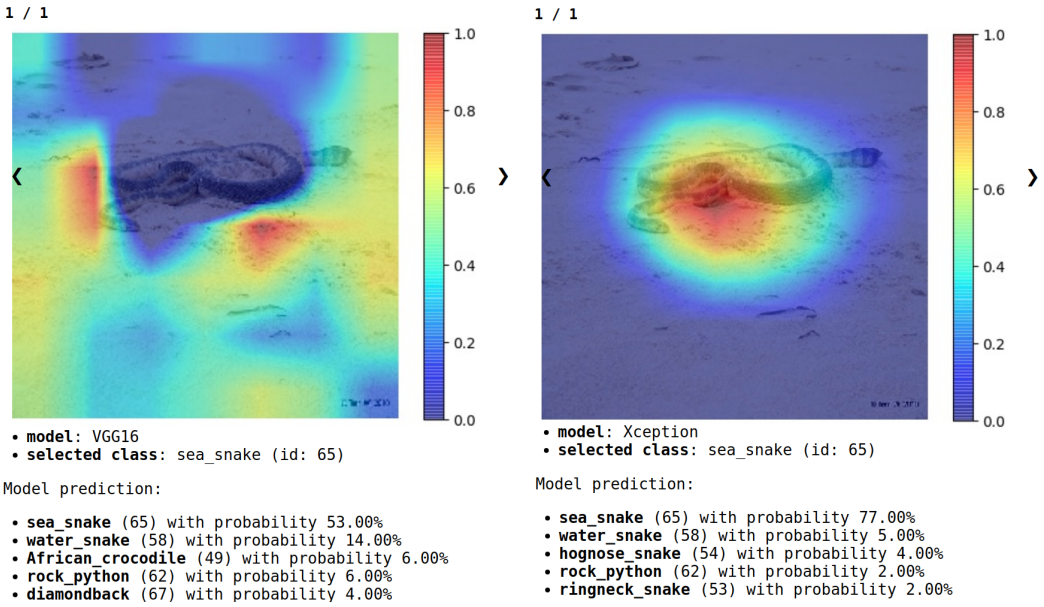


Fig. 3.1 Comparison of different model predictions for one of the images included in T1. The two models exploit different visual features to make the classification. The overlapping heatmaps have been produced using the Grad-CAM technique.

Additionally, the top predictions and their scores for each data sample and DNN are displayed next to the visualization output, as shown in Fig. 3.1.

Suggesting Useful Data Samples for Analysis The iNNvestigate GUI enables users to easily identify relevant data samples for analysis, leveraging the Suggestion panel (see Fig. 3.2). The approach assumes that it is beneficial to examine predictions across a range of data instances with diverse properties. For example, erroneous classifications offer insight into potential sources of error. To this end, the Suggestion panel categorizes available samples and enables the selection of a mix of samples with varying properties for inspection. Two operational modes are identified based on (i) whether one or multiple models are compared and (ii) whether ground truth labels are available.

In the *Multiple Model* scenario, the Suggestion panel displays a scatter plot of input samples according to the agreement and confidence levels of different models, as depicted in Fig. 3.2. The mean prediction score across all models is plotted on the x-axis, while the number of predicted classes is shown on the y-axis. By hovering

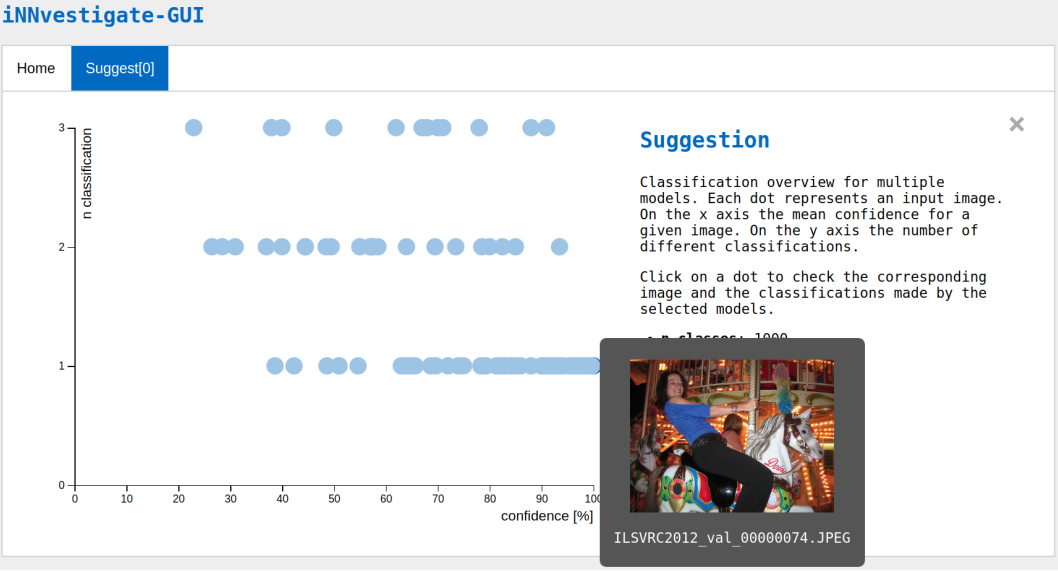


Fig. 3.2 *Suggestion* panel of iNNvestigate-GUI. A scatter plot represents the input dataset processed by multiple neural networks and guides the user towards selecting meaningful samples for further analysis. The average prediction score (x axis) is plotted against the number of different classes (y axis) predicted by the models. This visualization allows to identify data samples with high/low agreement among different models, as well as those predicted with high/low confidence.

the cursor over a data point, it is possible to preview examples visually. The scatter plot enables the distinction of different types of data instances.

Samples located in the bottom-right corner correspond to images classified in the same category by all evaluated models, with high prediction scores. These instances belong to a class that is predicted with high confidence across all models, and are likely to be correctly classified. However, it may still be worthwhile to inspect them, as this can help identify systematic biases in the dataset.

Data samples located in the upper-left corner of the scatter plot (low agreement/low confidence) likely represent borderline cases, or instances that are correctly classified by only a subset of the DNNs. Conversely, data samples located in the upper-right corner are predicted to belong to different categories (low agreement), but with high prediction scores. Such samples may include out-of-distribution data on which DNNs may misbehave, data samples that can easily fool one or more of the models (including adversarial data samples), or data samples that are correctly

classified only by a selection of the models. In this manner, users can concentrate on analyzing the upper-left and upper-right quadrants.

In the *Single Model* scenario, the Suggestion panel displays a simpler histogram plot. Users can opt to visualize the distribution of activations for a predefined layer, for instance, to select images that mostly excite a specific layer. Alternatively, for labeled data, users can plot the distribution of the difference between the prediction and the correct label (typically 1.0 for classification models) to identify data samples that are correctly or incorrectly classified.

3.2.4 Usability Results

To assess the usability of the tool, a group of 9 individuals with prior knowledge in DL were given two tasks to complete at Politecnico di Torino. The tasks were accompanied by a set of questions to guide the users, and after completion of the tasks, a questionnaire following the SUS (System Usability Scale) approach was provided to the users.

The two tasks given to the individuals are as follows:

T1. Compare the visual features used by different models to predict a subset of input images. This task was designed to emulate the comparison of various trained models. A subset of 100 images from the ImageNet ILSVRC2012 dataset was selected for this task.

T2. Evaluate whether multiple models correctly use visual features to classify a subset of inputs belonging to the same class. This task was designed to emulate the search for visual biases. For instance, a network may learn to predict an object based on co-occurring background features. For this task, a subset of 15 images from the *golden_retriever* class was randomly selected from the ImageNet ILSVRC2012 dataset.

The individuals were asked to compare three popular Keras models (VGG16, ResNet50, and Xception) with varying levels of complexity. To reduce the time required to complete the task, only a limited set of visualization algorithms were made available, including Gradient/Saliency, Guided Backpropagation, Grad-CAM, and LRP-z.

Task T1 was completed successfully by all participants, who were able to use the Suggestion panel to identify image agreement/disagreement and prediction confidence. The right-lower quadrant (high agreement/high confidence) and left-top quadrant (low agreement/low confidence) were the areas of focus for the users, with Grad-CAM being the preferred interpretation method for 66.7% and 55.6% of the users respectively (6 out of 9 and 5 out of 9).

For task T2, correctly classified samples by all models and samples with inconsistent behavior had to be identified by the users. The histogram chart was the main tool for 77.8% (7/9) of the participants to identify both, while 22.2% (2/9) also relied on the scatter plot. Guided Backpropagation was the most intuitive algorithm for analyzing visual features of correctly classified images by most models (55.6% of the votes, i.e. 5/9). Grad-CAM was again the most popular choice for analyzing incorrectly classified images (66.7% of users, i.e. 6/9).

In task T2, the visual explanations were rated by the users on a scale from 1 (completely agreeing) to 5 (completely disagreeing). For VGG16 and ResNet50 models, 55.6% (5/9) of the participants were completely satisfied with the visual explanations (meaning they found the proposed attributions appropriate for the label), whereas only 44.4% (4/9) were satisfied with the Xception network.

According to the SUS, the mean usability score was 73.34%, which is above average (any value above 68% is considered above average). Additionally, 77.8% (7/9) of the participants declared that the tasks could not have been completed without iNNvestigate-GUI, while only 22.2% (2/9) stated that the same tasks could be solved by writing ad hoc code.

3.3 HOLMES: Exaplaining CNNs Through HOLonym-MEronym Relationships

There is evidence that CNNs can learn human-interpretable concepts that are useful for detecting classes for which labels are provided, even though they are not explicitly labelled in the training set. For example, scene classification networks can learn to detect objects present in scenes, and individual units may even emerge as objects or texture detectors [121]. However, CNNs may take shortcuts, relying on contextual or unwanted features for their final classification [44]. Other studies

Fig. 3.3 Visual comparison of the predictions made by the three models in T1. The overlapping heatmap has been produced by the Grad-CAM algorithm. This is image was selected by the majority of the participants from the *Suggestion* view as an example of high confidence and high agreement classification. All the models are focusing on similar visual features to classify the frame. Best seen in RGB; consider brighter areas of the frames in case of gray scale visualization.

have found that CNNs are over-reliant on texture rather than shape for their final classification [122]. This chapter investigates how post-hoc explanations can be linked to underlying, human-interpretable concepts implicitly learned by a network with minimal annotation and supervision.

Therefore, the research question is as follows: *can a given label be broke down into a set of related concepts and a component-level explanation be provided for a DL model for image classification?*

The HOLonym-MEronym based Semantic inspection technique (HOLMES) dissects labels into a group of interconnected concepts and provides detailed explanations at the component-level for image classification models. Specifically, HOLMES uses ontologies, web scraping, and transfer learning to automatically generate part-based detectors for a given class. It then generates meronym-level heatmaps and highlights the significance of each part on the classification output by probing the holonym CNN with occluded images. Compared to state-of-the-art saliency methods, HOLMES goes a step further and provides information about both *what* and *where* the holonym CNN is focusing on. It accomplishes this without relying on extensively annotated datasets and without requiring concepts to be linked to single computational units.

3.3.1 Related Work

Feature Extraction and Transfer Learning In recent years, Deep CNNs have become the go-to models for CV applications [123]. These networks consist of multiple convolutional layers that extract features, followed by dense layers used for classification. However, the large number of parameters in these models makes training from scratch computationally intensive and data-hungry [124]. To address this challenge, transfer learning has emerged as a popular technique [125], in which a pre-trained model for a particular task is reused as the starting point for a new model. For CV tasks, this often involves selecting a CNN that was pre-trained on the widely-used Imagenet dataset [85], and then fine-tuning the last dense layers on the new task. The underlying intuition is that the convolutional layers of the pre-trained model have already learned a hierarchical representation of features that is useful for other tasks [125].

Interpretable and Explainable ML The field of XAI aims to make modern ML models more interpretable to humans. There are two main paradigms in XAI: interpretability and post-hoc explainability [126, 127]. Interpretable models are designed and trained to be transparent to some extent, meaning that they provide understandable information about their inner workings without the need for additional algorithms. In contrast, post-hoc explainability is performed after the model has been trained and possibly deployed. External algorithms are applied to the trained model to extract human-understandable information about the decision-making process.

Explainability methods can be categorized based on two binary attributes: local/global and model-agnostic/aware. Local methods provide explanations for individual data points, while global methods aim to explain the behavior of the model as a whole. Model-agnostic methods can explain any black-box model, regardless of its architecture, whereas model-aware methods require access to internal details of specific types of models. For example, model-aware methods may use gradients to explain the decision-making process of a particular model [126, 127].

XAI for CV Two of the most widely recognized XAI techniques are LIME [94] and SHAP [128], which are both local and model-agnostic. However, for the task of CV, most approaches are model-aware and based on saliency. In explaining image classification models, saliency techniques calculate relevance scores at the pixel level

for the model's final output. These scores can be visualized as heat-maps overlaid on classified images for human inspection.

One model-aware, local, post-hoc XAI approach for CV is Grad-CAM [102]. Grad-CAM uses the gradient information flowing into the last convolutional layer of the CNN to assign importance values to each neuron for a specific decision of interest, such as a target concept like *dog*. For a given input image and class, Grad-CAM calculates the gradient of the soft scores for that class with respect to the feature map activations of the last convolutional layer. A global heatmap is then computed as a weighted sum of the feature maps, where each map is weighted by an alpha value obtained by global average-pooling the corresponding gradients. ReLU is applied to emphasize only the features that have a positive influence on the class of interest.

While saliency maps methods such as Grad-CAM ask *where* a network looks when making a decision, the network dissection approach goes further and asks *what* a network is looking for. In [121], the authors discovered that a trained network contains units that correspond to high-level visual concepts that were not explicitly labeled in the training data. For instance, when trained to classify or generate natural scene images, both types of networks learn individual units that match the visual concept of a *tree*, even though the network was never taught the tree concept during training. The authors explored this phenomenon by first identifying which individual components strongly correlate with given concepts (taken from a labeled segmentation dataset), and then turning off each component to measure its impact on the overall classification task. The approach described in this chapter differs from the network dissection literature in several ways: (i) it allows for sparse representations of concepts, capturing concepts without forcing them to be represented by a single computational unit; (ii) it does not need additional, domain-specific ground truth sources and does rely instead on web scraping and general-purpose ontologies; and (iii) it focuses on the part-of relationships of labels in the more general image classification task, rather than the specific scene recognition task.

Ontologies and Image Recognition Ontologies and structured representation of knowledge, in general, are often disregarded in most DL for image processing papers [123]. Nonetheless, notable exceptions exist where efforts have been made to combine sub-symbolic ML models with ontologies.

In [129], the authors take advantage of the fact that ImageNet labels are WordNet nodes to introduce quantitative and ontology-based techniques and metrics to enrich and compare different explanations and XAI algorithms. For example, the concept of semantic distance between actual and predicted labels for an image classification task allows differentiation between a labrador VS husky misclassification as milder with respect to a labrador VS airplane case.

In [130], the authors present a hybrid learning system designed to learn both symbolic and deep representations, along with an explainability metric to assess the alignment level of machine and human expert explanations. The ultimate objective is to combine DL representations with expert domain knowledge during the learning process to provide a sound basis for explainability.

TREPAN [131] is a global approach for XAI that can generate decision trees from a trained neural network. The authors propose to use an ontology to map the feature space, using the ontological depth of features as a heuristic to guide the construction of the decision tree. This approach prefers to split over more general concepts, ensuring that the decision tree is more aligned with the ontology.

3.3.2 Methodology

HOLMES, a novel approach for explaining CNN image classification, leverages holonyms and meronyms to provide a model-dependent semantic inspection. HOLMES retrieves the parts of the image based on its predicted class and explains the classification outcome in terms of these parts. The proposed method is specifically designed for CNNs and takes the predicted image class as input.

Problem Formulation Consider the image classifier $\mathscr{H}$ as a function $\mathscr{H} : \mathbf{x} \in \mathbb{R}^{h \times w \times ch} \mapsto c \in \mathscr{C}$, where $\mathbf{x}$ represents an image with dimensions $h \times w \times ch$, and $\mathscr{C}$ is the set of image classes. Let $f_F^{\mathscr{H}}$ denote a feature extractor that maps $\mathbf{x}$ to a feature vector $\mathbf{f}$, and $f_C^{\mathscr{H}}$ be a feedforward classifier that maps $\mathbf{f}$ to c. Given a holonym-meronym relationship mapping HolMe : $c \in \mathscr{C} \mapsto p_1, ..., p_n$, where $P_c = p_1, ..., p_n$ denotes the meronyms of class c, a novel method to explain the image classification in terms of its parts is proposed. HOLMES outputs a function $\xi : \mathbf{x} \mapsto (\mathbf{x}^{(p_i)}, s_i),\ p_i \in P_c$, where $\mathbf{x}^{(p_i)}$ is a saliency map that highlights the meronym p_i in the original image $\mathbf{x}$, and s_i is an explanation score associated to $\mathbf{x}^{(p_i)}$.

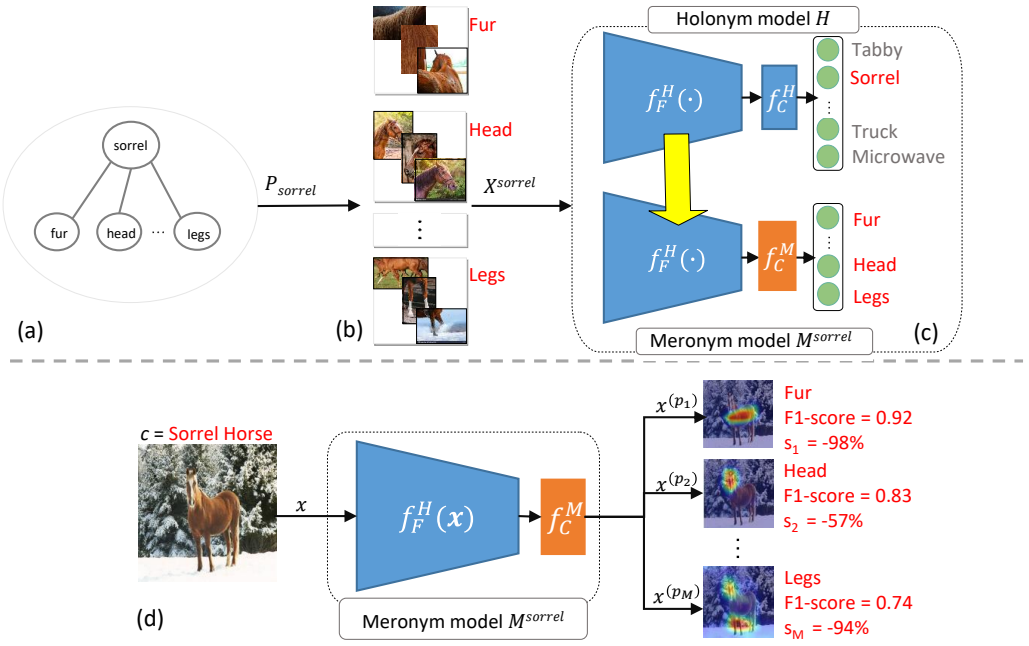


Fig. 3.4 HOLMES pipeline. Given an input image of class c, its parts (meronyms) are extracted from a Knowledge Base (KB) (a). Images depicting each part are either extracted from a densely annotated dataset or collected through Web scraping (b), and then used to train a meronym model by exploiting, through transfer learning, the implicit knowledge embedded in the original holonym model (c). The meronym model then produces part-based explanations, highlighting the most relevant parts for the class prediction (d).

To achieve this, HOLMES trains a new meronym classifier $\mathscr{M}^c : \mathbf{x} \mapsto p_i \in P_c$, which is a combination of the same feature extractor $f_F^{\mathscr{H}}$ and a new feedforward classifier $f_{P_c}^{\mathscr{M}}$ that maps $\mathbf{f}$ to $p_i \in P_c$. The meronym classifier identifies the presence of the parts p_i in the input image $\mathbf{x}$, and generates saliency maps for each part. A mask is created on $\mathbf{x}$ for each saliency map $\mathbf{x}^{(p_i)}$, which is then classified by $\mathscr{H}$. The drop in the classifier confidence for class c is used to determine the importance of the selected parts.

The proposed HOLMES framework involves four main steps in the pipeline (Fig 3.4):

1. **Meronyms Extraction**: given an image $\mathbf{x}$ and its predicted class c, the object parts P_c are extracted.

2. **Meronyms Image Data Collection**: a dataset for each part in P_c is created.

3. **Meronyms model Training**: auxiliary meronyms models $\mathscr{M}^c$ are trained to recognize the parts P_c using knowledge from the original CNN $\mathscr{H}$.

4. **Explanations**: part-based explanations are produced, highlighting the most relevant parts for the class prediction.

Holonym-Meronym Relationship Extraction The first step in the process of extracting meronyms is to create a mapping of holonym-meronym relationships. This mapping, denoted as HolMe : $c \in \mathscr{C} \mapsto P_c$, involves retrieving visible parts P_c associated with a holonym concept c.

To extract meronyms, an external KB that includes part-of relationships is used. The KB contains class concepts, such as animals, and their respective lists of parts, like legs and head, along with information on their visibility. To obtain the parts of a holonym concept, a selected KB is queried for the desired holonym concept and its associated visible meronyms are retrieved.

If a concept is not present in the chosen reference KB, it is mapped to the corresponding WordNet[132] ontology concept. The hypernym/hyponym relationship is then utilized by climbing the WordNet semantic hierarchy up to the first hypernym, such as a broader class like birds for seagulls, that occurs in the reference KB. The associated parts of the hypernym, which are more generic, are assigned to the initial holonym concept.

After the meronyms are extracted, they are divided into two categories: hyper-meronyms and hypo-meronyms. Given a list of meronyms $P = p_1, p_2, \ldots, p_n$, hypo-meronyms are parts that are entirely within the visual space of any other part in P. The other parts, whose visual space is not wholly contained in any other part in P, are hyper-meronyms.

For example, for the holonym concept *cat*, the hyper-meronyms are *head, legs, feet,* and *tail*, since none of them is visually contained in any other part. Hypo-meronyms, such as *mouth* and *whiskers*, can only be contained in hyper-meronyms.

Only the hyper-meronyms are retained in the final list of meronyms, while the hypo-meronyms are discarded. Therefore, the final list of parts is defined as $P_c = p_1, p_2, \ldots, p_n$.

Meronyms Image Dataset Generation After obtaining the part set P_c for the target class c, the following step is to create an image dataset $X^c = (\mathbf{x}_0, y_0), ..., (\mathbf{x}_n, y_n)$, where $\mathbf{x}_0, ..., \mathbf{x}_n$ are images corresponding to parts $y_0, ..., y_n \in P_c$. To generate such dataset, HOLMES can either use a pre-existing labelled dataset or build one incrementally by scraping web images for each meronym. In the latter case, HOLMES searches different web search engines for each part by appending the corresponding holonym (e.g. *sorrel fur*, *sorrel head*, etc.) and downloads the associated images.

However, search engine results are not always reliable (as discussed in Section 3.3.3). Some obtained images could still be unrelated to the desired part concept. Moreover, duplicates may exist in the scraped parts' datasets. To address these issues, HOLMES integrates two additional sub-steps:

- **deduplication**: using the pHash[133] hash-based deduplication method, duplicates are identified and removed from each meronym dataset.
- **outlier removal**: to identify outliers in the meronym datasets, meronyms images are converted to feature vectors (e.g. using the output of the feature extractor or the activations of one of the feedforward layers of the classifier). The feature vectors are then used to detect outliers. The identified outliers are removed from the meronym dataset.

Meronyms model Training The core of the HOLMES method lies in the training phase. In this step, the visual understanding of concept parts is acquired to use them as explanations later on. An auxiliary CNN model $\mathscr{M}^c$ is trained and evaluated on the collected dataset of meronyms X^c. The training and evaluation are conducted on disjoint sets.

To clarify, the objective of HOLMES is to explain the output of the target holonym CNN $\mathscr{H}(\mathbf{x}) = \hat{y}$, where $\mathbf{x}$ is an image of the holonym class c and $\hat{y}$ is its predicted class. It is worth noting that the CNN can be represented as a combination of two functions, i.e., $\mathscr{H}(\mathbf{x}) = f_C^{\mathscr{H}}(f_F^{\mathscr{H}}(\mathbf{x}))$, where $f_F^{\mathscr{H}}(\cdot)$ is a feature extractor, and $f_C^{\mathscr{H}}(\cdot)$ is a feedforward classifier.

Research has already shown that the last convolutional layers of a CNN tend to represent objects and their parts [134, 121]. HOLMES leverages this fact to train the model to identify the parts. Specifically, the meronym model is defined as $\mathscr{M}^c(\mathbf{x}) = f_{P_c}^{\mathscr{M}}(f_F^{\mathscr{H}}(\mathbf{x}))$, where the feature extraction $f_F^{\mathscr{H}}(\cdot)$ is shared between

the holonym $\mathscr{H}$ and meronym $\mathscr{M}^c$ models. However, a new feedforward classifier $f_{P_c}^{\mathscr{M}}(\cdot)$ is trained for each class c and part list P_c.

The idea is to teach the parts concepts using the same features learned by the original reference CNN model $\mathscr{H}$. Thus, HOLMES relies on transfer learning [125] for learning the objects' parts. Assuming that the characteristic parts of an object and their associated features are helpful in classifying the entire object, the same units that activate in the presence of a part will activate in the presence of the object. For instance, if a unit activates in the presence of a wheel, it is likely to activate in the presence of a wheeled vehicle such as a car. Therefore, training the meronym model $\mathscr{M}$ while keeping the feature extractor $f_F^{\mathscr{H}}$ intact will help determine if the knowledge of the parts was already present and embedded in the original model $\mathscr{H}$. The feature maps obtained in the presence of individual parts will be useful in creating a visual explanation for the (holonym) predictions of the original model.

To evaluate the performance of the meronym model, a held-out test set is used to calculate the per-part calibrated F1-score [135]. This score determines how well each part was learned and distinguished from the others. The F1-score is calibrated to be made invariant to the class prior, so that the performance of models trained on different numbers of meronyms can be compared.

Explanations After completing the previous stage, a CNN model for meronyms, denoted by $\mathscr{M}^c$, is trained. Given an input holonym image represented as $\mathbf{x}$, this model generates a set of prediction scores $Y_p = y_{p_1}, \ldots, y_{p_n}$, where n is the number of parts the model was trained on, and $y_{p_1}, \ldots, y_{p_n}$ are the scores assigned to each of the different parts. Consequently, when an input image of a holonym, such as an image of a car, is passed to the network, a score is produced for each of its parts, such as the wheels and the bumper. These scores reflect the degree to which each part is present in the input holonym image. By leveraging the ability of the network to recognize the part concepts in an input holonym image, it is possible to identify the specific region of the input image where each part is located.

In particular, the visualization of each part in the holonym image is generated by using the state-of-the-art saliency method Grad-CAM [102]. After generating a saliency map $\mathbf{x}^{(p_i)}$ for each part recognized by the network, each saliency map $x^{(p_i)}$ is converted to a binary segmentation mask $m^{(p_i)} \equiv (x^{(p_i)} \geq T^{(p_i)})$, where $T^{(p_i)}$ is set to the q^{th} percentile of the corresponding saliency map pixel distribution. Subsequently,

the same input holonym image is fed into the original CNN model and an assessment is made on the importance of each part for the original network prediction by ablating one part at a time based on the meronyms masks $m^{(p_i)}$. By observing the percentage drop in the original predicted holonym class label, it is possible to determine the degree to which the removed meronym was important for predicting that class label. A more consistent drop implies a greater visual presence of the part in the image and, therefore, greater significance for the original model.

At this point, the input image $\mathbf{x}$ is linked to a set of saliency maps $\mathbf{x}^{(p_i)}$ for each part $p_i \in P_c$, and each saliency map is associated with a score drop $s_i \in S = s_1, s_2, \ldots, s_n$.

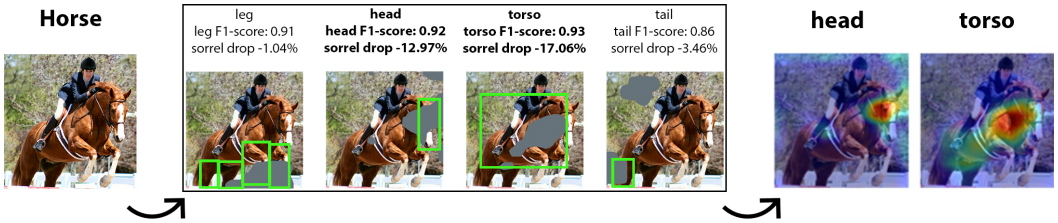
Additionally, the per-part calibrated F1-score, which measures the accuracy of the part identification, is used to assess the reliability of the part identification. The most promising hypothesis is that a meronyms model that had difficulty learning and distinguishing a part would have obtained a low F1-score for that part. Consequently, the parts whose holonym score drop s_i exceeds a threshold T_s and whose meronyms model has an F1-score above a threshold T_{F1} are provided as part-based explanations for the original model prediction, as this would imply that those parts are correctly detected by the meronyms model and considered relevant for the classification of the holonym.

3.3.3 Experimental Settings

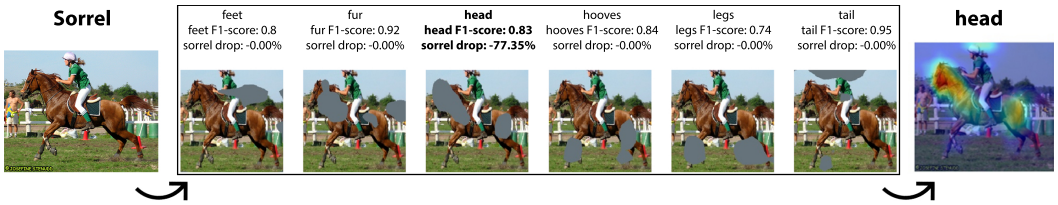
HOLMES provides part-based explanations for any classification model that consists of a feature extractor and a feed forward classifier. To demonstrate the effectiveness of HOLMES, it was applied to explain the outputs of a pre-trained VGG16 [124] image classifier on the ImageNet [85] dataset. In this section, two experimental settings are presented where HOLMES was applied to explain the predictions of the VGG16 model.

For the first experiment, the chose dataset was PASCAL-Part, which provides annotated bounding boxes for objects and their parts, to generate explanations and evaluate their validity. In the second experiment, a part-based mapping was created for several classes in the ImageNet dataset, a dataset for training meronyms models was constructed through web scraping, and part-based explanations were generated using HOLMES. The quality of the explanations was evaluated using

insertion, deletion, and preservation curves. Examples of the part-based explanations generated using HOLMES for both experiments are shown in Fig. 3.5.



(a) PASCAL-Part



(b) ImageNet

Fig. 3.5 HOLMES Explanation example for the *horse* class – PASCAL-Part (a) and *sorrel* class – ImageNet (b). For each part, the corresponding ablation mask (grey), the per-part calibrated F1-score and the holonym score drop are shown. For PASCAL-Part, the ablation masks are compared against the ground truth bounding boxes (green). The final heatmap(s) show the part-based explanations. Two and one part are included in the explanations for examples (a) and (b), respectively, as they exceed both the holonym score drop threshold T_s (0.1) and the calibrated F1-score threshold T_{F1} (0.7).

PASCAL-Part dataset The PASCAL-Part dataset [136] includes bounding box annotations for objects and their parts, which can be used to create holonym-meronym relationship mappings (HolMe). For the experiments, a set of 50 images per holonym and their corresponding cropped meronym images were held out to evaluate HOLMES's explainability and part localization performance. The remaining images were used to train the meronym models $\mathscr{M}^c$.

Meronyms Extraction: The Pascal VOC 2010 dataset includes 20 classes, which were divided into four macro-classes: (1) people, (2) animals (birds, cats, cows, dogs, horses, sheep), (3) vehicles (aeroplanes, bicycles, boats, buses, cars, motorbikes, trains), and (4) indoor objects (bottles, chairs, dining tables, potted plants, sofas, TV monitors). Two classes (people and potted plants) were excluded because they have no corresponding class in the ImageNet 1000 classes. Additionally, five classes

without part-based annotations were excluded (boats, chairs, dining tables, sofas, and TV monitors).

The meronyms P_c for each of the remaining 13 classes were extracted from the PASCAL-Part parts list. Hyper-meronym selection was performed for the six animal classes, as in [137]. For the remaining classes, hyper-meronyms were selected by majority voting. The final HolMe mapping is as follows:

- $P_{bird} = P_{cat} = P_{dog} = P_{horse}$ =head, torso, leg, tail
- P_{cow} = head, torso, leg, horn
- P_{sheep} = head, torso, leg
- $P_{aeroplane}$ = stern, wheel, artifact wing, body, engine
- $P_{bicycle}$ =saddle, wheel, handlebar
- $P_{motorbike}$ =saddle, wheel, handlebar, headlight
- $P_{car} = P_{bus}$ = window, wheel, headlight, mirror, door, bodywork, license plate
- P_{train} = coach, locomotive, headlight
- P_{bottle} = body, cap

Meronyms image extraction settings: Given the HolMe relationship mapping, the training of meronym models requires the extraction of image crops for each part associated with a holonym. The bounding box coordinates for each part in P_c for a given holonym image were obtained and used to crop the respective part images.

To ensure compatibility with the VGG16 model input format, each cropped part image was extended in the x or y direction to produce a square crop that preserved the aspect ratio and shape of the original part. The avoidance of overlapping bounding boxes in the same image was ensured, and padding was applied to the final square crop if the aspect ratio was not 1:1.

To address the high class imbalance resulting from a variable number of crops extracted for each meronym, data augmentation techniques were applied to balance the number of samples in each class. Specifically, each cropped image was randomly rotated and sheared, and one of the following transformations was applied: gaussian blur, emboss, or gaussian noise.

The extracted meronym samples for each holonym class were split into three folds for training, validation, and testing, respectively. The ratio of samples in each fold was 0.81/0.09/0.1.

Training and Explanations settings: To build a meronym model for each holonym, a feed forward classifier $f_{P_c}^{\mathscr{M}}(\cdot)$ with the same structure as the original VGG16 classifier was trained, using common data augmentation techniques such as horizontal flipping, rotation, cropping, color jittering, and random grayscale. Each meronym model was trained for 100 epochs, with a batch size of 64 and learning rate of 0.001, determined experimentally. To avoid overfitting, an early stopping policy with a patience of 5 was employed.

For generating the Grad-CAM meronym heatmaps, the activations of the last convolutional layer of VGG16 were used. These heatmaps were binarized using a threshold $T^{(p_i)}$, which was found to be the 83^{rd} percentile by performing a grid search on the $[75,90]$ percentile values. This threshold was chosen to strike the best trade-off between different causal metrics performance on the whole PASCAL-Part training set, which comprises all training holonym image samples. To retain natural image statistics, the masked pixels were ablated by replacing them with the gray RGB value (as the ImageNet mean pixel is gray). Finally, the T_s and T_{F1} thresholds were set to 10 and 0.7, respectively.

Evaluation settings: Each selected class underwent testing using the entire HOLMES pipeline on validation image samples. The localization performance of meronyms was evaluated by computing the per-pixel AUC score of the HOLMES meronym heatmap versus the meronym ground truth. True positive pixels correspond to those belonging to the actual part within the bounding box, while false positives refer to the remaining pixels. To establish the baseline, the per-pixel AUC score of the Grad-CAM holonym heatmap was compared to HOLMES. The faithfulness of HOLMES explanations was evaluated using common causal metrics based on the deletion/insertion/preservation curves [138, 139].

HOLMES produces a set of saliency maps $\mathbf{X}^{(P_c)} = \mathbf{x}^{(p_i)}$, $p_i \in P_c$ associated with a specific part p_i. However, a global quality assessment of these part-based explanations requires merging all saliency maps into a unique heatmap for all parts. The HOLMES global heatmap is generated through a weighted linear combination of the part-based saliency maps using normalized score drops $Z = z_1, ..., z_n$, which are calculated by dividing each score drop by the L1-Norm of $S = s_1, s_2, ..., s_n$ associated

with the ablation of each part. The global heatmap is then obtained by summing each weighted heatmap element-wise: $G = \sum_{i \in n} \mathbf{x}^{(p_i)} z_i$. This weighting scheme emphasizes the parts whose ablation causes a significant holonym class score drop.

Using the global heatmap G, it is possible to assess the overall quality of the part-based explanations by computing causal metrics such as the areas under the insertion [138], deletion [138], and preservation curves [139]. These metrics were calculated for all held-out PASCAL-Part validation images.

ImageNet In certain scenarios, HOLMES can be applied even if the datasets with part-level annotations are not available. In these cases, ontologies and image scraping are used to construct the required meronym datasets. To this end, the relationship between Imagenet[85] labels and WordNet[132] nodes is utilized to obtain a list of parts of the object-label, based on the holonym-meronym (whole-part) relationship.

Meronyms extraction settings: Out of the 1000 class concepts in ImageNet, 81 were selected as holonym classes for this process. The chosen holonym classes belong to two major categories:

1. Medium- or large-sized animals
2. Medium- or large-sized man-made objects

The size constraint was implemented to obtain suitable training sets. Retrieving distinct images of parts by querying web search engines becomes increasingly difficult for smaller holonyms (e.g., bugs in the animals category), as the engines tend to return images of the whole holonym concept instead of specific parts (e.g., the whole butterfly instead of a butterfly head). This results in highly similar meronyms datasets with a significant visual overlap, which can considerably impede the performance of the associated meronym model.

Therefore, for each of the 81 classes, their respective meronyms were extracted from the VISA[140] dataset. Hyper-meronyms were manually filtered to obtain a set of meronyms: meronyms extracted from the ontology were classified manually into hyper-meronyms and their respective hypo-meronyms were filtered automatically. Consequently, for every instance of a hyper-meronym, the associated hypo-meronyms were eliminated.

Meronyms image scraping settings: Two search engines, Google and Bing, were utilized for image scraping to generate the necessary meronym datasets. To avoid downloading irrelevant images, the number of downloads per part across all search engines was restricted, with a general guideline of limiting downloads to the first 100 items [141, 142]. Download limits of 40 and 60 images were set for Google and Bing, respectively, as Bing has been found to provide more relevant images for part concepts. In order to expand the dataset size for each part, the "Visually Similar Images" feature of Google was utilized to locate and download similar images for each downloaded image. To remove duplicates and near-duplicates [143], a hash-based deduplication method called pHash [133] was used. Additionally, outlier removal was performed using the PCA outlier detection algorithm[144]. Meronym images were transformed into a feature vector utilizing the activations of the penultimate FC layer of VGG16[124]. The outlier contamination rate hyper-parameter was set to 0.15. The meronym datasets were split into training, validation, and test sets with proportions of 0.81, 0.09, and 0.1, respectively.

Training and Explanations settings: The training and explanation procedures are performed using the identical configurations outlined for the PASCAL-Part dataset.

Evaluation settings: The overall heatmap is assessed using the same insertion, deletion, and preservation curves as described for the PASCAL-Part dataset.

3.3.4 Results

This study evaluates the HOLMES pipeline in all its steps to determine its effectiveness in identifying and locating meronyms, attributing classification scores to individual meronyms, and generating explanations

RQ1: How well can HOLMES classify and locate meronyms?

To answer this research question, the PASCAL-Part dataset was used, which includes 13 classes with an average of approximately four visible parts per class. A total of around 750 samples per meronym were collected, and approximately 1400 samples were obtained after data augmentation, resulting in a total of 74,772 training samples. For the ImageNet dataset, 81 classes were selected, with an average of approximately seven visible parts per image. A web scraping process was carried out to obtain a total of 559 meronyms, resulting in an average of around 450 images per

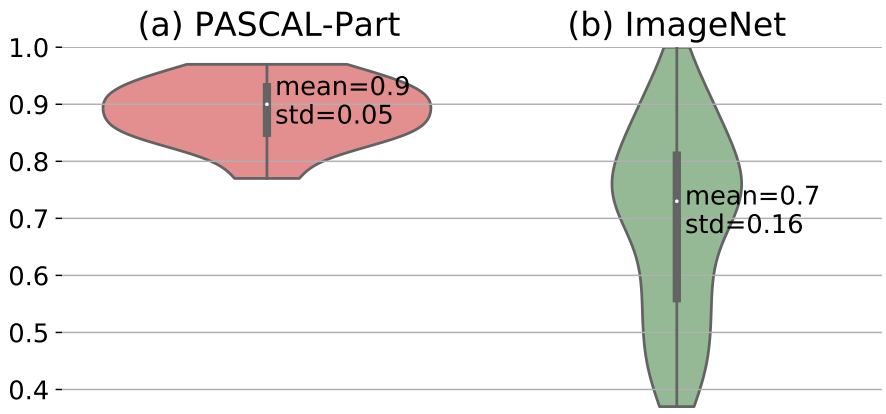


Fig. 3.6 Distribution (violin plot) of the average per-part calibrated F1-score.

part. After eliminating duplicates and outliers, the final average number of images per part was approximately 320.

The distribution of the calibrated F1-scores of the $\mathscr{M}^c$ models, which were trained upon each training set X_c for each of the selected classes, is reported in Fig. 3.6 for both PASCAL-Part and ImageNet dataset. The average F1-score was good in both cases, but higher on PASCAL-Part (0.9 ± 0.05 vs. 0.7 ± 0.16). This difference can be attributed, at least partially, to the higher precision of the PASCAL-Part reference standard, which includes bounding boxes.

Table 3.3 Per-pixel AUC results.

Method	horse	cat	bird	cow	dog	sheep	aeroplane	bicycle	bottle	bus	car	motorbike	train	Avg
HOLMES	0.77	0.74	0.8	0.77	0.74	0.75	0.74	0.76	0.67	0.75	0.68	0.74	0.71	**0.74**
Grad-CAM	0.68	0.68	0.71	0.66	0.71	0.62	0.76	0.63	0.6	0.65	0.62	0.66	0.62	0.66

On the contrary, as shown in Fig. 3.7, the performance deteriorates in direct proportion to the number of components in each category. This effect is particularly evident for the ImageNet dataset, which has a richer ontology with more classes and more components per class. In fact, as the number of components increases, the probability that images belonging to different components are visually similar increases, which can negatively impact the performance of the trained $\mathscr{M}^c$ model. Additionally, different categories tend to have a varying number of meronyms associated with them. For example, tools usually have between one and four components, animals between three and eight, and vehicles more than eight. Thus, the category

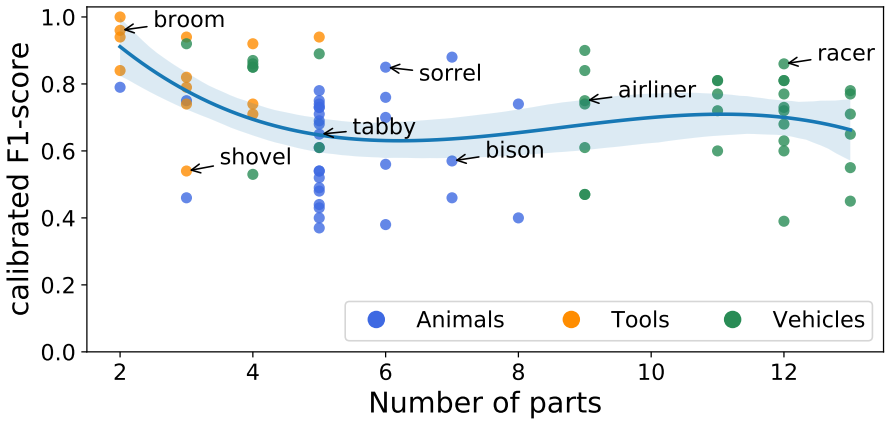


Fig. 3.7 Average per-part calibrated score as a function of the number of parts per holonym class (colored dots represent a holonym, blue line is the mean average per-part F1-score).

may also influence the quality of the scraping or the differentiation of the meronyms themselves.

To evaluate the localization performance of HOLMES meronyms, the per-pixel AUC score of each HOLMES meronym heatmap against their PASCAL-Part ground truth bounding boxes was computed. As a baseline, two cases were considered: the first case randomly assigns the meronym to any region of the image (AUC=0.5), while the second case focuses on the actual holonym extracted by the Grad-CAM algorithm and assumes that the meronym is inside the region of the Grad-CAM holonym heatmap. While the second scenario presents a more challenging baseline to surpass, the experiments in Table 3.3 consistently showcase that HOLMES meronyms offer improved and more precise localization of parts in comparison to Grad-CAM heatmaps, which solely localize the entire object. Notably, in an extreme case, it can be observed that even with the minimum number of parts present (2 in the instance of the *bottle* class), HOLMES outperforms Grad-CAM.

RQ2: To what extent the classification score can be attributed to individual meronyms?

After successfully classifying meronyms, the next step is to investigate their influence on the holonym classifier $\mathscr{H}$, as depicted in Fig. 3.5.

In Fig. 3.8, the distribution of the *per-meronym score decrease* is presented, which measures the decline in score observed for the holonym when the corresponding

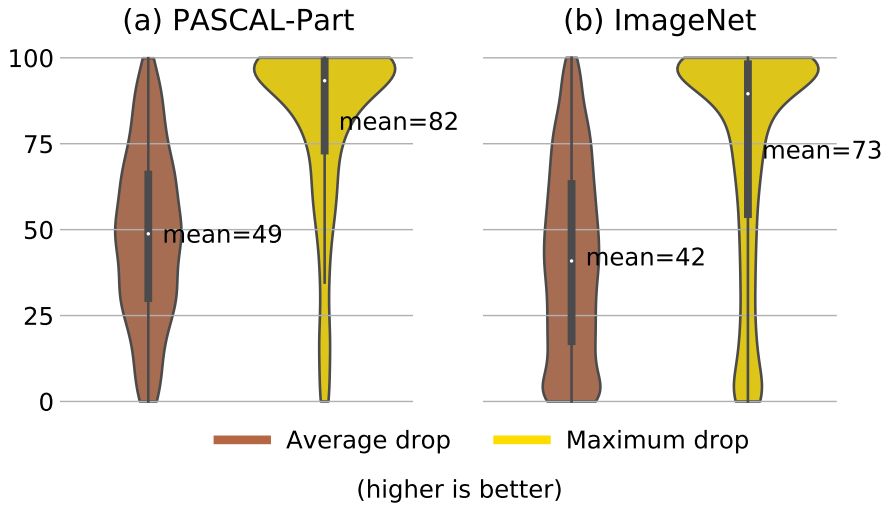


Fig. 3.8 Distribution (violin plots) of the average score drop and maximum score drop (in percentages) per image on the PASCAL-Part (a) and ImageNet (b) validation sets. The score drop is calculated for each image and meronym by ablating the corresponding mask; then, the average and maximum score drop are computed over all meronyms appearing in an image.

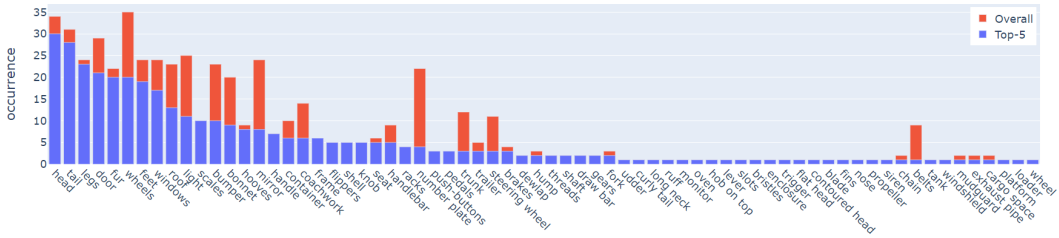


Fig. 3.9 Top-5 meronyms distribution. For each class, the top-5 meronyms are computed (i.e., the meronyms inducing on average the highest score drops). For each meronym, the total number of associated holonyms (blue) is compared with the frequency they appear in the Top-5 meronym list (red). Meronyms are listed in decreasing order of frequency.

meronym is removed. On average, a single meronym ablation roughly cuts the holonym model confidence in half, with an average score drop of 49% for PASCAL-Part and 42% for ImageNet. For PASCAL-Part, considering only the most significant part (i.e., the one with the highest score drop for each test image) yields an increase in the score drop to 82%. In this dataset, individual meronyms strongly influence the classifier output, with the majority of classifications being attributable to a single meronym. Conversely, for ImageNet, the mean maximum drop is lower (73%). The

mean ($\pm$ standard deviation) number of meronyms included in each explanation was 2.28 ± 2.46.

However, at the holonym class level, the mean average drop and mean maximum drop for PASCAL-Part were 50% and 80%, respectively, and 46% and 73% for ImageNet. The score distribution in Fig. 3.8 characterizes HOLMES behavior at the instance (image) level, whereas the mean values offer a glimpse into the general characteristics of the holonym classes. On PASCAL-Part, the mean maximum score drop presents a bimodal distribution, with animal classes experiencing a higher mean maximum drop than vehicles and man-made objects (78.5% vs. 62.1%). As a result, explanations for images belonging to animal classes are often highly focused, with a single meronym nearly describing the entire holonym concept. Explanations for other classes require the combination of two or more meronyms. On ImageNet, little difference was found between animals (70%), tools (73%), and vehicles (74%). However, there is a wider range of mean maximum drops between classes, with a low of 23% (zebra) and a high of 95% (Persian cat).

Some holonym classes may share a common set of meronyms. This is particularly evident for the richer ImageNet ontology. For example, most animal classes have a head and a tail, although each class will have its own training dataset and classifier $\mathbf{M}^c$. Based on this observation, it was sought to determine if certain parts consistently and significantly affected the holonym score drop (Fig. 3.9). In the case of animal classes, the meronyms head, tail, and legs frequently cause a consistent score drop when removed from the image. Similarly, for vehicles, the door, wheels, and window meronyms have the most significant impact on the holonym class prediction.

RQ3: How good are the explanations generated by HOLMES?

In the final stage of the evaluation, the overall quality of the generated explanations is assessed by comparing HOLMES with Grad-CAM, a method that does not provide part-level explanations. In order to make a fair comparison, part-level heatmaps are combined into a global heatmap using a linear combination approach, as illustrated in Fig. 3.10. The AUCs for the insertion, deletion, and preservation metrics are used to evaluate the performance, and the results are summarized in Table 3.4.

On PASCAL-Part, the average HOLMES insertion AUC is 0.96 times the Grad-CAM insertion AUC, while the average deletion AUC is 0.95 and the average preservation AUC is 1.03 times the corresponding Grad-CAM score. Similarly, on

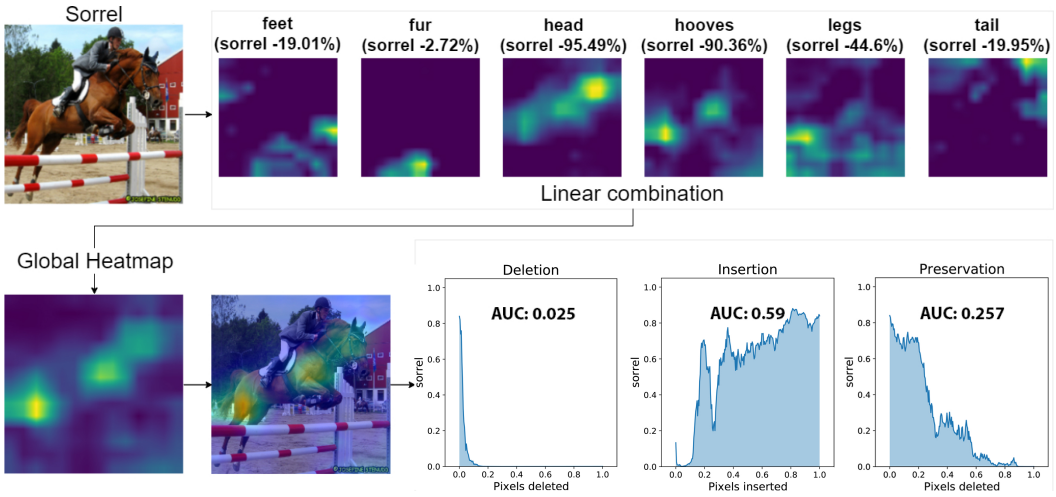


Fig. 3.10 HOLMES Global Explanation. Starting from an input image, the per-part heatmaps and the respective holonym score drops are obtained. Then, by a linear combination of the heatmaps, the global heatmap is obtained, and its quality is measured by means of the insertion/deletion/preservation metrics.

Table 3.4 Deletion/Insertion/Preservation AUCs for HOLMES and Grad-CAM.

Dataset	Method	Deletion ↓	Insertion ↑	Preservation ↑
PASCAL Part	HOLMES	0.050±0.053	0.487±0.269	0.392±0.255
	Grad-CAM	0.052±0.060	0.505±0.277	0.381±0.264
ImageNet	HOLMES	0.112±0.113	0.660±0.252	0.538±0.257
	Grad-CAM	0.111±0.107	0.684±0.242	0.539±0.261

ImageNet, the average insertion, deletion, and preservation AUCs are 0.96, 1.01, and 0.99 times the corresponding Grad-CAM scores.

Furthermore, a random baseline is used as a comparison to account for object scale, where images are divided into super-pixels and erased in random order. As shown in Fig. 3.11, HOLMES significantly outperforms the random baseline, with an average insertion AUC 0.58 lower and an average deletion AUC 1.77 higher than the baseline. Additional examples of part-based explanations extracted by HOLMES are reported in Fig. 3.12, 3.13, 3.14, 3.15 and 3.16.

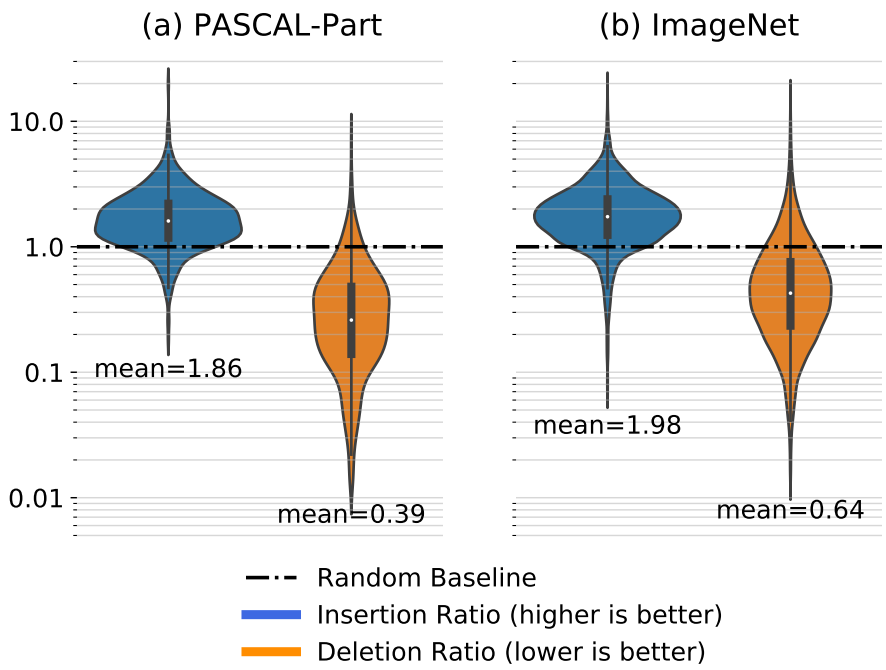


Fig. 3.11 Insertion/Deletion Ratio distribution (violin plot) for the PASCAL-Part (a) and ImageNet (b) datasets. The average insertion ratio (left) and the average deletion ratio (right) are calculated with respect to the random baseline (dotted black line).

3.3.5 Discussion

In contrast to existing approaches [121], HOLMES does not rely on an extensively annotated dataset with pixel-level annotations. Instead, it can be trained using weak annotations, such as bounding boxes available in the PASCAL-Part dataset [136], or web scraping, which drastically reduces the annotation effort and eliminates the closed-world assumption intrinsic to traditional labeled datasets. Previous works have established the efficacy of web scraping for object recognition [142, 145], which HOLMES builds upon and expands through deduplication and outlier removal to reduce noise and enhance variety in the training dataset.

However, obtaining high-quality images for meronyms, as opposed to holonyms, presents additional challenges that may impact the quality of the dataset and, therefore, the meronym models. While the relevance of the retrieved images is generally high, it may decrease depending on the popularity of the meronym as a search term. Moreover, visual overlap poses a specific challenge since it is difficult to find images that isolate a single meronym precisely.

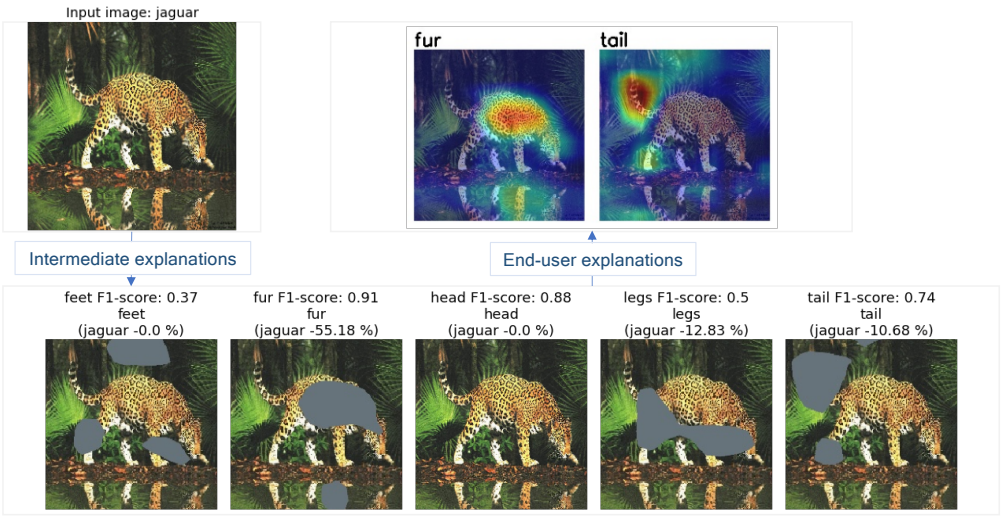


Fig. 3.12 HOLMES explanations example. The image portions associated to the meronyms (feet, fur, head, legs and tail) of the predicted holonym class (jaguar) are identified in the input image and ablated one at time. The feet and the legs calibrated F1-scores are below the calibrated F1-score threshold (0.7) and so these meronyms are immediately discarded: in fact, for such meronyms, the highlighted regions do not precisely encircle the respective part, while also including extraneous regions like background (vegetation) and other parts (the head and the tail are partially masked in the legs case). Among the remaining meronyms (fur, head and tail), only the fur and the tail score drops are above the holonym score drop threshold (10), hence only for these meronyms (in red) the associated saliency maps are interpolated with the input image and provided as end-user explanations. Notice as for this specific 'jaguar' input image the head is not recognized: its associated image portion is not highlighted, hence a zero holonym score drop is recorded and consequently the part in not included in the end-user explanations.

Despite the acknowledged limitations regarding repeatability, the results of the data collection approach for various holonyms demonstrate a remarkably positive outcome. This comprehensive analysis has successfully encompassed a substantial number of images spanning a diverse array of classes, showcasing promising results across the board. The incorporation of such a wide spectrum of visual content underscores the methodology's versatility and potential utility.

Furthermore, in addressing the challenge posed by varying image availability and quality through web scraping, it is essential to recognize the dynamic nature of online resources. This consideration underscores the importance of acknowledging the evolving nature of data collection, which inherently introduces variations in

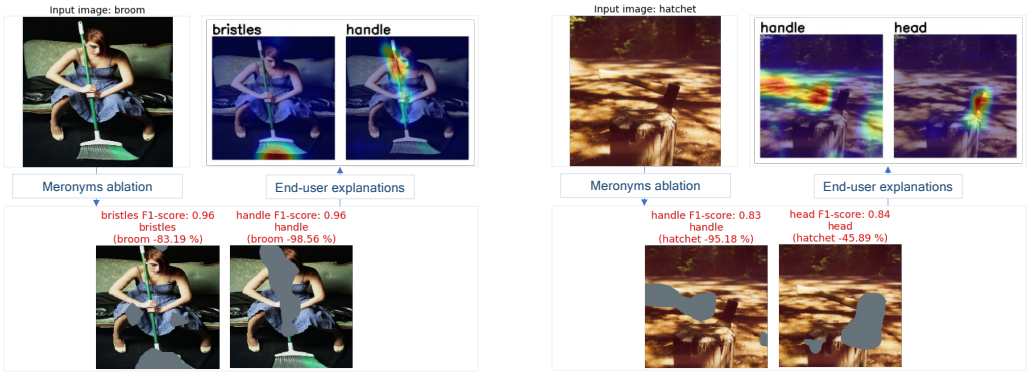


Fig. 3.13 HOLMES explanations example. Two different correctly predicted input images (broom on the left and hatchet on the right) get their respective parts identified and ablated. Since for all their meronyms both the calibrated F1-score threshold and the holonym score drop threshold is exceeded, all the parts (in red) are included in the end-user explanations. Notice as in both cases all the meronyms calibrated F1-scores are quite high (0.83 to 0.96), hence the image portions which are highlighted effectively describe the meronyms.

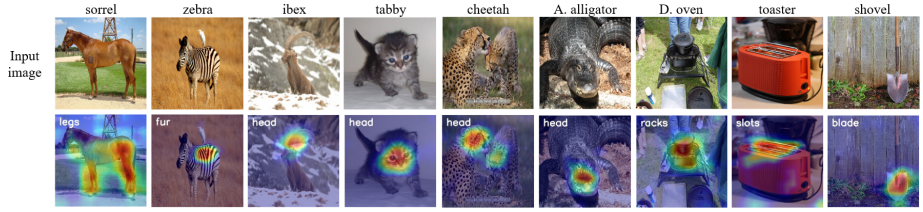


Fig. 3.14 Examples of end-user explanations comprising 1 meronym.

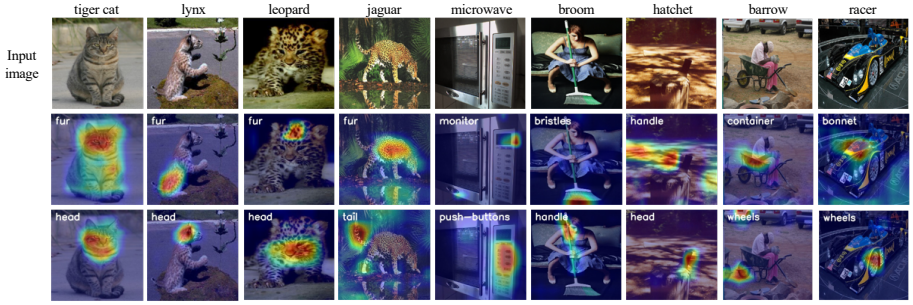


Fig. 3.15 Examples of end-user explanations comprising 2 meronyms.

acquired datasets over time. Moving forward, it becomes crucial to address this specific aspect in future research endeavors, thereby ensuring the sustained relevance and robustness of the meronym models. By confronting these challenges directly, the aim is to enhance the longevity and applicability of this approach within the context of a continually evolving digital landscape.

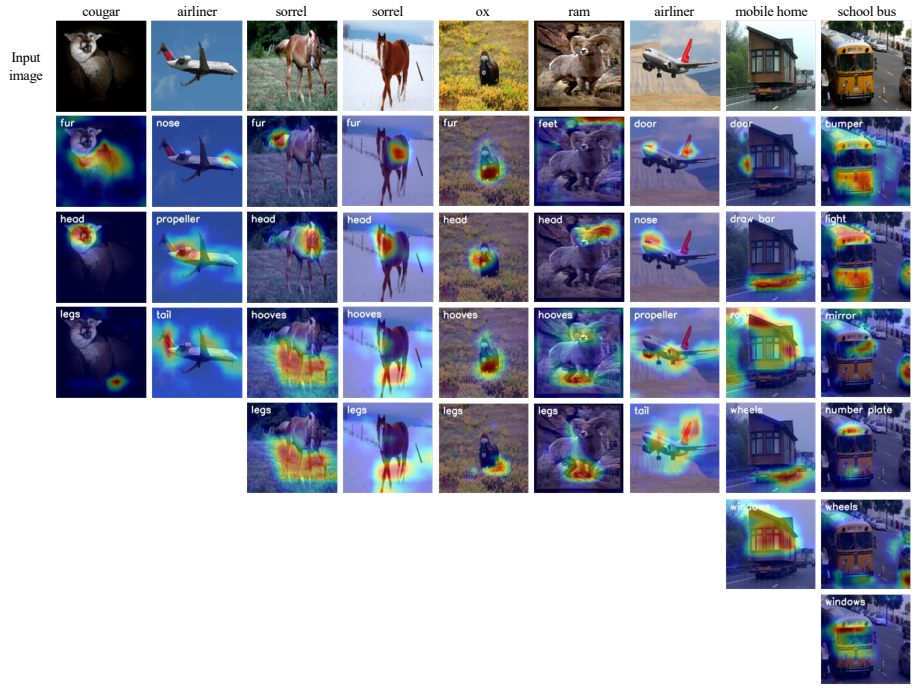


Fig. 3.16 Examples of end-user explanations comprising 3 or more meronyms.

Overall, quantitative evaluation on PASCAL-Part confirms that the meronym models can identify object parts and determine their position within the image. This is accomplished by exploiting the features learned by the holonym model without the need for retraining or fine-tuning, further supporting the notion that DNNs implicitly embed knowledge about object parts [146, 147, 121].

Due to the imperfect background, the ablated mask may not provide a perfect segmentation of the part, as shown in Fig. 3.5. The ablated masks for different parts may be very similar, especially for meronyms that are physically adjacent. For instance, the meronyms *legs* and *hooves* for the holonym *sorrel* exhibit visual overlap, as illustrated in Fig. 3.5. Less frequently, the ablated part may include a portion of the background; for example, legs may include some terrain or grass. This could affect the score drop observed when deleting the corresponding part and the resulting metrics.

The F1-scores for most holonyms vary between 0.6 and 1.0 for ImageNet and 0.8 and 1.0 for PASCAL-Part, encompassing multiple object categories such as animals, tools, and vehicles, and up to 14 visible parts per class. The quality of the ground truth affects the F1-score, but the number of meronyms that constitute

each object also plays a role. Improving the quality of the scraping process, and consequently the meronym model, could enable HOLMES to recognize and include an even greater number of meronyms in its explanations. Furthermore, HOLMES offers intrinsic safeguards against this type of noise by only incorporating meronyms with satisfactory F1-scores into its explanations, and by allowing users to easily inspect the heatmaps associated with each individual meronym.

The effectiveness of the part-based explanations provided by HOLMES in identifying the most relevant parts for the final classification is confirmed by quantitative causal metrics based on the deletion/insertion/preservation curves. These results are comparable to those of the state-of-the-art Grad-CAM method, and substantially better than chance level. Unlike Grad-CAM, HOLMES generates a set of articulate heatmaps associated with human-interpretable concepts, and enables exploration of the impact of individual meronyms on holonym classification at both the instance and class levels.

HOLMES was evaluated on two separate datasets, PASCAL-Part and ImageNet, both of which use the same classifier. Absolute differences in the insertion/deletion/preservation curves are attributed to the datasets themselves, including how the images were obtained, and possibly the domain shift between ImageNet and PASCAL-Part, since the holonym classifier $\mathscr{H}$ was trained on ImageNet. However, despite substantial differences in the sourcing of the meronym datasets X^c, the relative performance of HOLMES with respect to both baselines remains similar. HOLMES performs slightly better on PASCAL-Part, particularly in terms of the deletion and preservation curves. Additionally, explanations on PASCAL-Part tend to focus on fewer parts on average than those on ImageNet. These differences could be due to other factors besides the meronym datasets X^c. On the one hand, PASCAL-Part contains fewer and more distinct classes than ImageNet, potentially including images that are easier to classify. On the other hand, the KB derived from PASCAL-Part annotations is simpler, with fewer meronyms (approximately 4 versus approximately 7 parts per class) and less visual overlap. HOLMES is demonstrated to be resilient to variations in experimental settings, and performs well even when utilizing more cost-effective annotations obtained through general-purpose KBs and web scraping.

3.3.6 Limitations and Future Directions

While HOLMES demonstrates impressive performance within the context of a feedforward CNN trained for classification, limiting the application of this approach to a single type of neural network would hinder its broader potential. Therefore, in this section, we discuss acknowledged limitations and propose directions for future research to expand the applicability of HOLMES.

First and foremost, although the experiments in this chapter focus exclusively on classification networks, HOLMES could be extended to encompass various networks designed for different tasks. Regardless of a model's primary function, the latent representations acquired by its foundational layers can be utilized to train meronym models, enabling part-based explanations using the HOLMES framework.

Secondly, an important question arises about the adaptability of the HOLMES approach across different network architectures beyond CNNs. While this chapter primarily addresses feedforward neural networks, exploring the integration of HOLMES with other architectures like RNNs or transformers shows promise. This adaptation would require a thorough analysis of the internal representations and learning mechanisms inherent to these architectures. Such future investigation involves understanding the hierarchical features learned by RNNs or transformers to establish analogous holonym-meronym relationships.

Finally, a notable limitation of the current methodology relates to the analysis of samples lacking inherent hierarchical correlations. This situation often arises in fields like the medical domain, where samples could represent complex tissues or fine-grained patches extracted from MRI scans. In such cases, dividing a holonym into corresponding meronyms might not be feasible, making texture- or pattern-based approaches more suitable for understanding model decision-making processes. To address this, a potential modification to the HOLMES methodology could involve training texture- or pattern-based models using a strategy similar to the one outlined in this chapter. In this scenario, the training data for these auxiliary models could consist of small patches representing prevalent patterns or textures crucial for specific predictions. However, this approach would require the involvement of an expert capable of labeling important features within the samples needing explanations. Furthermore, since in feedforward models like VGG16, filters responsible for detecting

basic patterns are typically learned in the initial layers, utilizing features extracted from these layers for training auxiliary models could be explored.

3.4 Concluding Remarks

The first section of this chapter introduces iNNvestigate-GUI, a comprehensive approach to interpret the behavior of DNNs using visualization methods. The tool provides an interactive panel to analyze and compare the behavior of multiple networks. Additionally, the Suggestion panel categorizes available samples and enables users to select a mix of samples with varying properties for inspection. The iNNvestigate-GUI approach assumes that it is beneficial for users to examine predictions across a range of data instances with diverse properties. To assess the usability of the tool, a user study was conducted, which involved participants from different backgrounds, including ML experts and novices. The results show that the tool is user-friendly and intuitive, and it can be used by both expert and non-expert users to analyze the behavior of DNNs. Moreover, the study demonstrates that the tool's visualization methods are effective in identifying model behaviors and suggesting useful data samples for analysis. Overall, the iNNvestigate-GUI provides a powerful, yet easy-to-use solution for interpreting DNNs. This tool holds significant potential for diverse applications, such as model debugging, feature engineering, and ensuring algorithmic fairness. In the context of fairness, upcoming experimental configurations should showcase how the GUI recommends and manages samples when confronted with model biases. Furthermore, it is important to note that the capabilities of the proposed iNNvestigate-GUI extend beyond just plain CNNs. While the initial implementation focuses on CNNs, the framework has been designed to be adaptable and extensible, allowing for integration with a wider range of neural network architectures. This adaptability paves the way for the future extension of the tool to encompass more complex models like RNNs and Transformers, which are extensively used in tasks like sequential data analysis and natural language processing.

In the second section of the chapter, HOLMES is introduced, which is a novel XAI approach capable of enhancing image classification tasks by providing detailed part-level explanations. The method represents a significant advancement beyond the traditional label-level heatmaps that have been the state-of-the-art in eXplainable AI

for image classification. The approach offers several benefits, including the ability to integrate image classification models into decision support systems, enabling developers to debug classifiers before deployment and helping end-users to assess the classifier's reliability when working with previously unseen data.

Additionally, HOLMES offers valuable insights into how holonyms are learned and stored within CNNs during and after the training phase. Other recent studies, such as [121, 148], have proposed significant contributions in this area, but have additional requirements, such as a focus on scene recognition or the need for a segmented ground truth. Furthermore, this approach connects DL models with a symbolic KB, such as an ontology, which is a unique feature. Moreover, the approach opted to capture concepts without constraining them to a single computational unit, an approach that more naturally captures the robust learning of DL models while also providing greater expressive power on the symbolic side. As demonstrated in [147], models with the same discriminative power can have varying degrees of interpretability, depending on several factors, including the network structure, regularization, and supervision/task.

Given the novelty of the approach, there is ample room for further research on alternative components. First, a more refined scraping algorithm could be employed to increase both the training sample size and the sample quality. This might involve using more complex semantic expansion techniques, robust outlier detection algorithms like Robust and Kernel PCA, or novel data purification techniques like those described in [149]. Secondly, it may be worthwhile to investigate the effects of using the activations of units from different convolutional layers, or even sets of layers, to generate HOLMES explanations. Units belonging to different convolutional layers may better match specific (part) concepts, and their activations could be used to generate more accurate explanations for those concepts. Additionally, more model architectures could be tested to see how this method performs with different models, such as shallower (e.g., VGG13) or deeper (e.g., VGG19) models, or different types of networks (e.g., Deep Residual Networks). Thirdly, alternative perturbation techniques could be explored for removing the relevant pixels of the parts. It was observed that substituting pixels with constant values introduces contiguous shapes in the image, which may bias the prediction towards certain types of objects with similar shapes. Additionally, other types of semantic relationships could be studied to retrieve the desired (visible) parts of a specific concept and map different

concepts between different KBs (in alternative to the proposed hypernym-hyponym relationship).

Finally, it was demonstrated that the method performs better when a small number of parts for an object are considered, preferably spaced enough to minimize visual overlap. Therefore, a new strategy for selecting and filtering the meronyms of an object can be explored. In conclusion, HOLMES is a promising approach for enhancing image classification models with part-level explanations. There is still much to explore in terms of alternative components, and it is arguably believed that future research in this area will further improve the reliability and interpretability of DL models.

Chapter 4

Monitoring Data Changes

Part of the work described in this chapter was originally presented in [150, 151].

4.1 Introduction

One of the basic assumptions in ML is that the data points at test time come from the same distribution as the training set. However, this assumption is often violated in most ML-based applications, as they operate on continuous data streams whose nature is likely to change over time [23, 152, 153]. This phenomenon, known as *dataset drift or shift*, causes the performance of an ML model to *degrade* compared to that measured before deployment. Dataset drift has been investigated in a variety of fields, including predictive maintenance, recommender systems, autonomous driving, weather forecasting, etc. [23, 152, 154, 155]. Deviations from the training data can take many forms, from changes in the distribution of the input data to the emergence of entirely new concepts or classes. Hence, from this general problem two fundamental issues can be identified.

On the one hand, the emerging field of MLOps aims to establish best practices for effective, safe, and trustworthy maintenance of the models. A key aspect of successful MLOps strategies is the ability to monitor models in the field and identify when retraining is needed to maintain an adequate performance level [23]. However, directly measuring model performance requires continuously collecting labeled data, which can be costly and time-consuming. As a result, several techniques have been proposed to detect drift in an unsupervised fashion, such as detecting changes in the

input data distribution. However, most techniques have been proposed and tested for tabular or sensor data, whereas here extension and discussion to DL is discussed.

On the other hand, it is possible to focus on individual samples in order to determine whether they fall within the training distribution. This setting is related to drift detection, but focuses on the individual rather than the population. An observation that differs abnormally from other values in a population's random sample is known as an outlier in statistics [156]. Since every data population differs from the others in some ways, there is sadly no agreed-upon definition of what is "normal". [157] noted that it is not sufficient to determine whether a point should be classified as an outlier or not based just on its distance from the distribution's mean and that not all outliers require the same handling. Although they are not always used correctly and consistently, a variety of names have been used to refer to this issue in the context of DL in its many features and situations. The following is a typical classification ([158], [159]):

- Detection of anomalies: During training, a dataset consisting of just one class is employed, and during testing, the detector eliminates inputs that are anomalous due to a covariate or semantic shift.
- Novelty detection: the detector recognizes inputs that belong to any other class after being trained on a single-class or multi-class dataset. Samples aren't classified based on the semantic category to which they belong; instead, the only output is a binary distinction between distribution points that are in and distribution points that are out.
- Open set recognition: the model can properly identify samples from a specified set of categories while rejecting those from unidentified classes.
- Outliers detection: The model eliminates entries that are "far" from the other entries given a fixed set of data points. In contrast to the preceding settings, there is no separation between training and testing in outliers identification.

Even though they were typically treated differently and separately in earlier literature, these cases have characteristics that allow applying solutions to one to the other fairly simple and efficient. For instance, a novelty detector can be trivially converted into an open set recognition model by pairing it with a separate classifier able to discriminate between in-distribution (ID) categories. Given these affinities, a

broader term that includes all the various settings mentioned above may be defined: OOD Detection, the act of telling apart points belonging to a given distribution from the others. In this context, ID data refers to the set of examples used to train a model, originating from the same data distribution that the model is meant to encounter during testing or deployment. On the other hand, OOD data represents instances that lie outside the scope of the training data distribution, exhibiting different patterns, characteristics, or contexts.

Interest in OOD detection began to rise in the DL research community after Amodei et al. [160] listed robustness to distributional changes among the main problems concerning AI safety. ML agents should be able to precisely measure the confidence in their predictions in to deal with undesirable inputs and prevent failures of deployed models, which is a challenging task otherwise because to the general lack of model explainability. The posterior probability that neural networks produce, which may be understood as the distance of the point from the decision border, is used by neural networks as a measurement of confidence. The effort needed to accomplish their purpose is, however, minimized by ML models, as the learnt decision boundaries can only distinguish between ID classes and cannot deal with unknown distributions. Furthermore, networks have been shown to often be poorly *calibrated*, i.e. to have large gaps between predicted confidence and actual error probability [161], and to be able to output arbitrarily high confidence for observations far away from the training data [162].

In both cases, benchmark datasets, in which known drifts occur, are needed to design and evaluate drift/OOD detection algorithms. However, many current benchmarks are not representative of typical industrial and business use cases. In this chapter, the design of new techniques and benchmarks for drift detection and OOD detection in CV will be discussed, with the goal of introducing more complex and realistic benchmarks than toy problems commonly used in CV, such as CIFAR10, CIFAR100 or MNIST and their variations. The rest of this Chapter is organized as follows. Section 4.2 describes techniques for unsupervised concept drift detection. The study on OOD detection is reported in the following Section 4.3. Finally, in Section 4.4 conclusions as well as possible future directions are discussed.

4.2 Drift Detection: Ensuring Model Robustness and Performance

The trustworthy deployment of DL models is challenging due to their complex, black-box nature and their ability to process raw high-dimensional data (e.g., images or text). To achieve unsupervised drift detection, several issues need to be addressed, such as how to measure changes in high-dimensional multivariate distributions and how to effectively reduce input dimensionality [163].

In addition, benchmark datasets, in which known drifts occur, are needed to design and evaluate drift detection algorithms. However, many datasets collected for research purposes are not representative of typical industrial and business use cases. One potential solution is the use of synthetic datasets, which are rapidly expanding as they allow for the quick generation of large quantities of labeled training data [164]. In this chapter, the use of synthetic data for testing drift detection algorithms in controlled but practical dataset drift scenarios is proposed. These scenarios are specifically designed for document segmentation and analysis, a problem class with numerous practical applications.

The present section reviews available methods for dataset drift detection, analyzing their benefits and drawbacks. Supervised and unsupervised drift detection techniques are experimentally evaluated on a practical case study in the domain of semi-structured document segmentation, simulating dataset drift using a customized synthetic data generator. The remaining section is structured as follows. Section 4.2.1 provides a review of recent literature on the concept of drift. In Section 4.2.2, the problem of detecting drift is defined mathematically. Sections 4.2.3 and 4.2.4 present a simulated real-life scenario for detecting drift, as well as the methodology utilized to detect it. Finally, Section 4.2.5 provides a summary and discussion of the experiment results, along with potential directions for future research.

4.2.1 Related Work

Drift detection techniques can be classified into two categories: supervised and unsupervised [23]. Supervised techniques aim to detect changes in the error rate, but they rely on the assumption that labeled data is continuously available at test time. Unsupervised techniques, on the other hand, aim to bypass this requirement

by modeling the input data distribution, either directly or through auxiliary models or algorithms. In this section, an overview of the most representative approaches is provided, and the reader is directed to previous surveys and books [23, 165] for additional information.

One class of supervised techniques is based on monitoring changes in the error rate. Statistical learning theory suggests that a change in the data distribution will lead to an increase in the error rate [21]. The simplest implementations set one or more alarm thresholds for the raw error rate, while more sophisticated techniques track the distance between errors computed at different times [21] or use statistical change-detector tests (CDTs) [166].

Unsupervised techniques, on the other hand, attempt to detect changes in the distribution of the input data over time. These techniques often divide streaming data into time windows and compare the distribution in the current window to a reference distribution, which may be obtained from the training set or from typical operating conditions (e.g., immediately after deployment) [152, 167]. A challenge in these techniques is how to effectively compare multi-variate data distributions, as most modern ML/DL techniques operate on high-dimensional datasets. Once a measure of diversity is computed, drift can be detected using a fixed or adaptive threshold [168].

One approach is to use statistical methods to test the null hypothesis that the two distributions are the same, treating the test score as a "driftiness score" [163]. Both univariate tests on individual features (with corrections for multiple tests) and multivariate tests such as the Maximum Mean Discrepancy (MMD) have been employed. These techniques work best when the intrinsic dimensionality of the input data is low [163, 168].

An alternative is to compute the distance between the two distributions using a measure such as the Hellinger distance or Kullback-Leibler divergence [167]. Another strategy is to use XAI techniques to compute multiple model explanations over time and observe how they change [169].

A second challenge in unsupervised drift detection is how to reduce the dimensionality of the input data. Most drift detection techniques are designed for tabular data or temporal series [152, 167], while DL models operate on raw data such as images or text. Dimensionality reduction is necessary to make the problem numerically tractable and enable a semantically meaningful comparison between different data points [163]. Rabanser et al. [163] compared different dimensionality reduction

methods for image datasets, including Principal Component Analysis (PCA) and deep auto-encoders, but their experimental results were based on toy datasets and may not generalize to more realistic settings.

Unsupervised approaches offer several benefits, such as not requiring labeled data during testing, being computationally efficient, and having readily available open-source implementations [168]. However, it is important to note that a change in the distribution of input data does not necessarily result in model degradation.

It is worth mentioning that recently proposed techniques have introduced an auxiliary model to assess the consistency of an instance with the training set distribution [170]. These models can be trained on the available dataset, and do not require prior knowledge of OOD samples. However, pre-trained models may not have access to the training set due to privacy or commercial limitations, which necessitates the collection of a dedicated dataset.

Finally certain types of dataset drifts can be addressed by drawing from the field of open-set recognition, which assumes that new, previously unseen classes will emerge during testing [171–173]. This issue has been extensively explored in the CV domain, such as object classification/detection. Typically, these techniques do not require labeled samples during testing, but rather necessitate modifying the model or training procedures, incorporating additional losses, or relying on auxiliary models to differentiate between ID and OOD samples.

4.2.2 Problem Definition

A model is trained on a source distribution S and tested on a target distribution T, represented by the joint distribution $P(x,y)$ of the input data x and the label(s) y, which can be further decomposed as:

$$P(x,y) = P(x)P(y|x). \tag{4.1}$$

The dataset drift phenomenon can be defined as a change in the distribution of the source and/or target data [23]. Dataset drift may degrade the performance of models trained on historical data, and hence must be detected to trigger a model update. In mathematical terms, it can be formalized as a change in the distribution, $P(x_S,y_S) \neq P(x_T,y_T)$. Researchers distinguish between different types of drifts

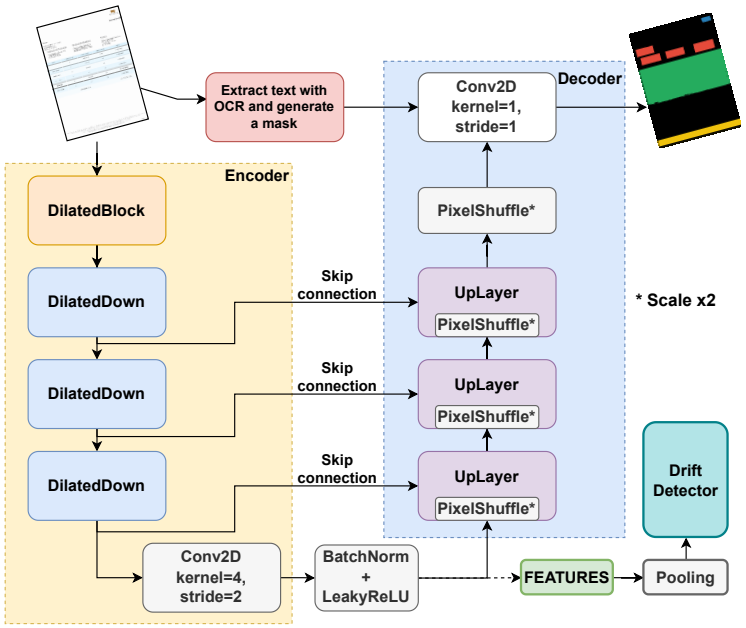


Fig. 4.1 The DL model utilized for document segmentation employs the following structure. The encoder compresses the input image into a low-dimensional feature space, which is subsequently upsampled in the final output space by the decoder. The final classifier layer takes a binary mask indicating text presence, detected by the OCR, as an additional input. The model is language-agnostic and does not rely on the text or semantic embedding. The final output image is segregated into five classes: logo, header, table, footer, and background. The approach used in this work diverges from that of [174] in that convolution and pixel shuffle are employed for downsampling (encoder) and upsampling (decoder), as opposed to pooling and unpooling. Inference involves the acceptance of the encoder's features, which are further reduced in dimensionality through pooling, as input by the drift detector.

(covariate and concept drift) depending on which aspect of the data distribution is most affected [165]. *Covariate drift* occurs when the distribution of the input features changes, but without changing the relationship between the input and output variables of the model:

$$P(x_S) \neq P(x_T) \text{ and } P(y_S|x_S) = P(y_T|x_T). \tag{4.2}$$

On the other hand, *concept drift* refers to a change in the relationship between the input and output variables:

$$P(x_S) = P(x_T) \text{ and } P(y_S|x_S) \neq P(y_T|x_T). \tag{4.3}$$

In some cases, both covariate drift and concept drift can occur simultaneously.

In this work, the focus is on unsupervised drift detection techniques that do not necessitate access to labeled data from the target distribution. Unsupervised drift detection can be accomplished by monitoring the performance of a model on the target data or by detecting changes in the input data distribution.

Several methods have been proposed to detect changes in the input data distribution, including statistical tests, distance measures, and density estimators. Statistical tests, such as the Kolmogorov-Smirnov test, compare the distribution of the input data in the source and target domains, and can be used to detect both covariate and concept drift. Distance measures, such as the Kullback-Leibler divergence, compare the similarity between the distributions of the input data in the source and target domains. Density estimators, such as kernel density estimates, estimate the probability density function of the input data, and can be used to detect changes in the distribution over time.

In addition to these unsupervised drift detection methods, there are also supervised drift detection techniques that rely on labeled data from the target distribution. These methods can be more accurate, but are often impractical due to the cost and difficulty of collecting labeled data in the target domain.

In particular, the performance of several unsupervised drift detection methods is evaluated on a practical case study in the domain of semi-structured document segmentation, utilizing a customized synthetic data generator to simulate dataset drift. The results show that the performance of these methods is dependent on the specific characteristics of the dataset drift, and that no single method is universally superior.

4.2.3 Simulating Data Drift in Scanned Document Segmentation

The case study being discussed involves the task of segmenting scanned documents, such as invoices, into their various components (logo, header, table, footer, background). The DL model used in this study (shown in Fig, 4.1) is based on previous work in [174], with some modifications. It consists of an encoder and a decoder, both of which are essential for the model to function. The encoder is responsible for learning a feature representation of the input data, while the decoder takes this learned representation and uses it to reconstruct the input and generate the output

segmentation masks. The decoder also receives as input a binary mask generated by running Optical Character Recognition (OCR) on the original document. It's worth noting that the model does not use the recognized words or their embeddings as input, only the presence or absence of text.

For training the network, a separate dataset of 100,000 invoices (which is not the dataset utilized in the experiments described here) was employed. As obtaining pixel-level labels is expensive, a synthetic document generator (shown in Fig, 4.2) was developed to produce the required data. The document structure is defined by customizable templates, which are filled with fake data and rendered utilizing an HTML rendering engine. Subsequently, the physical printing process is simulated by applying data augmentation techniques such as rotation and adding Gaussian noise.

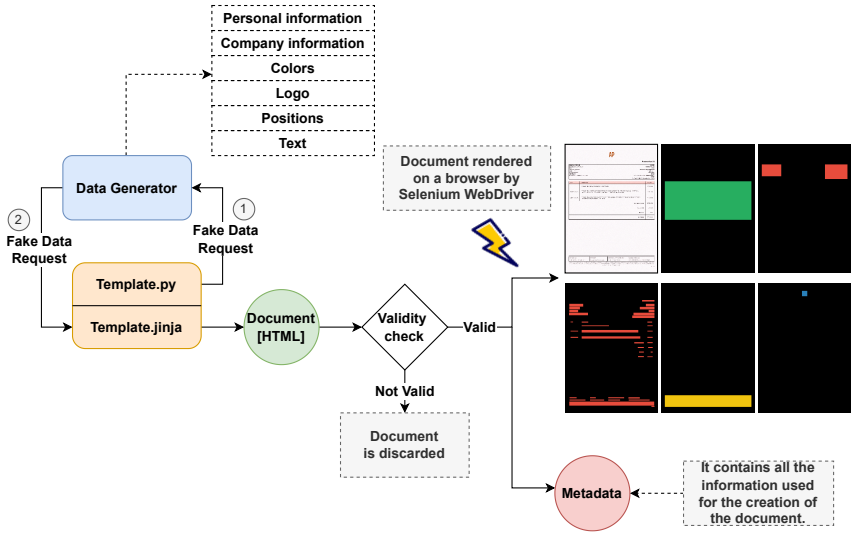


Fig. 4.2 The structure of the artificial document generator allows for the creation of various dataset shifts, such as the addition of new templates. All document components are examined to confirm proper rendering, ensuring no significant overlaps or deviations from printing margins. The generator automatically produces the ground truth, which includes a separate mask for each class, in conjunction with the input image.

To model data drift, two OOD datasets were created by introducing two novel features not observed during training: a different background color or an additional background image or logo. Additionally, a dataset with no drift was created. For each scenario, 1000 documents (without replacement) were sampled from the ID and one of the two OOD sets to create 11 time windows in which the percentage of

OOD samples increased from 0% to 100% in increments of 10%. This process was repeated 100 times to calculate confidence intervals.

4.2.4 Methodology

The performance of a supervised drift detection model was measured using the IoU metric, which quantifies the overlap between predicted segmentations and ground truth [174]. For each time window, the mean IoU was calculated over N documents. Mathematically,

$$IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union}$$

Unsupervised drift detection was achieved by calculating the Hellinger distance [167] using the following formula:

$$H(p,q) = \frac{1}{\sqrt{2}}\sqrt{\sum_{i}^{n}(\sqrt{p_i} - \sqrt{q_i})^2} \tag{4.4}$$

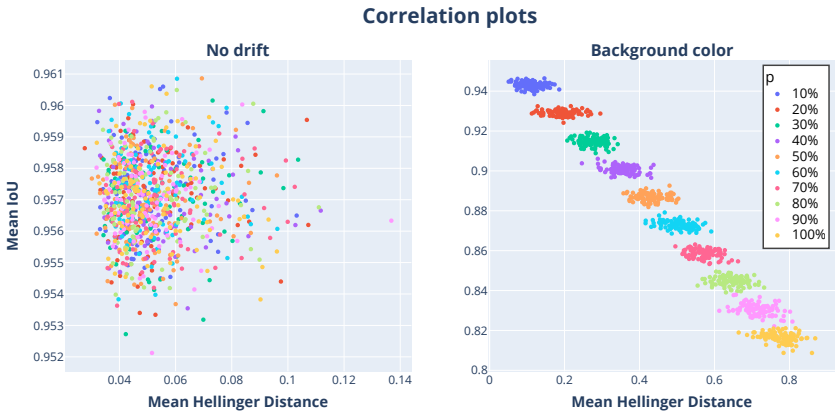
where p and q are the discrete distributions of the current and reference time windows, respectively.

High-resolution images were first compressed into compact feature vectors using the encoder of a DL model. The encoder reduces an image of size 512x365x3 to a feature map of size 32x32x520 and further to a feature vector of size 520 using either Global Max Pooling or Global Average Pooling. The latter was found to provide slightly better results. The feature distribution (histogram) was then computed for each time window of size N. The window's "driftiness score" was calculated as the Hellinger distance between the current and reference time windows (p = 0).

The correlation between the mean IoU and Hellinger distance was studied to understand if a change in the feature distribution is indicative of model degradation. Additionally, the classification performance of the unsupervised drift detection algorithm was evaluated by comparing the results of applying a fixed threshold on the Hellinger distance to the ground truth. The AUC provides a global measure of the algorithm's ability to distinguish between dataset drift and natural data variance.

(a)

(b)



(c)

Fig. 4.3 The metrics for supervised (a) and unsupervised (b) drift detection are visualized via distribution violin plots. The mean IoU (a) assesses the quality of output segmentations, while the Hellinger distance (b) quantifies the difference between the current time window and a reference time window with no drift ($p = 0$). To generate the distribution, $Z = 100$ time windows are sampled for each drift scenario and fraction p of OOD samples. The correlation plots (c) depict the correlation between the IoU and Hellinger distance for both no drift and background color covariate drift scenarios.

The unsupervised drift detection algorithm's classification performance was evaluated by comparing the results obtained by applying a fixed threshold on the Hellinger distance to the ground truth. The algorithm classified each time window as either "drift detected" or "no drift" and the results were compared against the known values of "drift present" ($p > 0$) or "no drift" ($p = 0$). For each configuration, a sample of 100 time windows was taken, and true and false positive rates were

determined based on the percentage of correctly and incorrectly classified time windows.

4.2.5 Results

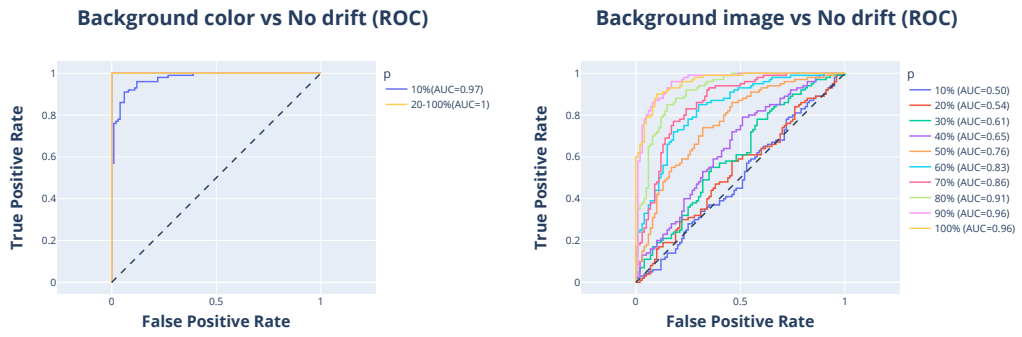
A comparison of the distribution of supervised and unsupervised metrics in two types of drift scenarios (background color and background image) is presented. Fig. 4.3 is used to demonstrate this comparison, where both metrics are represented by violin plots. The effect of these drift scenarios on the model performance is evaluated by varying the proportion of OOD samples, represented by the variable p, from 0% to 100%. A set of documents from the ID set is sampled at each time point as a baseline to control for fluctuations that can be attributed solely to the variance in data.

The results in Fig. 4.3 illustrate that different novel features can significantly impact the performance of the DL model. It can be observed that the mean IoU drops substantially (-0.14) in the background color scenario, while it remains relatively stable (-0.02) in the case of background images. This is in line with previous literature, which suggests that DL models can be fairly robust to certain sources of covariate shifts, depending on the regularization and data augmentation techniques applied during training.

It is also observed that the Hellinger distance increases proportionally as the fraction of OOD samples increases. However, it is worth noting that the distance grows faster in the background color scenario, where the model degradation is stronger. Additionally, the correlation between the Hellinger distance and the mean IoU is weak without drift ($\rho = 2.51 \times 10^{-1}$), but strong in the background image ($\rho = -0.904$) and background color ($\rho = -0.998$) scenarios.

The accuracy of the drift detector was evaluated using ROC curves, as shown in Fig. 4.4. The results indicate that the detector can effectively distinguish drift caused by background color from random noise resulting from data sampling, with almost perfect performance across all values of p. On the other hand, drift due to background images can only be detected with acceptable performance (AUC$>$ 0.85) when $p > 70\%$. When a threshold of 0.1 is set on the Hellinger distance, drift due to background color can be detected with a true positive rate of 100%. However, drift due to background images is only detected when $p > 60\%$, and with low confidence (true positive rate between 12% and 50%). The performance of the detector is

dependent on the size of the time window used in the experiments ($N = 1000$), and it may vary with changes in the window size.



(a) (b)

Fig. 4.4 The ROC curves for drift types are plotted against the percentage p of OOD samples. In the case of background color vs. no drift classification (a), an AUC of 1.0 is attained for $p \geq 0.2$.

4.3 OOD Detection in DL

There is a strong interest in making CNN classifiers more robust by endowing them with the capability to separate samples to a given distribution from the others [158–160, 162].

Several methods were proposed in the literature for OOD detection [158]. However, the comparison of different methods is complicated by the broad definition of OOD and the wide variety of settings under which they are tested: the best method intrinsically depends on the experimental settings and its underlying assumptions. For instance, distance-based methods were found to yield better performance than those based on prediction scores depending on whether the OOD samples are far away or close to the decision boundary between classes [175].

In real-world scenarios OOD samples can be both largely different or fairly similar with respect to those considered as ID. [176] distinguishes among *Far-OOD* and *Near-OOD* samples. In particular, Far-OOD samples are those that are significantly different from the ID samples and are often from a completely different distribution. For example, a model trained on natural images may encounter Far-

OOD samples from medical images. On the other hand, Near-OOD samples are those that are similar to the ID samples but have subtle differences. For example, a model trained on natural images may encounter Near-OOD samples that are heavily occluded or have been digitally altered. The ability to accurately detect both types of OOD samples is important in real-world applications where the model may encounter unexpected data. In general, performance on Far-OOD detection is usually satisfactory, Near-OOD detection is still challenging for many state-of-the-art methods [177]. Another recent study [175] shows how, even on handcrafted toy datasets, no single state-of-the-art method can be identified as out-performing the others. Additionally, the authors demonstrate a correlation between the size of the dataset and the performance of OOD detection techniques, thus underlining how the problem is strongly dataset-dependent. Other studies [178–180] demonstrate how a large number of classes and real-world images can prove challenging for many well-established OOD detection techniques. Finally others [181, 182], discuss the open issue of the selection of a proper benchmark for OOD-Detection, demonstrating how choosing two different datasets as ID and OOD (as performed by a large amount of the literature) can result either in an overly simplified problem of Far-OOD detection or in a large semantic overlap between the two distributions.

In this context, the current work aims to describe an innovative methodology adopted during the construction of a novel large-scale and realistic OOD Detection benchmark. Moreover, a performance comparison of several state-of-the-art techniques for OOD detection on the proposed dataset is reported. In particular, the study focuses on techniques that do not require the retraining of the target model to detect OOD samples. As highlighted by recent research works [171], current solutions for OOD detection encounter difficulties when applied to real-world image datasets with numerous classes, making it important to evaluate the effectiveness of OOD detection techniques on a challenging benchmark. As such, this study proposes a set of datasets with increasing levels of semantic overlapping, spanning from Far-OOD samples to Near-OOD samples, and evaluated state-of-the-art techniques for OOD detection on these benchmarks. By conducting a comprehensive comparison of these techniques on a realistic datasets, more insights into the strengths and limitations of different approaches are provided, as well as a contribution towards improving the state-of-the-art in OOD detection. Finally, the novel methodology to perform a semantic mapping between concepts from different datasets is described in details.

The rest of the Chapter is organized as follows. Section 4.3.1 provides an overview of the related work on OOD detection, including existing techniques and benchmark datasets. In Section 4.3.2, the problem is defined and a brief explanation of the OOD techniques compared in the experiments is provided. Section 4.3.3 details the experimental settings and the process of creating several realistic benchmarks for OOD detection. The results obtained from the experiments are presented and discussed in Section 4.3.6.

4.3.1 Related Work

OOD Detection Techniques In light of recent attention given to the detection of OOD samples, numerous approaches have been proposed to tackle the issue. These solutions can be broadly categorized into two groups: classifier-based and generative techniques. While the latter aims to explicitly model the target distribution to reject OOD samples, the former concentrates on strengthening the current model's robustness or combining it with an external classifier for OOD detection. Classifier-based methods refer to a broad family of solutions that tend to prioritize discriminative solutions to the problem of OOD detection, although in significantly diverse ways. Confidence-based techniques aim to produce a numerical score that can accurately evaluate the probability of a prediction being accurate. Common choices for this metric include the softmax and the logit, with potential modifications in their definition or network to increase OOD detection performance.

Despite considerable attention paid to related challenges during the 1990s, Hendrycks and Gimpel [183] are widely recognized as the first to define OOD detection, present assessment criteria, and propose a baseline solution. In their work, the output of the softmax score relative to the predicted class (Max Softmax Probability, MSP) is selected as the OOD detection score. Liang et al. [171] proposed an improved version of MSP called ODIN, which suggests scaling the softmax score by a constant temperature parameter T. Additionally, the final score is computed on a preprocessed version of the input image, via a constant magnitude shift in the direction described by the gradient of the loss function, obtained through a preliminary forward pass. These techniques help to expand the difference between the ID and OOD samples' softmax scores. The dependence on OOD data, which was used in the original work to choose appropriate values for the temperature T and the magnitude, is eliminated by generalized ODIN (Hsu et al. [184]). It achieves this

Table 4.1 Usage statistics for the 10 most popular data sources used as both ID and OOD distributions on papers performing experiments related to OOD detection.

	CIFAR-10	CIFAR-100	SVHN	MNIST	TinyImageNet
Popularity	87%	57%	57%	56%	38%
ID	55	26	20	32	13
OOD	39	24	34	29	21

	LSUN	Fashion-MNIST	Gaussian Noise	Textures	Places365
Popularity	37%	19%	22%	17%	17%
ID	0	9	1	0	1
OOD	23	9	14	11	11

by swapping temperature scaling for a function of the input and selecting the perturbation magnitude value that maximizes the softmax for ID samples. Conventional OOD identification algorithms encounter difficulties with large semantic spaces, as evidenced by MOS (Minimum Others Score, Huang and Li [180]), which suggests grouping related categories to address the issue. The classifier in charge of each group benefits from a reduced complexity of the decision boundary between known and unknown samples, resulting in better OOD detection abilities. To combat the issue of the huge semantic space, Maximum Logit Value (MLV [179]) eliminates the normalizing impact of the softmax function and utilizes raw logits as the OOD detection score, preventing probability mass from spreading between similar classes. Bendale and Boult [185] propose OpenMAX as a substitute for the softmax, believing it to be an unsatisfactory option for OOD detection. The intuition behind their approach relies on the interdependence among the outputs of different classes, grouped in an Activation Vector (AV). Each category is then represented by a Mean Activation Vector (MAV), whose position to the AV of an input image is used for OOD detection. Meinke and Hein [186] introduced CCU, a model able to provide close-to-uniform prediction confidence for inputs that are sufficiently different from the samples seen during training.

OOD Detection Datasets Previous studies have frequently employed a common approach, which is to designate one dataset as the ID set and multiple other datasets as OOD. The choice of datasets is often similar across research studies and driven by convenience in retrieving or processing well-established research benchmarks. This approach allows for easy comparison of results with other methods tested on the same benchmark.

To retrieve common choices for ID and OOD datasets, a set of 63 papers published on the topic of OOD detection in recent years (spanning from 2015 to 2021, with a single outlier from 1999) were analyzed. Usage statistics for the 10 most commonly selected datasets are reported in Table 4.1. For each dataset, the popularity (measured as the percentage of papers in which the dataset is used) is reported along with the number of papers using the dataset as an ID or OOD source of data. Many popular datasets are toy datasets such as CIFAR-10, CIFAR-100, MNIST and Fashion-MNIST, containing a small number of classes and low-resolution images, allowing for quick and rapid experiments. CIFAR-10 [187] is the most commonly used dataset, appearing in 87% of the analyzed papers. The dataset contains a large number of small resolution images (60,000) from 10 classes, making it suitable for evaluating the performance of DNNs on a variety of classification tasks. CIFAR-100 [187] and SVHN [188] are the next most commonly used datasets, appearing in 57% of the papers. CIFAR-100 has 100 classes, and each class contains 600 images. SVHN is a street view house number dataset, consisting of over 600,000 digit images. MNIST [8] is a widely used dataset for digit recognition tasks, and is included in many introductory ML courses. Despite its relatively small size (10,000 training images and 60,000 test images), it is still used in 56% of the papers analyzed, due in part to its simplicity, which allows researchers to quickly evaluate new methods or architectures. Fashion-MNIST [189] is a relatively new alternative to MNIST appearing in 19% of the papers analyzed. It contains 10 classes of clothing items, and consists of 60,000 training low-resolution, grayscale images. It was created to provide a more challenging toy problem than MNIST while retaining similar characteristics in sample and dataset size.

TinyImageNet [190] and LSUN [191] are larger datasets, containing 200 and 10 classes, respectively. They are less commonly used, appearing in only 38% and 37% of the papers, respectively, representing in general more complex dataset to work with. TinyImageNet contains 100,000 images representing a subset of the complete ImageNet dataset. On the other hand LSUN is made of a large number of images from 10 scene classes.

Finally, Gaussian Noise and Textures [192] datasets are less commonly used, appearing in only 22% and 17% of the papers, respectively. These datasets are often used to evaluate the robustness of ML algorithms to noise or texture variations. The Gaussian Noise dataset consists of images from the ID datasets with Gaussian noise added at various levels of intensity, while the Textures dataset contains 5640 images,

each depicting one of 47 different texture categories such as brick, grass, or sand. Finally, Places365 [193] is a large-scale scene recognition dataset, consisting of over 1.8 million images in 365 categories. It is used less frequently than the other datasets, appearing in only 17% of the papers analyzed.

Interestingly, several datasets were used more frequently as OOD sources than ID sources. For example, Gaussian Noise datasets and Places365 were used as ID sources in only one paper, but are common choices for OOD sources, appearing in 14 and 11 papers, respectively. Similarly, Gaussian Noise generated datasets and LSUN were not used as ID sources in any of the papers analyzed, but were used as OOD sources in 23 and 11 papers, respectively. The vast majority of works use different datasets as sources of ID or OOD data. For example, Hendrycks and Gimpel [183] used, CIFAR-10 and SVHN as ID data, while SUN and Gaussian Noise datasets are used as OOD data. Similarly, in a paper by Liang et al. [171], CIFAR-10 and CIFAR-100 were used as ID data, while SUN, LSUN and Gaussian Noise datasets were used as OOD data.

Only a small amount of works attempt to mix classes from different datasets. The work by Roady et al. [178] represents an attempt to provide a standardized set of evaluation problems for testing the scalability of OOD detection methods. In particular, the authors evaluate the ability of OOD detection methods to scale by experimenting on two large-scale image classification datasets, namely ImageNet-1K and Places-434. They create three separate OOD problems of varying difficulty using these datasets. For the first problem, called *Noise*, they generate synthetic images from a Gaussian distribution to match the normalization scheme of training and test images. The second problem, called *Inter-Dataset*, is of intermediate difficulty and involves testing each method's ability to detect outlier samples drawn from another large-scale dataset. For the third and most challenging problem, called *Intra-Dataset* novel classes are made up of the remaining classes in each dataset. This is difficult because the image features of a class are often very similar to the features of other classes in the dataset. The authors keep the training set and models fixed across the three paradigms, but the test sets vary across them. They construct the OOD evaluation sets for each problem/dataset by randomly choosing 10,000 ID samples evenly among the ID classes and 10,000 outlier samples evenly among the OOD classes within each respective dataset validation set. Both Inter- and Intra-dataset settings, however, have conceptual or practical drawbacks: inter-dataset comparison, relying solely on the correspondence between the names of the categories between

the ID and OOD datasets, may ignore or under-estimate semantic overlap between classes from both datasets. On the other hand, intra-dataset comparison affects the training set and thus cannot be applied as is to existing pre-trained models.

4.3.2 OOD Detection: Problem Definition

In this section, the OOD detection problem is briefly introduced along with a description of the techniques included in the experiments. The OOD detection problem can be defined as the task of identifying inputs that are different from the distribution of training data. In other words, given a model trained on a specific dataset, OOD detection aims to determine if a new sample comes from the same distribution as the training data or not.

Definition of OOD Detection Let $\mathscr{D}_I$ be the probability distribution of ID data, including K_I different classes, and $\mathscr{D}_O = \{\mathscr{D}_O^i\}_{i=1}^{\infty}$ the set of distributions of OOD data. In the most general setting, at training time a dataset $\mathscr{T} = \mathscr{T}_I \cup \mathscr{T}_O$ is available, where

$$\mathscr{T}_I = \{(x_i, y_i, z_i)\}_{i=1}^{N_I},$$

$$\text{with } x_i \sim \mathscr{D}_I,\ y_i \in \{1,\ldots,K_I\},\ z_i = 0$$

is the set of ID training samples and

$$\mathscr{T}_O = \{(x_i, y_i, z_i)\}_{i=1}^{N_O},$$

$$\text{with } x_i \sim \mathscr{D}_O^j \in \mathscr{D}_O,\ y_i \in \{1,\ldots,K_O^j\},\ z_i > 0,\ j \in \{1,\ldots,J\}$$

is the set of OOD training points. In particular, each x_i is a training image drawn from a distribution, y_i is the corresponding class index, K_O^j is the number of classes modeled by distribution $\mathscr{D}_O^j$, J the number of OOD distributions represented in $\mathscr{T}_O$, and z_i the ground truth label for OOD detection (henceforth referred to as *OODness*) of the i-th sample, which is 0 if ID and higher otherwise.

The classifier is then defined as a vector function

$$f : \mathbb{R}^d \to \mathbb{R}^{K_I}, \mathbf{x} \mapsto \mathbf{l},$$

where $\mathbf{x}$ is a d-dimensional input and $\mathbf{l}$ the vector of class scores, referred to as *logits*. The model f is composed of L concatenated layers, whose output $\mathbf{l^i}$ is computed as

$$\mathbf{l^i} = f^i(\mathbf{x}), i \in \{1, \dots, L\},$$

having defined f^i as the composition of layers $1, \dots, i$. The whole classifier f then coincides with f^L, and the logits vector $\mathbf{l}$ with $\mathbf{l^L}$. Class scores $\mathbf{l}$ are converted to class probabilities $\mathbf{p}$ via the application of the softmax function

$$p_i = S(\mathbf{l})_i = \frac{e^{\mathbf{l}_i}}{\sum_{j=1}^{K_l} e^{\mathbf{l}_j}} \approx \mathbb{P}[y = i | x = \mathbf{x}], \tag{4.5}$$

where $\mathbf{l} = f(\mathbf{x})$ and y is the true label for input $\mathbf{x}$. The predicted class c is then chosen to be the one for which the assigned probability is the highest: $c = \text{argmax}(\mathbf{p})$. The network is usually trained by minimizing the cross-entropy loss $\mathscr{L}_{CE}$ between the predicted probabilities and the one-hot-encoded vector $\mathbf{y}$ representing the ground truth.

$$\mathscr{L}_{CE}(\mathbf{y}, \mathbf{p}) = -\sum_{i=1}^{K_l} y_i \log p_i = -\log p_y \tag{4.6}$$

The ability to reject undesired inputs is provided by an additional function s, that returns the OOD score $o = s(\mathbf{x})$ for sample $\mathbf{x}$. This score is then discretized by picking a suitable threshold, in order to obtain the binary prediction between ID and OOD.

$$O(\mathbf{x}) = \begin{cases} OOD & o < \theta \\ ID & o \geq \theta \end{cases} \tag{4.7}$$

One of the main differences among research works lays in the choice of the scoring function s, which can be obtained by extracting some internal network outputs, by means of an additional classifier or by explicit density estimation. The threshold θ is usually selected with the aim of optimizing a relevant metric for the specific task and dataset.

Scoring Methods for OOD Detection Although methods that use OOD samples during training have been successful, the infinite variety of possible unseen distributions poses a theoretical limitation on them. These methods tackle a classification problem with a well-defined set of outlier distributions, and increasing their diversity

may not be helpful for dealing with future observations. This chapter focuses on OOD detection methods that require minimal to no supervision, typically limited to hyperparameter selection.

Post-hoc techniques are preferred to avoid expensive training operations and to enable OOD detection on a large number of pre-existing models. These techniques can be applied to existing networks without the need for retraining from scratch. The most commonly used method as a baseline in the literature is MSP [183]. It utilizes the output of the softmax function S (4.5) to obtain an OOD score for a given prediction on sample $\mathbf{x}$:

$$o = s(\mathbf{x}) = \max_{i \in 1 \ldots K_I} S(\mathbf{x})_i$$

This technique does not require any addition to the existing network, as the softmax output is already computed to select the predicted class and is assumed to represent the prediction confidence. No training phase is necessary, and a single forward pass is sufficient for performance evaluation, as there are no hyperparameters to select.

A more recent improvement called Maximum Logit Value (MLV) uses raw network outputs instead of the normalized ones used by MSP [179]. This has been helpful in cases where applying the softmax function significantly changes the class scores.

$$o = s(\mathbf{x}) = \max_{i \in 1 \ldots K_I} \mathbf{l_i} = \max i \in 1 \ldots K_I f(\mathbf{x})_i$$

MSP can leverage the confidence scores typically given as output by neural classifiers, while MLV requires some slight changes to the code to save the logit value corresponding to the predicted class. An entire dataset can be scored in a single pass through the network.

ODIN [171] is a simple but effective method that enhances MSP by slightly altering the model workflow. It proposes two different strategies, temperature scaling and input preprocessing, that are combined, while the softmax is chosen as the scoring function. Temperature scaling involves dividing the logits by a positive integer constant T before applying the softmax, which increases the gap between in and OOD points.

$$S_T(\mathbf{x}) = S(f(\mathbf{x})/T)$$

Input preprocessing, inspired by adversarial attacks, involves adding a small perturbation to a sample $\mathbf{x}$. Each sample is propagated through the network, and the sign of the loss gradient with respect to $\mathbf{x}$ is computed and added to the original point,

scaled by a positive constant ε. This trick should move input points closer to a peak of the softmax, increasing the gap between those that were already nearby (ID) and the remaining ones (OOD). The final score for sample $\mathbf{x}$ is given by:

$$o = s(\mathbf{x}) = \max_{i \in 1 \ldots K_I} S_T(\tilde{\mathbf{x}})_i$$

Finally, OODL [194] avoids the expensive retraining of the original model f, but does not exploit its confidence scores in order to produce the OODness value. Instead, it leverages an external one-class-classifier g to decide whether a point is ID or not. Aiming to reuse as much as possible the existing model and keeping g simple, the additional classifier does not act on raw samples, but on intermediate outputs from the original network f. These network features $\mathbf{z}$ are extracted from one of the L layers of the model f, called $oodl$, and later compressed to reduce their dimensionality.

$$\mathbf{z} = f^{oodl}(\mathbf{x})$$

$$\mathbf{z} \in \mathbb{R}^{c_{oodl}, h_{oodl}, w_{oodl}}$$

$$\hat{\mathbf{z}} = \sum_{i=1}^{h_{oodl}} \sum_{j=1}^{w_{oodl}} z_{cij}$$

The resulting vector $\hat{\mathbf{z}} \in \mathbb{R}^{c_{oodl}}$ is then fed to the OOD detector g, which outputs the final score o.

$$g : \mathbb{R}^{c_{oodl}} \rightarrow \mathbb{R}$$

$$\hat{\mathbf{z}} \mapsto o$$

Following the official OODL implementation [195] a One-Class SVM can be used as an OOD Detector. The set of hyperparameters used for training includes among the others the choice of a function for kernel approximation (RBF or Nystroem) and the fraction of allowed training errors ν. Additional details on the training procedure for the OOD Detector g is are illustrated in Algorithm 1. Finally, the network layer from which features are extracted highly impacts the performance of the resulting OOD Detector. Following the approach defined by [194], candidate layers are chosen between the activation layers of a particular model. Algorithm 2 describes the process through which the optimal later is determined.

Algorithm 1 Training of the OOD Detector g

Require: training dataset features, $\mathscr{X}_{tf}$, validation dataset features $\mathscr{V}_f$

```
ss ← StandardScaler()
for data batch b ∈ X_tf do
    ss.partial_fit(b)
end for
γ ← 1/#features of X_tf
best_score ← 0
for ν ∈ {0.5, 0.1, 0.01} do
    for kernel ∈ {RBFSampler, Nystroem} do
        for average ∈ {True, False} do
            svm ← SGDOneClassSVM(ν, average)
            ker ← kernel(γ)
            for data batch b ∈ X_tf do
                x ← ss.transform(b)
                if ker not fit then
                    ker.fit(x)
                end if
                x ← ker.transform(x)
                svm.partial_fit(x)
            end for
            clf ← make_pipeline(ker, svm)
            auc ← AUROC(ss, clf, V_f)
            if auc > best_score then
                best_score ← auc
                best_clf ← clf
            end if
        end for
    end for
end for
return make_pipeline(ss, best_clf)
```

Algorithm 2 OODL Training

Require: training dataset features, $\mathscr{X}_{tf}$, validation dataset features $\mathscr{V}_f$
  $candidate\_layers \leftarrow$ {2, 5.0, 5.1, 10.0, 10.1, 15.0, 15.1, 22.0, 22.1, 27.0, 27.1, 34.0, 34.1, 39.0, 39.1, 46.0, 46.1}
  $oodl\_score \leftarrow 0$
  for $l \in candidate\_layers$ **do**
    $ft \leftarrow extract\_features\_of\_layer(\mathscr{X}_{tf}, l)$
    $fv \leftarrow extract\_features\_of\_layer(\mathscr{V}_f, l)$
    $d \leftarrow train\_ood\_detector(ft, fv)$ ▷ see Algorithm 1
    $auc \leftarrow AUROC(d, fv)$
    if $auc > oodl\_score$ **then**
      $oodl\_score \leftarrow auc$
      $oodl \leftarrow l$
    end if
  end for
  return $oodl, oodl\_score$

4.3.3 Proposed Benchmark

Several datasets with varying levels of complexity with respect to the task of OOD detection. This section introduces the research assumption and methodology used to create them.

Research Context and Motivating Example The selected benchmarks were designed having in mind, as target application, the automatic tagging of images from social media platforms such as Facebook or Instagram, with applications in social sciences and digital humanities, was considered. Specifically, research activities were carried out within the context of the FACETS[1] (acronym for *Face Aesthetics in Contemporary E-Technological Societies*) research project, carried out by a multidisciplinary team at Università degli studi di Torino with the goal of studying the meaning of the human face in relation to the fast-changing and diverse contexts in which people have been immersed since the beginning of the digital era. FRESCO (*Face Representations in E-Societies through Computational Observation*), led by Politecnico di Torino, is the quantitative branch of FACETS, with the goal of analysing profile pictures from multiple social-networks and extracting human-interpretable information. Data analytics tools and AI models can help in semiotics and social

[1]Website: http://www.facets-erc.eu/about/

studies, by uncovering hidden patterns in the data, driving the research activity on a different path that would have been otherwise neglected. To this aim, researchers may want to exploit readily available pre-trained models for automatic classification and tagging, which however requires operating under the open-world assumption. Hence, it is necessary to detect and, if necessary, exclude wrong predictions. Scene classification was selected as target application, and thus Places365 was selected as the ID dataset for the benchmark. In addition, as highlighted in Section 4.2.1, there is the need to investigate more complex settings for OOD detection, moving beyond toy problems.

Dataset Split Although the selected techniques do not require OOD samples to be available at training time, a selection of ID and OOD samples is occasionally required to perform hyperparameter selection. To avoid biases, a validation and test set was defined for every configuration for hyperparameter selection and OOD detection evaluation, respectively.

Baseline Dataset A *Baseline* dataset representing a basic level of complexity was constructed by following the most popular choice in literature, which involved choosing a different dataset as the source of OOD data. The validation split of Places365-Standard was selected as the ID data, with a total of 100 labeled images per each of the 365 categories. For OOD data, SVHN (in the multi-resolution version) was selected as it is semantically different from Places365 and representing one of the most popular choices in the literature for similar OOD experiments. To match the cardinality of ID samples (36.5K), a subset of images was sampled from SVHN test set, to integrate the 33.5K images of the training set. The stratification of the split between test and validation is done per-source to maintain the same proportions between the original datasets. On the other hand, SVHN labels were neglected and the splitting operation follows the order in which images appear in the original list. The composition of the dataset is reported in Table 4.2.

Inter-Dataset OOD Detection To further increase the complexity of the task, a second dataset named *Inter-Dataset OOD Detection* is proposed, inspired by the work of Roady et al. [178]. In this setting, a model is trained to distinguish between two distinct datasets, but the OOD images are taken from an object detection dataset

Baseline Dataset (val/test)				
	ID classes	OOD classes	ID samples	OOD samples
Places365-Standard (val)	365	0	18,250	0
SVHN (train)	0	1	0	16,701
SVHN (test)	0	1	0	1,549
Total	**365**	**2**	**18,250**	**18,250**

Table 4.2 Composition of the Baseline dataset. For SVHN samples, it is assumed to have a single category, e.g. "number", instead of the usual 10 digits.

Inter-Dataset (val/test)				
	ID classes	OOD classes	ID samples	OOD samples
Places365-Standard (val)	365	0	18,250	0
ImageNet (train)	0	968	0	18,108
Total	**365**	**968**	**18,250**	**18,108**

Table 4.3 Composition of the splits used for the Inter-Dataset OOD Detection. In this setting the task consists of discriminating among the two datasets, therefore one of them in considered entirely ID and the other as OOD. Common classes are removed from ImageNet.

spanning a large number of diverse categories that are closer to the target distribution if compared to those in the baseline dataset. In this setting, the OOD dataset is ImageNet-1K, specifically the ImageNet Large Scale Visual Recognition Challenge from 2012 [196], which includes 1.2 million training images from ImageNet and 50,000 validation samples collected from Flickr, spanning 1,000 categories. Common classes between ImageNet-1K and Places365-Standard are identified and removed, resulting in a 32 common categories removal. The validation split of Places-365 is maintained as the ID set, while the OOD samples are collected from the training set of ImageNet via stratified random sampling on all classes (except for those shared with Places365), until the cardinality of OOD and ID examples are equal. Refer to Table 4.3 for the details on the dataset composition.

WordNet ImageNet The Inter-dataset setting disregards the fact that categories from ImageNet-1K and Places365, while technically different concepts, may be highly correlated from a semantic point of view. In this case, a classifier trained on Places365 may yield entirely reasonable predictions, and thus it would be advisable to consider such samples as part of the ID distribution.

In addition, many categories can represent synonyms or subsets of ID concepts. Scene classifiers often rely on detecting objects within an image that are typically associated with a particular category. For example, an image of an empty room can still be classified as a "bedroom" if the room layout, wall color, and other contextual cues are consistent with a typical bedroom. However, the "bed" object will be present inside a "bedroom" scene in the vast majority of cases. By extension, the same image may be categorized as "bedroom" in Place365 and "bed" in ImageNet, and both classification should be considered appropriate, whereas in the standard Inter-dataset setting the "bed" class would be considered as OOD and "bedroom" as ID. Therefore, some categories from the ImageNet dataset should not be considered entirely OOD even if they are not included in the classes from Places365.

To select a suitable split, a reliable and objective method for measuring similarities between classes is proposed. The WordNet lexicon [132] can be used to link categories from different datasets by analyzing how these words are connected by semantic relations. WordNet groups nouns, verbs, adjectives, and adverbs into cognitive synonym sets (*synsets*), which express distinct concepts. Synsets are linked by conceptual-semantic and lexical relationships, making it easy to identify synonyms. Other types of relationships, such as hypernymy-hyponymy, where one synset is a more specific version of another, and holonymy-meronymy, where the meronym is a part of the holonym, are also considered in the proposed approach.

To compute similarity metrics between ImageNet-1K categories and Places365 classes, the concepts of Places365 were mapped onto WordNet synsets. While most of the mapping was automated, manual intervention was required for some scene labels that were not included in a WordNet synset. Since Places365 labels are constituted of words and not synsets, a single ID class could be correctly associated with more than one WordNet concept. To choose the best matching concepts between multiple possibilities, a distance metric between semantic concepts was necessary. Pedersen et al. [197] implemented six metrics for similarity measurement using WordNet, including Path (a baseline metric equal to the inverse of the shortest path between two concepts), Leacock-Chodorow [198], Wu-Palmer [199], Resnik [200], Lin [201], and Jiang-Conrath [202]. The first three metrics mainly depend on the shortest path among nodes of the concept tree, while the latter ones involve the information content of the nodes. The information content is obtained empirically by computing the frequency of a given word in a large collection of texts (a corpus) and can be referenced by similarity formulas. For each class in ImageNet, distances

WordNet ImageNet T40 (val/test)				
	ID classes	OOD classes	ID samples	OOD samples
Places365-Standard (val)	365	0	18,250	0
ImageNet (train)	56	944	2,800	21,332
Total	**421**	**944**	**21,050**	**21,332**
WordNet ImageNet T45 (val/test)				
	ID classes	OOD classes	ID samples	OOD samples
Places365-Standard (val)	365	0	18,250	0
ImageNet (train)	90	910	4,500	22,750
Total	**455**	**910**	**22,750**	**22,750**
WordNet ImageNet T50 (val/test)				
	ID classes	OOD classes	ID samples	OOD samples
Places365-Standard (val)	365	0	18,250	0
ImageNet (train)	140	860	7,000	25,560
Total	**505**	**860**	**25,250**	**25,560**

Table 4.4 Composition of the WordNet ImageNet datasets

from all previously identified synsets of Places365 were computed using the six aforementioned metrics. The closest synset was selected by taking the maximum value of the similarity score, separately for each metric, and min-max scaling it between 0 and 1 afterwards. After evaluating the scores, path-based metrics were found to perform better in identifying common categories and to adhere better to expectations. To account for all of them with a single score, an average of the three path-based metrics (Path, Leacock-Chodorow, and Wu-Palmer) was used as the final metric of similarity. Three threshold values of 0.40, 0.45, and 0.50, determined experimentally, were proposed for discretization between ID and OOD labels. The resulting datasets are described in Table 4.4.

FACETS OOD Detection T1 and T2 To better represent real-world data and focus on semantic content the ImageNet and SUN397 datasets were manually annotated, starting from three different datasets, with the goal of assessing whether each individual class can be considered ID or OOD w.r.t. the source distribution of Places365. The validation split of Places365-Standard was maintained for ID samples as it is considered the most reliable option due to its shared database with the training set. On the other hand, SUN397 [203] was added to the composition of the last two datasets for both ID and OOD data. This dataset is a subset of the

Scene UNderstanding (SUN) database and contains 108,754 images categorized into 397 classes. The SUN397 dataset has a strong connection with Places365, with 294 classes being shared between the two datasets, and some others being similar. Incorporating SUN397 into the composition of the datasets enabled the gathering of more samples from a different data source for the categories belonging to the ID distribution. This strengthened the evaluation capabilities of these datasets in terms of generalization. However, not all of the non-shared categories can be clearly considered OOD, and manual inspection is required to identify and match similar classes. Labelling operations of this kind are risky and can potentially introduce biases or adversely affect the performance of the detector. To mitigate this problem, classes in SUN397 are not assigned a binary label but instead given an OODness value ranging from 0 to 3 based on the following criteria:

- 0: classes from that appear in Places365 with the exact same label
- 1: classes that are semantically close to a class from Places365
- 2: classes which tend to include features typical of one or more ID classes
- 3: the remaining categories

The availability of fine-grained labelling enables experimentation with different dataset configurations. Two main versions of the dataset were investigated: T1, in which classes labelled as 0 or 1 are considered ID while 2 and 3 are considered OOD, and T2, where 0, 1 and 2 classes are considered ID and categories labelled as 3 are considered OOD.

In contrast, each category in ImageNet-1K is manually given a binary OODness label of 0 or 1, with classes that appear in Places365 or represent objects typically associated with a scene from the target dataset being assigned the label 0 (ID), 1 (OOD) otherwise. The choice of the threshold for the classes of SUN does not affect the way ImageNet classes are considered. The statistical properties of datasets T1 and T2 are reported in Table 4.5.

4.3.4 Preliminary Analysis

In the considered research setting, the underlying motivating question is not necessarily whether ImageNet-1K samples can be distinguished from Places365 images, but

FACETS OOD Detection T1 (val/test)				
	ID classes	OOD classes	ID samples	OOD samples
Places365-Standard (val)	365	0	18,250	0
SUN397	319	78	46,851	7,530
ImageNet (val)	356	644	8,900	16,100
ImageNet (train)	0	644	0	50,232
Total	**1,040**	**1,366**	**74,001**	**73,862**

FACETS OOD Detection T2 (val/test)				
	ID classes	OOD classes	ID samples	OOD samples
vv heightPlaces365-Standard (val)	365	0	18,250	0
SUN397	351	46	49,827	4,554
ImageNet (val)	356	644	8,900	16,100
ImageNet (train)	0	644	0	56,606
Total	**1,072**	**1,334**	**76,977**	**77,260**

Table 4.5 Composition of the FACETS OOD detection dataset

rather whether incorrect or misleading outputs can be discriminated from plausible predictions, irrespective of the specific set of labels used to annotate the dataset. As a preliminary step to better characterize the behaviour of scene classifier (Resnet50 pre-trained on Places365), as well as the datasets introduced in Section 4.3.3, the network predictions were analyzed to understand if specific or consistent patterns of misclassification emerged. The analysis was conducted on the OOD Detection T1 dataset, including examples from both ImageNet and SUN.

Top-5 accuracy is the most common metric used on Places-365. However, when dealing with potentially OOD samples, classification accuracy cannot be applied because the target label sets are generally different from the predicted labels. Therefore, to obtain a similar measurement for OOD detection datasets, a slightly different approach must be followed. Specifically, the present analysis deals with the question *"how similar is the predicted class to the correct one?"*. To address this question, a strategy similar to the one used to label the WordNet-ImageNet datasets is adopted, which involves exploiting semantic similarity between class labels.

Similarly to the mapping procedure described in Section 4.3.3, a mapping was performed for SVHN and SUN397 datasets to ensure methodological consistency. Only one class is available for SVHN, and it was manually mapped to *digit.n.01*. For SUN397, the associations were performed in a partially automated way, following the same procedure used for Places365. This process was eased by the large number

of shared classes between the two datasets, allowing to re-use 294 out of the 397 required mappings, as these categories are shared with Places365. Algorithm 3 shows how the similarity score was calculated given the ground truth class and the predicted class, using the average among Wu-Palmer and Path similarity as the metric. Other similarity measures were discarded due to their unbounded nature, which could make them prevail over those limited between 0 and 1.

Algorithm 3 Prediction similarity computation

Require: ground truth class gt_label, predicted class $pred\_label$
  $gt\_synsets \leftarrow get\_synsets\_for\_class(gt\_label)$
  $pred\_synsets \leftarrow get\_synsets\_for\_class(pred\_label)$
  $max\_sim \leftarrow 0$
  for $s1 \in gt\_synsets$ **do**
    for $s2 \in pred\_synsets$ **do**
      $sim \leftarrow \frac{Wu\_Palmer\_similarity(s1,s2)+path\_similarity(s1,s2)}{2}$
      if $sim > max\_sim$ **then**
        $max\_sim \leftarrow sim$
      end if
    end for
  end for
  return max_sim

The newly defined scoring function was utilized to conduct multiple aggregated measurements aimed at evaluating the behavior of the pre-trained model. The average similarity between the ground truth labels of each OOD sample from the original dataset and the predicted classes from the classifier trained on Places365 is reported in Fig. 4.5. The plot is restricted to the 10 best/worst performing classes for visualization purposes. Interestingly, no class from the Places365 validation set appears among the top 10 best performing categories, underlining a strong semantic similarity between SUN and Places365 categories. ImageNet classes are in general the ones which perform the worst, as the semantic similarity between an object and its background is usually low.

Likewise, the 10 best/worst ID classes (i.e., from the Places365 dataset) based on the average similarity between the predicted ID label and the ground truth are presented in Fig. 4.6. As an example, the *conference_room* class from Places365 shows a high similarity w.r.t. the ground truth labels of the samples that were assigned such label, meaning that the model is assigning to OOD samples a category that is semantically near to the ground truth. In general, it appears that indoor and

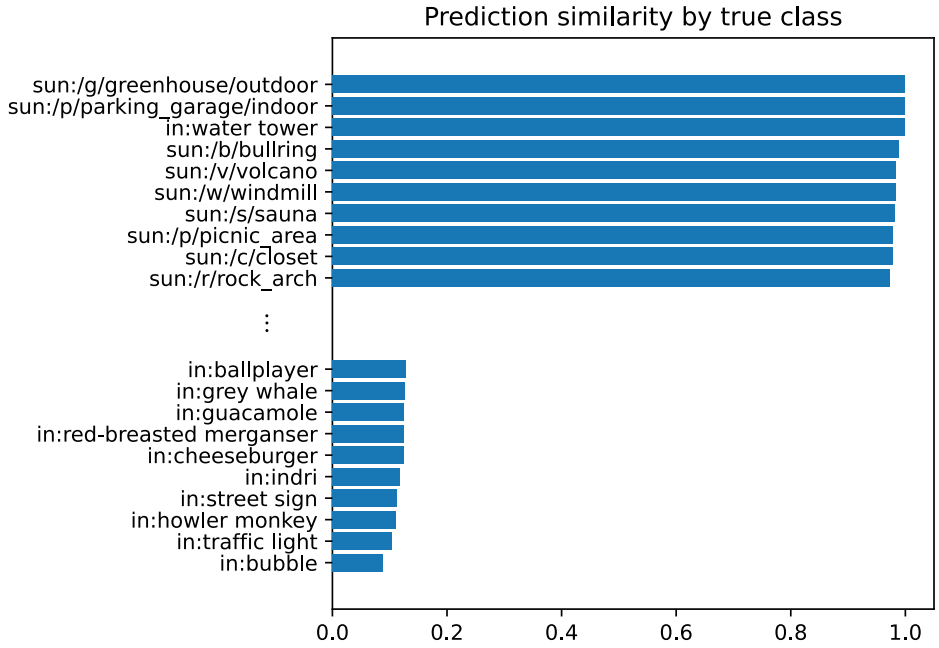


Fig. 4.5 Average semantic similarity between the ground truth category and the predicted ID labels, as computed on the validation split of the FACETS OOD Detection T1. SUN ground truth labels (*sun* prefix) are generally semantically similar to the predicted ID class if compared to ImageNet ground truth labels (*in* prefix).

man-made environments are more likely to be assigned plausible labels whereas the same is not true for natural scenes. There could be several explanations for such a phenomenon: artificial settings may be less ambiguous in terms of labelling with respect to landscapes, in which multiple categories could be represented (e.g. hill, sky or forest).

An additional way of visualizing the current network behaviour without relying on the described similarity metric is to model relationships as a directed graph. Each class is therefore represented as an independent node, and an edge going from node a to node b indicates that at least one image belonging to class a was predicted as class b. The analysis of graphs can represent a powerful technique to spot macroscopic tendencies in the classification. In order to incorporate the additional information of frequency and relevance of every association, the weight of each edge in the graph was assigned to represent the strength of the bond between the two nodes. Given a set of predictions $\{(y_i, \hat{y}_i, o_i)\}_{i=1}^{N}$, where a sample belonging to class y_i is predicted

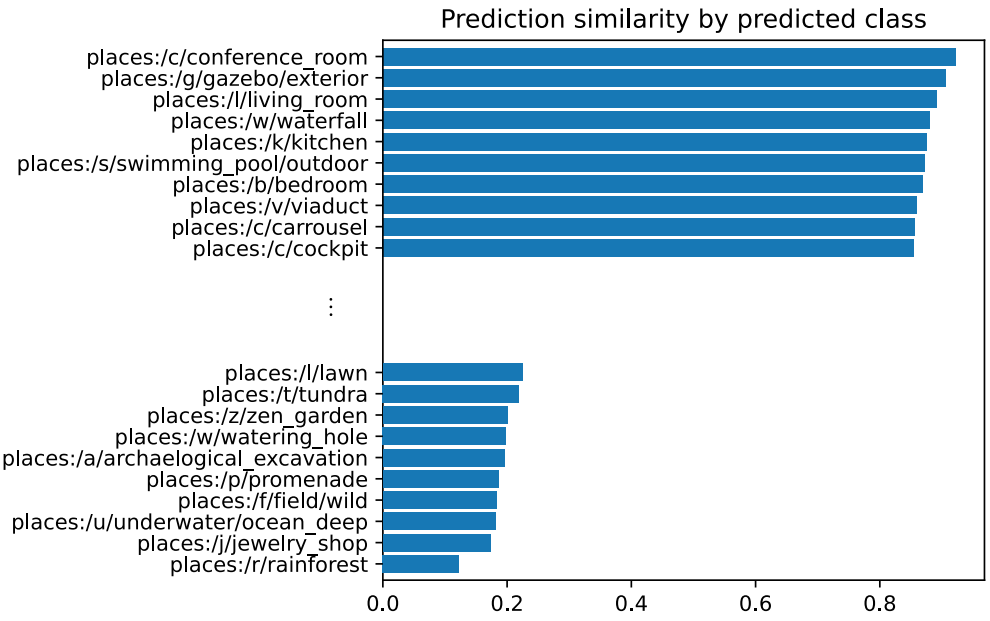


Fig. 4.6 Average semantic similarity between the and the predicted ID labels, and the OOD ground truth classes, as computed on the validation split of the FACETS OOD Detection T1. Man-made environments seems to be less ambiguous and are most likely to be semantically similar to the OOD ground truth classes.

as class $\hat{y}_i$ with an OOD score o_i, the the weight of the edge from node a to node b has been computed as:

$$w_{ab} = \frac{\sum_{i=1}^{N} \mathbb{1}[y_i = a, \hat{y}_i = b] o_i}{\sum_{i=1}^{N} \mathbb{1}[y_i = a]}$$

In particular, w_{ab} represents the sum of the prediction scores scaled by the number of samples of the true class.

However, the raw graph for FACETS OOD Detection T1, obtained using MSP as OOD detection score, is very large, including $1,760$ nodes and $41,397$ edges. Several pruning operations were performed to remove uninformative information, such as obvious links (correct classifications, obviously ID categories, etc.) and noise (rare and low-scored associations), in order to allow for easier niche inspections of the graph. This was achieved by filtering self-loops (correct classifications that are not informative for OOD detection), as well as any other link among obviously-ID samples. For this reason, edges whose source node belongs to Places365 (val) were removed and the remaining ones always have source and destination class

belonging to different datasets. Several other associations could easily be identified as correct, such as links among classes that have the same name (e.g. *sun:/f/fountain*, *in:fountain* and *places:/f/fountain*) and thus removed. Finally, edges with low weight were filtered out, as they were likely not carrying any useful information while significantly increasing the complexity of the graph. On top of these operations, nodes whose updated degree is zero were removed. The resulting graph (reported in Fig. 4.7), visualized using the open source software Gephi [204] is considerably smaller, having 989 nodes and 1022 edges. In the pruned version of the graph, several isolated clusters, representing independent semantic groups were identified and are reported in Fig. 4.8.

Secondly, several edges with a large weight emerged from the inspection of the graph. Many of them represent links that were somehow neglected by the simple algorithm used for matching class names. For instance, an underscore instead of a white space or a different word order are the sufficient conditions for two classes to be considered different (see Fig. 4.9).

Several others groups model meronym-holonym or scene-object relationships, which are not easily captured by the WordNet similarity metrics largely adopted in this work. Some examples are observable in Fig. 4.10.

Additionally a large cluster, mainly composed of classes representing animals, can be identified by inspecting the graph. The two nodes that tend to draw a large portion of the arrows are *veterinarian_office* and *underwater/ocean_deep*, respectively for terrestrial and aquatic species. Other categories that appear to be frequently selected as output when classifying animals are: *aquarium*, *field/wild*, *watering_hole*, *lawn*, *kennel/outdoor*, *pet_shop*, *tundra* and *rainforest*. While all of the previous are acceptable scenes for an animal, and very often the species matches the habitat, the strength of some relations suggests possible network biases. The fact that most dogs and domestic species are associated with *veterinarian_office* or *kennel* while their typical environment should be a house or a garden suggests that the model is likely to classify most scenes where a pet is present without focusing enough on the background, possibly due to the significantly larger presence of pets in *veterinarian_office* with respect to the other classes in Places365.

Finally, undesirable behaviour emerge from clusters of unrelated concepts. Fig. 4.11 represents two clusters which are very likely originated by visual rather

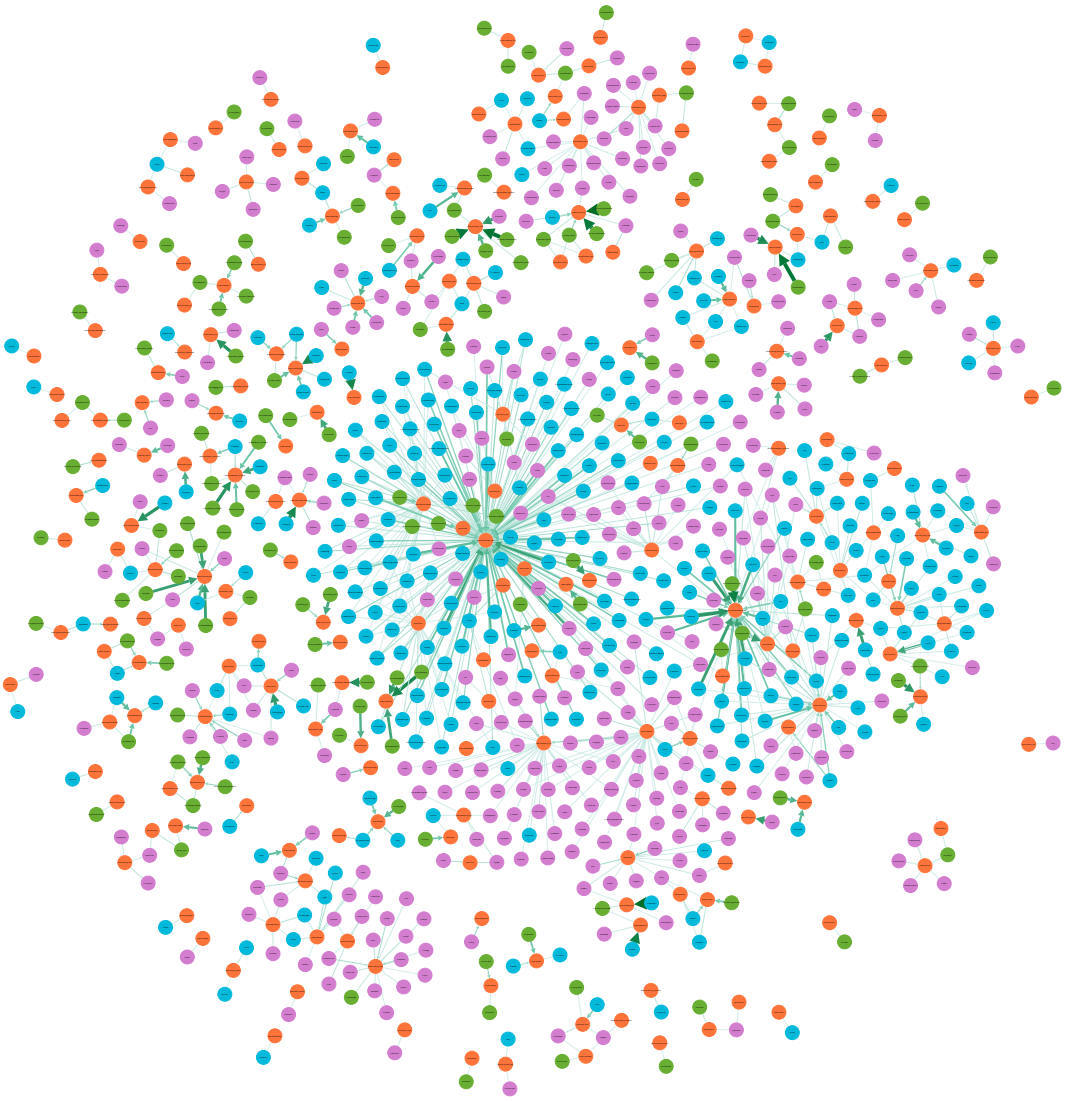


Fig. 4.7 Pruned version of the original graph, with no Intra-Dataset, obviously correct nor noisy edges. Width and darkness of a link are proportional to the weight. Each node's position is determined by applying the ForceAtlas2 algorithm for graph visualization [205].

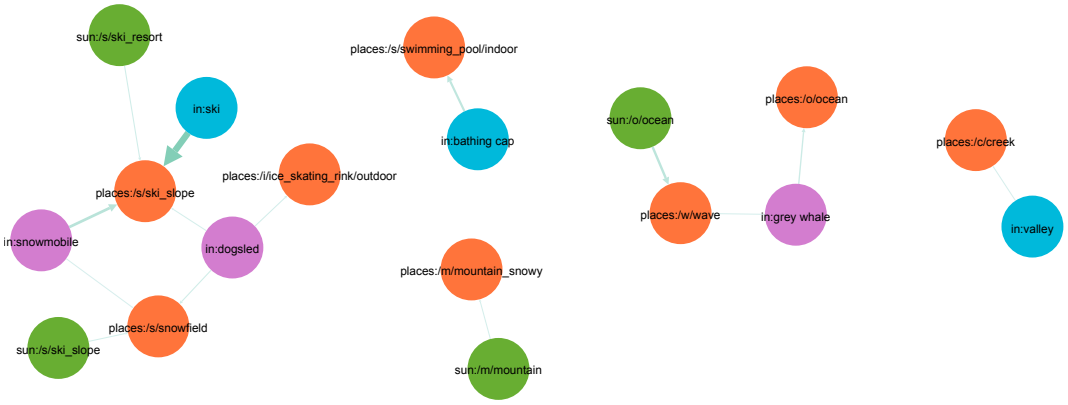


Fig. 4.8 Examples of isolated clusters consisting of few nodes.

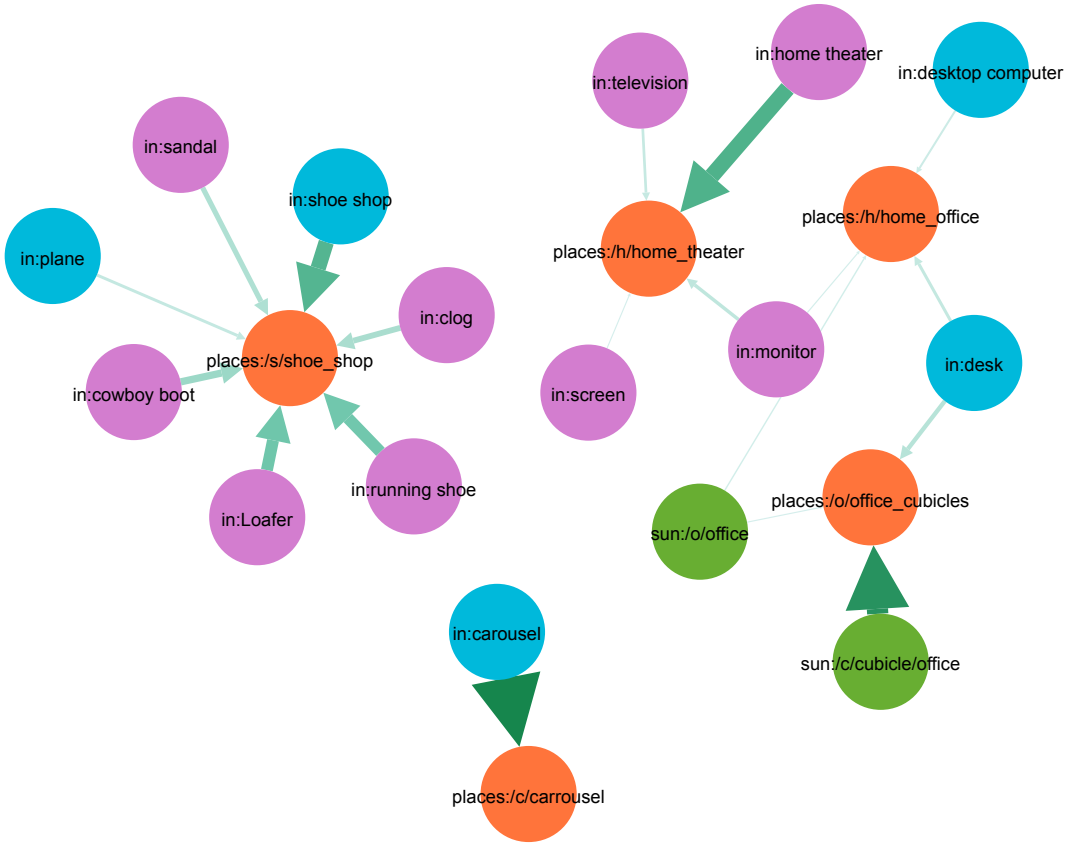


Fig. 4.9 Examples of strong edges between classes representing the same concepts with slightly different names. The underscore prevented *shoe shop* and *home theater* to be paired with their counterparts, whereas different wording or spelling were responsible for mismatches in the case of *cubicle/office* and *carrousel*.

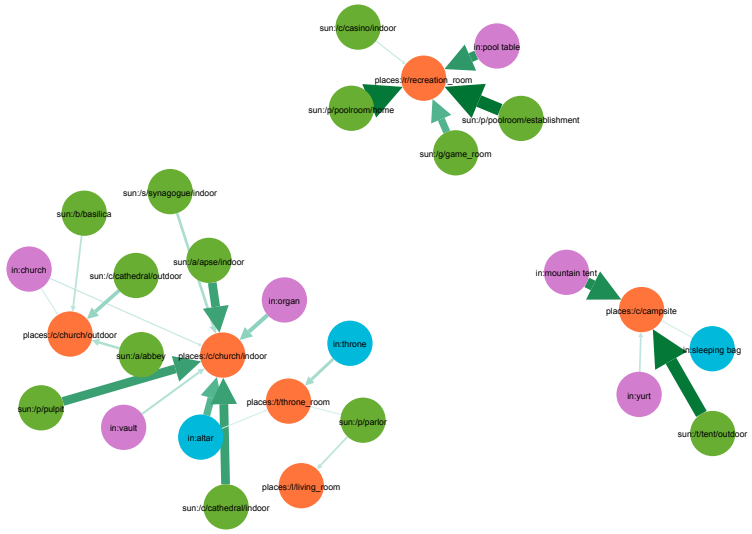


Fig. 4.10 Examples of strong edges between different concepts, linked by meronym-holonym and similar relationships.

than semantic similarity: towers, lighthouses and telescopes are grouped due to their similar shape despite their different use.

Fig. 4.11 Examples of network biases that affect classification results.

Features Extraction Examining the computed features at different layers in the network can also be useful for interpreting the performance of OOD detection techniques. To this extent, the validation splits of the datasets were chosen to perform a visualization of the features, as extracted by the pre-trained model at several layers. The visualization of the features was done using the T-SNE technique on a

random sample of $10,000$ observations. Fig. 4.12, 4.13 and 4.14 show the resulting distributions for some of the datasets considered. The remaining ones are omitted for simplicity. As expected, the baseline setting presents a clear distinction between ID and OOD data at most network levels, thanks to the different characteristics of the two adopted datasets. The remaining OOD detection settings are significantly harder, as all samples depict real-world objects or scenes and are highly interleaved in feature space.

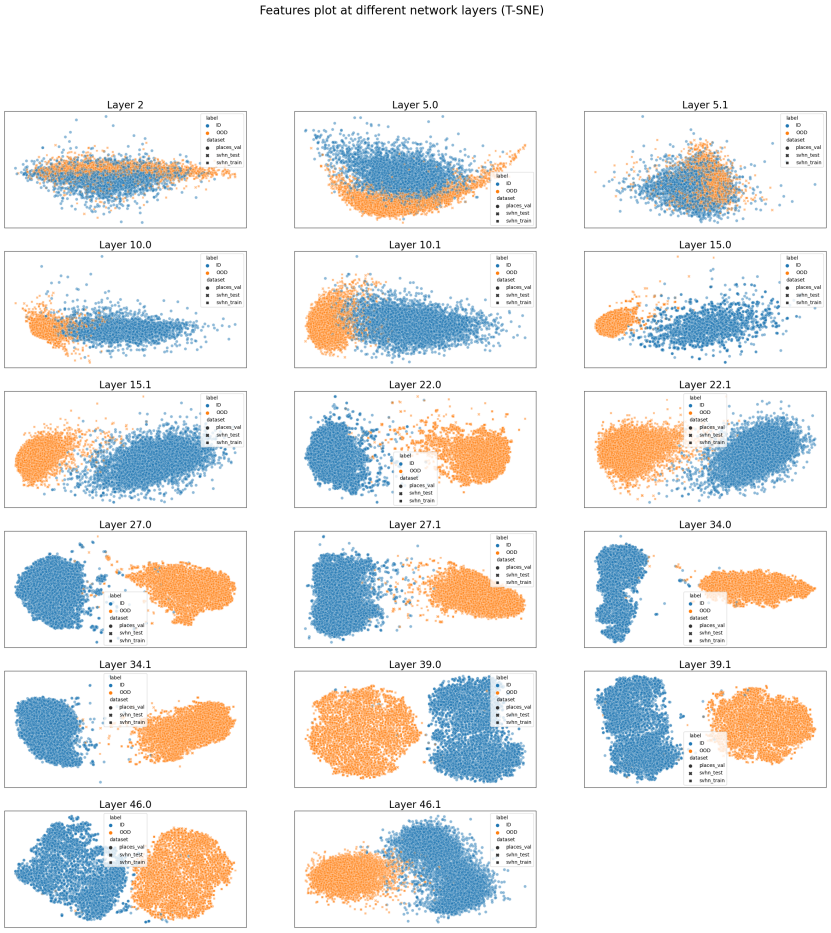


Fig. 4.12 T-SNE visualization of the features extracted from the Baseline dataset at different network layers.

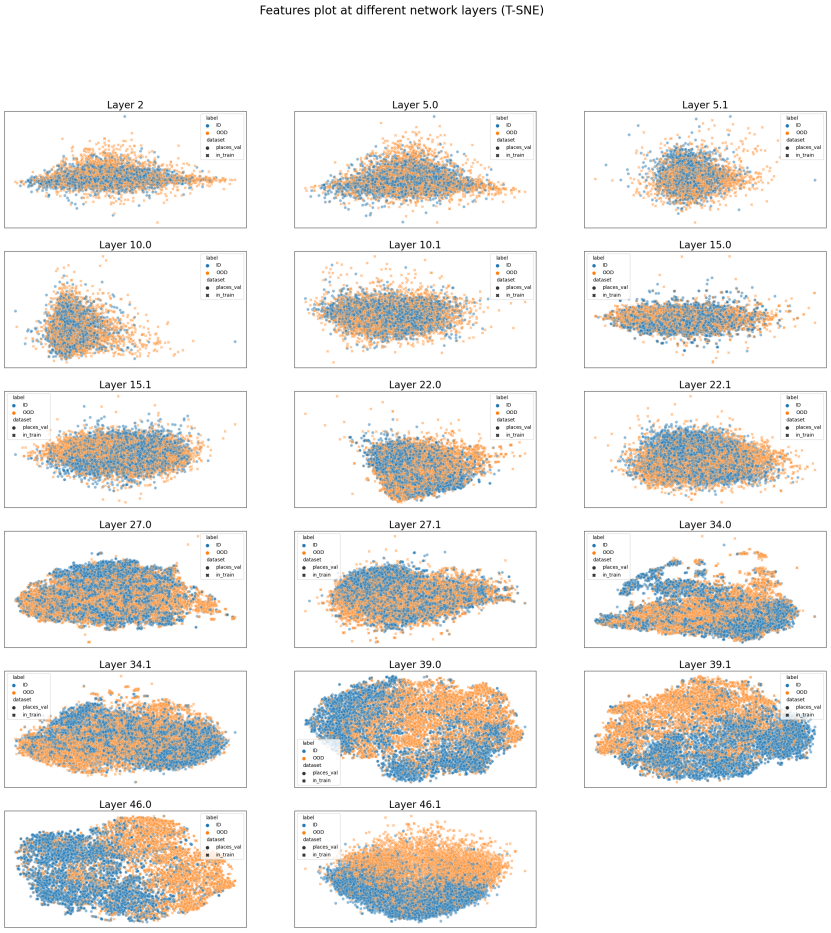


Fig. 4.13 T-SNE visualization of the features extracted from the Inter-Dataset OOD Detection dataset at different network layers.

4.3.5 Experimental Settings

In this section, the implementation and evaluation of the selected OOD detection techniques is detailed.

ODIN and OODL Evaluation Although the original ODIN paper provides hyper-parameter values that best suit the ODIN technique on the analyzed datasets [171], these values may differ from the ideal ones for the network and data used in the current work. Therefore, a validation procedure was conducted to determine the

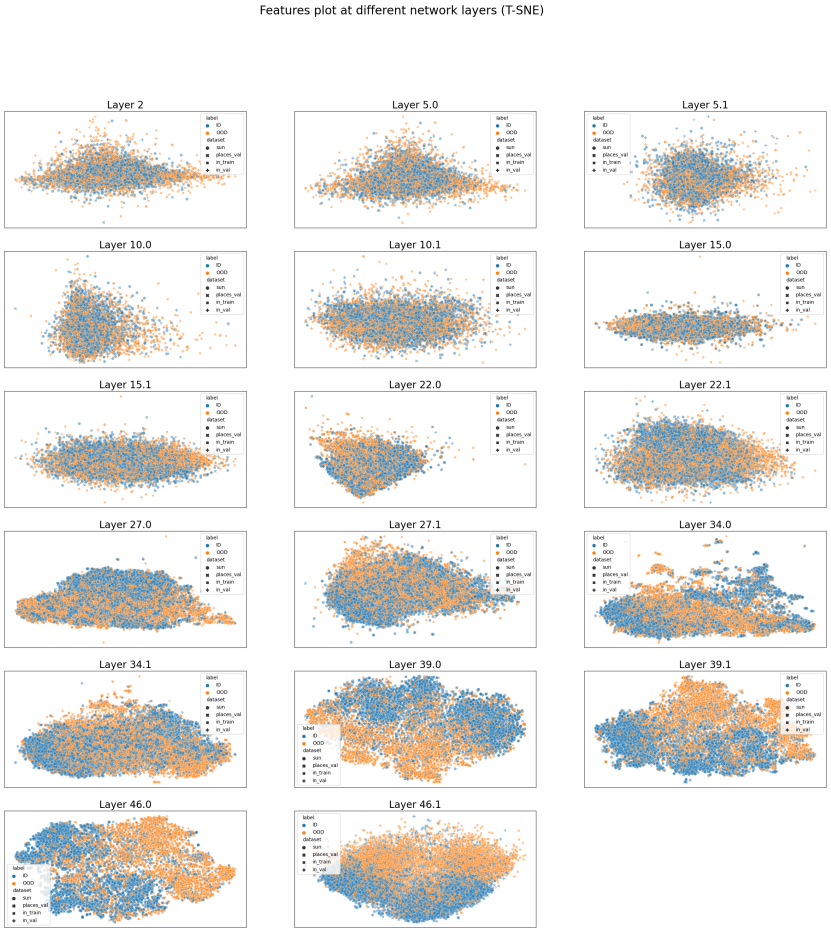


Fig. 4.14 T-SNE visualization of the features extracted from the FACETS OOD Detection T1 dataset at different network layers.

optimal set of values for ε and T. Each hyperparameter was varied while keeping the other one fixed. Specifically, ε was tested on linearly spaced values in the range [0.0002, 0.004] with step 0.0002. On the other hand, T was chosen from the set of values {2, 5, 10, 20, 50, 100, 200, 500, 1000}. Experimental results (see Fig. 4.15) demonstrate the effectiveness of temperature scaling for OOD detection, even though the performance did not improve for high values of T. This is not true for the perturbation magnitude, as performance appeared to consistently worsen with an increase in the value of ε, with the exception of the Baseline dataset. This difference may be attributed to the nature of the Baseline dataset. As the dataset is composed

of Far-OOD samples, it is likely that the OOD samples are already far from regions where the softmax is very high and are marginally affected by input preprocessing. Conversely, if an input sample is very similar to the ID classes, its gradient could be highly influenced by the presence of several local maxima. The perturbed point might end up moving in the wrong direction or overshooting the desired region. Furthermore, none of the selected values for the hyperparameter ε outperformed the setting where no input preprocessing was used.

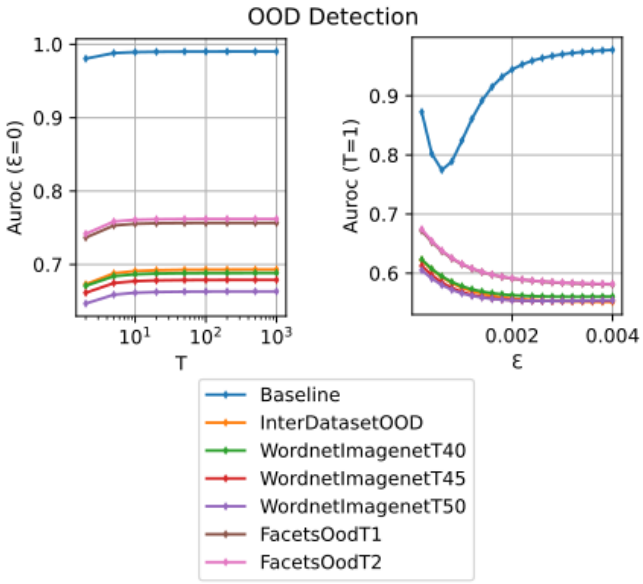


Fig. 4.15 ODIN validation procedure. AUROC of the OOD detection performance when varying the temperature T and the ε hyperparameters.

In the following set of experiments, three different methods derived from the ODIN algorithm will be considered. The first method is called TS (Temperature Scaling), which will be evaluated in isolation. The best hyperparameters combination resulting from the previous validation procedure will be used, which are $T = 1000$ and $\varepsilon = 0$. The second method is the full ODIN algorithm with the parameters selected by its authors and commonly used in literature, which are $T = 1000$ and $\varepsilon = 0.0014$. This method will be tested for comparison purposes. The third method is a mixed solution named IP TS MLV (Input Preprocessing, Temperature Scaling, Max Logit Value). This approach combines MLV with ODIN by using the maximum logit of the perturbed sample, scaled by the temperature parameter $T = 1000$, as the OOD detection score. The preprocessing magnitude is $\varepsilon = 0.0014$.

OODL requires a more complex procedure in order to be adapted to the dataset and model it will be evaluated on. Two main steps can be identified: learning a decision function and applying it to the model. The original OODL paper [194] does not provide any detailed information on how the detector is implemented, besides the choice of the One-Class SVM and the presence of the hyperparameters k (the chosen kernel function) and ν (fraction the allowed training errors). The official code ([195]) reveals that these hyperparameters are fixed to $\nu = 0.001$ and k = 'RBF'. The other hyperparameters of the OODL implementation are the followings:

- Average: parameter of the *SGDOneClassSVM*, when *True* averages the model coefficients over different calls to partial_fit instead of replacing older ones. Values assumed: {*True*, *False*}
- Kernel Approximator: the function used in order to approximate a Radial Basis Function, required in order to apply the kernel trick to the otherwise linear *SGDOneClassSVM*. Values assumed: {*'Nystroem'*, *'RBFSampler'*}
- γ: parameter of the RBF kernel, needed by both the aforementioned approximators. Its value is fixed to the reciprocal of the number of features, the same solution implemented as *'auto'* in *OneClassSVM*.
- ν: parameter of *SGDOneClassSVM*, fixes the ratio of training samples that can be incorrectly classified, selected among $\{0.01, 0.1, 0.5\}$.

For each of the 12 possible sets of hyperparameters values a Support Vector Machine is trained and tested on the features extracted by specific layers of the model on the validation dataset. Eventually, the model yielding the highest AUROC is selected and returned as the resulting classifier. To this extent, it is required to determine the layer from which features are extracted. AUROC scores for each layer were evaluated experimentally and are reported in Fig. 4.16. The optimal OOD detection layers, together with the hyperparameters of the corresponding OCSVM were collected for each dataset. Results are shown in Table 4.6.

These results are in accordance with two findings from [194]: the optimal layer tends to be one of the early modules of the network, as deeper ones are more focused on class separation, and does not depend on the choice of the OOD dataset. Both of these properties are preserved when moving from the small datasets tested by the authors to more complex ones such as ours, the only exception being the baseline.

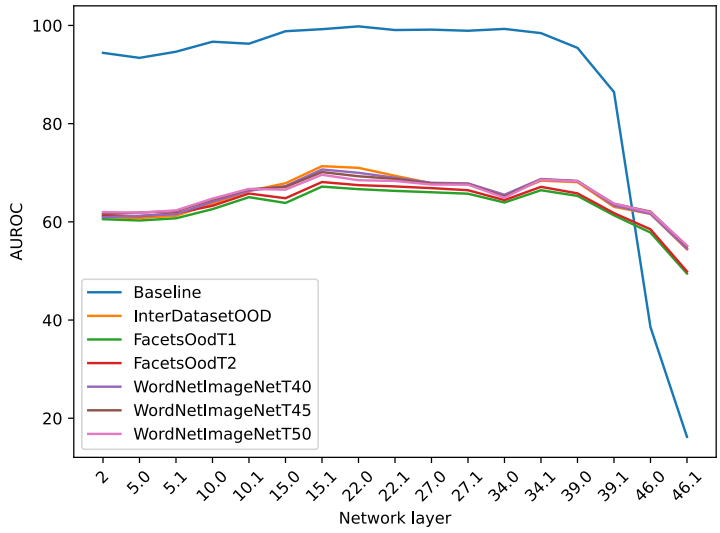


Fig. 4.16 OODL validation AUROC for each candidate layer.

OOD Dataset	Layer	Average	Kernel approximator	ν	γ
Baseline	22.0	False	Nystroem	0.5	0.0078125
InterDataset OOD Detection	15.1	True	Nystroem	0.01	0.0078125
WordNet ImageNet T40	15.1	True	Nystroem	0.01	0.0078125
WordNet ImageNet T45	15.1	True	Nystroem	0.01	0.0078125
WordNet ImageNet T50	15.1	True	Nystroem	0.01	0.0078125
FACETS OOD Detection T1	15.1	True	Nystroem	0.01	0.0078125
FACETS OOD Detection T2	15.1	True	Nystroem	0.01	0.0078125

Table 4.6 OODL Validation results

Evaluation and metrics The aforementioned methods for OOD detection, with the hyperparameters and configuration resulting from the validation step, were evaluated on the test splits of the datasets described in Section 4.3.3. The results of the OOD detection are reported by means of three scoring functions, namely FPR@95TPR ↓, Detection Error ↓ and AUROC ↑. The arrows describe whether a higher (↑) or lower (↓) value is an indicator of a higher performance.

4.3.6 Results

Performance of the selected OOD detection techniques are shown in Table 4.7, and the best score for each dataset is highlighted in bold.

	FPR@95%TPR ↓ / Detection Error ↓ / AUROC ↑					
	MSP	TS	MLV	ODIN	IP TS MLV	OODL
Baseline	22.36/13.68/94.79	4.50/ 4.75/99.02	4.50/ 4.74/99.02	10.34/ 7.67/97.82	10.34/ 7.67/97.82	**0.37/ 1.61/99.81**
InterDataset OOD Detection	86.85/45.91/64.40	79.69/42.34/69.79	**79.67/42.33**/69.79	84.43/44.72/65.73	84.46/44.72/65.74	80.83/42.92/**71.30**
WordNet ImageNet T40	87.22/46.11/63.99	81.28/**43.14**/68.91	**81.27/43.14**/68.91	85.27/45.13/64.84	85.26/45.13/64.85	82.06/43.53/**70.93**
WordNet ImageNet T45	87.64/46.32/63.44	82.16/**43.58**/68.05	**82.15/43.58**/68.05	86.44/45.72/63.80	86.43/45.71/63.81	83.19/44.09/**70.24**
WordNet ImageNet T50	88.66/46.83/62.28	**85.30/45.15**/66.29	**85.30/45.15**/66.29	88.14/46.56/62.30	88.14/46.56/62.30	85.37/45.18/**69.54**
FACETS OOD Detection T1	82.64/43.82/69.56	**74.19**/39.60/**75.87**	**74.19/39.59/75.87**	79.11/42.05/72.39	79.10/42.05/72.40	82.24/43.62/67.49
FACETS OOD Detection T2	82.44/43.72/69.83	**74.01/39.51/76.38**	74.02/**39.51/76.38**	78.92/41.96/72.70	78.93/41.96/72.70	81.49/43.25/68.43

Table 4.7 OOD detection results: the best score is highlighted in bold. Specifically, for each dataset, the OOD detection techniques that score the smallest FPR@95%TPR, the smallest detection error, and the largest AUROC are highlighted. If multiple techniques have equal performance, both scores are highlighted in bold.

Differences among datasets are evident: most methods are able to deal with the Baseline providing good results, but struggle as the complexity of the problem increases. The first sharp decrease is observable when moving to InterDataset OOD Detection, as even the basic statistics of ID and OOD images start to become very similar. The need for discrimination based on the semantic content further affects the ability of the model to detect outliers, as proven by the worse performance on the WordNet ImageNet datasets. This is even clearer when an additional data source and more categories are added, such as with FACETS OOD Detection datasets. These behaviours are particularly evident from the scores achieved by OODL, considering it is the only method to rely on a decision function which is specifically trained on each dataset.

The MSP technique proves to be a solid baseline, but is generally outperformed by more complex solutions. Test results confirm the disadvantage for methods relying on input perturbation: ODIN and IP TS MLV do not improve over their basic versions, TS and MLV. OODL appears to be a suitable solution for simpler datasets, especially in terms of AUROC. Classifier-based scores such as TS and MLV, however, tend to largely outperform the SVM on the FACETS OOD Detection datasets, providing even better results with respect to simpler benchmarks like WordNet ImageNet.

The difference with respect to the adopted method could be due to the difficulties faced by SVMs when the required decision boundary becomes very complex, while the one with respect to WordNet datasets is likely to be a consequence of the manual labelling: it is possible that some of the labels pander to existing network biases,

rewarding it for its mistakes rather than applying penalties. For instance, most of the categories representing aquatic animals and fishes in ImageNet are considered ID in the manual labelling procedure, as they are typically depicted in environments like *ocean_deep*, *aquarium* or *coral_reef*, which are represented in Places365. At the same time, since these species are usually associated with the aforementioned scenes, the classifier might tend to use them as a shortcut to identify them. As a consequence, test images representing one of these animals in an unusual context or on a blank background are wrongly labelled as ID, but are likely to be given high scores by softmax and logit-based methods. This is considered to be a correct OOD detection result and improves the performance metrics accordingly, however it is the effect of both the ground truth label and the classifier output being wrong.

Finally, the reliability and robustness of the model pre-trained on Places365 in rejecting strongly misclassified ID samples was investigated. An important factor in the choice of a method over the others can therefore be the performance in detecting ID data, labelled with the negative class when wrongly classified and with the positive when assigned to the correct category. Results are reported in Table 4.8.

FPR@95%TPR ↓ / Detection Error ↓ / AUROC ↑					
MSP	TS	MLV	ODIN	IP TS MLV	OODL
76.33/40.66/77.74	81.92/43.45/70.84	81.91/43.45/70.84	81.83/43.41/71.23	81.84/43.41/71.23	96.40/49.97/47.24

Table 4.8 Misclassification detection results on Places365-Standard (val)

While MSP was proven to be suboptimal when used for OOD detection, it is by far the best of the evaluated methods in the misclassification detection scenario. The softmax approach can in this case benefit from the lack of unknown categories, which means that its main hypothesis (the *closed world assumption*) is met. Indeed, the normalization step forces the total probability of the output classes to be 1, enhancing the confidence estimates for ID samples although disregarding any additional concept and negatively affecting OOD detection capabilities. Methods specifically designed to identify outliers do not shine in this setting, especially OODL which was trained in a discriminative way using a different kind of labelling. It is then predictable (and desirable) for it to be so far from MSP in terms of performance, even achieving sub-random AUROC scores.

4.3.7 Discussion

As shown by [175], different OOD detection techniques do not behave consistently: each of them might be better suited for a given type of dataset and not able to generalize to every possible novel distribution. In particular, distance-based methods are effective on outliers that are far from the ID data, while softmax-based ones are better when OOD samples are close to the decision boundaries. This kind of behavior can be observed on the custom datasets (Table 4.7): the one-class SVM outperforms the remaining techniques on the Baseline dataset, which presents well-separated clusters (Fig. 4.12). However, it starts to struggle when ID and OOD data are overlapped (Fig. 4.14).

Several past works have addressed issues that are combined in the setting under consideration: large datasets [178], real world images [177], large number of classes [180], high inter-category similarity and semantic content as OOD detection criterion [181]. The overlapping of different problems is likely to have been the main cause of misalignment between the results and previous findings. For example, with respect to ODIN, it was demonstrated that input preprocessing had a negative impact on most of the datasets in this study, while being beneficial for the Baseline and the approach used in [171]. On the other hand, although heavily modified from its original implementation, OODL [194] matched the expectations by confirming the intuitions that motivated its authors and by outperforming ODIN in most OOD detection occasions, especially in terms of AUROC. Despite the fact that the results obtained in the study are significantly worse than what is typically achieved by the same methods on simpler datasets, they are still better than those obtained by [178] in a very similar scenario: Table 4.9 compares the AUROC scores for Inter-Dataset OOD detection (Places vs ImageNet).

Current Work		Roady et al. [178]	
ODIN	OODL	ODIN	OCSVM
65.73	71.30	49.9	62.4

Table 4.9 Comparison of AUROC scores for ODIN and OCSVM applied to the Places vs ImageNet task (Inter-Dataset OOD Detection), as reported by [178] and in this study. Although the dataset was inspired by [178], it is important to note that the implementation of both the dataset and the detection techniques differs. For instance, Places365 was adopted as the ID, while Places434 was originally used by the authors.

4.4 Concluding Remarks

Dataset drift, or changes in the distribution of the input data, is a common and challenging problem in the field of ML and DL applications [153]. These changes can significantly degrade the performance of ML/DL models, making it difficult for practitioners to adopt these blackbox models in practical settings. One cost-effective solution for addressing dataset drift is the detection of changes in the incoming data distribution, which can be achieved by monitoring the distribution of features learned by the model itself [170, 163]. This approach is computationally efficient, easy to deploy, and does not require continuous labeling of data. The case study described in this work demonstrates that there is a strong correlation between performance degradation and the distance from a reference feature distribution. It was found that stronger performance deviation corresponds to larger distances [155]. Future work will involve extending the comparison to different types of CDTs with the aim of further improving the detection accuracy, as well as investigating the generalizability of this approach to different models and applications under more complex drift patterns and scenarios. These findings have practical implications for MLOps [170, 163]. The proposed strategy is readily applicable to models that are developed in-house and to open-source implementations. However, many companies rely on models that are trained by third-party and commercial providers, which do not typically enable access to inner model features for various reasons, including intellectual property concerns [155]. In such cases, end-users of blackbox models could resort to training an auxiliary model, such as an auto-encoder [170, 163], to learn the data distribution before or when the model is put in production. It should be noted that this approach incurs additional costs and the features learnt by auxiliary models could differ from those learnt by the prediction model. Finally, it was also found that the use of synthetic data has the potential to be a powerful tool for benchmarking concept drift detection algorithms and tailoring them to specific use cases [155].

The work presented in the second part of the Chapter focused the problem of OOD detection applied to a scene classifier pre-trained on Places365, evaluating several techniques commonly adopted in past literature. The main difference with respect to previous comparisons lies in the selection of benchmarks: the most common choices in literature, such as CIFAR10, CIFAR100 or TinyImageNet, have been evaluated as not sufficiently representative of a realistic scenario and were thus

superseded by custom datasets. Similarly to [181] and [182], the focus was brought to the semantic content of the images rather than their appearance or the source they were taken from. Differently from their approach of manually selecting images or categories that are not at risk of semantic ambiguity from a subset of the original data, each class was labeled as ID or OOD, taking into account the entirety of the data. Furthermore, the feasibility of automating this operation was evaluated using the WordNet database and concept similarity, leading to the three WordNet-ImageNet datasets.

Some experimental choices were driven by practical convenience or constraints: for instance, the focus on OOD detection techniques that do not require re-training the classifier was motivated by the opportunity to use pre-trained models made available by third-parties. However, significant room for future developments is open. The main limitation of the proposed experiments is the adoption of a single ID distribution and model. Testing the same procedure on a different setting would strengthen and complement the results. When defining the datasets on which the analysis was carried out, aiming to develop a more fine-grained labelling than the typical ones (which are based on the data source), each class was declared ID or OOD. While this is clearly a step in the right direction, it is far from being optimal. Especially when dealing with object-centric data, images belonging to the same category might depict highly diverse scenes, meaning that some of them should be considered ID and others OOD. For instance, a real-world picture of a chair in a living room would be ID, while a chair on a blank background is obviously OOD. Manually inspecting every image is not feasible, thus the operation must be automated. Hendrycks *et al.* [206] proposed an adversarial procedure to separate ID and OOD images, however this is likely to introduce a strong bias that prevents the same network to be used at test time. A possible alternative would be to run an image captioning model and later measure the distance between the captions and the ID labels. To measure instead the distance among concepts, the change that would improve the most the results of this work is probably the selection of a better metric with respect to the synset similarity that was adopted. As already discussed, similarity is not able to model several types of semantic relationship which are instead highly relevant when assessing closeness between ideas. Improving the distance metric will have consequences on the labelling of WordNet-ImageNet datasets, on the Prediction Similarity values and on the comparisons between concept closeness and output

OOD scores. Additional benefits could come from testing other OOD detection methods, as well as investigating an ensemble of metrics.

Chapter 5

Conclusions

This document outlines research conducted during my Ph.D. path aimed at developing innovative solutions to enhance the robustness and interpretability of AI in the field of CV. Specifically, novel data-centric approaches are presented to preprocess, maintain, and monitor data quality, as well as extended methods for model interpretability to offer more comprehensive and user-friendly explanations. Providing more reliable and interpretable AI models can pave the way for wider adoption and implementation of AI technologies in various industries, potentially leading to significant advancements and improvements. Ensuring the trustworthiness of AI models can help mitigate the risks associated with biased or erroneous decision-making, enhancing their credibility and acceptability. Ultimately, advancing the state-of-the-art in data-centric solutions for AI can unlock the full potential of this transformative technology while maintaining a high level of accountability and transparency.

The content of this document has been organized to reflect the stages of a typical DL pipeline, with each part building upon the previous one to achieve a more robust and interpretable AI model. The first part focuses on data preprocessing and domain understanding; this is a crucial step that involves collecting, cleaning, and labeling data to ensure its quality and reliability. Moving on from data preprocessing, the next stage involves applying XAI to gain a better understanding of the inner workings of the AI model, understand its strengths and weaknesses, and improve its interpretability. Finally, the last part of the pipeline involves data monitoring, which is an essential aspect of maintaining the quality and reliability of AI models. With

the increasing complexity of AI models and the vast amount of data that they operate on, it is critical to monitor the model's performance continuously.

Chapter 2 explored the integration of causal models in DL as a data-centric systematic approach to document and identify potential dataset biases that could affect the training and deployment of neural networks [28]. The methodology was applied to a real-life example from an industrial research project, demonstrating the potential benefits of using causal diagrams to model datasets. The suggestion put forth in this study is therefore a best practice to mitigate or avoid the presence of biases in the data, and may be adopted by practitioners in various domains to improve the robustness of trained models. The case study supports the potential benefits of causal analysis in improving the performance and generalization ability of trained models.

In Chapter 3, two studies on the topic of XAI were presented: iNNvestigate-GUI [97] and HOLMES. The iNNvestigate-GUI tool provides an interactive approach for analyzing and comparing the behavior of multiple DNNs using visualization methods. A user study showed that the tool is user-friendly and effective in identifying model behaviors and suggesting useful data samples for analysis. HOLMES, on the other hand, offers detailed part-level explanations for image classification tasks, which is a significant advancement beyond the traditional label-level heatmaps. The approach connects DNN models with a symbolic knowledge base, such as an ontology, and captures concepts without constraining them to a single computational unit. Results demonstrate that HOLMES performs better when considering a small number of parts for an object, and there is still much to explore in terms of alternative components. The iNNvestigate-GUI approach and HOLMES provide promising solutions to interpret the behavior of DNNs and enhance image classification models with part-level explanations, respectively. These approaches demonstrate the potential of XAI to improve the interpretability and reliability of DL models, which is becoming increasingly important as these models are being deployed in critical applications. The user study of iNNvestigate-GUI showed that even non-expert users can effectively use the tool to analyze the behavior of DNNs. HOLMES, on the other hand, offers valuable insights into how holonyms are learned and stored within CNNs during and after the training phase, and connects DNN models with a symbolic knowledge base. Despite the promising results of both approaches, there is still much to explore in terms of alternative components. For iNNvestigate-GUI, future research could investigate the effects of using the acti-

vations of units from different convolutional layers to generate explanations. For HOLMES, future research could explore more refined scraping algorithms, alternative perturbation techniques, and new strategies for selecting and filtering the meronyms of an object. Overall, both the studies represent significant contributions to XAI and provide a foundation for future research towards trustworthy AI.

Finally, in Chapter 4, two studies were presented focused on concept drift detection [150] and OOD detection. The first study showed that the distance from a reference feature distribution is strongly correlated with performance degradation, suggesting that monitoring the distribution of features learned by the model itself is a cost-effective solution for addressing dataset drift. However, the proposed strategy is not readily applicable to models that are trained by third-party and commercial providers, which do not typically enable access to inner model features. In such cases, end-users of blackbox models could resort to training an auxiliary model to learn the data distribution before or when the model is put in production. Moreover, it was found that the use of synthetic data has the potential to be a powerful tool for benchmarking concept drift detection algorithms and tailoring them to specific use cases. The second study evaluated several techniques commonly adopted in literature for OOD detection: the main difference with respect to previous comparisons lies in the selection of benchmarks. Each class was labeled as ID or OOD, taking into account the entirety of the data. Furthermore, the feasibility of automating this operation was evaluated using the WordNet database and concept similarity, leading to the three WordNet-ImageNet datasets. Despite being significantly worse than what is typically achieved by the same methods on simpler datasets, the results obtained are better than the ones obtained by Roady et al. [178] in a very similar scenario.

Both studies have limitations that should be addressed in future research. For the first study, it is planned to extend the comparison to different types of concept drift detectors to further improve the detection accuracy. Additionally, the aim is to investigate the generalizability of this approach to different models and applications, as well as more complex drift patterns and scenarios. For the second study, the results support the hypothesis that more comprehensive and realistic benchmarks are needed to evaluate the effectiveness of OOD detection methods, and that per-class analysis can provide more realistic benchmarks than intra-dataset comparison. These findings have practical implications for MLOps, as they provide cost-effective solutions for addressing dataset drift and detecting OOD data. Overall, it is believed

that the insights gained from these studies will contribute to advancing the research on data-centric approaches towards robust and reliable AI.

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[3] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems*, 28, 2015.

[4] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[6] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[7] Richard Szeliski. *Computer vision: Algorithms and applications*. Springer Nature, 2022.

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT Press, 2016.

[10] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[11] Sheraz Naseer, Yasir Saleem, Shehzad Khalid, Muhammad Khawar Bashir, Jihun Han, Muhammad Munwar Iqbal, and Kijun Han. Enhanced network anomaly detection based on deep neural networks. *IEEE Access*, 6:48231–48246, 2018.

[12] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, Eftychios Protopapadakis, et al. Deep learning for computer vision: A brief review. *Computational Intelligence and Neuroscience*, 2018.

[13] Xin Huang, Xiaopeng Han, Song Ma, Tianjia Lin, and Jianya Gong. Monitoring ecosystem service change in the city of shenzhen by the use of high-resolution remotely sensed imagery and deep learning. *Land Degradation & Development*, 30(12):1490–1501, 2019.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[15] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1134–1141, 2018.

[16] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

[17] Nathalie A Smuha. The eu approach to ethics guidelines for trustworthy artificial intelligence. *Computer Law Review International*, 20(4):97–106, 2019.

[18] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. *Proceedings of Machine Learning Research*, 81:1–15, 2018.

[19] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

[20] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340, 2018.

[21] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295, 2004.

[22] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[23] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12):2346–2363, 2018.

[24] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[25] Anas Bodor, Meriem Hnida, and Daoudi Najima. Mlops: Overview of current state and future directions. In *Proceedings of the 7th International Conference on Smart City Applications*, pages 156–165, 2023.

[26] Georgios Symeonidis, Evangelos Nerantzis, Apostolos Kazakis, and George A Papakostas. Mlops-definitions, tools and challenges. In *Proceedings of the IEEE 12th Annual Computing and Communication Workshop and Conference*, pages 0453–0460, 2022.

[27] Data-centric AI Resource Hub. https://datacentricai.org/. Accessed: 2023-04-26.

[28] Fabio Garcea, Lia Morra, and Fabrizio Lamberti. On the use of causal models to build better datasets. In *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 1514–1519. IEEE, 2021.

[29] Fabio Garcea, Giacomo Blanco, Alberto Croci, Fabrizio Lamberti, Riccardo Mamone, Ruben Ricupero, Lia Morra, and Paola Allamano. Self-supervised and semi-supervised learning for road condition estimation from distributed road-side cameras. *Scientific reports*, 12(1):22341, 2022.

[30] Judea Pearl. Causal inference in statistics: An overview. 2009.

[31] Md Nasim Khan and Mohamed M. Ahmed. Weather and surface condition detection based on road-side webcams: Application of pre-trained convolutional neural network. *International Journal of Transportation Science and Technology*, 11(3):468–483, 2022.

[32] Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C de Albuquerque. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[33] Angelo Ziletti, Devinder Kumar, Matthias Scheffler, and Luca M Ghiringhelli. Insightful classification of crystal structures using deep learning. *Nature communications*, 9(1):1–10, 2018.

[34] Lia Morra, Silvia Delsanto, and Loredana Correale. *Artificial intelligence in medical imaging: From theory to clinical practice*. CRC Press, 2019.

[35] Big Data To Good Data: Andrew Ng Urges ML Community To Be More Data-Centric And Less Model-Centric. https://tinyurl.com/3t6dp5n3. Accessed: 2023-04-29.

[36] Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528, 2011.

[37] Mariachiara Mecati, Flavio Emanuele Cannavò, Antonio Vetrò, and Marco Torchiano. Identifying risks in datasets for automated decision–making. In *International Conference on Electronic Government*, pages 332–344. Springer, 2020.

[38] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *arXiv preprint arXiv:1803.09010*, 2018.

[39] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. A deeper look at dataset bias. In *Domain adaptation in computer vision applications*, pages 37–55. Springer, 2017.

[40] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

[41] Toon Calders. Machine-learning discrimination: bias in, bias out. In *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 27–27, 2019.

[42] Nicholas Petrick, Berkman Sahiner, Samuel G Armato III, et al. Evaluation of computer-aided detection and diagnosis systems. *Medical physics*, 40(8):087001, 2013.

[43] Markus Borg, Cristofer Englund, Krzysztof Wnuk, et al. Safely entering the deep: A review of verification and validation for machine learning and a challenge elicitation in the automotive industry. *arXiv preprint arXiv:1812.05389*, 2018.

[44] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, Nov 2020.

[45] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards. *arXiv preprint arXiv:1805.03677*, 2018.

[46] Elena Beretta, Antonio Vetrò, Bruno Lepri, and Juan Carlos De Martin. Detecting discriminatory risk through data annotation based on bayesian inferences. In *Proc. of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 794–804, 2021.

[47] Lu Zhang, Yongkai Wu, and Xintao Wu. Achieving non-discrimination in data release. In *Proc. of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1335–1344, 2017.

[48] Sahil Verma, John Dickerson, and Keegan Hines. Counterfactual explanations for machine learning: A review. *arXiv preprint arXiv:2010.10596*, 2020.

[49] Viktoriia Sharmanska, Lisa Anne Hendricks, Trevor Darrell, and Novi Quadrianto. Contrastive examples for addressing the tyranny of the majority. *arXiv preprint arXiv:2004.06524*, 2020.

[50] Bradley Butcher, Vincent S Huang, Christopher Robinson, Jeremy Reffin, Sema K Sgaier, Grace Charles, and Novi Quadrianto. Causal datasheet for datasets: An evaluation guide for real-world data analysis and data collection design using bayesian networks. *Frontiers in Artificial Intelligence*, 4:18, 2021.

[51] Adam Kortylewski, Bernhard Egger, Andreas Schneider, Thomas Gerig, Andreas Morel-Forster, and Thomas Vetter. Analyzing and reducing the damage of dataset bias to face recognition with synthetic data. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[52] Judea Pearl and Elias Bareinboim. External validity: From do-calculus to transportability across populations. *Statistical Science*, pages 579–595, 2014.

[53] Daniel C Castro, Ian Walker, and Ben Glocker. Causality matters in medical imaging. *Nature Communications*, 11(1):1–10, 2020.

[54] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. In *Proceedings of the 29th International Conference on Machine Learning*, pages 459–466, 2012.

[55] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[56] Gareth J Griffith, Tim T Morris, Matthew J Tudball, Annie Herbert, Giulia Mancano, Lindsey Pike, et al. Collider bias undermines our understanding of covid-19 disease risk and severity. *Nature communications*, 11(1):1–12, 2020.

[57] Sheela Ramanna, Cenker Sengoz, Scott Kehler, and Dat Pham. Near real-time map building with multi-class image set labeling and classification of road conditions using convolutional neural networks. *Applied Artificial Intelligence*, pages 1–31, 2021.

[58] Juan Carrillo, Mark Crowley, Guangyuan Pan, and Liping Fu. Comparison of deep learning models for determining road surface condition from roadside camera images and weather data. In *Transportation Association of Canada and Intelligent Transportation Systems Canada Joint Conference*, pages 1–16, 2019.

[59] Saira Jabeen, AbdulGhaffar Malkana, Ali Farooq, and Usman Ghani Khan. Weather classification on roads for drivers assistance using deep transferred features. In *2019 International Conference on Frontiers of Information Technology (FIT)*, pages 221–2215. IEEE, 2019.

[60] P. Jonsson. Road condition discrimination using weather data and camera images. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1616–1621, Oct 2011.

[61] Raouf Babari, Nicolas Hautière, Éric Dumont, Nicolas Paparoditis, and James Misener. Visibility monitoring using conventional roadside cameras–emerging applications. *Transportation research part C: emerging technologies*, 22:17–28, 2012.

[62] P Allamano, A Croci, and F Laio. Toward the camera rain gauge. *Water Resources Research*, 51(3):1744–1757, 2015.

[63] Kshitiz Garg and Shree K Nayar. Vision and rain. *International Journal of Computer Vision*, 75(1):3–27, 2007.

[64] Jérémie Bossu, Nicolas Hautiere, and Jean-Philippe Tarel. Rain or snow detection in image sequences through use of a histogram of orientation of streaks. *International journal of computer vision*, 93(3):348–367, 2011.

[65] Kshitiz Garg and Shree K Nayar. When does a camera see rain? In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, pages 1067–1074. IEEE, 2005.

[66] Lushan Cheng, Xu Zhang, and Jie Shen. Road surface condition classification using deep learning. *Journal of Visual Communication and Image Representation*, 64:102638, 2019.

[67] Bin Zhao, Xuelong Li, Xiaoqiang Lu, and Zhigang Wang. A CNN–RNN architecture for multi-label weather recognition. *Neurocomputing*, 322:47–57, 2018.

[68] Marcus Nolte, Nikita Kister, and Markus Maurer. Assessment of deep convolutional neural networks for road surface classification. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 381–386. IEEE, 2018.

[69] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

[70] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4L: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485, 2019.

[71] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[72] Ke Yan, Xiaosong Wang, Le Lu, Ling Zhang, Adam P Harrison, Mohammadhadi Bagheri, and Ronald M Summers. Deep lesion graphs in the wild: relationship learning and organization of significant radiology image findings in a diverse large-scale lesion database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9261–9270, 2018.

[73] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[74] Rui Qian, Tianjian Meng, Boqing Gong, Ming-Hsuan Yang, Huisheng Wang, Serge Belongie, and Yin Cui. Spatiotemporal contrastive video representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6964–6974, 2021.

[75] Kumar Ayush, Burak Uzkent, Chenlin Meng, Kumar Tanmay, Marshall Burke, David Lobell, and Stefano Ermon. Geography-aware self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10181–10190, 2021.

[76] Vladan Stojnic and Vladimir Risojevic. Self-supervised learning of remote sensing scene representations using contrastive multiview coding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1182–1191, 2021.

[77] Omar Elharrouss, Noor Almaadeed, and Somaya Al-Maadeed. A review of video surveillance systems. *Journal of Visual Communication and Image Representation*, 77:103116, 2021.

[78] David W Hughes, BD Yallop, and CY Hohenkerk. The equation of time. *Monthly Notices of the Royal Astronomical Society*, 238(4):1529–1535, 1989.

[79] Julia M Rohrer. Thinking clearly about correlations and causation: Graphical causal models for observational data. *Advances in Methods and Practices in Psychological Science*, 1(1):27–42, 2018.

[80] Judea Pearl and Dana Mackenzie. *The book of why: the new science of cause and effect*. Basic Books, 2018.

[81] Luke Darlow, Stanisław Jastrzębski, and Amos Storkey. Latent adversarial debiasing: Mitigating collider bias in deep neural networks. *arXiv preprint arXiv:2011.11486*, 2020.

[82] Computer Vision Annotation Tool (CVAT). https://github.com/openvinotoolkit/cvat. Accessed: 2023-04-29.

[83] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[84] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019.

[85] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[86] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

[87] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin Raffel. Mixmatch: A holistic approach to semi-supervised learning. *arXiv preprint arXiv:1905.02249*, 2019.

[88] Zhongzheng Ren, Raymond Yeh, and Alexander Schwing. Not all unlabeled data are equal: Learning to weight data in semi-supervised learning. *Advances in Neural Information Processing Systems*, 33, 2020.

[89] Fabio Garcea, Alessandro Cucco, Lia Morra, and Fabrizio Lamberti. Object tracking through residual and dense lstms. In *International Conference on Image Analysis and Recognition*, pages 100–111. Springer, 2020.

[90] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[91] Dennis D Boos. Introduction to the bootstrap world. *Statistical science*, 18(2):168–174, 2003.

[92] Elizabeth R DeLong, David M DeLong, and Daniel L Clarke-Pearson. Comparing the areas under two or more correlated receiver operating characteristic curves: a nonparametric approach. *Biometrics*, pages 837–845, 1988.

[93] Elahe Vahdani and Yingli Tian. Deep learning-based action detection in untrimmed videos: A survey. *arXiv preprint arXiv:2110.00111*, 2021.

[94] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "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*, pages 1135–1144, 2016.

[95] Guglielmo Camporese, Pasquale Coscia, Antonino Furnari, Giovanni Maria Farinella, and Lamberto Ballan. Knowledge distillation for action anticipation via label smoothing. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 3312–3319. IEEE, 2021.

[96] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 6450–6459, 2018.

[97] Fabio Garcea, Sina Famouri, Davide Valentino, Lia Morra, and Fabrizio Lamberti. innvestigate-gui-explaining neural networks through an interactive visualization tool. In *Artificial Neural Networks in Pattern Recognition: 9th IAPR TC3 Workshop, ANNPR 2020, Winterthur, Switzerland, September 2–4, 2020, Proceedings 9*, pages 291–303. Springer, 2020.

[98] Francesco Dibitonto, Fabio Garcea, Panisson André, Perotti Alan, Lia Morra, et al. Holmes: Holonym-meronym based semantic inspection for convolutional image classifiers. *COMMUNICATIONS IN COMPUTER AND INFORMATION SCIENCE*, 2023.

[99] Waddah Saeed and Christian Omlin. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, page 110273, 2023.

[100] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[101] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[102] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, 2019.

[103] Christin Seifert, Aisha Aamir, Aparna Balagopalan, Dhruv Jain, Abhinav Sharma, Sebastian Grottel, and Stefan Gumhold. Visualizations of deep neural networks in computer vision: A survey. In *Transparent Data Mining for Big and Small Data*, pages 123–144. Springer, 2017.

[104] Minsuk Kahng, Pierre Y Andrews, Aditya Kalro, and Duen Horng Polo Chau. Activis: Visual exploration of industry-scale deep neural network models. *IEEE transactions on visualization and computer graphics*, 24(1):88–97, 2017.

[105] Maximilian Alber, Sebastian Lapuschkin, Philipp Seegerer, Miriam Hägele, Kristof T Schütt, Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller, Sven Dähne, and Pieter-Jan Kindermans. iNNvestigate neural networks! *J. Mach. Learn. Res.*, 20(93):1–8, 2019.

[106] Kanit Wongsuphasawat, Daniel Smilkov, James Wexler, Jimbo Wilson, Dandelion Mane, Doug Fritz, Dilip Krishnan, Fernanda B Viégas, and Martin Wattenberg. Visualizing dataflow graphs of deep learning models in tensorflow. *IEEE transactions on visualization and computer graphics*, 24(1):1–12, 2017.

[107] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B. Viégas, and Martin Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings, 2016.

[108] Sunghyo Chung, Sangho Suh, Cheonbok Park, Kyeongpil Kang, Jaegul Choo, and Bum Chul Kwon. Revacnn: Real-time visual analytics for convolutional neural network. In *KDD 16 Workshop on Interactive Data Exploration and Analytics*, pages 30–36, 2016.

[109] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[110] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Polo Chau. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE transactions on visualization and computer graphics*, 26(1):1096–1106, 2019.

[111] Nicola Pezzotti, Thomas Höllt, Jan Van Gemert, Boudewijn PF Lelieveldt, Elmar Eisemann, and Anna Vilanova. Deepeyes: Progressive visual analytics for designing deep neural networks. *IEEE transactions on visualization and computer graphics*, 24(1):98–108, 2017.

[112] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M. Rush. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks, 2016.

[113] Junpeng Wang, Liang Gou, Hao Yang, and Han-Wei Shen. Ganviz: A visual analytics approach to understand the adversarial game. *IEEE transactions on visualization and computer graphics*, 24(6):1905–1917, 2018.

[114] Mengchen Liu, Jiaxin Shi, Zhen Li, Chongxuan Li, Jun Zhu, and Shixia Liu. Towards better analysis of deep convolutional neural networks. *IEEE transactions on visualization and computer graphics*, 23(1):91–100, 2016.

[115] Daniel Smilkov, Shan Carter, D Sculley, Fernanda B Viégas, and Martin Wattenberg. Direct-manipulation visualization of deep networks. *arXiv:1708.03788*, 2017.

[116] Zijie J Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Chau. Cnn explainer: Learning convolutional neural networks with interactive visualization. *arXiv preprint arXiv:2004.15004*, 2020.

[117] Fred Hohman, Minsuk Kahng, Robert Pienta, and Duen Horng Chau. Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE transactions on visualization and computer graphics*, 25(8):2674–2693, 2018.

[118] Keras Explain. https://github.com/primozgodec/keras-explain. Accessed: 2023-04-29.

[119] DeepExplain. https://github.com/marcoancona/DeepExplain. Accessed: 2023-04-29.

[120] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org, 2017.

[121] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Agata Lapedriza, Bolei Zhou, and Antonio Torralba. Understanding the role of individual units in a deep neural network. *Proceedings of the National Academy of Sciences*, 117(48):30071–30078, 2020.

[122] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations*, 2019.

[123] Arman Sarraf, Mohammad Azhdari, and Saman Sarraf. A comprehensive review of deep learning architectures for computer vision applications. *American Scientific Research Journal for Engineering, Technology, and Sciences*, 77:1–29, 03 2021.

[124] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[125] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, volume 27, pages 3320–3328, 2014.

[126] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[127] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys*, 51(5):1–42, 2018.

[128] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

[129] Valentina Ghidini, Alan Perotti, and Rossano Schifanella. Quantitative and ontology-based comparison of explanations for image classification. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 58–70. Springer, 2019.

[130] Natalia Díaz Rodríguez, Alberto Lamas, Jules Sanchez, Gianni Franchi, Ivan Donadello, Siham Tabik, David Filliat, Policarpo Cruz, Rosana Montes, and Francisco Herrera. Explainable neural-symbolic learning (*X-NeSyL*) methodology to fuse deep learning representations with expert knowledge graphs: The monumai cultural heritage use case. *Information Fusion*, 79:58–83, 2022.

[131] Roberto Confalonieri, Fermín Moscoso del Prado, Sebastia Agramunt, Daniel Malagarriga, Daniele Faggion, Tillman Weyde, and Tarek R. Besold. An ontology-based approach to explaining artificial neural networks. *CoRR*, abs/1906.08362, 2019.

[132] G. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[133] Priyanka Samanta and Shweta Jain. Analysis of perceptual hashing algorithms in image manipulation detection. *Procedia Computer Science*, 185:203–212, 2021.

[134] Abel Gonzalez-Garcia, Davide Modolo, and Vittorio Ferrari. Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision*, 126, 2018.

[135] Wissam Siblini, Jordan Fréry, Liyun He-Guelton, Frédéric Oblé, and Yi-Qing Wang. Master your metrics with calibration. *Advances in Intelligent Data Analysis XVIII*, page 457–469, 2020.

[136] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Loddon Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1979–1986, 2014.

[137] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8827–8836, 2018.

[138] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. In *British Machine Vision Conference*, 2018.

[139] Dohun Lim, Hyeonseok Lee, and Sungchan Kim. Building reliable explanations of unreliable neural networks: Locally smoothing perspective of model interpretation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6464–6473, 2021.

[140] C. Silberer, V. Ferrari, and M. Lapata. Visually grounded meaning representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11):2284–2297, 2017.

[141] Dario Molinari, Giulia Pasquale, L. Natale, and B. Caputo. Automatic creation of large scale object databases from web resources: A case study in robot vision. In *International Conference on Image Analysis and Processing*, 2019.

[142] Nizar Massouh, Francesca Babiloni, Tatiana Tommasi, Jay Young, Nick Hawes, and Barbara Caputo. Learning deep visual object models from noisy web data: How to make it work. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5564–5571. IEEE, 2017.

[143] Lia Morra and Fabrizio Lamberti. Benchmarking unsupervised near-duplicate image detection. *Expert Systems with Applications*, 135:313–326, 2019.

[144] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and Liwu Chang. A novel anomaly detection scheme based on principal component classifier. In *Proceedings of International Conference on Data Mining*, 01 2003.

[145] Yazhou Yao, Fumin Shen, Guosen Xie, Li Liu, Fan Zhu, Jian Zhang, and Heng Tao Shen. Exploiting web images for multi-output classification: From category to subcategories. *IEEE transactions on neural networks and learning systems*, 31(7):2348–2360, 2020.

[146] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *International Conference on Learning Representations*, abs/1412.6856, 2015.

[147] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition*, 2017.

[148] C. Chen, O. Li, Alina Barnett, Jonathan Su, and C. Rudin. This looks like that: deep learning for interpretable image recognition. In *Advances in neural information processing systems*, 2019.

[149] Chuanyi Zhang, Qiong Wang, Guosen Xie, Qi Wu, Fumin Shen, and Zhenmin Tang. Robust learning from noisy web images via data purification for fine-grained recognition. *IEEE Transactions on Multimedia*, 2021.

[150] Luca Piano, Fabio Garcea, Valentina Gatteschi, Fabrizio Lamberti, and Lia Morra. Detecting drift in deep learning: A methodology primer. *IT Professional*, 24(5):53–60, 2022.

[151] Pietro Recalcati, Fabio Garcea, Luca Piano, Fabrizio Lamberti, Lia Morra, et al. Toward a realistic benchmark for out-of-distribution detection. In *Proceedings of the 10th IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 2023.

[152] Tania Cerquitelli, Nikolaos Nikolakis, Lia Morra, Andrea Bellagarda, Matteo Orlando, Riku Salokangas, Olli Saarela, Jani Hietala, Petri Kaarmila, and Enrico Macii. Data-driven predictive maintenance: A methodology primer. In *Predictive Maintenance in Smart Factories: Architectures, Methodologies, and Use-cases*, pages 39–73. Springer Singapore, Singapore, Republic of Singapore, 2021.

[153] Alexander Lavin, Ciarán M Gilligan-Lee, Alessya Visnjic, Siddha Ganju, Dava Newman, Sujoy Ganguly, Danny Lange, Atılım Güneş Baydin, Amit Sharma, Adam Gibson, et al. Technology readiness levels for machine learning systems. *arXiv:2101.03989*, 2021.

[154] Angelos Filos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *ICML*, pages 3145–3153, 2020.

[155] Zhenyi Liu, Trisha Lian, Joyce Farrell, and Brian A Wandell. Neural network generalization: The impact of camera parameters. *IEEE Access*, 8:10443–10454, 2020.

[156] N Alan Heckert, James J Filliben, C M Croarkin, B Hembree, William F Guthrie, P Tobias, J Prinz, et al. Handbook 151: Nist/sematech e-handbook of statistical methods. 2002.

[157] Frank E Grubbs. Procedures for detecting outlying observations in samples. *Technometrics*, 11(1):1–21, 1969.

[158] Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*, 2021.

[159] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[160] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[161] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. *arXiv e-prints*, page arXiv:1706.04599, June 2017.

[162] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. *CoRR*, abs/1812.05720, 2018.

[163] Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32:1396–1408, 2019.

[164] Sara Castellanos. Fake it to make it: Companies beef up AI models with synthetic data, Jul 2021.

[165] Joaquin Quiñonero-Candela, Masashi Sugiyama, Neil D Lawrence, and Anton Schwaighofer. *Dataset shift in machine learning*. Mit Press, 2009.

[166] Simone Disabato and Manuel Roveri. Learning convolutional neural networks in presence of concept drift. In *IJCNN*, pages 1–8, 2019.

[167] Gregory Ditzler and Robi Polikar. Hellinger distance based drift detection for nonstationary environments. In *IEEE Symposium on Computational Intelligence in Dynamic and Uncertain Environments (CIDUE)*, pages 41–48, 2011.

[168] Thomas Viehmann, Luca Antiga, Daniele Cortinovis, and Lisa Lozza. Torchdrift: Drift detection for pytorch, 2019.

[169] Jaka Demšar and Zoran Bosnić. Detecting concept drift in data streams using model explanation. *Expert Systems with Applications*, 92:546–559, 2018.

[170] Tong Che, Xiaofeng Liu, Site Li, Yubin Ge, Ruixiang Zhang, Caiming Xiong, and Yoshua Bengio. Deep verifier networks: Verification of deep discriminative models with deep generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7002–7010, 2021.

[171] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. 2018.

[172] Dario Fontanel, Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulo, Elisa Ricci, and Barbara Caputo. Boosting deep open world recognition by clustering. *IEEE Robotics and Automation Letters*, 5(4):5985–5992, 2020.

[173] Chuanxing Geng, Sheng-jun Huang, and Songcan Chen. Recent advances in open set recognition: A survey. *IEEE Transactions on Pattern Analysis and Mchine Intelligence*, 43:3614–3631, 2021.

[174] Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *CVPR*, pages 5315–5324, 2017.

[175] Fahim Tajwar, Ananya Kumar, Sang Michael Xie, and Percy Liang. No true state-of-the-art? OOD detection methods are inconsistent across datasets. *CoRR*, abs/2109.05554, 2021.

[176] Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R. Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon A. A. Kohl, taylan. cemgil, S. M. Ali Eslami, and Olaf Ronneberger. Contrastive training for improved out-of-distribution detection. *ArXiv*, abs/2007.05566, 2020.

[177] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. Exploring the limits of out-of-distribution detection. *CoRR*, abs/2106.03004, 2021.

[178] Ryne Roady, Tyler L Hayes, Ronald Kemker, Ayesha Gonzales, and Christopher Kanan. Are out-of-distribution detection methods effective on large-scale datasets? *arXiv preprint arXiv:1910.14034*, 2019.

[179] Dan Hendrycks, Steven Basart, Mantas Mazeika, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. A benchmark for anomaly segmentation. *CoRR*, abs/1911.11132, 2019.

[180] Rui Huang and Yixuan Li. Mos: Towards scaling out-of-distribution detection for large semantic space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8710–8719, 2021.

[181] Faruk Ahmed and Aaron C. Courville. Detecting semantic anomalies. *CoRR*, abs/1908.04388, 2019.

[182] Jingkang Yang, Haoqi Wang, Litong Feng, Xiaopeng Yan, Huabin Zheng, Wayne Zhang, and Ziwei Liu. Semantically coherent out-of-distribution detection. *CoRR*, abs/2108.11941, 2021.

[183] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[184] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized ODIN: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10951–10960, 2020.

[185] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1563–1572, 2016.

[186] Julian Bitterwolf, Alexander Meinke, and Matthias Hein. Certifiably adversarially robust detection of out-of-distribution data. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 16085–16095. Curran Associates, Inc., 2020.

[187] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[188] Yuval Netzer, Tao Wang, Adam Coates, A. Bissacco, Bo Wu, and A. Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[189] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[190] Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. 2015.

[191] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.

[192] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[193] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[194] Vahdat Abdelzad, K. Czarnecki, Rick Salay, Taylor Denouden, Sachin Vernekar, and Buu Phan. Detecting out-of-distribution inputs in deep neural networks using an early-layer output. *ArXiv*, abs/1910.10307, 2019.

[195] OODL Github Repository. https://github.com/vahdat-ab/OODL. Accessed: 2023-04-29.

[196] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[197] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet::similarity - measuring the relatedness of concepts. 04 2004.

[198] Claudia Leacock and Martin Chodorow. Combining local context and wordnet similarity for word sense identification. *WordNet: An electronic lexical database*, 49(2):265–283, 1998.

[199] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. *arXiv preprint cmp-lg/9406033*, 1994.

[200] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. *arXiv preprint cmp-lg/9511007*, 1995.

[201] Dekang Lin et al. An information-theoretic definition of similarity. In *Icml*, volume 98, pages 296–304, 1998.

[202] Jay J Jiang and David W Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*, 1997.

[203] Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492, 2010.

[204] Gephi The Open Graph Viz Platform. https://gephi.org/. Accessed: 2023-04-29.

[205] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one*, 9(6):e98679, 2014.

[206] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Xiaodong Song. Natural adversarial examples. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15257–15266, 2021.