Explaining Deep Neural Networks through Knowledge Extraction and Graph Analysis

Vitor A. C. Horta M.Sc.

Supervised by Prof. Alessandra Mileo

A thesis presented for the degree of Doctor of Philosophy (Ph.D.)

School of Computing

Dublin City University

August 2023

Vitor A. C. Horta

ID No.: 18215150

August 2023

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Alessandra Mileo, for her invaluable guidance, expertise and continuous support throughout the course of these four years.

I would also like to thank my wife, Léia, for her constant love and support during the challenging times of moving abroad and adapting to a new environment.

I am also grateful to my parents, Pedro Paulo and Marcia, and my three brothers, Bruno, Pedro and André, for their constant encouragement and belief in me, which contributed enormously to each achievement during my PhD journey.

Finally, I would like to express my appreciation to all those who have directly or indirectly contributed to this work, including my colleagues and fellow researchers. Thank you for your support and collaboration.

Contents

List of Figures

List of Tables

Abstract

Explaining Deep Neural Networks through Knowledge Extraction and Graph Analysis

Vitor A. C. Horta

Explainable Artificial Intelligence (XAI) has recently become an active research field due to the need for transparency and accountability when deploying AI models for high-stake decision making. Despite the success of Deep Neural Networks (DNNs), understanding their decision processes is still a known challenge. The research direction presented in this thesis stems from the idea that combining knowledge with deep representations can be the key to more transparent decision making. Specifically, we have focused on Computer Vision tasks and Convolutional Neural Networks (CNNs) and we have proposed a graph representation, called co-activation graph, that serves as an intermediate representation between knowledge encoded within a CNN and the semantics contained in external knowledge bases. Given a trained CNN, we first show how a co-activation graph can be created and exploited to generate global insights for the model's inner-workings. Then, we propose a taxonomy extraction method that captures how symbolic class concepts and their hypernyms in a given domain are hierarchically organised in the model's subsymbolic representation. We then illustrate how background knowledge can be connected to the graph in order to generate textual local factual and counterfactual explanations. Our results indicate that graph analysis approaches applied to co-activation graphs can reveal important insights into how CNNs work and enable both global and local semantic explanations. Despite focusing on CNN architectures, we believe that our approach can be adapted to other architectures which would make it possible to apply the same methodology in other domains such as Natural Language Processing.

At the end of the thesis we will discuss interesting research directions that are being investigated in the area of using knowledge graphs and graph analysis for explainability of deep learning models, and we outline opportunities for the development of this research area.

Chapter 1

Introduction

Explainable Artificial Intelligence (XAI) is the research field that focuses on bringing more transparency to the decision making process of AI systems [41]. With the success of Deep Neural Networks (DNNs), and the desire to deploy these models for high-stake decision making, researchers in XAI are focusing on the need to be able to explain the decisions made by DNNs [5]. Providing such explanations is crucial to guarantee their reliability [94], and to ensure compliance with the EU's new AI regulation [1]. This can be crucial in situations where the ability to explain decisions and understand misjudgements is key, which include medical diagnosis, law enforcement, financial analysis and more. However, given their black box nature, deriving explanations for DNNs is not trivial, and understanding why a DNN makes a particular decision is still a open challenge.

The desire to understand the black box inside which Artificial Neural Networks (ANNs) work is not new. There were already attempts to explain ANNs even when the quality and availability of learning data and the lack of computational resources did not yet enable full exploitation of deep learning approaches [95, 4, 73]. More recently the number of papers that tackle this problem has increased considerably [36] and strategies with different perspectives have been developed to help understand the decision making process of DNNs. Although existing approaches have made a considerable progress in explaining the high level behaviour of DNNs,

[1]https://ec.europa.eu/commission/presscorner/detail/en/IP_21_1682)

most of them derive their explanations solely based on the information contained in the same feature space in which the DNN was trained. In this sense, existing methods do not leverage human domain knowledge when explaining DNNs, which usually translates into explanations that are not self-explanatory in terms of human expert knowledge, and therefore lack transparency.

In this thesis, we follow the idea that combining knowledge and deep representations can be the key to more transparent decision making [28][66]. We focus on explaining models designed for tasks in Computer Vision, more specifically Convolutional Neural Networks (CNNs) since they are the most used architecture in such visual tasks. If we consider knowledge graphs, for example, the benefits of combining the learning capabilities from CNNs and the explicit and structured representation from knowledge graphs is extensively discussed by [66]. Unlike CNN representations, knowledge graphs represent information explicitly so that each decision taken can be explained in semantic terms. Also, the knowledge represented by such graphs can be automatically analysed using a variety of well studied graph analysis methods. Combining the two approaches, however, is not trivial due to the opaque nature of neural networks and the difficulty in integrating semantics to their hidden neurons [66].

To enable this combination, we propose a method that takes inspiration from recent studies in Neuroscience [30, 72], where graph representations and network analysis are used to interpret brain structure. These studies have shown that analysing brain networks using graph theory methods and concepts like communities in graphs and node centrality, can give important insights on the architecture, development, and evolution of brain networks. More specifically, researchers in neuroscience use functional graphs to study the brain by representing and analysing statistical dependencies among neuronal activities. Inspired by these approaches, we propose a representation that is similar to a functional graph, called co-activation graph, that explores statistical correlations among artificial neural activities in order to explain the inner workings of CNNs. Our approach could also be extended to other types of deep neural networks, which is relevant given the recent success of transformers in computer vision tasks [24, 58] and other application fields such as natural language processing and audio processing [70]. We discuss in more detail in Chapter 3.

The co-activation graph is a knowledge graph representation that connects the knowledge learned by a given CNN during the training phase with human knowledge available in external knowledge bases. It can be used to generate different types of explanations (local and global) as well as a means to compare the semantic quality of different models.

In this introductory chapter, we will first present some background and concepts from Explainable AI. Then, we will discuss the different areas that motivated our work as well as the research questions associated to them.

1.1 Introduction to Explainable AI

With the latest advancements in Deep Learning, the field of Artificial Intelligence (AI) has been relying on the use of Deep Neural Networks (DNNs) for a variety of applications, such as computer vision, natural language processing, speech recognition, and others. Despite the considerable gain in performance acquired by DNNs in these tasks, the lack of interpretability and traceability still hinder wider adoption of AI systems based on such black box models in real-world applications, specifically for high-risk decision making where the notion of trust is paramount. [107].

In order to overcome these limitations, research in the field of Explainable AI (XAI) aiming to bring more transparency to DNNs gained lots of attention in recent years. The development of techniques for interpreting and explaining DNNs is also motivated by the Ethical Guidelines for Trustworth AI [46], which states that AI models need to be lawful, ethical and robust, in order to be considered trustworthy. This challenges the current paradigm of deep learning models, since their black box nature makes it difficult to guarantee that these requirements are achievable.

According to existing systematic literature reviews [99, 51], methods in XAI can be categorised based on their scope (local or global), stage (ante-hoc or post-hoc), and input and output format (tabular, visual, textual, etc). The characteristics pertaining to each of these dimensions are described in what follows.

1.1.1 Scope

Regarding the scope, a local explanation focuses on a single data point. For example, given a prediction in image classification, a visual local interpretation may highlight parts that most influenced the prediction for that specific image [69]. On other hand, global explanations aim to understand the overall behaviour of the model, such as identifying concepts that impact the generalisation for an entire class.

1.1.2 Stage

Ante-hoc methods usually change the model architecture or the training process in order to achieve a more interpretable model. In constrast, post-hoc methods are generally aimed to explain trained models without intervening in the training phase.

1.1.3 Input Format

An explainable model is often designed for dealing with a specific input type. For example, methods designed for explaining predictions around images may not be applied to tabular data.

1.1.4 Output Format

The output format is how the explanation is communicated with the end user. It may or may not be the same as the input format. For example, a prediction around an image may be explained in a visual form but also in natural language or logical rules.

In this thesis we propose a method for generating post-hoc explanations for models that receive images as the input format. Our method is capable of generating both local and global explanations. The local explanations are exposed in a textual format aiming to provide a self-explainable output to the end user. The global explanations are provided using semantic structures (e.g. taxonomies and knowledge graphs) to facilitate evaluating the knowledge contained within the model with external domain knowledge.

1.2 Introduction to Convolutional Neural Networks

Since their inception in 1989 [67], Convolutional Neural Neworks (CNNs) are still widely used and considered to be the state of the art in computer vision tasks. Each year, more sophisticated and accurate CNN architectures are developed by researchers in both academia and industry [57] and more real-world applications are making use of them [3].

A CNN consists of a number of stacked convolutional layers and each convolutional layer contains k convolutional filters. A convolutional filter has three dimensions: height n, width n and depth q, such that the height is equal to the width. When data is passed through a convolutional filter, it performs a convolutional operation that consists of sliding the filter over the data horizontally and vertically while calculating the dot product between the filter's parameters (weight and bias) and the respective region in the data. In computer vision, each region usually corresponds to pixels from an image. When this operation finishes (no further sliding is possible), the results are passed through an activation function and finally to a pooling layer that generates a feature map, which is the output of a single convolutional filter. The output of a whole convolutional layer are k feature maps, which are passed as input to the next layer in the chain. The output of the last convolutional layer is usually flattened and passed through a fully connected layer which uses this information to perform some task, such as a classification or regression.

One major benefit of using CNN is that this type of architecture is capable of extracting features from data automatically. This is especially important for computer vision given the challenging of performing feature engineering over image data. In this regard, existing evidence shows that convolutional layers closer to the input are responsible for detecting low-level features, such as dots or lines, while layers closer to the output detect high-level features, such as objects and more complex shapes [42]. In addition to that, CNNs have a reduced number of trainable parameters, which makes them more efficient and less prone to overfitting [3].

Despite the success of CNNs, their adoption in critical domains such as in healthcare [26, 22] is yet to reach its full potential, mainly due to limitations in terms of transparency. In these cases, it is important to understand not only how good a model is in terms of accuracy, but how reasonable is the decision making process adopted by the model. The method proposed in this thesis will tackle this issue by generating both global and local explanations for CNNs, considering state-of-the-art deep architectures trained over large datasets.

1.3 Introduction to Graph theory

Graph theory is a research field that focuses on studying relationships between objects as they emerge in nature or society. A graph is a representation $G = \{V, E\}$, where V is a set of nodes (or vertices) v and E is a set of edges $e(i, j)$ (relationships) between pairs of nodes v_i and v_j. It is called weighted graph if its edges contain numerical weights associated to them, which can be used to represent distance or similarity between objects. When no weights are defined, the graph is unweighted. Graphs can also be directed or undirected. In undirected graphs, for each existing edge e we can assume that $e_{i,j} = e_{j,i}$, while directed graphs allow for $e(i, j) \neq e(j, i)$.

The study of graphs has a wide range of applications, such as biological, chemical and social networks [76, 96, 29], especially because of graphs' flexibility in representing data from different domains. For example, in social networks, nodes represent people and edges may represent any social interaction or relationship between them, while in biological networks, graphs can be used to represent protein-protein interactions. Another reason for the success of graphs is the presence of well defined graph metrics and algorithms, which can be used to automatically extract patterns from complex highly connected data. In this sense, one important concept is the notion of node centrality. Given a graph, there are different ways to compute the centrality of each node in the graph. A node with high centrality can be seen as "more important" while a less central node is "less important". The definition of important depends on the data domain but also on the type of centrality that is being calculated. Common metrics for node centrality which are used in the thesis are:

- Degree centrality: the degree of a node is equal to the number of edges connecting it to other nodes in the graph. A high degree centrality indicates that the node is connected to many other nodes in the graph while low degree centrality indicates that the node has only few connections.
- Betweenness centrality: the betweenness centrality of a node indicates the number of paths in the graph that passes through that node. Nodes with high betweenness centrality are usually perceived as being like bridges in the graph. Such nodes are also known for keeping distinct groups or subgraphs connected through them. Nodes with low betweenness centrality are usually peripheral nodes and are less important for keeping the overall graph connectivity.
- PageRank centrality: a node with a high PageRank centrality is either a highly connected node or a node that is linked to highly connected neighbours. This type of centrality is calculated using the PageRank algorithm [79], which became famous for its use in search engines but it can be applied to many other application domains. The algorithm performs iterative random walks on the graph in order to measure the centrality of each node and the intuition is that a node becomes important if many different paths lead to it. The PageRank centrality of a node n_i can be mathematically expressed as shown in Equation 1.1 where N is the number of nodes in the graph, d is the dumping factor and c_{n_j} is the number of outbound links on node n_j:

$$PR_{n_i} = \frac{1-d}{N} + d \sum_{j \in \{1,\ldots,N\}} \frac{PR_{n_j}}{c_{n_j}} \tag{1.1}$$

Another way to extract interesting patterns from graphs is by exploring the communities that emerge as the graph evolves. Communities are groups of nodes such that nodes within the same community are more connected than nodes from different communities. Commonly used algorithms for detecting communities are:

- Louvain: the Louvain method [8] is a well known community detection algorithm that finds communities by optimising their modularity index. The modularity measures how dense the connections inside a community are with respect to the density of the connections outside the community. Considering a weighted graph, the modularity Q can be calculated using Equation 1.2, where A_{ij} is the edge weight between nodes i and j, m is the sum of all edge weights in the graph, k_i and k_j are the sum of edge weights for nodes i and j, c_i and c_j are their respective communities and $\delta(c_i, c_j)$ is the Kronecker delta which indicates whether $c_i = c_j$. The louvain algorithm can then be understood as an optimisation algorithm for Q.

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{1.2}$$

- Label Propagation: this algorithm [108] works by propagating labels through the graph iteratively until the process converges or a predefined maximum number of iterations is reached. In the first iteration, each node i receives its own label i. Then, the node labels are updated to the most frequent label in their neighbors. The final set of labels correspond to the communities in the graph and nodes are assigned to these communities based on their final label.

Because community structure is so important in the study of graphs and networks, there is a vast amount of methods for detecting them. For a more detailed overview of the existing methods please refer to [77, 53].

A third important concept when analysing a graph is the ability to predict missing relationships between objects. Link prediction algorithms are designed for this purpose and there are many applications for them, such as recommendations systems in social networks or drug re-purposing in biology. Once again, there are different existing strategies for predicting such missing links and they usually differ in how much of the graph is considered when calculating the probabilities for new links. For example, methods like Common Neighbours and Adamic Adar consider only the local neighbourhood of a given node, while other methods such as Personalised PageRank leverages more the global topology of the graph.

- Common Neighbours: one of the simplest approach to predict a link between two nodes n_i and n_j is to count the number of common neighbours between them, as shown in Equation 1.3. The main limitation of this method is that it only leverages one-hop connections between the nodes and the rest of the graph is not taken into account.

$$CN(n_i, n_j) = |n_i \cap n_j| \tag{1.3}$$

- Adamic Adar: the adamic adar measure A_{ij} [1] computes the likelihood of a link between nodes n_i and n_j using Equation 1.4, where $N(x)$ is the neighborhood of a node x. This method considers two-hop connections since it leverages the neighborhood of the intersected node n_u. The intuition for using the inverse logarithm is that nodes that are connected through highly connected intersected nodes are less significant when compared with intersected nodes with a smaller neighborhood.

$$A(x, y) = \sum_{u \in N(x) \cap N(y)} \frac{1}{\log |N(u)|} \tag{1.4}$$

- Personalised PageRank: in order to calculate the likelihood of a link existing between nodes n_i and n_j, this algorithm calculates the PageRank centrality of n_j considering n_i as a starting node. The algorithm follows a similar approach as the original PageRank algorithm, but it uses a fixed starting point for the random walk process. One advantage of this method is that it is not limited by a fixed number of hops from the source node, which makes it appealing for recommendation systems or search engines, as discussed in [79].

1.4 Introduction to Functional Graphs

In order to apply graph analytics to address a given problem, the first step is to define a graph model for the respective data domain. In some cases the relationship between objects to be analysed are explicit, such as connections between users in a social network online platform. In other cases, it may be more valuable to analyse implicit relationships that are not necessarily structurally connected.

In network neuroscience, a branch of study in the neuroscience research field, graph representations and graph methods are used to study the human brain. In order to create a graph representation for the human brain, some neuroscience studies use a concept called functional graphs. Functional graphs are created to connect objects based on the statistical interdependence of their activities over time. In the case of brain networks, nodes are neurons or regions in the brain and the edges are statistical correlations between their activity, which can be calculated based on data coming from functional magnetic resonance imaging (fMRI) [10].

The use of functional graphs is motivated when the interaction between objects are key to understand the underlying system, but there are no explicit connections between such objects or the structural connections provide only limited understanding of the data. This is the case for CNNs, considering that the neural connections are responsible for the decision making process but their structural weights do not facilitate understanding their *thinking* process.

In this thesis, we propose a method to extract functional graphs for CNNs, called co-activation graphs, which is capable of representing and connecting neurons in both convolutional and fully connected layers. We show how this representation can be generated and analysed using graph metrics and algorithms in order to provide global and local explanations for CNNs.

1.5 Introduction to Knowledge Graphs

A knowledge graph (KG) is a knowledge base that uses a graph representation to describe real world entities and their interrelations [18]. There is a wide range of applications that can take benefit from the use of knowledge graphs especially because of their flexibility in representing data from different domains.

Some known benefits of using knowledge graphs to describe a given domain are their ability to integrate data from different sources and gain insights from data through the use of graph metrics and algorithms. In addition to that, the decision making process of methods designed for KGs are transparent and self-explanatory due to their explicit semantic representation. For these reasons, researchers in XAI suggest that a more transparent decision support for Deep Learning architectures can be achieved by combining their deep representations with knowledge graphs [66, 93].

According to [66], the potential benefits of integrating knowledge graphs with deep learning systems can be manifold and affect different types of applications at different areas, like robotics, natural language processing and computer vision. The authors point that, in computer vision, for example, the use of KGs to explain CNNs could lead to more semantic based explanations, as opposed to existing explanations based solely on visual artifacts.

However, the challenge of combining KGs with CNNs is not trivial, especially because a CNN is a computational graph, which is layered, cyclical and lacks semantic information about both the neurons and their relationships. In this thesis, we aim to overcome this challenge using co-activation graphs as an intermediate representation for CNNs. By using co-activation graphs, neurons in a CNN are connected to each other based on statistical correlations between their activation values, instead of relying on their opaque connectionist architecture. This not only enables the use of graph metrics and algorithms, as discussed in the previous section, but it also allows the knowledge acquired by the CNN to be integrated with external knowledge bases designed by humans. We show how to extend co-activation graphs from purely functional graphs to knowledge graphs by connecting nodes in the graph with external knowledge bases. This extended co-activation graph can be used to predict semantic attributes of unseen images which allows our method to generate semantic and textual local explanations for CNNs in image classification.

1.6 Hypotheses and Research Questions

The first phase in our research consists of studying whether it is feasible to build a representation similar to functional graphs, but for deep learning models. Therefore, a first hypothesis in our research is as follows:

Hypothesis 1 (H1) *The knowledge contained in a Deep Neural Network can be captured by extracting and representing statistical dependencies among neural activities.*

To guide the evaluation and assessment of H1, we formulate two Research Questions:

- **RQ1**: How can we extract and represent statistical dependencies among neural activities in DNNs?
- **RQ2**: How well can functional graphs based on statistical dependencies among neural activities represent the knowledge acquired by DNNs and how can we measure their suitability?

The second phase of our research focuses on studying how graph theory methods applied to the aforementioned graphs can give interesting insights on the inner-working of trained DNNs. This leads to our second hypothesis:

Hypothesis 2 (H2) *Given a graph representing statistical dependencies among neural activities, graph theory methods can be used to give insightful interpretation on how the represented DNN works.*

Two research questions were elaborated to help assessing H2:

- **RQ3**: How can we use graph analysis to interpret the origin of common mistakes in deep models?
- **RQ4**: How can we use graph analysis to understand weaknesses or biases in deep models?

A third hypothesis in this thesis focus on extracting taxonomies from co-activation graphs in order to provide a quantitative assessment on how well semantic concepts and their taxonomic relationships are captured by the model's internal representation. The third hypothesis is formulated as follows:

Hypothesis 3 (H3) *Given a deep model, it is possible to extract a taxonomy from its respective co-activation graph in order to measure how well the model has captured the hierarchical relationships between classes from a given domain.*

Two research questions were elaborated to help assessing H3:

- **RQ5**: How can we extract a taxonomy for co-activation graphs representing deep models?
- **RQ6**: How can taxonomies extracted from co-activation graphs help in assessing the semantic adequacy of deep models?

A fourth hypothesis in this thesis focuses on exploring the use of co-activation graphs as enablers for integrating the knowledge contained in deep representations with external knowledge bases. The fourth hypothesis is formulated as follows:

Hypothesis 4 (H4) *Given a co-activation graph which is representing statistical dependencies among neural activities, if we can connect entities in such graph with external knowledge bases, we can generate human understandable local explanations based on semantic attribute values.*

Three research questions were elaborated to help assessing H4:

- **RQ7**: How can we use co-activation graphs as an intermediate representation to connect deep models with external knowledge bases?
- **RQ8**: How can we use graph algorithms and external knowledge to automatically discover semantic attributes from unseen data?
- **RQ9**: How can we leverage knowledge from external sources to generate semantic explanations for deep models?

To test the four hypotheses and answer the respective research questions, we propose a method based on a graph we introduced in [49], called co-activation graph. The co-activation graph is a graph representation in which nodes represent neurons in a CNN and weighted relationships indicate a statistical correlation between their activation values. This representation enables the use of graph theory methods to exploit the inner-workings of DNNs. This representation is general enough to represent connections between pairs of neurons in any layer of the neural network, including hidden (convolutional and dense) layers and the output layer. As a result, and unlike previous approaches, this makes it possible to study relationships between pairs of neurons in the hidden layers.

1.7 Thesis Structure

This thesis consists of the following six chapters:

(i) **Introduction**: the current chapter, which presents some of the context, motivation and background knowledge for this work.

(ii) **Literature Review** (Chapter 2): we review state-of-the-art research in Explainable AI considering the topics of (i) global explanations; (ii) local explanations; (iii) graph-based explainability methods and; (iv) knowledge-informed explanations.

(iii) **Global Explanations using Knowledge Graphs** (Chapter 3): this chapter introduces the core of our methodology, which is the co-activation graph representation. We formalise the graph representation and perform experiments to validate it as method for generating global explanations.

(iv) **Extracting taxonomies from CNNs using Knowledge Graphs** (Chapter 4): building on the previous chapter we explore how co-activation graphs can be used for extracting taxonomies from CNNs. The taxonomies are used to measure the semantic adequacy of a given trained network and expose how well the model captured the hierarchical semantic relationships between classes.

(v) **Generating Local Textual Explanations for CNNs: A Semantic Approach based on Knowledge Graphs** (Chapter 5): in this chapter we enrich the original co-activation graph with external knowledge and input data points in order to generate semantic explanations in a textual form. We show how the proposed graph representation along with a link prediction method enable the prediction of semantic attributes for unseen images, which helps generating factual and counterfactual explanations.

(vi) **Conclusions** (Chapter 6): this final chapter summarises the research presented in this thesis by revisiting the hypotheses and research questions. Finally, we discuss current limitations with the proposed methods and present future directions for further research.

Chapter 2

Literature Review

In this chapter we provide more details on relevant state-of-the-art literature regarding Explainable AI (XAI) techniques for deep learning models, with a specific focus on explaining Convolutional Neural Networks (CNNs). We first review the most commonly used methods designed for global explanations, which include neuron contribution, rule-extraction methods, visual methods, concept-based explanations and class similarity. For local explanations we review approaches based on visualisation, case-based explanations, factual and counterfactual generators and explanations based on the association between semantic concepts and specific neurons. Then, we present explainability methods that leverage graph structures and thus are more similar to ours. Finally, we discuss other works that integrate external domain knowledge with deep representations in order to generate semantic explanations.

2.1 Global Explanations

The problem of understanding and explaining the decision making process of neural networks has been explored by researchers since the 1990s when the models were still shallow (models with only a few hidden layers) and consisted of only fully connected layers. Even with these simplifications compared to Deep Neural Networks (DNNs), providing meaningful explanations for such models proved to be a challenging task.

One of the first approaches proposed in [4] measured the contribution of input variables to the output layer to provide explanations for the model. This approach can be extended for measuring the contribution of neurons in the hidden layer [73] but, because it can only compute the contributions for neurons in fully connected layers, it cannot be directly applied to modern architectures such as convolutional or recurrent neural networks.

After the development of more modern deep architectures, different strategies for explaining DNNs beyond neuron contribution were developed. One such strategy is based on rule extraction and the goal is to extract sets of rules that are sufficient to explain the decisions of complex black box models. A comprehensive review of these methods can be found in [43]. There are three main types of rule extraction methods: decompositional, pedagogical and hybrid or eclectic strategy. Methods based on a decompositional strategy extract rules by examining activation and weights in the neural networks [14, 60, 62] but they also target only fully connected layers and cannot be applied to CNNs. The pedagogical strategy uses the decisions made by DNNs to extract rules without exploring the model architecture [55]. Unlike methods based on decompositional strategy, pure pedagogical methods can be applied to any neural network because they do not rely on the architecture. As a drawback, however, these methods are not able to explain the inner working of hidden layers as they use the DNN as a black box. It is also possible to use a hybrid of these two approaches, which is the eclectic strategy [75, 9], but current works apply the technique only on top of fully-connected layers.

Another common approach to explain DNNs is to use visualisation techniques [25, 13]. Global visual explanations can help understanding, for example, the role of convolutional filters or entire convolutional layers by looking at the visual representations they provide. It is also possible to understand which visual patterns are usually more relevant for each output class. Extensive surveys of visualisation methods are provided in the literature [40, 80, 15]. Despite visualisation techniques playing an important role in understanding the behaviour of DNNs, most visual analyses require human interpretation, which limits their scalability.

Global explanations can also be provided in terms of the semantic concepts that impacts the prediction of each output class. One of such methods is TCAV [59], which was developed to identify the most important concepts for predicting each output class in a classification problem. TCAV was able to interpret classification outcomes in terms of semantic concepts but it did not work well when such concepts are correlated with each other, which often occurs in real-world image datasets. This issue is addressed by the authors in [37], where they develop a method to detect cause-effect between concepts and the models' predictions. In [104] the authors propose a way to discover concepts and measure the importance of each concept for each prediction. Although useful, interpreting such cause-effect relations between concepts and output classes does not allow for a direct quantitative comparison between models, since each concept needs to be measured individually.

Using a different approach, [7] explores how classes are hierarchically organised by CNNs. Authors first use a visual approach to discover hierarchical relationships between classes and then propose a hierarchical-aware CNN architecture. This work is the most related to ours in the sense that we also explore hierarchical structures. However, while their method requires visual and interactive analysis, our approach extracts the hierarchical relationships automatically from the model, which is important in order to provide a quantitative assessment of the quality of the semantic structure learned by a CNN. In addition, our approach for taxonomy extraction explores neural relations within the CNN internal representation, as opposed to the work in [7] which relies only on the confusion matrix. This makes our approach potentially suitable for interpreting different layers in the model and understand the role of such layers in the overall decision making process.

2.2 Local Explanations

When it comes to supporting local explanations for the outcomes of a CNN, a common practice is to use visualisation techniques to highlight parts of the image that contributed the most for that outcome. Differently from global visual approaches, the methods for local explanations focus on providing visual clues for explaining each single image and not entire class representations. There is an extensive number of works around visualisation techniques, as well as several surveys that summarise the main approaches in this category [40, 80, 15]. Despite their popularity, visual explanations, need to be interpreted by a human case by case, which limits the scalability and can add human bias to the process. Saliency maps, for example, do not reveal the model's inference process, making them hard to interpret or misleading, especially when they refer to a mistake such as a misclassified input [100].

A more flexible way to analyse locally the outcome of a Deep Neural Network is by using case-based approaches, which have been widely used also to interpret the network's mistake. In this type of techniques, an explanation can be derived by comparing the mistaken input data with its closest neighbors. Case-based approaches have been used to generate factual and counterfactual explanations on both tabular data [90] as well as for image classification [56], but their explanations is solely based on the input feature space. In addition, case-based explanations also require human interpretation especially when dealing with highly unstructured data such as raw images, thus being time-consuming and prone to misinterpretation and human bias. Surrogate models, such as LIME [83], suffer from the same issue when it comes to explanations for computer vision. Although they can provide high-level feature based explanations for tasks based on tabular data, their explanations in computer vision are still provided in the pixel level, which leads to the same limitations as the methods above.

By using a phrase-critic model, authors in [45] provide a method to extract factual and counterfactual explanations for image classification models. Their method however makes use of auxiliary black box models, like LSTMs, adding an additional level of opacity in the explainability process.

In [6], network dissection is introduced and used to identify neurons that are responsible for detecting semantic concepts for object detection tasks. After mapping neurons to semantic concepts, they show that explanations around such concepts can be achieved by analysing the behaviour of the respective neurons for a new image. In [27] this idea is extended, and instead of mapping individual neurons to semantic concepts, the authors relate them with patterns of filter activities. The main difference between these works and the one we proposed in this thesis is that they rely on pixel-level segmented datasets, such as BRODEN, and their ability to produce meaningful explanations is limited to the concepts existing in the dataset. Although the BRODEN dataset contains a variety of concepts, it mainly contains information about objects and it is not suitable for supporting explanations based on semantic attributes of such objects.

We show that our method works even for fine-grained image classification tasks, in which the amount of annotated data is limited and an auxiliary dataset such as BRODEN is not available. In addition, although [6, 27] make use of semantic annotations, none of the above methods is able to leverage external knowledge bases to generate explanations, which is another desirable property of our approach.

2.3 XAI approaches using graph structures

The potential of using graphs for explaining the behaviour of a trained deep neural network can be summarised by two key motivations. The first is the vast amount of well-known graph methods that can help when analysing complex systems from many different domains, as mentioned in Section 1.3. The second motivation is that background domain knowledge is often described using graph structures, such as knowledge graphs. Therefore, finding a suitable intermediate graph representation may be a first step to integrate the hidden knowledge from deep learning systems with background knowledge in order to provide knowledge-informed explanations.

A deep learning model can be seen as a graph structure, called computational graph, where the nodes in this graph are artificial neurons linked by directed edges that describe mathematical operations between them. Such a computational graph is used merely to compute arithmetic operations and thus it is not commonly studied by means of graph theory methods, especially because of its layered and acyclic topology which would prevent the discovery of meaningful insights.

Analysing deep architectures using graph theory methods therefore requires extracting a suitable intermediate graph representation in a way that the resulting graph reflect somehow the knowledge contained within the deep representation and at the same time can lead to interesting insights when used as input to graph algorithms.

An existing approach that extracts a graph structure from CNNs is called explanatory graphs [106]. The authors propose a method that disentangles object-part patterns for each convolutional filter in a CNN. The patterns and their spatial relationships are represented in a graph, which helps revealing patterns that activate together as well as which filters are responsible for detecting them. The explanatory graph however is still a layered, acyclic and bipartite graph that serves as a dictionary of object parts, and thus not suitable for graph algorithms and metrics.

In [33], authors propose a graph representation based on vector embeddings that can represent both data instances and convolutional filters in the same structure. In their preliminary experiments, they show how community detection algorithms over the proposed graph can be used to find clusters of images based solely on the graph topology. One issue that remains is that their representation is still layered, acyclic, directed and bipartite when considering only the nodes and connections related to neurons in the hidden layers, which limits the potential of graph methods.

A third representation, called relational graph, is proposed by the authors in [105]. The relational graph is constructed based on a message passing strategy and the resulting graph differs from a computational graph since it is not layered, acyclic, directed nor bipartite. The objective of the relational graph however is to provide insights on the predictive performance of different deep architectures, and not on generating global or local post-hoc explanations.

The graph representation proposed in this thesis, called co-activation graph, differs from the existing alternatives in two ways. First, the co-activation graph enables the use of any method from graph theory or network science, since it resembles the topology of complex systems, such as brain functional graphs, instead of computational graphs. The second is that a co-activation makes it possible to combine neural knowledge with symbolic knowledge via input data and class labels, as reported in [48] and detailed in Chapter 5.

2.4 XAI approaches using external domain knowledge

Most of the methods presented so far are designed to provide explanations based only on the information that is available for the deep model during the training phase or at inference time. However, there is an increasing amount of domain knowledge available in existing knowledge bases such as DBPedia or Wikidata, which could in principle be leveraged when generating explanations, especially if we want them to be self-explanatory and directly understandable in human terms. Not only that, but there are also several annotated datasets, such as CoCo and CUB-200, that contain relevant semantic information about each data instance, which could be leveraged by explainability methods. Despite that, integrating external domain knowledge with deep representations is still an open challenge, especially due to the opaque and black box nature of deep learning models.

A first step towards this integration was proposed in [88], where the authors use the Suggested Upper Merged Ontology (SUMO) along with a DL-learner to generate explanations for image classifiers. Authors in [2] also propose an ontology-based reasoning approach to explain decisions made by a convolutional autoencoders for satellite images. More recently, [26] proposed a method that generates counterfactual explanations for image classifiers using external domain knowledge. The authors show how their method can generate both local and global explanations using semantic concepts. One issue shared by all the above methods is that they only consider the information in the input and in the output layer of the deep architecture, i.e, they do not leverage any information contained in the hidden layers.

Two comprehensive surveys of works that use knowledge graphs and semantic technologies for XAI are [93, 92]. In these surveys, the authors present several challenges and opportunities for knowledge-informed explainability methods that are tightly connected to the objectives of this thesis. The first opportunity is referred as automated knowledge extraction from graphs and it emphasizes the importance of providing novel graph-based methods and network analysis for the explainability task, which is widely explored in this thesis. The second relates to understanding the human role in the explainability process. This connects highly with our scope in this work since most of all explanations are human-centered and designed to facilitate how humans perceive such explanations. The third point refers to capturing meaning in the form, for example, of causal relationships, which connects especially with the methods developed in Chapter 5 where we generate local explanations based on the impact of semantic attributes in misclassifications.

Our work differs from the existing ones especially because it creates a novel representation that connects all parts of the deep models with existing domain knowledge. Because of this, the explanations generated by our method are produced based on the neural interactions, which may help, for example, in revealing what parts of the model are responsible for specific behaviours.

Chapter 3

Global Explanations using Knowledge Graphs

3.1 Introduction

This chapter introduces a new method to extract and represent knowledge from trained DNNs in order to better understand how the hidden part of the model works. Inspired by the use of functional graphs in the field of Neuroscience, we propose a graph representation, called co-activation graph, that connects every pair of neurons in a DNN based on statistical correlations of their activities. This chapter provides a formal definition of the co-activation graph and a step-by-step procedure to clarify how it is constructed.

We can summarise the key contributions of this chapter as follows: (i) we provide a formal definition of co-activation graphs so that we can identify properties and graph analytics processes for extracting meaningful knowledge from the deep learning model **(RQ1)**; (ii) we evaluate the proposed graph representation by analysing if there is an association between the statistical relationships encoded by co-activation graphs and the neural behaviour of their respective DNNs **(RQ2)**; (iii) we apply graph analysis beyond class similarity, for detecting groups of neurons that work together to predict similar classes on models and datasets at different levels of complexity **(RQ3)**; (iv) we explore and apply the notion of graph centrality to detect central nodes that represent the most important neurons in hidden layers, which may indicate possible biases **(RQ4)**.

3.2 (RQ1): Co-activation Graph: Definition and Construction

A co-activation graph is a graph representation that connects every pair of neurons of any type (fully connected or convolutional) and located in any layer of the neural network. This section provides a formal definition of the co-activation graph and recalls the general idea on how it is constructed.

Nodes of the co-activation graph correspond to neurons in the DNN and weighted edges represent a statistical correlation between them based on their activation values. We refer to this graph as a *co-activation graph*, since the relationships between its nodes represent how their activation values are correlated. The main idea of the co-activation graph is to create a relation between pairs of neurons in any depth of the hidden layers and neurons in the output classes, since the latter are more comprehensible for humans.

Definition 1 *A co-activation graph can be represented as an undirected graph* $G = (V, E)$ *where* $V = \{v_0, v_1, ..., v_n\}$ *is the set of* n *nodes that represent the neurons in the DNN and* E *is the set of weighted relationships (edges)* $e_{ij} = (v_i, v_j, w)$ *between pairs of neurons* v_i *and* v_j *with weight* w_{ij}*, considering* $0 \leq i < n$ *and* $0 \leq j < n$*. The weight* w_{ij} *represents a statistical correlation between the activation values of* v_i *and* v_j*.*

Given a data sample S, the first step to calculate w_{ij} is to feed the DNN with S. Then extract the sets of activation values $A(v_i, S) = \{a_{i0}, a_{i1}, ..., a_{in}\}$ and $A(v_j, S) = \{a_{j0}, a_{j1}, ..., a_{jn}\}$, where a_{i0} is a single activation value of neuron v_i for a single data input $s_0 \in S$. The weight w_{ij} is then obtained by applying a statistical correlation on $A(v_i, S)$ and $A(v_j, S)$, as shown in Equation 1:

$$w_{ij} = Spearman\_corr(A(v_i, S), A(v_j, S)). \tag{3.1}$$

We chose to use Spearman coefficient since we do not expect linear relationships between neurons' activation values. Therefore, edge weights vary in the range of [-1,1]. To clarify this process we recall the three steps below on how to generate a co-activation graph for a given trained DNN. For this, consider a DNN with n neurons and a data sample $S = \{s_0, s_1, ..., s_m\}$.

1. **Extract activation values**: The first step to build a co-activation graph is to feed the given DNN with S. Then, for each neuron v_i and each data input $s_h \in S$ where $0 \leq h < m$, extract a single activation value a_{ih}. The result is the set $\{A(v_0, S)$,$A(v_1, S)$,...,$A(v_n, S)\}$, where $A(v_i, S)$ represent all activation values of each neuron v_i for the whole data sample S.

   For dense layers this process is straightforward because each neuron outputs a single activation value. Filters in convolutional layers will output multiple values since they have different activation values for each region in the input. To overcome this and make our approach work for convolutional layers, the average pooling technique is applied to extract a single value for convolutional filters. Although some spatial information is lost in this process, feature maps tend to be sparse as most activations are zero after being passed through a ReLU activation function [12]. Therefore, in this thesis we use the spatial pooling strategy since it allows representing convolutional filters in the graph while keeping the dimensionality low [31].

2. **Define and calculate edge weights**: After collecting the activation values $A(v_i, S)$ for each neuron v_i in the DNN, the next step is to define the relationships between pairs of neurons. For each pair of neurons v_i and v_j, Equation 1 is applied using the activation values $A(v_i, S)$ and $A(v_j, S)$ to calculate a statistical correlation that will define the relationship weights w_{ij} between each pair of neurons. The result of this step is a matrix containing weights w_{ij} for

every neuron pair v_i and v_j that can be used to construct the set of edges E.

3. **Build and analyse the co-activation graph**: Given the set of edges E that describes the relationship between every pair of neurons, the co-activation graph can be built and analysed using any suitable computational tool for graph structures. In this work, we chose to represent the co-activation graphs in the graph database Neo4j[1] to facilitate data persistence and enhance reproducibility. The result of this final step is a graph where nodes represent neurons of any layer in the DNN and weighted edges indicate the correlation between their activation values. Our evaluation in Section 3.3 demonstrates that this graph correctly encodes the knowledge contained in the hidden layers of the DNN and we can then explore the graph structure and use graph analysis tools to understand relationships between neuron pairs in hidden layers and output classes.

It is easy to imagine how the result of this process is a very dense graph where each possible pair of nodes is connected. Analysing and visualising such a graph would be both difficult and computationally expensive. However, it is possible to define a threshold to remove relationships with small weights. Although we do not provide a systematic way for determining the threshold, it is feasible to define it empirically given that the relationship weights vary in a constrained range of [-1,1] and can be interpreted as statistical correlations. By doing this, the resulting process will be a less dense graph where only relationships with significant positive correlations are kept.

The formal definition together with the step-by-step procedure on how to build co-activation graphs help answering **(RQ1)**, since it is a systematic approach to build graph representations based on statistical dependencies between neural activities. In next section, we validate the approach based on co-activation graphs to interpret DNNs by conducting experiments on models and datasets of different levels of complexity.

[1]https://neo4j.com

3.3 Evaluation: Explaining DNNs through Co-activation Graphs

To evaluate our approach, three experiments were conducted. This section first introduces the model architectures, datasets and how the experiments were conducted. Then, a two-part evaluation is performed. The first part aims to understand how well co-activation graphs can represent the knowledge contained in DNNs, thus tackling **(RQ2)**.

In the second part, we begin to show how the use of graph theory methods over co-activation graphs can help interpreting DNNs. For this, we first use a community detection algorithm to detect groups of neurons that act together for predicting similar classes in the DNN, which tackles **(RQ3)**. Then, we study the presence of central nodes in a co-activation graph and show that they can be used to automatically detect important neurons in the DNN, thus addressing **(RQ4)**.

3.3.1 Building the Co-activation Graph: Datasets and Neural Architectures

Three well known datasets were used for the experiments: *MNIST handwritten digits* [68], *MNIST fashion* [103] and *CIFAR-10* [63]. For the MNIST variants, two shallow DNNs were trained from scratch. For CIFAR-10, a state-of-the-art model was used, as detailed below.

MNIST-handwritten and MNIST-fashion

The classes in *handwritten digits* dataset refer to digits from 0 to 9 and classes in *fashion* dataset are related to clothes. Both datasets contain ten classes, with 60000 training images and 10000 testing images. The DNN used for *handwritten digits* contains two convolutional layers and three fully connected layers and the DNN used for the *fashion* dataset has three convolutional layers and two fully connected layers. The Adam optimiser was used with learning rate 0.002. After training for 10 epochs, these models achieved an accuracy greater than 97%. Although it is possible to obtain higher accuracy with different architectures, we chose to use these models since we also want to analyse the reason behind mispredictions.

The co-activation graph was built for each DNN following the three steps discussed in Section 3.2. We first fed the DNNs with data samples from the testing set to extract activation values for each neuron. Then, we calculated the Spearman correlations between those neurons and built a co-activation graph for each DNN. In order to keep only relevant relationships in our graph, we applied a threshold of 0.3 so that only neurons with some significant correlation are connected. This threshold value was chosen empirically based on two observations. The first is many graph methods are not designed for negative edge weights, such as the Louvain and PageRank algorithm and there only connections with a significant positive correlation were kept. In second, if was observed that a higher threshold would separate some output classes into different graph components. Therefore, a threshold of 0.3 was chosen in order to maintain only significant positive correlations without partitioning the graph into multiple components.

CIFAR-10

The *CIFAR-10* dataset contains ten classes: *Dog; Cat; Horse; Frog; Bird; Deer; Airplane; Ship; Truck; Automobile.* There are 50,000 training images and 10,000 testing images. The DNN used for *CIFAR-10* was the *MobileNetV2* [87] that was designed to achieve state-of-the-art results on the challenging *ImageNet* dataset.

The *MobileNetV2* architecture contains an initial convolutional layer, 19 residual bottleneck layers followed by another convolutional layer and a fully connected layer. The model used in this experiment achieved 94.43% accuracy on the *CIFAR-10* dataset.

To build a co-activation graph for this model the first step was to extract the activation values from each neuron in every layer. For the convolutional and fully connected layers we extracted the activation values using the same approach described in Section 3.2. For the bottleneck layers we had to adapt our strategy as these layers are formed by depthwise and pointwise convolutions. In this case we collected the RELU activation values from the last pointwise convolution, since this is the last transformation that is performed on each bottleneck layer.

After collecting the activation values we calculated the Spearman correlation between each pair of neurons and connected them to generate the co-activation graph. Finally, we applied the same threshold of 0.3 to keep only relevant relationships in our graph. With these settings the *MobileNetV2* co-activation graph for *CIFAR-10* contains 4,012 nodes and 184,144 edges.

3.3.2 (RQ2): Evaluating the co-activation graph representation

After constructing a co-activation graph for each of the three models and datasets described above, the first part of our evaluation was conducted to analyse how well these graphs are being able to represent knowledge from their respective DNNs. This first evaluation addresses **(RQ2)** and, for this, we began by analysing if is there an association between the statistical relationships encoded by co-activation graphs and the neural connections of their respective neurons in the DNN.

In a DNN, each neuron in the last hidden layer has a neural connection with each output class. The respective co-activation graph also contains relationships between these neurons and classes, but with a different meaning, which is a statistical correlation between their activation values. Then, the first step in this analysis is to check if the statistical correlations encoded by the co-activation graph are compatible with the neural connections.

By querying the co-activation graph it is possible to collect, for each class, a set of its most correlated neurons based on their relationship weights. Since strong relationships indicate high positive correlations in the activation values, it is expected that when these highly correlated neurons are activated in the DNN, they will have a high contribution to increase the prediction value of their classes. For example, if we pick from the *MNIST-fashion* co-activation graph the neurons most correlated to a class *sandal* and activate them, it is expected that the DNN will output *sandal* as a prediction. However, it is important to note that, activating a single or just few neurons might not result in the expected class, since the absence of the other neurons can have a high influence in the final prediction.

Considering this, we extracted and manually activated the k most correlated neurons for each class. These neurons were extracted from the first fully connected hidden layer after the convolutional layers, which is not directly connected to the output and thus more difficult to interpret. Since activating a single neuron ($k = 1$) could not be enough to predict the expected class, we tested multiples values of k to see what is the minimum number of neurons we need to activate in order to predict the expected class. We expect that when activating the *top* k neurons (k most correlated neurons), the model will output a prediction with the respective class. Figure 3.1 shows the results for *handwritten* and *fashion* datasets. It is possible to see that, for the *fashion* dataset, when we activate the *top* 7 neurons we get the expected prediction for all the classes. For *handwritten* dataset, the model predicts the expected class 80% of the time.

This result indicates that we can query the co-activation graph to find neurons with a high impact over the prediction values for each class, which is a first evidence towards **(RQ2)**.

After this first evidence that the co-activation graph is being able to represent DNNs knowledge, we performed a second step in the analysis to understand more about the association between co-activation graphs and DNNs. We investigated the similarity between classes in co-activation graphs to see if there is an association between similar classes in co-activation graphs and pairwise mistakes in the DNN predictions. For this, we first calculated the jaccard similarity coefficient for each pair of classes in the co-activation graphs. The jaccard coefficient was calculated based on the overlaps (shared nodes) between each pair of classes. Then, we collected the number of mistakes between them by counting for every pair of classes A and B

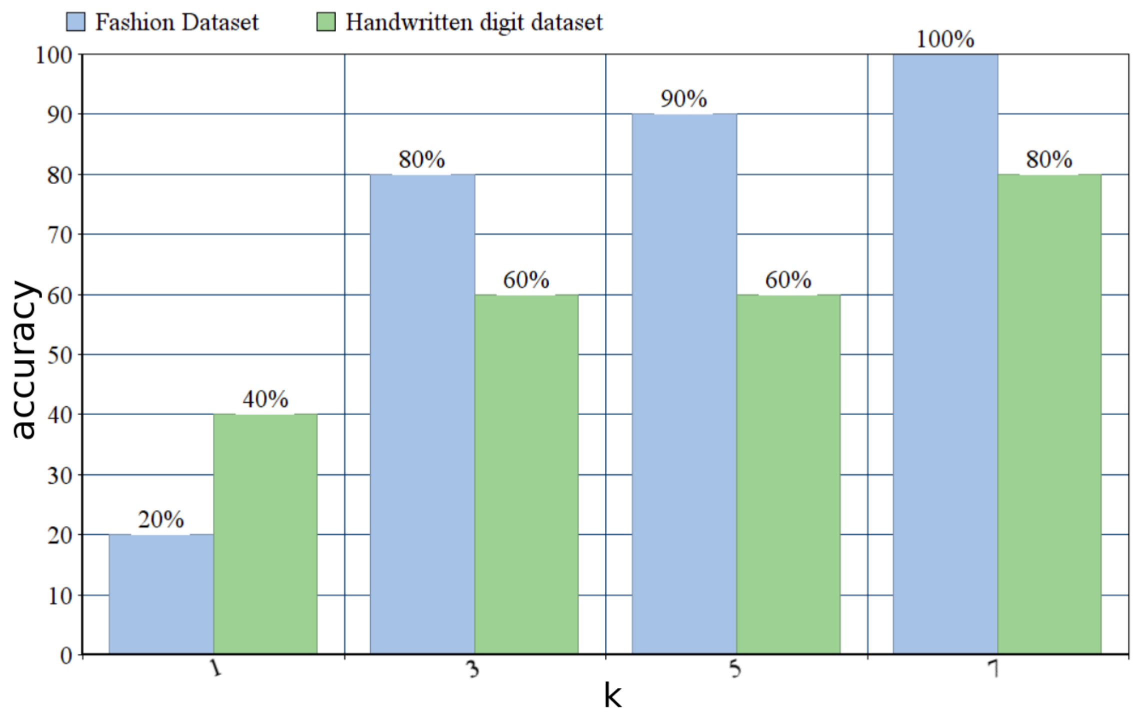


Figure 3.1: Result acquired after extracting *top k* most correlated neurons for each class in the co-activation graph and activating then in the DNN. The accuracy indicates how many times the expected class was predicted.

how many times the model predicted the wrong class A when the the correct answer was B, or the other way around.

Figure 3.2 shows that there is a positive correlation between class similarity and the number of mistakes involving them, showing that, pairs of classes with many overlapping nodes in the co-activation graph tend to cause a high number of mistakes in the DNN. It also suggests that overlapping nodes between two classes might be the origin of mistakes in the DNN, which is an initial step towards **(RQ3)**. We follow this intuition in next subsection to study more about class similarity and origin of mistakes.

The strong correlation between class similarity in co-activation graphs and pair-wise mistakes in the DNN is a second evidence towards **(RQ2)**, since it shows that the co-activation graphs are in some way representing the knowledge acquired from the DNNs. Next, the second part of the evaluation process is performed to show how the use of graph theory methods over co-activation graphs can help interpreting DNNs.

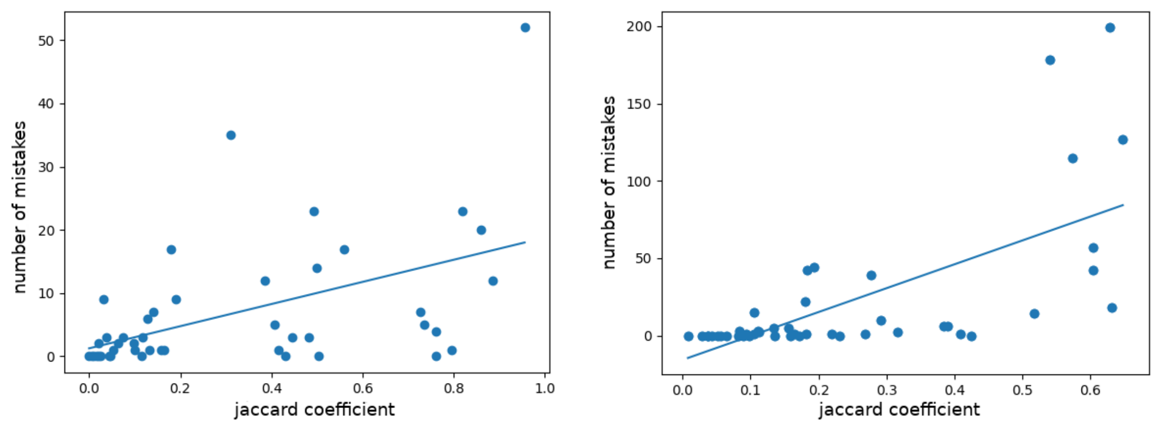


Figure 3.2: Correlations of mistakes between two classes and their jaccard similarity. The Spearman correlation is 0.5042 for handwritten digits (left) and 0.6625 for fashion (right)

3.3.3 (RQ3): Interpreting DNNs' mistakes using Community Detection Analysis

The presence of community structure in graphs and networks is a key aspect worth being investigated to understand interesting properties of the graph itself and the knowledge it represents. When applied over co-activation graphs, communities represent groups containing neurons and classes that tend to activate together in the DNN. We analysed the community structure of the three co-activation graphs (*fashion, handwritten and CIFAR-10*) to see if it can help detecting classes that are similar from the DNN point of view, which would help answering **(RQ3)**.

The Louvain community detection algorithm [8] was chosen because, besides being a well established algorithm, it outputs a modularity coefficient that indicates how the community structure differs from random graphs. The value of modularity varies in range [-1,1] and higher values indicate that connections between nodes in same community are stronger than nodes in different ones. The only required parameter for the Louvain algorithm is the resolution, which can be adjusted to detect a higher number of smaller communities or a lower number of bigger communities. For this experiment, the resolution was set to the default value 1.0.

Table 3.1 shows the detected communities and classes contained in each of them, for *fashion* and *handwritten* co-activation graphs. For the *fashion* dataset it can be noted that classes in same communities have a similar semantic meaning. We can notice this because classes like *pullover* and *coat* were put in same community while *sandals* and *sneakers* are grouped in a different one, somehow reflecting the semantic alignment among classes grouped in the same community structure for that dataset. For the *handwritten* dataset it is not possible to conclude the same, since semantics of digits are less clear. It is important to note that for both datasets the modularity was higher than 0.4 which means that found communities are denser regions although they are not totally distinguishable.

These results are reported in our preliminary investigation on [49] which demonstrated to some extent that community analysis over the co-activation graphs can help identifying classes that are similar from the point of view of the DNN.

Table 3.1: Communities on MNIST Fashion and MNIST Handwritten.

Community	MNIST Fashion	MNIST Handwritten
C1	T-shirt/Top; Pullover; Coat; Shirt	0; 2; 4; 6
C2	Trouser; Dress;	5; 7; 8; 9
C3	Sandal; Sneaker; Bag; Ankle Boot	1; 3

We then extended and elaborated on the validity of this result by analysing the community structure of a deeper model (*MobileNetV2*) on a more complex dataset (*CIFAR-10*). As described before, the *CIFAR-10* dataset contains ten classes: *Dog; Cat; Horse; Frog; Bird; Deer; Airplane; Ship; Truck; Automobile*. As humans, it would be reasonable to separate these classes in two groups: Animals and Vehicles. To understand if *MobileNetV2* also organises these classes into different and semantically aligned groups, the community detection algorithm was run on *CIFAR-10* in the same way as done in [49] for the *MNIST* datasets.

After following the same procedure and applying the Louvain algorithm with default resolution, the detected communities for the co-activation graph on *CIFAR-10* can be seen from Table 3.2.

As expected, Animal and Vehicle classes were put in different communities showing once more that classes in the same communities are semantically aligned, and

Table 3.2: Communities on MobileNetV2 for CIFAR-10 with default resolution.

Community	Classes
C1	Deer; Dog; Horse
C2	Frog; Bird; Cat
C3	Airplane; Ship; Truck; Automobile
Modularity	0.489

confirming previous experiments. However, this first result also shows that the model separated Animal classes in two different groups, one containing {Deer; Dog; Horse} and the other containing {Frog; Bird; Cat}.

To further study these communities we analysed the similarity between them. In this case, it is reasonable to expect that $C1$ and $C2$ are more similar between each other than they are with $C3$, since the former contains only animals and the latter only vehicles. To verify this, we increased the resolution parameter to achieve fewer communities (i.e. bigger ones). We observed that when we set the resolution parameter to 1.2, $C1$ and $C2$ are merged by the Louvain algorithm into one single community, indicating that not only classes in the same community have a semantic similarity but also there is a semantic hierarchy reflected by the analysis as these merged communities are semantically aligned, as shown in Table 3.3.

Table 3.3: Communities on MobileNetV2 for CIFAR-10 with increased resolution.

Community	Classes
C1	Deer; Dog; Horse; Frog; Bird; Cat
C2	Airplane; Ship; Truck; Automobile
Modularity	0.489

Another important aspect to check in this analysis is how the communities' quality depends on the knowledge encoded within the CNN. In order to test this, we repeated the previous experiment but this time we built the co-activation graph for an untrained model with randomly initialised parameters. The results shown in Table 3.4 show that the communities detected for the untrained model do not

Table 3.4: Communities on an untrained MobileNetV2 for CIFAR-10.

Community	Classes
C1	Airplane, Cat, Dog, Ship
C2	Automobile, Bird
C3	Deer, Frog, Horse, Truck
Modularity	0.22

maintain the expected semantic alignment between their contained classes, which is reflected by the presence of both vehicles and animals in all three communities. This is an important evidence towards **RQ2** since it shows that the knowledge contained in the co-activation graph is only as good as the knowledge from its respective CNN.

Detecting these groups and having the evidence that they carry a semantic meaning is important for different reasons. First, mistakes are more likely to occur between classes that are semantically similar and therefore belong to the same community. This is also intuitive, since from the human's perspective misclassifications between two animals or two vehicles should be more frequent than mistakes between *Frogs* and *Trucks*, for example. Second, if we consider that neurons in convolutional layers work as feature extractors, it is plausible to expect that neurons for extracting animal parts are more likely to be found in $C1$, while vehicle parts are in $C2$. This claim however would require a more rigorous evaluation which will be considered in future work.

Although a more rigorous evaluation over the detected communities is required, the detected communities seem to represent groups of similar classes from the point of view of the DNN. This analysis gives a step towards answering **(RQ3)**, since analysing the neurons within these communities might reveal the origin of mistakes for classes with high similarity.

3.3.4 (RQ4): Interpreting DNNs bias using Node Centrality Analysis

In graph analysis, the centrality of a node provides valuable insights on how important this node is in a graph. The notion of importance depends on both the graph domain and the chosen centrality measure. In our context, the graph domain refers to neurons in a DNN, and the objective is to discover whether centrality measures can help revealing neurons that are more important for the DNN. If central nodes in co-activation graphs are somehow associated with important neurons in the DNN, this could serve as an automatic way to detect key steps in the DNN decision process, thus revealing possible weaknesses or biases and addressing **(RQ4)**. The following analysis was conducted to see if there is a notion of node centrality in co-activation graphs, and what is the association between central nodes and their respective neurons in the DNN. For this, consider that a specific neuron is redundant or less important if we can remove it from the DNN without having a significant accuracy drop.

Two different centralities are used for this analysis: Degree centrality and PageRank centrality. The Degree centrality of a node indicates the number of direct relationships this node has with other nodes. In the case of co-activation graphs, a node with high degree corresponds to a neuron that correlates with many other neurons. The PageRank centrality of a node indicates its importance by leveraging not only its direct relationships but also the importance of its neighbors.

To check if there is an association between node centrality and neuron importance, the following steps were performed:

1. Calculate node centrality in the co-activation graph
2. Pick the node with the highest (or lowest) centrality
3. Remove the respective neuron from the DNN and from the co-activation graph by zeroing its weights and bias parameters

4. Check the accuracy loss

5. Repeat

Table 3.5 shows the result of the above procedure for both Degree and PageRank centralities [79] for the three models: *CIFAR-10*, *MNIST-handwritten* and *MNIST-fashion*. In the case of PageRank centrality, it is possible to see that there is an interesting correlation between the PageRank centrality in the co-activation graph and the neuron importance in the DNN. This can be observed from the third column of Table 3.5, which shows that when we remove neurons ordered by higher PageRank, there is a strong accuracy loss in the DNN. In contrast, if neurons with lower PageRank are removed, the accuracy is stable for a longer period. This behaviour is consistent through all three datasets, and it indicates that analyzing PageRank centrality over co-activation graphs has the potential to reveal important neurons in the respective DNN.

The same consistency cannot be observed for the Degree Centrality. The second column of Table 3.5 shows that for *CIFAR-10*, nodes with high degree have a high impact when removed from the DNN. However, for *MNIST-fashion*, the opposite behaviour is observed. In addition, when analyzing the impact of Degree Centrality for the *MNIST-handwritten* dataset, removing neurons for either lower or higher degree shows a similar behaviour, indicating that Degree Centrality is not very informative for this model.

It is important to note that a node with a low degree might still have a high PageRank, and thus these results are not contradictory. Instead, this can be considered as initial evidence that a co-activation graph might be used as a strategy for pruning algorithms on pre-trained models, since removing a considerable amount of nodes with low PageRank caused only a minor accuracy drop in all three DNNs in this experiment.

The centrality analysis shows that there is an association between node centrality in the co-activation graph and neuron importance in the DNN, which is a step towards answering **(RQ4)**. This also represents further evidence that the knowledge

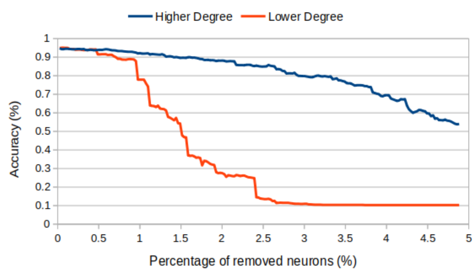
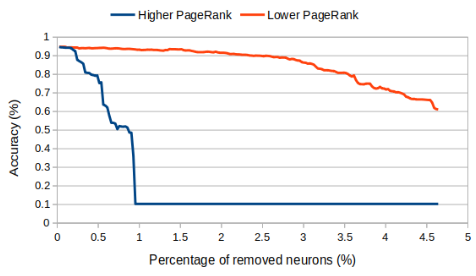
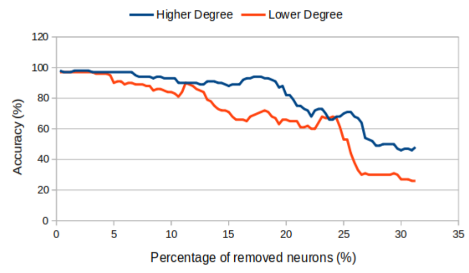
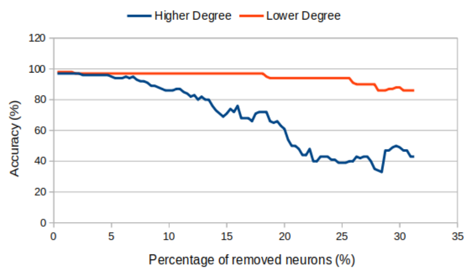
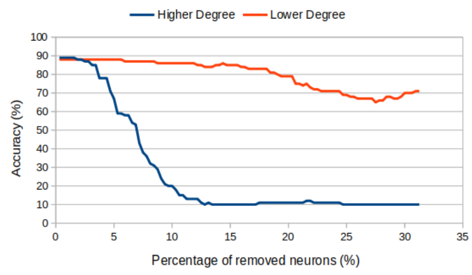
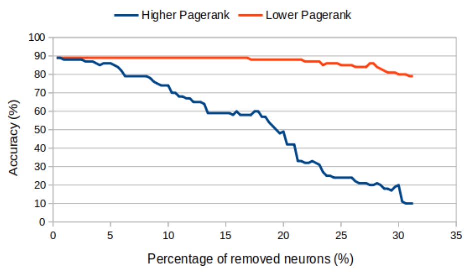
Dataset	Pruning by Degree	Pruning by PageRank
CIFAR-10		
Handwritten		
Fashion		

Table 3.5: Model accuracy after removing neurons based on their node centrality in the co-activation graph.

contained in the co-activation graph is compatible with the knowledge encoded in the DNN, improving our previous answers to **(RQ2)**. However, one current challenge is to understand the real meaning of each node centrality in co-activation graphs. In the next section, we discuss possible strategies to investigate this issue, as well as possible limitations of the proposed method.

3.3.5 Visualising Co-activation Graphs

Graph visualisations are useful to derive new insights from data and to formulate new hypotheses based on how the graph is visually distributed. This section shows how graph visualisations can help understand the results achieved by each of the graph theory methods applied previously and provides a visual way to exploit co-activation graphs to interpret the knowledge contained in the DNN. For this the *MNIST-*

fashion dataset is used as an example and visual representations are presented for the communities and node centralities that were obtained and analysed in Section 3.3.3 and Section 3.3.4, respectively.

When visualising a co-activation graph it is useful to distinguish nodes in the hidden layers from nodes in the output layers (target classes), since the latter have a clear interpretation for humans. In Figure 3.3 blue nodes represent neurons in the hidden layers while yellow and labelled ones represent the output classes. After distinguishing these nodes it is easier to visualise each step of the previous analyses.

From Figure 3.3, it is possible to see the impact of applying a specific threshold to the co-activation graph. After applying a threshold of 0.3 to the *MNIST-fashion* dataset, the density of the graph is visibly reduced if compared to a fully-connected graph but the graph still consists of a single component. It can also be seen from this figure that the visualisation algorithm, *ForceAtlas 2* [52], has placed some classes close to each other (e.g *Sandal, Sneaker* and *Ankle Boot*) intuitively indicating that the graph has a community-like structure, which is shown in more detail in Figure 3.4.

In Figure 3.4 the communities discovered in Section 3.3.3 are represented by different colors. From this community structure a first observation is that the node representing class *Bag* is quite close to the red community although it belongs to the green one. This could be an indication that the *Bag* class is at the intersection of these two communities and probably should not be considered as part of a single community. This behaviour could be captured by community detection algorithms that consider overlapping nodes, such as [22, 47], and we plan to investigate this further. In addition, classes *Dress* and *Trouser*, although they belong to the pink community were placed close to the red community by the *ForceAtlas2* visualisation algorithm. This means that they are probably more similar to classes *Coat, Pullover, Shirt* and *T-shirt* than to *Sneaker, Ankle Boot, Sandal* and *Bag*. In fact this observation corroborates with the results presented in Section 3.3.3 which showed that the community containing *Dress* and *Trouser* was merged with the red community

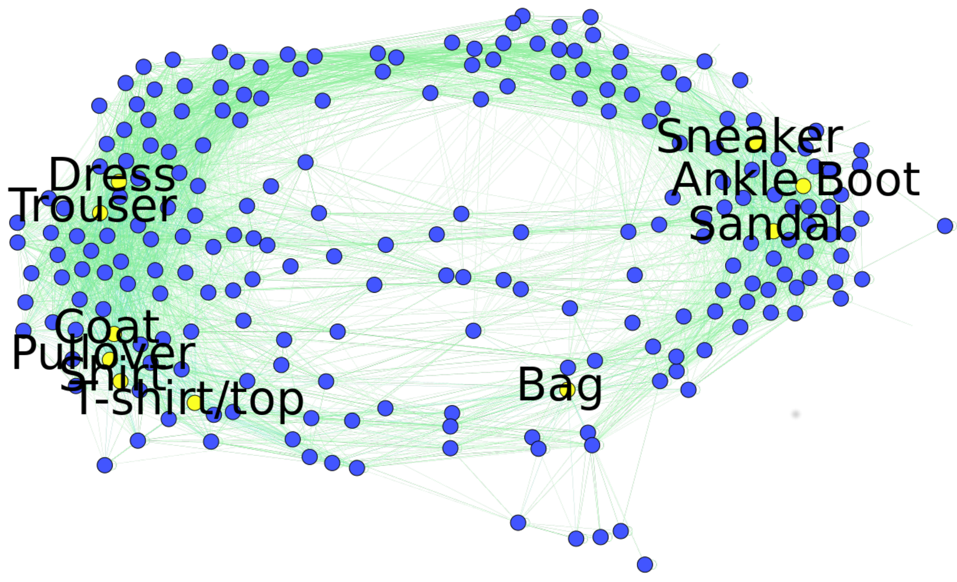


Figure 3.3: Visualisation for *MNIST-fashion* co-activation graph made on Gephi. Blue nodes represent neurons in hidden layers and yellow and labelled nodes represent neurons in the last layer (output classes).

when we increased the resolution parameters for the Louvain algorithm.

As a third step in the visual analysis Figure 3.5 illustrates nodes according to their PageRank centrality represented by the size of the nodes. We have already seen from Section 3.3.4 that central nodes tend to be important neurons in the DNN. Now, according to the visualisation, the nodes with high centrality seem to belong to dense regions in the graph while less central nodes are either peripheral nodes or nodes in less dense regions. This supports the intuition that, by removing the most central nodes, the model will suffer a considerable accuracy loss since large number of neurons (dense region) can be impacted.

The visualisations for the communities and central nodes also suggest that if we combine the community structure with the centrality analysis we might be able to automatically detect which neurons are more important for the prediction of classes within each of the communities. For example, if we remove central nodes related to the green community (big green nodes), this should result in higher impact on accuracy for classes in the green community (e.g *Sandal*) than classes in the red

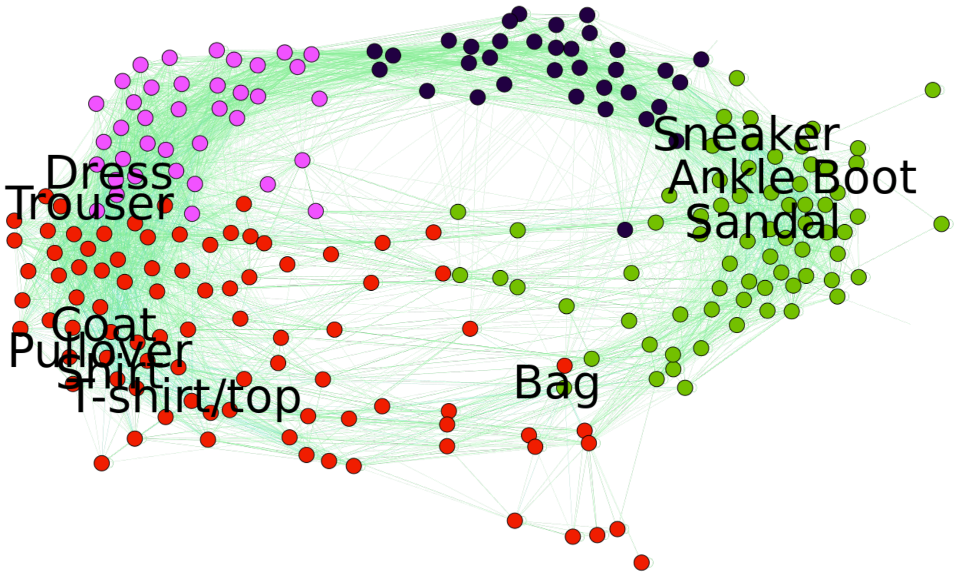


Figure 3.4: Visualising the communities detected by the Louvain algorithm. Communities are represented by different colors.

community (e.g. *Coat*). This is an intuition that should be further investigated.

We can see that by using visualisation techniques we can improve our understanding of the results obtained by the different graph analyses, like the community structure and the positioning of central nodes. This can be considered as another advantage of extracting knowledge from DNNs and representing it in co-activation graphs since each analysis performed in the latter can be evaluated and visualised in a transparent manner. In the next section, we present the limitations we have identified for the current approach and discuss possible strategies to overcome them.

3.4 Discussion

The evaluation conducted in Section 3.3 demonstrated that results obtained by our approach are consistent across datasets and neural architectures of different complexity. However, in this section we discuss some limitations of our approach that should be taken into account and indicate possible ways to overcome them. We will also discuss other interesting open issues worth considering in the next steps of

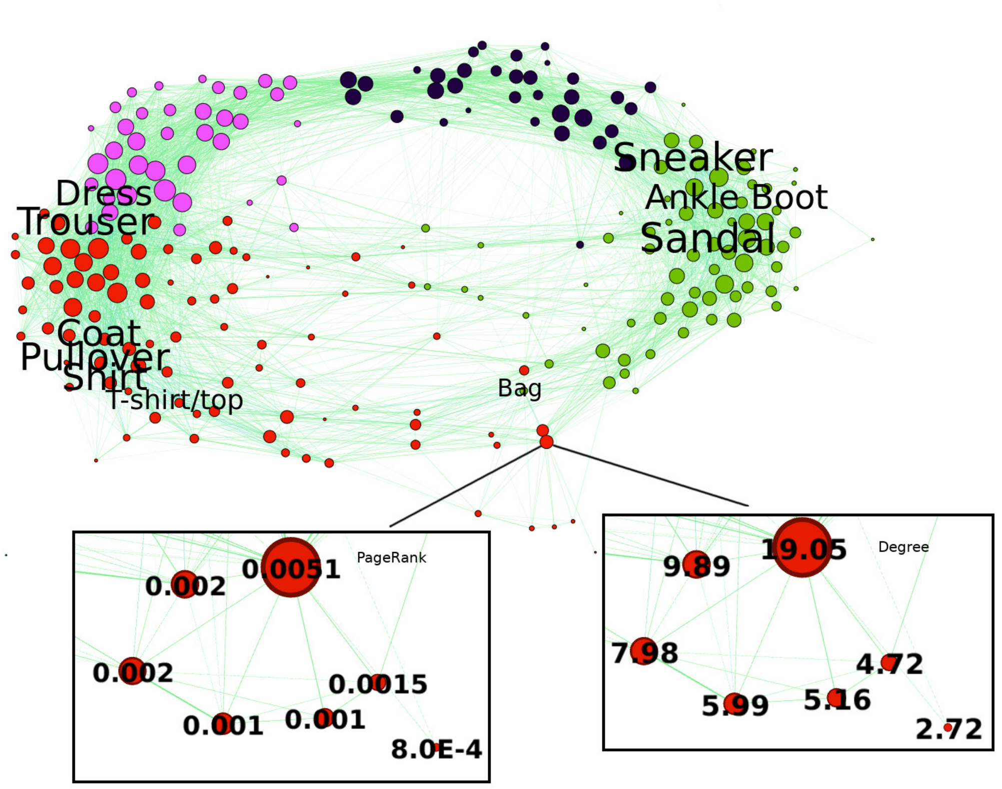


Figure 3.5: Visualising nodes according to their PageRank centrality. Bigger nodes represent the most central ones.

this research.

One limitation is that both the community and centrality analysis can be biased by the chosen threshold. Choosing this threshold is not a trivial task. A possible way around this would be to use a deeper understanding of the graph distribution to help define it.

A second limitation is the strategy used to extract activation values from convolutional layers. In our experiments we used spatial pooling. This makes it easier to extract and represent convolutional filters but it causes some information loss. One possible strategy to overcome this is to use multiple nodes to represent all possible slices covered by each convolutional filter.

A third point is that although the method is designed for any deep architecture we have evaluated it only on fully connected and convolutional layers. Extending the construction of co-activation graphs for other feed-forward architectures, such as transformers, would require a mechanism to extract activation values from neurons in layers such as the multihead attention layers. This could possibly be achieved by applying a global pooling at the output of either the attention layer or the last feedforward layer of the encoder, which is currently being investigated. Once this mechanism is established, the method for computing correlations based on activation values would remain the same. On the other hand, architectures that consider long term dependencies such as recurrent neural networks may present a bigger challenge, because the statistical correlations might not be sufficient to accurately represent interactions between neurons. In this case, future work should consider either representing the state of neurons without exploding the dimensionality of the graph or the use of dynamic graphs to represent interactions over time.

When comparing with alternative approaches it is important to note that the method proposed in this chapter aims to explain Deep Neural Networks (DNNs) from a novel perspective and thus a direct and quantitative comparison is not possible at the moment. However, we believe that both community and centrality analysis could be combined with other explanation techniques to generate mutual benefits for better understanding of DNNs. For example, visualisation techniques might use information about communities and node centrality to decide which neurons are more important and therefore should be analysed. Visualisation techniques might also help in understanding why communities were formed in this particular way as well as the role of central nodes. By combining the method proposed in this work with existing approaches to enhancing explainability (such as visualisation techniques) we expect to achieve richer and complementary explanations that may improve understanding of how our approach compares to existing ones such as [13, 78].

The observation that the detected communities contain classes with a similar semantic meaning is also first evidence that our approach might be used to integrate the knowledge acquired by DNNs with external knowledge. In this regard an interesting avenue for investigation is to use information from knowledge bases such as ConceptNet or DBpedia to identify properties shared among classes in a community. In our experiments, for example, even though our datasets contain a small number of classes, it would be possible to formalise that C1 is related to *dbo:Animal* while C2 is related to *dbc:Transport*. While this is a trivial example, Chapter 4 looks into using richer datasets such as CIFAR-100 [64] and ImageNet [20] to explore this knowledge linkage and how it leads to a more valuable community structure with potential for a richer semantic interpretation.

Finally, to include more semantic information in our approach we are considering the addition of connections between the input layer and the co-activation graph. This is important particularly due to the existence of datasets with contextualised images, such as CoCo [71] and ImageNet, and could potentially be done based on recent work in feature extraction [31, 33].

3.5 Summary

In this chapter, we propose the notion of co-activation graph [49, 50] to extract and represent statistical dependencies between neurons in DNNs, which addresses **(RQ1)**. In the co-activation graph, nodes represent neurons in a DNN and weighted relationships indicate a statistical correlation between their activation values. This representation connects neurons in any layer of the neural network, including hidden (convolutional and dense) layers and the output layer.

From our experiments, we acquired evidence that statistical dependencies encoded in the proposed graph capture the knowledge contained in the DNNs, which answers **(RQ2)** and supports verification of **H1**. In particular, we first demonstrated how we can find neurons that have a high contribution for predicting some classes. Then, we observed how classes that are highly overlapping in the co-activation graph are responsible for most mistakes in the model. This is an interesting finding as it indicates that overlapping nodes in the graph are the ones that should be operated on in order to improve the model's ability to better distinguish similar classes.

In addition, we started to verify **H2** by evaluating how our approach can help interpreting DNNs. For this, a community detection analysis was performed, showing that it is possible to identify groups of similar classes from the point of view of the DNN. This is an important step to understand the origin of mistakes and answer **(RQ3)**, since mistakes tend to occur between classes with high similarity and within the same community.

In addition, a node centrality analysis revealed that there is an association between central nodes in co-activation graphs and important neurons in DNNs. We showed that removing nodes with high PageRank centrality from the DNN causes a high accuracy drop in the model. This is an important step towards answering **(RQ4)** as it is an initial evidence towards finding possible weaknesses or biases within DNNs, although further investigation is required.

In the next chapter, we show how we can leverage knowledge from external sources into our approach: considering that communities detected in co-activation graphs already group together classes with high semantic similarity, we conduct experiments over ImageNet to see if there is a relation between groups of similar classes and the WordNet hierarchy.

Chapter 4

Extracting taxonomies from CNNs using Knowledge Graphs

4.1 Introduction

In the field of Explainable AI (XAI), the issue of evaluating and comparing deep learning models beyond their accuracy over a test set has been tackled by methods that provide global interpretations. Examples include detection of concept importance [59, 35] and class hierarchy visualisation [7]. While providing some form of explanation, methods like these can not be sufficient for an objective comparison between different models, since the resulting interpretations do not provide a metric that compares CNN decision making with some external semantic ground truth.

In this chapter, we discuss an approach to semantic interpretability of CNNs, based on how hierarchical relationships between semantic concepts are captured by the model's internal representation. To this aim, we propose a taxonomy extraction method to derive a domain taxonomy from a trained CNN. For example, for a model trained over the ImageNet dataset [21], our method is capable of extracting taxonomic axioms such as ($GermanShepherd \subset Dog \subset Mammal \subset Vertebrate \subset Organism \subset Entity$).

By comparing such taxonomic axioms with a relevant ground truth class hierarchy (e.g., WordNet for ImageNet) we can evaluate quantitatively how faithful the extracted taxonomy is to the selected groundtruth. In the current context this metric is referred to as *semantic adequacy* and its purpose is to evaluate the entire extracted taxonomy with respect to the groundtruth. Note that, independent of model accuracy, a higher semantic adequacy score implies better transparency by allowing interpretation of model behaviour in terms of user-accepted concepts and their taxonomic relationships. Moreover, it makes objective semantic comparison between models possible, provided that their training data is drawn from classes in similar or related taxonomies.

In order to achieve these goals, we build upon the idea of co-activation graph in Definition 1, a data structure based on correlation coefficients of neuron activation values. Semantic notions are introduced by relating CNN output neurons to nodes in the graph.

We combine co-activation graphs with a taxonomy extraction method originally designed for knowledge graphs by [74]. An overview of the approach can be seen in Figure 4.2. Given the co-activation graph of a trained CNN, and following [74], we calculate node embeddings for the nodes in the co-activation graph using well known methods such as Node2Vec [39] and FastRP [16]. We then run a hierarchical clustering algorithm over the node embeddings and extract the CNN taxonomy. Finally, the resulting taxonomy can be compared to a ground truth taxonomy to measure the semantic adequacy of the CNN classifier. The main contributions of the chapter are:

- A vector representation for semantic relationships of output classes, obtained by co-activation analysis of a trained model. Such class embeddings are general purpose representations. We use them for taxonomy extraction by means of hierarchical clustering **(RQ5)**.

- A method for taxonomy extraction from a trained model. The resulting taxonomies provide symbolic explanations for sub-symbolic decision making **(RQ5)**.

- An evaluation metric for comparison between extracted and ground truth taxonomies, measuring what we call the semantic adequacy of a model. In turn, this metric allows us to compare different models in semantic terms, instead of merely their performance metrics **(RQ6)**.

The rest of this chapter is organised as follows. Section 2 presents background knowledge that is key to the development of the proposed method. Section 3 elaborates on the methodology for extracting taxonomies from co-activation graphs built for CNNs. In Section 4 an experiment using CNNs trained for CIFAR-100 and ImageNet is conducted. In Section 5 we present our conclusions and discuss the obtained results.

4.2 Background Knowledge

4.2.1 Semantic Global Interpretation of CNNs

In XAI, global interpretations aim to provide insights into the behaviour of a model as a whole, instead of explaining individual inference events. E.g., TCAV [59] identifies the most important concepts for predicting each output class in a classification problem. TCAV is able to interpret classification outcomes in terms of semantic concepts but fails to work well in treating correlations between concepts, often present in real-world image datasets. This issue is addressed by [37], with a method to detect cause-effect relations between concepts and the model predictions. Then, [104] proposes a method to discover concepts and measure concept importance for predictions. Interpreting cause-effect relationships between concepts and output classes in this way is useful, but requires each concept to be measured individually. As a result it does not easily allow for a direct and quantitative comparison between models.

Using a different approach, [7] explores how classes are hierarchically organised by CNNs. A visual approach to discover hierarchical relationships between classes is combined with an hierarchical-aware CNN architecture. This work is closely related to ours in the way we both explore hierarchical structures. However, while their method requires visual and interactive analysis, our approach extracts the hierarchical relationships automatically from the model.

This is an important prerequisite for the assessment of the semantic adequacy: without automatic extraction of hierarchical information, the scale of modern CNNs makes semantic global interpretation infeasible. In addition, while [7] relies only on the confusion matrix, our taxonomy extraction explores the internal representations of a model. This makes our approach suitable for interpreting separate layers in the model and understanding the role of such layers in the overall decision making process.

4.2.2 Taxonomy extraction from graph structures

In complex domains, constructing taxonomies by hand is an expensive task. To address this, researchers have studied how to extract taxonomies automatically from data [101]. The most common approaches focus on extracting hypernym (Is-A) relationships directly from text [91, 19]. The goal is to organise a hierarchy of concept terms such that subtype concept relationships are reflected by the hierarchy tree structure while remaining faithful to observations from the corpus.

In recent years, some approaches have looked into taxonomy extraction from non-textual, structured and semi-structured data, such as knowledge graphs. Authors in [74, 84] propose an unsupervised method to find hypernym axioms. Vector embeddings for each node in the knowledge graph are calculated, resulting in a class centroid and radius in the latent space. Then, based on the distance between the centroids, they form axioms and construct their transitive closures to build a taxonomy.

While such methods are relevant to our work, we go further and focus on visual data. The unstructured nature of raw visual data is a formidable challenge for global XAI methods. We therefore intend to leverage knowledge captured by a neural network's co-activation graph as source data for our extraction method. Co-activation graphs are a suitable intermediate representation from which to extract semantic structures that function as knowledge graphs, e.g., as described in Chapter 3. In addition, we argue here that this use of co-activation graphs enables extraction of semantic taxonomies for understanding global model behaviour and XAI.

The main drawback of the approach in [84] is that concepts need to be directly represented as nodes in the knowledge graph. This is not generally the case in co-activation graphs, considering that the only explicit semantic information contained in the CNN and present in co-activation graphs are the class labels.

The method proposed in [74] can help overcome this issue. Based on node embeddings, their approach uses hierarchical clustering over the latent space to find a hierarchical structure. By mapping concepts (or types) to cluster-nodes in this structure, a typed tree is formed from which to construct a taxonomy. The advantage is that the clustering phase does not use any information regarding the types. This makes it more suitable in our setup, where these types are typically not represented by nodes in the graph.

Our focus is therefore on the novel application of node embedding to co-activation graphs and its combination with node embedding clusterings for semantic global XAI by means of taxonomy extraction.

4.2.3 Hierarchy-aware architectures

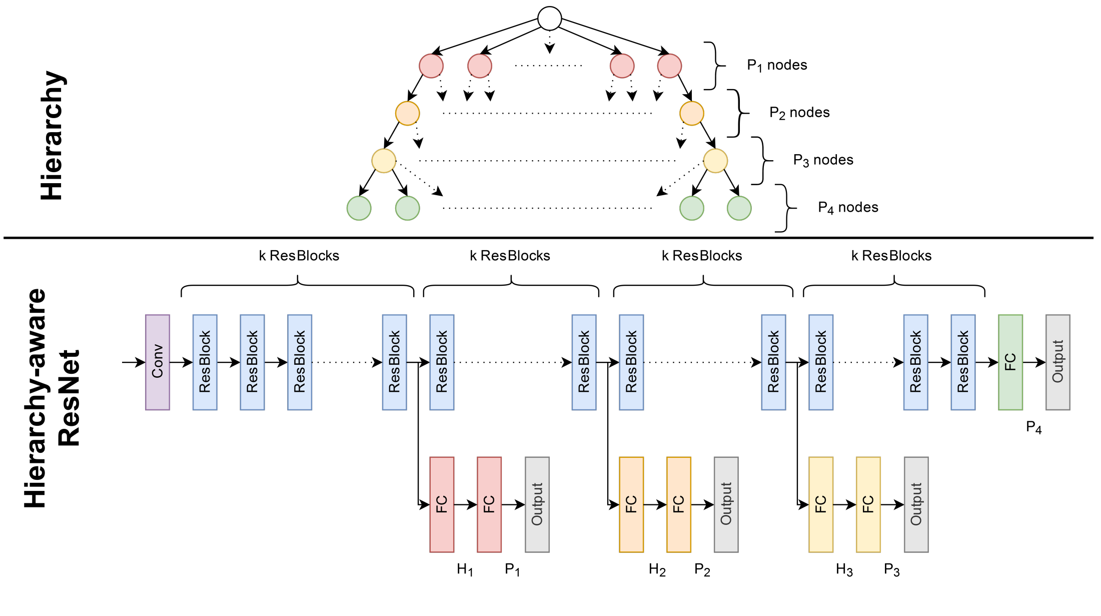
Recent works have shown that CNNs can learn the underlying hierarchical structure between classes during the training phase even though they were not explicitly trained for this specific task. Motivated by this observation, authors in [7, 38] have shown how a CNN architecture can be modified in order to help the model in the learning of such hierarchical structure during the training phase.

In their work, [7] show how a modified Alex-Net architecture [65] can improve accuracy and accelerate the process of learning class hierarchies. Their method works by adding extra classification branches between some of the convolutional layers from the original architecture.

On other hand, the method proposed by [38] adds extra classification layers after the final classification layer of the original architecture. We consider the method from [7] more suitable for a benchmark on semantic adequacy because the hierarchical structure is learned directly from the convolutional layers and not from the final classification layer.

Following [7] and by adapting their method for ResNets, a hierarchy-aware ResNet can be constructed as follows: Given a hierarchy of depth n (root excluded), we add $n-1$ branches in the architecture that learn group level classifications and optimise for error at each level of the class hierarchy. The branches are each composed of two fully-connected layers, and are evenly spread along the residual blocks (Figure 4.1). These additions are sufficient for our current experiments. We leave further architectural optimisations (e.g., dimension of extra modules) to future work.

Figure 4.1: Hierarchy-aware ResNet for hierarchy of depth 4. H_1, H_2 and H_3 are hyperparameters



One hypothesis that can emerge from this is that hierarchy-aware CNNs may lead to better taxonomies than their corresponding original architectures, since they were explicitly trained for learning the hierarchical relationships between classes. To test this hypothesis, we compare the semantic adequacy from hierarchy-aware CNNs constructed following [7] against their corresponding original architecture.

4.3 Methodology: Extracting taxonomies from CNNs

We assume class concepts are organised in a taxonomy, and that a neural network was optimised only to discriminate between leaf node classes. Our goal is to reconstruct, into symbolic form, the full taxonomy from the internal sub-symbolic structure of the model. To this end, we introduce a novel extraction method, designed such that the extracted taxonomy reflects how the model organises output classes and their hypernyms hierarchically in its internal representation.

We use co-activation graphs [50] as an intermediate representation to correlate neuron activities with classes in a trained model. Embedding graph nodes as vectors results in a latent representation of semantic relationships between classes as learned by the model. We then build on the taxonomy extraction method of [74] and hierarchical clustering to transform latent representations back into a semantic structure. An overview of the procedure is shown in Figure 4.2. The remainder of this section is devoted to describing each step in further detail.

4.3.1 From deep representations to co-activation graph

Remember that, given a trained neural network model, the nodes of its co-activation graph stand in one-to-one correspondence with its neurons. As described in Definition 1, each pair of nodes in the graph is connected by a weighted edge determined by the Spearman correlation coefficients between neuron activation values on a test data set. For neurons in dense layers this process is straightforward: there is a single activation value per data sample. For neurons in convolutional layers, a single activation value is obtained by applying average pooling on the feature map.

Restriction to significantly large and positive correlations is sufficient for our purposes. Hence, after observing activation values on the data and computing pairwise correlations, a threshold is applied to obtain an edge sparse graph.

The resulting graph provides relevant information on how dependencies between internal representations impact the global behaviour of a classifier. This makes co-activation graphs a suitable intermediate representation to analyse how a model

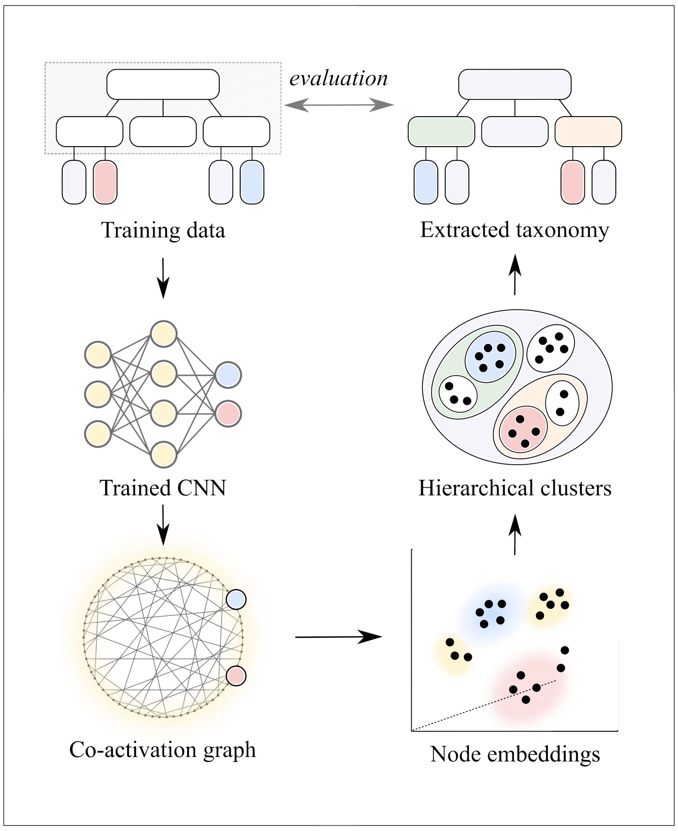


Figure 4.2: Taxonomy extraction. After optimizing for class discrimination without hierarchical structure, we extract a taxonomy by means of co-activation analysis and hierarchical clustering. This enables a comparison between extracted taxonomy and original semantic class relationships.

transforms sub-symbolic information from pixels, via its internal representations, into an assignment to a semantic class. Next, we will discuss how to transform the statistical information in the co-activation graph into a semantic structure.

4.3.2 From co-activation graph to taxonomy

We modify the extraction method in [74] such that it applies to co-activation graphs instead of knowledge graphs. Three phases can be distinguished: embedding of graph nodes into vector representation, followed by agglomerative clustering, and finally assignment of semantic types to clusters.

For the modification to co-activation graphs two important aspects need to be highlighted: First, node embeddings for output classes require taking into consideration their relationships with neurons in the hidden layers. However, for clustering and type assignment, neurons from hidden layers no longer provide relevant semantic content, and can therefore be ignored.

Second, note that hyperclass types are not represented in co-activation graphs but can nonetheless be assigned to clusters in the third phase. For example, although there is no single node represented in the co-activation graph for the taxonomic parent of an output class, it is possible to find the optimal cluster for this type by means of the F-scores. This is crucial: it allows a taxonomy to be generated that includes all hyperclass types without further modification of the co-activation graph. We avoid the risk of adding external bias to our method, which can now expose what a trained CNN has learned itself about taxonomic relationships.

We now discuss further details of the method. For the embedding of nodes from the co-activation graph to vector representations we make use of existing embedding functions [16, 39]. The results from the first phase is a data set D containing a vector for each node in the co-activation graph.

Agglomerative clustering starts by creating a leaf cluster for every vector in D. At each iteration, the two closest clusters are merged according to some chosen distance metric. Clustering terminates when there is a single cluster containing every vector in D, resulting in a tree structure over the vectors. Assigning semantic types to the tree will turn it into a taxonomy.

In order to assign types to clusters, we first extract a list of types $T = \{t_0, t_1, ..., t_{n-1}\}$ where n is the total number of types t. T can be created manually or retrieved from an existing taxonomy such as the WordNet hierarchy. Then, we map each output class to a flat list of types based on their known hierarchical relationships, which allows computing the F-scores between clusters and types as explained next. It is important to note that the list of types is not used neither during the training phase nor during the clustering phase in order to avoid adding possible biases to the final taxonomy. Finally, we calculate the F-score $F(c, t)$ for each cluster c and type t, which indicates how well c represents the entities in t.

The F-score can be calculated as shown in Equation 4.1, where $N_{c,t}$ is the number of entities with type t in cluster c, N_c is the number of entities in cluster c and N_t

is the number of entities with type t. A high F-score(c,t) indicates that cluster c contains mostly entities of type t and the highest number of entities of type t is contained in c.

$$F(c,t) = 2 \cdot \frac{N_{c,t}}{N_c + N_t} \tag{4.1}$$

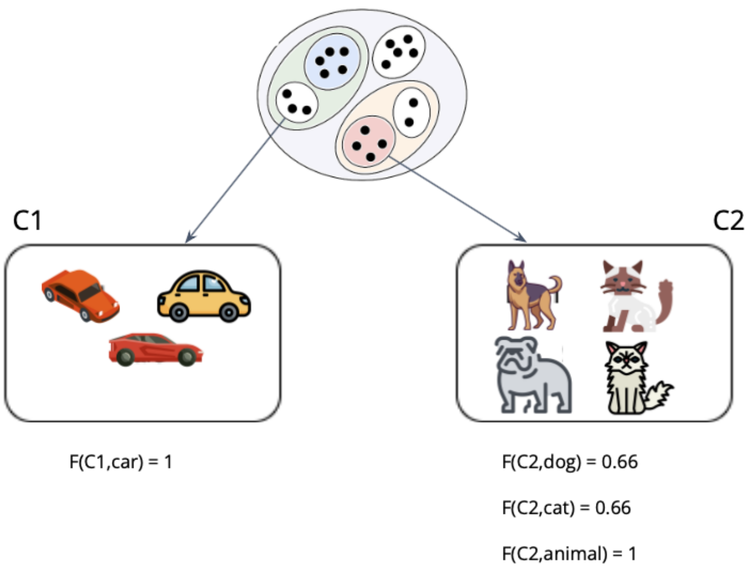


Figure 4.3: Example on how to assign types to clusters. In this example a hypothetical type car is best represented by cluster c_1 (left) since F(c_1, car) = 1 which receives type car. On the other hand, cluster c_2 (right) both cats and dogs but it is best represented by type animal since it contains all the animals in the dataset.

The process of assigning types to clusters using the F-score is illustrated in Figure 4.3 using a hypothetical example. In this example, c_1 is a cluster containing three different classes of cars, which are hypothetically all the cars in the dataset. Because c_1 only contains cars and all the cars in the dataset are in c_1, then F(c_1, car) = 1, which is the maximum possible F-score and the type car is assigned to c_1. On the other hand, c_2 contains two classes of dogs and two classes of cats. Using Equation 4.1, F(c_2, dog) = F(c_2,cat) = 0.66. However, F(c_2, animal) = 1 because this cluster only contains animals and all the animal in the dataset are in c_2 and thus the type animal is assigned to c_2.

Mapping each type to the cluster with maximum F-score may result into assignment of two types to the same cluster. The authors of [74] suggest using a

Maximum Linear Sum Assignment to avoid this. Instead, we use a modified Jonker-Volgenant algorithm [54] to solve this problem, with a randomized priority order for assignment. This gives unique solutions while avoiding bias. This solution does not ensure that every cluster has a type nor that every type is represented by a cluster. In our experiment, we ensure having more clusters than types, ensuring then that every type is mapped. We therefore remove clusters that are not associated to a type, leading to a ground truth and an extracted hierarchy with the exact same set of labels. Finally we evaluate the extracted taxonomy by precision, recall and F-score over edges of the taxonomy ("direct" evaluation) and its transitive closure ("transitive" evaluation).

The proposed method for extracting taxonomies from co-activation graphs representing CNNs helps answering **(RQ5)**, which evaluates the feasibility of such taxonomy extraction. In next section we perform experiments to evaluate the method and to validate how the extracted taxonomies can be used to assess the semantic quality of their respective CNNs.

4.4 Experimental Analysis

The experimental evaluation has three distinct goals. First, to measure correlations of class similarity in the embedding space with confusion between classes by the model. This indicates the soundness of node embeddings capturing sub-symbolic representations of the model and its decision process. Second, to measure correlations of class similarity in the embedding space with semantic similarity in the ground truth taxonomy. This indicates the other side of the coin: the ability of learned representations to separate symbolic concepts in the taxonomy. Third, to evaluate the semantic adequacy of the extracted taxonomy with respect to its ground truth. This indicates the degree of transparency allowed by the particular models as expressed in semantic terms, and the suitability of our method for semantic interpretation in general.

4.4.1 Experiment Setup

Experiments were conducted using CIFAR-100 and ImageNet datasets. We extracted training hierarchies from those using the method from [38]: given a wanted depth of taxonomy n, we extract the first $n-1$ nodes that link the root to the labels in WordNet. For CIFAR-100 we trained VGG-16, ResNet-18 and hierarchy-aware ResNet-18 (noted HA-ResNet-18). For ImageNet we trained VGG-16, ResNet-152 and hierarchy-aware ResNet-152 (noted HA-ResNet-152). A detailed description on the datasets, the training hyperparameters and hierarchy used for training HA-ResNet-18 and HA-ResNet-152 can be found in Appendix B.

For each dataset and model, a co-activation graph containing only those connections with correlation higher than 0.3 was built. We extracted the taxonomy using the pipeline in Figure 4.2. However, due to computational limitations, some layers were excluded from the graph for VGG16, ResNet-152 and HA-ResNet-152, the exhaustive list of layers used being in Appendix B. Node embeddings were generated using two different embedding methods in order to check if the results are consistent across different strategies: Node2Vec [39] and Fast Random Projection (FastRP) [16]. For both Node2Vec and FastRP we used the algorithm implemented by Neo4j.

- Node2Vec [39]: A node embedding algorithm based on graph random walks. The results generated from the random walks are then passed to a single hidden layer neural network to predict the likelihood of stepping in to a node given the occurrence of another node. For this experiment we have used the Neo4j implementation.

- Fast Random Projection [16]: Also referred as FastRP, this method learns node embeddings by applying dimensionality reduction over the graph adjacency matrix. For this experiment we have also used the Neo4j implementation.

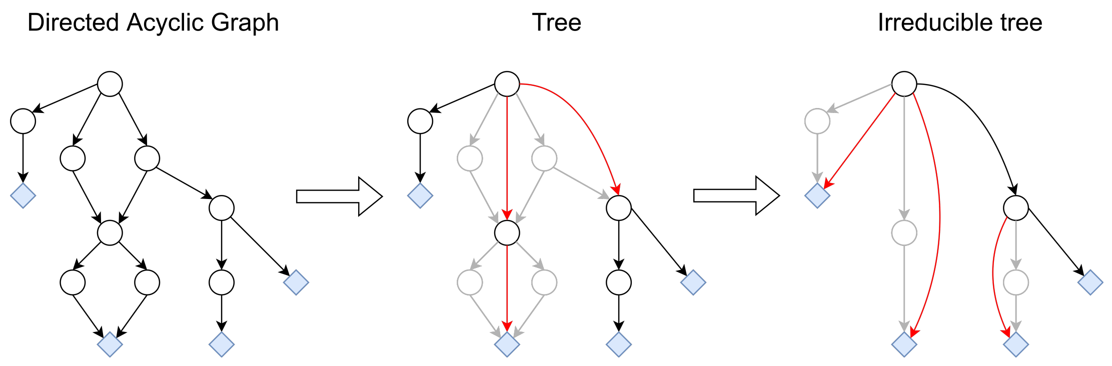
After calculating the node embeddings using the two methods above, the next phase is to apply the agglomerative clustering algorithm. For this phase we have tested different distance criteria and metrics, which influence the merging strategy of the agglomerative clustering. The metrics were euclidean and cosine (when applicable) while the distance criteria used were: *average (UPGMA)*, *weighted (WPGMA)*, *complete*, *centroid* and *ward*.

At this point, we end up with a hierarchical clustering tree, and the next phase is to assign types to the clusters that best represent the entities from each type. Because we want to compare the extracted taxonomy with the WordNet hierarchy, we extracted the types directly from WordNet using the *nltk* python package. However, WordNet does not provide a tree but a Directed Acyclic Graph, and trying to compare the extracted hierarchy to a DAG presents a number of drawbacks:

- If there are multiple path leading to a class, then it is not possible to decide which one(s) have been learned by the network.
- If a class contains a single subclass, then these two cannot be differentiated by the network. Also, as we are mainly looking for discriminating concepts to explain the network's decision making, such classes are of low interest in our context.

To face these limitations, we first convert the DAG to a tree by removing duplicated paths, then simplify nodes that have only one child. This method is illustrated on Figure 4.4. This process keeps the leaves and thus ensures that all base types are preserved. Also, the final number of classes and hyper-classes is lower than the number of clusters returned by the agglomerative clustering, ensuring then that every type is mapped during Linear Sum Assignment resolution.

Figure 4.4: Conversion from DAG to irreducible tree



Then, for each type t and each cluster C, we have calculated the $Fscore(C, t)$ using Equation 4.1 and assigned types to clusters by solving the corresponding Linear Sum Assignment problem. Finally, the clusters and types that are not associated are removed, and the resulting tree hierarchy represents the extracted taxonomy.

4.4.2 Correlating class embedding with class confusion

In order to evaluate if node embeddings from co-activation graphs can be used to represent output classes from a given CNN, we analysed if there is a connection between the similarity between pairs of classes in the embedding space with the number of mistakes between these classes in the model.

To perform this comparison, we calculated the correlation between the number of mistakes between any two classes and the cosine similarity between the corresponding classes in the vector embedding space. As shown in Table 1, there is a positive correlation on the number of mistakes between classes and their cosine similarity in the node embedding space using both FastRP and Node2Vec for models from both CIFAR-100 and ImageNet.

It is important to point out that Node2Vec achieves higher values for correlation than FastRP, whereas for ImageNet the results are comparable. The positive result in this first analysis shows that the node embeddings calculated from co-activation graphs can represent well the similarity between classes with respect to the CNN representation.

4.4.3 Correlating class embedding with class similarity

The goal of this analysis is to check whether the produced node embeddings can encode semantic similarity between classes represented in that space. If classes that are semantically similar are also closer in the embedding space, we consider that such embedding space is suitable for the taxonomy extraction process. For this analysis, we have first calculated the semantic similarity for every pair of classes in

Dataset	Model	Correlation using Node2Vec	Correlation using FastRP
CIFAR100	VGG16	**0.31**	0.24
CIFAR100	ResNet18	**0.41**	0.38
CIFAR100	HAResNet18	**0.48**	0.41
ImageNet	VGG16	0.14	**0.16**
ImageNet	ResNet152	0.12	**0.13**
ImageNet	HAResNet152	0.13	**0.14**

Table 4.1: Benchmark of node embeddings methods using the spearman correlation between pairwise class mistakes and their cosine similarity in the vector embedding space.

the dataset. This process was done by running the Floyd-Warshall algorithm[1] on the undirected irreducible tree extracted from WordNet.

We have then calculated the correlation between semantic similarity among classes and cosine similarity between the corresponding classes in the embedding space.

Dataset	Model	Correlation (Node2Vec)	Correlation (FastRP)
CIFAR100	VGG16	**0.13**	0.09
CIFAR100	ResNet18	**0.36**	0.27
CIFAR100	HAResNet18	**0.32**	0.30
ImageNet	VGG16	0.44	**0.45**
ImageNet	ResNet152	0.43	**0.54**
ImageNet	HAResNet152	0.48	**0.61**

Table 4.2: Benchmark of each node embeddings method using the spearman correlation between pairwise class semantic similarity and their cosine similarity in the embedding space.

In Table 2 it is possible to observe that there is a positive correlation between semantic similarity among classes and cosine similarity in the embedding space. Once again, the results from Node2Vec are higher than FastRP for CIFAR-100

[1] https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.shortest_paths.dense.floyd_warshall_numpy.html

while FastRP performs better for ImageNet. The positive correlations achieved in this analysis are encouraging evidences that the CNNs learned class semantic similarities.

However, the correlation only is not a strong metric to compare the semantic adequacy between each architecture, since they do not expose which semantic relationships were learned by the model. The semantic adequacy comparison between models is done in a more detailed way in the third analysis.

4.4.4 Evaluating extracted taxonomies

In this third analysis we evaluate the semantic adequacy of the taxonomies extracted using our method by comparing them with the ground truth hierarchy from WordNet. We evaluate the taxonomies extracted from all the models and the two datasets, using Node2Vec and FastRP embedding methods and five different clustering criterion with both euclidean and cosine distances. For this analysis, we also evaluated the semantic adequacy of an untrained model using the VGG-16 architecture initialised using random parameters with the purpose of providing a lower bound baseline.

This evaluation is conducted following the same principles used in [74], which uses both a direct and a transitive evaluation. In the direct evaluation, we measure how well the extracted taxonomy handles direct axioms. For example, an axiom such as $GermanShepherd \subset Dog$ is going to cause a negative effect in the direct evaluation, because, according to WordNet, the direct hypernym for a *GermanShepherd* is *ShepherdDog*. In the transitive evaluation, the goal is to evaluate high level axioms. Using the previous example, $GermanShepherd \subset Dog$ causes a positive effect in the transitive evaluation because there is a transitive relationship between types *GermanShepherd* and *Dog*, also according to WordNet.

The precision, recall and F-score are calculated using the edges of the graphs representing the extracted taxonomies and the edges of the graph representing the ground truth, as described in Equation 4.2. For the direct evaluation we calculate the

Best F-scores for CIFAR100				
	FastRP		Node2Vec	
Model	Transitive	Direct	Transitive	Direct
UntrainedVGG16	0.03	0.23	0.04	0.23
VGG16	0.28	0.40	0.24	0.37
ResNet18	0.34	0.44	0.38	0.50
HAResNet18	0.36	**0.44**	0.40	**0.55**

Table 4.3: Direct and transitive accuracies using different embedding methods and clustering criteria for CIFAR-100.

evaluation metrics based directly on the respective graphs whereas for the transitive evaluation we consider the transitive closure.

$$\text{Ground truth}: G_t = (V, E_t),\ \text{Experimental}: G_e = (V, E_e)$$
$$\begin{cases} precision = \dfrac{|E_t \cap E_e|}{|E_e|} \\ recall = \dfrac{|E_t \cap E_e|}{|E_t|} \\ f\text{-}score = 2 \cdot \dfrac{|E_t \cap E_e|}{|E_t| + |E_e|} \end{cases} \tag{4.2}$$

Table 4.3 reports the semantic adequacy of each model by means of the best F-scores achieved by their taxonomies for CIFAR-100 using both FastRP and Node2Vec. The untrained VGG-16 had the lowest semantic adequacy for both direct and transitive evaluations, which was expected since this model was provided as a lower bound baseline. It can be observed that HA-ResNet-18 achieves the highest scores for FastRP and Node2Vec, although ResNet-18 also matches the best value for the direct evaluation using FastRP. VGG-16 gets the lowest results from all the three models. It is also possible to observe that Node2Vec achieves better results than FastRP for all the three models in this dataset.

From Table 4.4 we can observe a similar behaviour when considering the setup

Best F-scores for ImageNet				
	FastRP		Node2Vec	
Model	Transitive	Direct	Transitive	Direct
UntrainedVGG16	0.02	0.22	0.03	0.22
VGG16	0.41	0.47	0.34	0.44
ResNet152	0.41	0.48	0.40	0.50
HAResNet152	0.43	**0.49**	0.46	**0.53**

Table 4.4: Direct and transitive accuracies using different embedding methods and clustering criteria for ImageNet.

for the ImageNet dataset. Again, the untrained VGG-16 achieved a much lower semantic adequacy for both direct and transitive evaluations, when compared to the trained models. The HA-ResNet variation had the best scores for both direct and transitive evaluations with VGG-16 getting the lowest values.

The results obtained from Table 4.3 and Table 4.4 suggest that the ResNet variations tend to perform better than VGG-16 in terms of semantic adequacy. However, these analyses consider only the best result achieved by each model, which can be biased towards outlier performances. In order to analyse which architectures are the most semantic adequate in a more systematic way, we have performed a statistical test to compare all the F-scores from the taxonomies obtained from every architecture using all the embedding methods, clustering criterion and distance metrics.

For this, we first obtained all the F-scores for each model. The F-scores were collected by varying the embedding methods, clustering criterion and distance metrics. The setups that produced F-score values lower than twice the standard deviation from the median were considered as outliers and thus removed from this analysis. Then, for each pair of models we tested if there were a difference in their F-score distribution using the student t-test. The null-hypothesis is that there is no difference between two distributions and p-value ≤ 0.05 rejects the null hypothesis and indicates there is a statistical difference between the two distributions. The results from the statistical analysis are presented in Table 4.5.

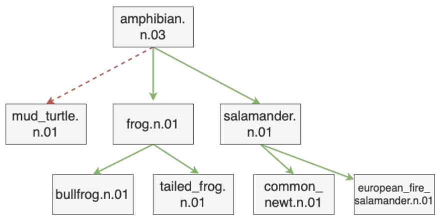
From Table 4.5 it is possible to observe that, for the transitive evaluation, the HA-ResNet variations are statistically more semantically adequate than pure ResNets and VGG-16. This is an encouraging evidence that the injection of external knowledge during the training phase may help in achieving more interpretable models. We can also see that VGG-16 does not perform better than any other model, which may indicate that the architecture depth can have an effect on semantic adequacy.

Dataset	Model one	Model two	direct pvalue	transitive pvalue
CIFAR100	ResNet18	VGG16	**<0.05**	**<0.05**
CIFAR100	HAResNet18	VGG16	**<0.05**	**<0.05**
CIFAR100	HAResNet18	ResNet-18	>0.05	**<0.05**
ImageNet	ResNet152	VGG16	>0.05	>0.05
ImageNet	HAResNet152	VGG16	>0.05	**<0.05**
ImageNet	HAResNet18	ResNet152	**<0.05**	**<0.05**

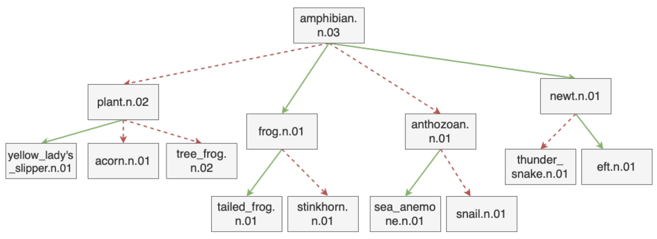
Table 4.5: Statistical comparison on the semantic adequacy between different architectures. HA-ResNets achieved the best values for semantic adequacy.

One important consideration is that, because the semantic adequacy is calculated for the entire taxonomy, it does not give fine-grained information about which parts of the taxonomy contributed positively or negatively for the final score. For this purpose, visualisation techniques can be used to analyse specific subtrees by checking their correct and incorrect axioms.

As an example, Figure 4.5 shows how different architectures can capture very different axioms for the same concept, which in this case is concept amphibian. In the visualisation, green edges and red dashed edges indicate correct and wrong axioms according to WordNet, respectively. It is visually clear that the amphibian subtree for HA-ResNet152, shown in Figure 4.5a captured the amphibian concept and its subclasses in a more appropriate way than VGG-16, which is shown in Figure 4.5b. The amphibian subtree for VGG-16 contains seven wrong axioms while HA-ResNet52 has only one wrong axiom. Also, some axioms from the VGG-16 subtree



(a)



(b)

Figure 4.5: (a) Concept amphibian.n.03 and its subtree for HA-ResNet-152. (b) Concept amphibian.n.03 and its subtree for VGG-16.

are hard to justify, such as plant being a subclass of amphibian.

Analysing the subtrees is relevant because it shows how well the CNN has learned a specific set of concepts. However, relying purely on visualisation techniques can be a cumbersome task, which motivates the search for a local semantic adequacy metric that can capture the quality of each subtree in the taxonomy. The development of such metric is included in the next steps for this research topic.

Overall, our evaluation shows that taxonomies extracted from CNNs using our method can achieve reasonable direct and transitive F-scores, even when the models were not trained explicitly for that. The best results in our analysis were generated by the hierarchical-aware architectures, as proposed by [7] and adapted for ResNets in this chapter. In contrast with that, the lower bound baseline provided using an untrained model showed that the semantic adequacy can be very low when there is no knowledge encoded within the deep model. Therefore, we obtained multiple evidences to show that the proposed semantic adequacy metric reflects how compatible the knowledge contained within a deep representation is to a ground truth taxonomy.

One possible drawback of our current approach is the need to project class node representations into vector representations which could lead to some information loss. An alternative would be to apply hierarchical community detection directly to the co-activation graph, instead of relying on agglomerative clustering methods. However, this would lead to a large number of clusters containing only neurons in the hidden layer which could increase complexity both in terms of computational time and interpretability of the final taxonomy.

The conducted experiments indicate that, by comparing the extracted taxonomy with a selected ground-truth, it is possible to measure the semantic adequacy by means of the direct and transitive F-scores. This result answers **(RQ6)** and gives evidences towards **H3** by showing that the extracted taxonomies can measure how well the model has captured the hierarchical relationships between classes from a given domain.

4.5 Summary

We proposed a method for semantic interpretability of deep representations by extracting taxonomies from the internal structure of trained CNNs. Our approach represents a CNN as a co-activation graph and adapts the taxonomy extraction method in [74] to such graph.

As indicated in Figure 4.2, we do this by calculating node embeddings for classes, performing hierarchical clustering, and assigning semantic types to the clusters that best represent the entities belonging to the corresponding types. This generates a typed tree that can then be transformed into a taxonomy, which then answers **(RQ5)**.

We introduce the concept of semantic adequacy to measure how well a model captures the hierarchical relationship between classes from a given domain by comparing its extracted taxonomy to a ground truth such as WordNet. Our experimental evaluation shows that the node embeddings preserve both class similarity with respect to the internal representations by the CNN and semantic similarity with respect to a ground truth taxonomy. This is an important evidence that the CNN has learned semantic relationship between classes and that these relationships are reflected in the vector representation calculated by our approach.

We then extracted the taxonomies for each model and each dataset to evaluate their semantic adequacy by comparing the taxonomies with the WordNet hierarchy. Some models achieved an F-score of 0.50 for direct and for transitive evaluation, which is a reasonable semantic adequacy considering that most models were not explicitly trained for recognising hierarchical relationships between classes.

A statistical comparison on the semantic adequacy between all the models indicated that ResNet variations achieved higher scores than VGG-16, which is in line with recent literature evidence that suggests that network depth may lead to more interpretable models [6]. The hierarchy-aware models achieved the best semantic adequacy scores when analysing individual results. Moreover, a statistical test showed that such hierarchy-aware models are statistically better than their original architectures, which motivates the use of external knowledge during the training phase of CNNs.

The results demonstrate the applicability of taxonomy extraction and the semantic adequacy metric for CNNs, which answers **(RQ6)** by showing the feasibility of our approach to semantically interpret some of the knowledge contained within a trained CNN so that it can be more easily related to its decision making processes.

Chapter 5

Generating Local Textual Explanations for CNNs: a Semantic Approach based on Knowledge Graphs

5.1 Introduction

In local explanations, the objective is to understand the decision making process for a specific data point. This type of explanation is useful because it helps discerning case-specific algorithmic fairness, as observed by [23]. For tasks in Computer Vision, such as image classification, the goal of a local explanation is to identify which elements from a given image have contributed to the prediction of a given class.

Current methods commonly use visual approaches, such as saliency maps, to highlight parts of the input image that has mostly influenced a given outcome [98]. Such visual clues can be misleading and make human interpretation particularly difficult, especially when trying to explain unexpected outcomes [66, 86].

In this chapter, our goal is to provide easy-to-understand textual explanations that do not need to be visually interpreted. We specifically focus on explaining mistakes as these can be particularly challenging to interpret, and we generate both factual and counterfactual explanations as known concepts in cognitive science [97] and widely used in the XAI community [11]. The factual explanations produced by our approach indicate in human terms what made the model classify an instance i as class c_{wrong} instead of $c_{correct}$. The counterfactuals indicate what could help the model to correctly classify i as $c_{correct}$ instead of c_{wrong}.

Building upon the idea of co-activation graphs described in Definition 1, we first augment co-activation graphs to link it with input data and external knowledge. We then use link prediction to predict attributes of unseen data and use such predictions to generate factual and counterfactual explanations of mistakes in image classification.

The potential of our method is demonstrated in an experiment on CUB200 birds dataset [102], used for fine-grained classification of birds images among very similar classes. We fine tune a VGG-16 model [89], construct the proposed knowledge graph form VGG-16 and use link prediction to derive explanations for classification mistakes in terms of semantic properties and their values, such as *"This bird was misclassified as Blue Jay instead of Black Footed Albatross because the model has identified blue as value for primary_color instead of brown."*

The main contributions in this chapter are: i) the extension of the co-activation graph to include input data and external knowledge bases ((**RQ7**)); ii) a graph-based approach to identify semantic attributes that might be responsible for a mistake using link prediction ((**RQ8**)); and iii) an algorithm for the automatic generation of easy-to-understand factual and counterfactual textual explanation for mistakes based on the results of link prediction ((**RQ9**)).

5.2 Methodology

Our approach focuses on providing textual post-hoc factual and counterfactual explanations in image classification, and we specifically focus on explaining mistakes as they are harder to explain.

In classification models whenever a mistake happens it involves two classes, the ground truth class $c_{correct}$ and the wrongly predicted class c_{wrong}. A factual explanation describes what influenced the model to classify a given input i as of class c_{wrong}, while a counterfactual explanation tells what changes to the input i could invert a wrong prediction from c_{wrong} to $c_{correct}$.

Our approach to generate these textual explanations is to first define and construct a knowledge graph that extends the co-activation graph described in Definition 1 by including input data and external knowledge bases, so that knowledge about such input (in terms of its properties) can be added to the graph. On such extended graph, we then apply link prediction to generate the explanations. In what follows we are going to provide details of each step.

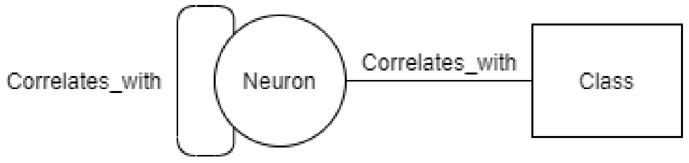
5.2.1 (RQ7): The Knowledge Graph Model

Since our Knowledge Graph Model extends the notion of co-activation graph, we first provide some preliminaries on co-activation graphs and then indicate how we extend their definition in this chapter to connect them with input data as well as external knowledge bases.

Preliminaries on Co-activation graphs.

In a co-activation graph, nodes represent neurons from a given trained CNN and edges connect pairs of nodes that tend to be activated together. The nodes are instantiated by extracting the neurons from the model and the edges can be created for each pair of nodes by calculating a statistical correlation on their respective neurons' activation values following the three-steps process detailed in Definition 1 from Chapter 3: first, the testing set is passed through the network and, for each data point, the activation value of each neuron is stored; second, given that each neuron is associated to a list of activation values, a statistical correlation is calculated for every possible pair of neurons; third a threshold is used to discard the non relevant statistical correlations. As a result of this process, in the co-activation graph nodes are neurons and edges represent statistical correlations between their activation values. If a neuron belongs to the output layer, it is represented using a node of type *class*, otherwise, we represent it using a node of type *Neuron*, as shown in Fig. 5.1.

Figure 5.1: Co-activation graph diagram.



We have already demonstrated in Chapter 3 how co-activation graphs provide a suitable representation for the knowledge embedded in a trained CNN, which can be used to generate global insights about the inner workings of the network using graph analysis. In this section, we extend this representation by integrating input data and external knowledge to the graph in order to provide local explanations by means of link prediction algorithms.

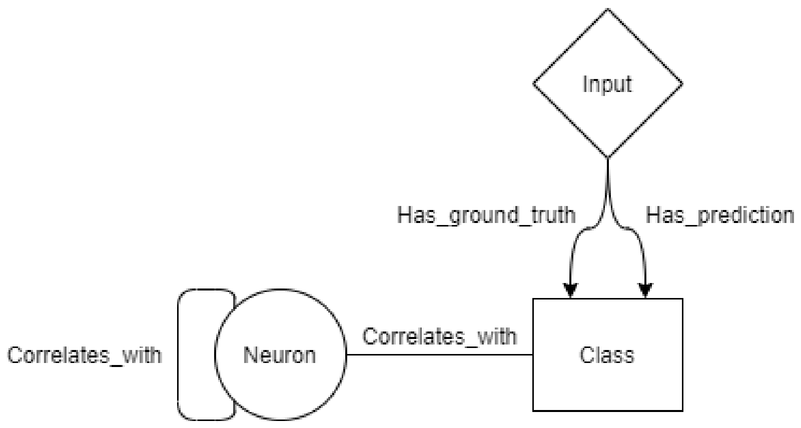
Connecting the Input Data.

In order to represent the input data in co-activation graphs, we create a new type of node, called *Input* node. An *Input* node i can be connected to both *Neuron* nodes and *Class* nodes. The connection between i and a *Class* node c can be created in two ways:

- if the CNN predicted c when classifying i, an edge of type *has_prediction* is created between i and c;
- if c is the ground truth for i, an edge of type *has_ground_truth* is created between i and c.

The resulting model can be seen in Fig 5.2. To create connections between *Input* and *Neuron* nodes we measure the relevance of the activation caused by each input to each neuron. When one input is passed through a CNN, neurons in dense layers will output an activation value and filters in convolutional layers will output

Figure 5.2: Diagram after connecting inputs to classes.

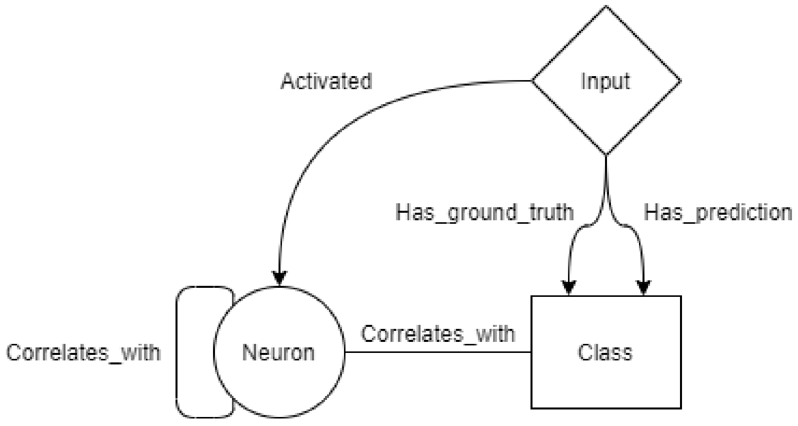


an activation map. As discussed in 3.2, it is possible to apply a global pooling to reduce an activation map into a single activation value. Considering that an activation value is accessible for each neuron, our goal is to be able to connect the respective input data to the generated graph.

A naive approach would be to create a link between the *Input* node and every *Neuron* node and to use the activation value as the edge weight. One issue with this approach is that it would generate a huge number of connections, which increases quadratically with respect to the number of neurons and data points. A second issue is that neurons at different layers activate with different magnitude and the different scales in the edge weight distribution could lead to biases when applying graph algorithms.

To address these two issues, we normalise and discretise the neuron activations. After extracting the activation values, we first calculate how relevant was the activation of each neuron for a given input data as proposed in [32]. To calculate the activation relevance we first compute the *zscore* of each activation, which gives the distance between each activation and the mean. We can then apply a threshold to the *zscores*, as explained by the same authors in [34], so that the final relevance score will assume the values of -1, 0 or 1. This eliminates the side-effects of variability of the activation values. If the final score is 1 we consider that the neuron activation was relevant to the input, otherwise we discard it. This way we do not have to deal with a quadratic number of generated connections, since we only connect inputs to neurons that were activated in relation to that input. After calculating the scores, we can extend the graph by creating a new edge of type *Activated*, denoted with $e_{in} = (i, n)$ where i is the input data and n is a neuron. The resulting model is illustrated in Fig 5.3.

Figure 5.3: Diagram after connecting inputs to neurons.



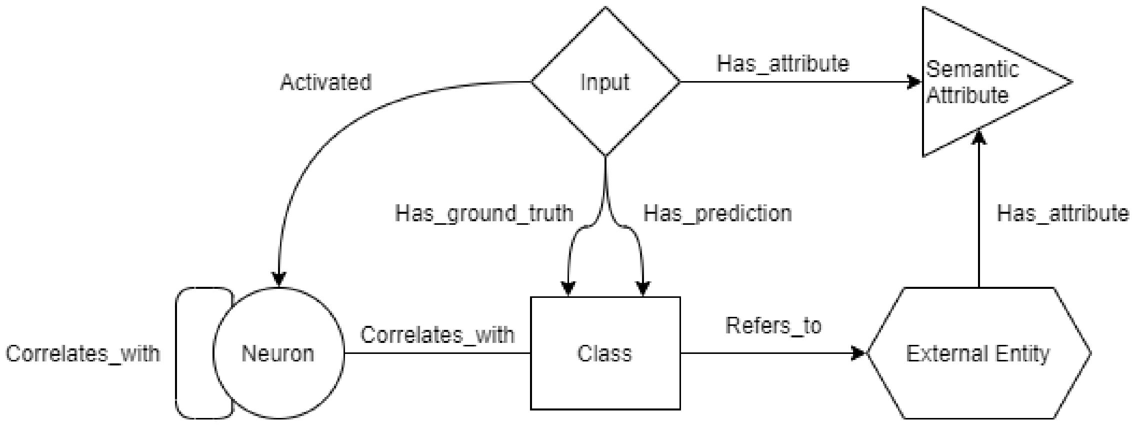
Connecting the graph with external knowledge bases.

In this phase of the approach, we consider external knowledge bases that contain real world entities (e.g animals) and their semantic attributes (e.g body shape and colors). We incorporate these entities and semantic attributes in our graph by creating two new node types: *External Entity* and *Semantic Attribute.*

External Entity nodes are the semantic entities corresponding to the output classes in the deep neural network. They can be easily connected to *Class* nodes when classes which are neurons in the output layer represent entities in the real world. In this case, we create unweighted edges of type *Refers_to* between a *Class* node in the co-activation graphs and their respective entities in the external knowledge base. When *External Entities* are related to *Semantic Attributes* in the external knowledge base, we can create an edge of type *Has_attribute* between the *External Entity* and each of its *Semantic Attribute.* Since modern datasets, like CUB200 birds, provide annotated data, we can create *Has_attribute* edges between *Input* nodes and *Semantic Attribute* nodes. This concludes the step of connecting co-activation graphs with external knowledge bases and the final diagram is illustrated by Fig 5.4.

By integrating external knowledge with co-activation graphs we are answering **(RQ7)**, since any analysis conducted over the resulting graph will leverage both the CNN deep representation and knowledge described in external knowledge bases. In next subsection, we validate this approach based on how the resulting knowledge graph based can be exploited to generate local explanations for CNNs.

Figure 5.4: Diagram after connecting inputs and classes to entities in external knowledge bases.



5.2.2 Generation of textual factual and counterfactual explanations for mistakes

We can now leverage the knowledge graph constructed in the previous section to generate textual factual and counterfactual explanations for mistakes in image classification.

Our method is divided in three phases: first, we generate hypotheses around possible semantic attributes that could have caused the mistake; second, each hypothesis is either accepted or rejected based on link prediction; third, textual explanations are automatically generated based on the accepted hypotheses. After describing each phase in details, we provide an example using real data from the CUB200 dataset and the *WhatBird* external knowledge base containing semantic information about the bird species found in the CUB200 dataset.

Generating hypotheses for explaining mistakes.

The first phase in our method is to generate hypotheses for all the possible semantic attributes that could have caused a mistake. Consider each image in the dataset as annotated with a set of attributes A, and each attribute $a \in A$ can assume different values $V_a = \{V_{a_1}, V_{a_2}, ..., V_{a_n}\}$, where n is the number of possible values for a. Given an output class c, the knowledge graph in Fig. 5.4 can be queried to find $V_a(c)$, which is the value attribute a assumes for class c. Given a misclassified input i, the ground truth class $c_{correct}$, the wrongly predicted class c_{wrong} and an attribute a, we consider that a may have caused the mistake if a assumes different values for $c_{correct}$ and c_{wrong}, i.e. $V_a(c_{correct}) \neq V_a(c_{wrong})$.

A hypothesis $h(a, i)$ can now be defined as follows:

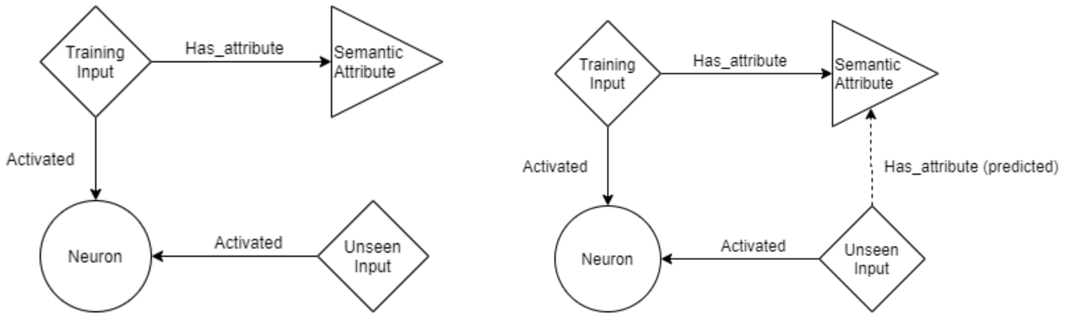
> $h(a, i)$: since attribute a assumes different values for classes $c_{correct}$ and c_{wrong}, a **has influenced** input i to be misclassified as belonging to class c_{wrong} instead of class $c_{correct}$

For each mistake made by the CNN, a list of hypotheses can be generated by querying the graph to obtain the semantic attributes that assume different values between $c_{correct}$ and c_{wrong}. Note that such a query is possible given the edges linking *Class*, *External Entity* and *Semantic Attribute* nodes, which gives all the information needed to compare $V_a(c_{correct})$ with $V_a(c_{wrong})$ for each attribute of each entity. Each hypothesis generated is then either accepted or rejected as detailed in the next phase.

From hypotheses to relevant features through link prediction.

In this second phase, we take as input a list of hypothesis on attributes that might have caused the mistake, and accept or discard them based on the semantic attributes of the misclassified input i.

A hypothesis $h(a, i)$ is accepted if the attribute a assumes the same value for input i and class c_{wrong}, i.e. $V_a(i) = V_a(c_{wrong})$. The intuition behind this is that



(a) (b)

Figure 5.5: (a) Alternative graph used for link prediction. (b) Resulting graph after applying link prediction.

when a has the same value for i and c_{wrong}, the model may have classified i as c_{wrong} because of a. As discussed previously, $V_a(c_{wrong})$ can be obtained by querying the graph. However, when i is an unseen input data, $V_a(i)$ is not directly available in the graph as we do not have information about its attributes. Our method leverage the connection between input i and the *Neuron* nodes in our knowledge graph and uses link prediction to discover its attributes.

Given the proposed knowledge graph contains seen input data from the training set, we can predict semantic attributes for unseen data using link prediction. The rationale of using a link prediction algorithm is to predict the semantic attributes of unseen input i based only on the behaviour of the neurons in the deep learning model. To guarantee that the link prediction is not biased by the connections between classes and external entities, an alternative graph is created which contains only the connections between neurons, inputs and semantic attributes. This alternative graph used only for the link prediction and illustrated in Fig. 5.5a, can be easily derived from the original knowledge graph by removing *Class* nodes, *External Entity* nodes and their respective relationships. Fig. 5.5b shows the resulting graph, in which the dashed connection results from the link prediction method.

Consider that $V_a(c_{correct})$ and $V_a(c_{wrong})$ are the values that a assumes for $c_{correct}$ and c_{wrong} respectively. The procedure for accepting or discarding a hypothesis $h(a, i)$ around an attribute a and a misclassified input i is described in Algorithm 1, where $h(a, i)$ is denoted as h for simplicity. In the algorithm, line 2 uses link prediction to calculate the probability $P_{correct}$ of input i having the attribute value $V_a(c_{correct})$. Line 3 predicts the probability P_{wrong} of input i having the attribute value $V_a(c_{wrong})$. Line 4 compares these two probabilities: if the condition is true, it means that $V_a(i) = V_a(c_{correct})$, so we discard this hypothesis because $V_a(c_{correct})$ is already the attribute of the ground truth class; if the condition is false it indicates that $V_a(i) = V_a(c_{wrong})$ and we accept this hypothesis in Line 8, because $V_a(c_{wrong})$ is the attribute of the wrongly predicted class and thus it could be the cause of the mistake. After applying Algorithm 1 to every hypothesis generated in the previous phase, the output of this phase is a list of accepted or discarded hypotheses as well as the predicted attributes values for input i.

Algorithm 1

```
1: procedure EVALUATE_HYPOTHESIS($h$, $graph$, $i$, $a$, $V_a$, $c_{correct}$, $c_{wrong}$)
2:     $P_{correct} \leftarrow$ LinkPrediction($graph$, $i$, $V_a(c_{correct})$)
3:     $P_{wrong} \leftarrow$ LinkPrediction($graph$, $i$, $V_a(c_{wrong})$)
4:     if $P_{correct} > P_{wrong}$ then
5:         discard($h$)
6:     else
7:         accept($h$)
8:     end if
9: end procedure
```

Generating factual and counterfactual explanations.

After evaluating the hypotheses, the third and final phase is used to generate the explanations. Given an accepted hypothesis $h(a, i)$, a factual statement can be automatically generated as:

> The input i was misclassified as c_{wrong} instead of $c_{correct}$ because the model has identified $V_a(c_{wrong})$ as value for a in i instead of $V_a(c_{correct})$.

It is important to note that, when multiple hypotheses are accepted, the final factual explanation will consist of multiple factual statements. Following the same logic, a counter factual statement can be automatically generated as:

If the attribute a of input i assumed the value $V_a(c_{correct})$ instead of $V_a(c_{wrong})$, input i would more likely be classified as $c_{correct}$ instead of c_{wrong}

Note that a final counterfactual explanation will consist of multiple counterfactual statements in case of more than one hypothesis is accepted.

Example using real data

To illustrate the three phases of our method, consider an input data sample $i = img001$ extracted from the CUB200 dataset and the attribute $a = primary\_color$. For this data sample, $c_{correct} = Black\ Footed\ Albatross$ and $c_{wrong} = Blue\ Jay$. According to *WhatBird*, used as a knowledge base, we have that:

$$V_{primary\_color}(Black\ Footed\ Albatross) = brown$$

$$V_{primary\_color}(Blue\ Jay) = blue$$

Since the attribute *primary_color* assumes different values for the two classes, a hypothesis $h(primary\_color, img001)$ can be created as follows:

$h(primary\_color, img001)$: since *primary_color* assumes different values for *Black Footed Albatross* and *Blue Jay*, *primary_color* **has influenced** $img001$ to be misclassified as *Blue Jay* instead of *Black Footed Albatross*.

In the second phase, we evaluate $h(primary\_color, img001)$ using Algorithm 1. If the algorithm finds that $P_{correct} > P_{wrong}$, our method would assumes that $img001$ has brown as *primary_color*, which is the expected value for a *Black Footed Albatross*. In this case we have that $V_a(i) = V_a(c_{correct})$ and we reject $h(primary\_color, img001)$. If $P_{wrong} > P_{correct}$ instead, we consider that $img001$ has blue as *primary_color* and we accept $h(primary\_color, i)$, since $V_a(i) = V_a(c_{wrong})$.

If the hypothesis is accepted, the third phase generates the factual and counterfactual explanations that can be written as follows:

- **Factual explanation**: "The input *img*001 was misclassified as *Blue Jay* instead of *Black Footed Albatross* because the model has identified *blue* as value for *primary_color* in *img*001 instead of the expected value *brown*.

- **Counterfactual explanation**: "If the attribute *primary_color* of input *img*001 assumed the value *brown*, input *img*001 would more likely be classified as a *Black Footed Albatross* instead of *Blue Jay*".

In this example, we illustrated the three phases using only a single attribute. However, as it will be shown in the next section, our explanations may contain more than one attribute if multiple hypotheses are accepted simultaneously.

In the next section we evaluate our method in two ways. First, we perform a quantitative evaluation over the link prediction method to analyse whether it is possible to predict the semantic attribute for a given input with reasonable accuracy. Then, we provide a visual qualitative evaluation to demonstrate how the automatically generated explanations can help understand the model's decisions.

5.3 Experimental evaluation

Our experiment is based on fine-grained image classification using the CUB200 dataset. The dataset contains 200 classes of bird species and 11,788 input images in total. Each image is annotated with a set of attributes A. In our experiment, specifically, we use five attributes such that $A = \{primary\_color, underparts\_color, upperparts\_color, nape\_color, bill\_shape\}$. Each attribute $a \in A$ can assume different values V_a. For example, for $a = underparts\_color$, we have $V_{underparts\_color} = \{blue, brown, iridescent, purple, rufous, grey, yellow, olive, green, pink, orange, black, white, red, buff\}$. The experiment was conducted as follows:

- A VGG-16 network pretrained over ImageNet was fine-tuned to achieve 86% accuracy in the classification task. Note that during the training the deep model receives only raw images as input in order to predict the output classes.

Therefore the model is not given any contextual information about neither the semantic attributes nor external entities.

- The knowledge graph was created for the model, following the steps described in Section 5.2.1.
- For each misclassified input, we have generated the hypotheses and performed link prediction as described in Section 5.2.2.
- Factual and counterfactual explanations were generated for the accepted hypothesis as described in Section 5.2.2.

Our knowledge graph was implemented in Neo4j[1], which facilitates reproducibility and provides access to many graph analysis algorithms. To predict links between *Input* and *Semantic Attribute* nodes we used the Personalised PageRank algorithm [79] implemented on Neo4j.

One issue that could affect the outcome of our link prediction algorithm is that the CUB200 dataset is not well balanced in terms of the semantic attributes. For example, there are 1443 birds with brown primary_color but only 27 with pink primary_color. To avoid link prediction to always output the most frequent attribute in the dataset, we normalised the Personalised PageRank score by the Global PageRank score.

The PageRank algorithm, also called Global PageRank, calculates a score that can be interpreted as the importance of each node in a graph by performing random walks in such a graph. Its variant, the Personalised PageRank, calculates the probability of a link between a source and a target node in the graph. In our experiments, the probability of a link between a source *Input* node i and a target *Semantic Attribute* node a, denoted $Link(i, a)$, was calculated as shown in Equation 1.

$$Link(i, a) = \frac{PersonalisedPageRank(i, a)}{GlobalPageRank(a)} \tag{5.1}$$

[1]https://neo4j.com

Since our method relies heavily on link prediction, in our experimental evaluation we first perform a quantitative analysis of the accuracy of the link prediction algorithm used to discover the attributes of unseen inputs from the testing set. Then, we visually evaluate the quality of the factual and counterfactual explanations for some of these inputs.

5.3.1 (RQ8): Evaluating link prediction for semantic attributes

The first part of our evaluation focuses on verifying **(RQ8)** and how accurately we can predict semantic attributes for unseen inputs using link prediction over our knowledge graph. For this, we used the alternative graph described in Section 5.2.1 containing only *Input*, *Neuron* and *Semantic Attribute* nodes. As already mentioned, this approach guarantees that the prediction of attributes relies only on the interactions between a given input and the neurons in the hidden layer of a deep learning model and is not biased by any external information.

In our approach we focus on classification mistakes between two classes. For each attribute $a \in A$ we evaluate the link prediction between pairs of attribute values in V_a. For this, we first remove the ground truth connections between inputs in testing set and *Semantic Attribute* nodes. Then, we calculate the link prediction using Equation 5.1 to obtain a score for each link between an input data and a semantic attribute. Finally, the link with the higher score is taken as the predicted one, which can be checked against the ground truth since the CUB200 dataset provides such annotations.

On average, the link prediction accuracies over all possible pairs of values for the attributes *primary_color*, *underparts_color*, *upperparts_color*, *nape_color* and *bill_shape* are: 81.4%, 83.4%, 82.5%, 81.5% and 77.4% respectively. There was however some variability in that the method is much more accurate for some pairs of attribute values than some others. For example, as shown in Table 5.1, the method obtained 99% accuracy when distinguishing between the values *pink* and *white* for the at-

Top Predictions								
Underparts color			Upperparts color			Bill shape		
Value one	Value two	Acc	Value one	Value two	Acc	Value one	Value two	Acc
pink	white	99%	purple	black	99%	hooked	all-purpose	95%
grey	green	94%	pink	white	98%	needle	all-purpose	93%
iriscident	black	94%	orange	black	96%	hooked	cone	93%
orange	white	92%	iriscident	black	96%	curved	cone	90%
blue	white	92%	grey	orange	94%	cone	specialized	89%

Table 5.1: Examples of pairs of attributes that can be well distinguishable by the proposed method.

Lowest Predictions								
Underparts color			Upperparts color			Bill shape		
Value one	Value two	Acc	Value one	Value two	Acc	Value one	Value two	Acc
purple	pink	45%	purple	pink	55%	curved	hooked	58%
orange	red	64%	orange	red	60%	all-purpose	cone	62%
olive	green	64%	green	orange	64%	hooked	needle	64%
grey	buff	66%	iriscident	orange	65%	hooked	specialised	64%
grey	black	66%	grey	black	66%	dagger	all-purpose	65%

Table 5.2: Examples of pairs of attributes that are not well distinguishable by the proposed method.

tribute *underparts_color*, which means that the model is almost perfect in detecting the *underparts color* of a bird if it considers in advance that the bird is either *pink* or *white*. Other very high accuracy values can be seen for the attribute *upperparts_color* between *black* and *purple*, as well as between *hooked* and *all-purpose* for the attribute *bill_shape*.

We can also observe that for some pairs of very similar attribute values our method does not perform so well, which is the case for *purple* and *pink* in attribute *underparts_color* and other pairs as shown in Table 5.2. But considering that classes in fine-grained image classification are already similar, it is expected that the method would produce lower accuracy for similar attributes. Looking into using more sophisticated link prediction strategies is something worth exploring in future work.

The obtained results answer **(RQ8)** because it shows that by using link prediction over the proposed knowledge graph model it is possible to predict the semantic attributes of unseen images.

5.3.2 (RQ9): A visual analysis for factual and counterfactual explanations

Now that we have provided a quantitative analysis of the performance of our link prediction approach, we proceed by visually analysing some of the factual and counterfactual explanations generated by our method. The explanations within this section were generated by considering only the pairs of attribute values in which our link prediction performs with accuracy higher than 80%.

Since there is no ground truth for such textual explanations, we provide visual analyses based on randomly selected examples. Visual analyses are used to check if the attributes used in our textual explanations are in fact present in the images. Note that in order to generate an explanation we first identify all attributes that could have caused a given mistake as hypothesis, and then we use link prediction to accept or deny each individual hypothesis, as explained in Section 5.2.

The first example of explanation presented below refers to the data sample used in the previous section and the hypotheses are shown in Table 5.3. Fig 5.6 shows the image for this data and the respective factual and counterfactual explanations.

	Black Footed Albatross	Blue Jay	Predicted for i
Bill shape	Hooked-seabird	All-purpose	Hooked-seabird
Primary Color	**Brown**	**Blue**	**Blue**
Underparts Color	**Brown**	**White**	**White**
Upperparts Color	**Brown**	**Blue**	**Blue**
Nape Color	Brown	Blue	Brown

Table 5.3: Predicted attributes for the data sample *img*001. Bold rows indicate accepted hypotheses.

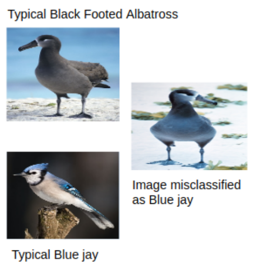

Typical Black Footed Albatross
Image misclassified as Blue jay
Typical Blue jay

Figure 5.6: Misclassification involving Black Footed Albatross (ground truth class) and Blue Jay (wrong predicted class).

- **Factual explanation**: "The input *img001* was misclassified as *Blue Jay* instead of *Black Footed Albatross* because the model has identified *blue* as value for *primary_color* in *img001* instead of *brown* and *blue* as value for *upperparts_color* instead of *brown* and *white* as value for *underparts_color* instead of *brown*.
- **Counterfactual explanation**: "If the attribute *primary_color* of input *img*001 assumed the value *brown* and *underparts_color* assumed the value *brown* and *upperparts_color* assumed the value *brown*, input *img*001 would more likely be classified as a *Black Footed Albatross* instead of *Blue Jay*".

According to our method, this first mistake was caused by the presence of blue and white colors, which are characteristic of the Blue Jay species. It is worth noting that, if the bird in the image had the value *brown* for its *underparts_color*, *upperparts_color* and *primary_color*, it would likely be classified as the correct class. In addition, our method discarded hypotheses around attributes *bill_shape* and *nape_color*. This happened because, according to our link predictions, this bird image has *hooked-seabird* as a value for attribute *bill_shape* and *brown* as a value for attribute *nape_color*, which are already the values for the correct class and thus are not likely to have caused the mistake.

In the second mistake, shown in Fig 5.7, our factual explanation tells that the detection of white colors are the causes of the misclassification. The counterfactual explanation indicates that if the bird was more black, it would likely be classified correctly. In this case we discard any hypotheses around the *bill_shape*, the *nape_color* and the *upperparts_color*.

It can be seen that in both cases the factual explanations are visually compatible with the real images. In the first case, the *blue primary_color*, *blue upperparts_color* and the *white underparts_color* are in fact visible. In the second, except for a small part in its wings, the bird is visibly more white than black. Although the known limitation of such visual analysis in terms of scaling, these examples give an indication that our method generates reasonable factual explanations.

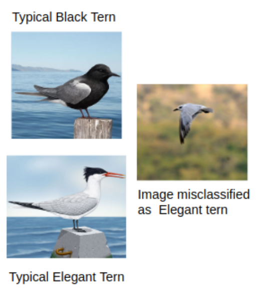


Figure 5.7: Misclassification involving Black Tern (ground truth class) and Elegant Tern (wrong predicted class)

- **Factual explanation**: "The input $img002$ was misclassified as *Elegant Tern* instead of *Black Tern* because the model has identified *white* as value for *underparts_color* in *img002* instead of *black* and *white* as value for *primary_color* instead of *black*.
- **Counterfactual explanation**: "If the attribute *underparts_color* of input $img002$ assumed the value *black* and *primary_color* assumed the value *black*, input $img002$ would more likely be classified as a *Black Tern* instead of *Elegant Tern*".

The counterfactuals generated by our approach also provide interesting insights about why the mistakes happened and what could invert the classifications. If we look back into Fig 5.6, it is intuitive that if the bird had brown colors instead of blue and white, this would probably change the classification to the correct class. A similar intuition is present in the second case, in Fig 5.7, in the sense that if the bird were black it would probably be correctly classified. In Appendix B.5, Figure B.1 presents eight additional factual and counterfactual explanations, providing a demonstration of our method's outcomes across various pairs of classes and attribute values.

The visual analysis on the generated factual and counterfactual explanations helps answering **(RQ9)** by showing that our approach can generate local explanations based on semantic attributes described in external knowledge bases. As discussed in the first part of our evaluation, currently the quality of our explanations is mainly associated with the accuracy of the link prediction algorithm used. More experiments are required to explore under which conditions we can guarantee sufficiently high accuracy and what adjustments can be made for cases in which the link prediction does not work as expected. Another point worth noting is that when generating counterfactuals with the current approach we cannot quantify the exact contribution of each attribute. This is another area of current investigation.

5.3.3 Towards a more systematic evaluation for counterfactual explanations

We have started investigating alternative ways to validate the counterfactuals in a more systematic manner. One challenge is that, being textual and not visual, our counterfactuals cannot be tested by passing them through the CNN and checking if they in fact invert the outcome to the correct class.

To tackle this problem, we experimented on the use of Generative Adversarial Networks (GANs) to modify a misclassified image based on a textual counterfactual. For this, we have experimented with different existing GAN architectures, namely DCGAN [81], StarGAN [17], AttGAN [44]. These models receive as input an image and an attribute to be changed in the image. This allow us to modify the misclassified images by changing the semantic attributes reflected in our counterfactual explanations.

Our preliminary results, reported in [61], indicated that the resulting counterfactual images generated affected the original misclassifications outcome but not as strongly as we expected. We believe this could be due to the known problem of poor resolution in GAN-generated images, and the challenge of multiple attribute modifications. In order to overcome these issues we are planning new experiments using more modern text-to-image generators such as DALL-E 2 [82] and Stable Diffusion [85].

5.4 Summary

In this chapter, we propose an approach for generating factual and counterfactual explanations of mistakes made by deep learning models in classification tasks.

In our method, we first construct a knowledge graph that connects the components of deep learning models (the neurons) with input data as well as external knowledge. Once created, the knowledge graph can be queried to generate hypotheses around semantic attributes that have potentially caused a given mistake. In order to accept or discard each hypothesis, a link prediction algorithm is used to predict the semantic attributes of unseen data based on the behaviour of the neurons in the deep learning model, as represented in the knowledge graph. Accepted hypotheses based on the results of link prediction are used to generate factual and counterfactual explanations.

Our evaluation on fine-grained image classification was conducted over the CUB200 dataset and using a VGG-16 pretrained on ImageNet and fine-tuned on CUB200. The results obtained show that, by applying the link prediction algorithm over our knowledge graph it is possible to identify semantic attributes of unseen data for different pairs of attribute values. This result is valuable especially if we consider the case of fine-grained image classification, where subtle differences among the classes contribute to make disambiguation a challenging task.

After evaluating the suitability of the approach for discovering attributes through link prediction, we present a visual inspection over selected mistakes in the dataset. The investigation showed that the factual explanations generated by our approach which are automatically generated can help understanding why a mistake happened in an easy-to-understand way and with no need for interpretation. The visual inspection also gave a first indication that the counterfactuals can indicate what semantic attribute would need to be learned differently to possibly correct the mistake.

The proposed approach assumes we have a dataset annotated with semantic attributes and an external knowledge base describing the relations between those attributes and real world entities. Given both, the approach is flexible in that it can be applied to any CNN.

Chapter 6

Conclusions

6.1 Introduction

This chapter concludes the thesis by revisiting each of the four studied hypotheses. It provides a summary of the key outcome of this thesis and its impact in the area of Explainable AI for computer vision and beyond. We first highlight the key properties of the co-activation graph representation, which is a main contribution of this thesis. Then we re-state its role in global explanations, taxonomy extraction and local explanations for deep learning models, as well as the limitations and open challenges in each of these contributions. Finally, in the last section we discuss the potential of this research in opening up new lines of investigation. We present multiple opportunities for building upon and learning from our results, we report on what we have learned from our experiments and discuss how this work could influence the direction in the field of Explainable AI.

6.2 The Co-activation Graph Representation

In Chapter 3 we formalised and experimented with a novel approach to analyse and explain the inner workings of deep learning models. The proposed methodology extracts and represents knowledge from a trained Deep Neural Network (DNN) in a graph representation called co-activation graph. In this graph, nodes represent neurons from a DNN and weighted relationships indicate a statistical correlation between their activation values. Neurons from both fully connected layers and convolutional layers can be represented in such graphs.

This representation can also be extended to include the input data and entities from external knowledge bases, as described in Chapter 5. By doing this, the co-activation graph acts as an intermediate representation between the knowledge encoded in the deep representation and human knowledge described in existing databases. It is a novel way to combine symbolic and sub-symbolic representations for Explainable AI that enables the generation of human-comprehensive global and local explanations for trained DNNs.

The extensive evaluation conducted across models and datasets at different levels of complexity corroborates initial results showing that the co-activation graph representation is compatible with the knowledge encoded in its corresponding DNN. We started the evaluation process by showing that co-activation graphs can help detecting nodes that contribute significantly to the prediction of specific classes. We also observed a positive correlation between the number of nodes shared by a pair of classes with the number of mistakes that occur between these classes. These were the first indications that co-activation graphs can play a role in providing explanations for deep learning models, which motivated further investigation on how graph algorithms and network science methods applied over these graphs can give insights about the inner workings of DNNs, which is reported in the subsequent sections.

The potential of co-activation graphs is yet to be explored in many ways that go beyond the scope of this thesis. One example is the need to adapt our representation to different architectures, such as transformers, which is crucial for applying the methods discussed in this thesis for other input data and different application domains like Natural Language Processing and Speech Recognition.

Another factor that can impact the analysis obtained using our approach is the choice of a threshold when building the graph. In this case, we consider that negative correlations can bring relevant knowledge to the graph since they indicate the absence of certain features which can be important for the task at hand [34]. However, integrating negative correlations to the graph is not trivial because many graph algorithms are not designed for edges with negative weights. One alternative is to represent negative edges by using a different edge type, which is something worth exploring in future work.

6.3 Generating Global Explanations using Knowledge Graphs

In Chapter 3 we studied the potential of using graph algorithms and metrics over co-activation graphs in order to provide global explanations for CNNs. Community detection and centrality analysis were used to explore the relationship between different output classes and the role of neurons in the hidden layers. For this, a state-of-the-art model MobileNet-V2 was trained on the CIFAR-10 dataset.

After building a co-activation graph for the trained model, a community analysis revealed that that the DNN not only has a good accuracy for the dataset but it also seems to be distinguishing groups of classes in an intelligent manner. In this case, for the CIFAR-10 dataset, the community analysis showed that the model learned how to separate classes into two groups: one group containing all the animals and a second group containing all the vehicle classes.

A step further in our analysis focused on exploring the notion of centrality. Our evaluation showed that there is an association between central nodes in a co-activation graph and their respective neurons in the DNN. It was demonstrated that the DNN drastically loses accuracy if neurons with high PageRank centrality are removed from the model, while the accuracy remains stable if neurons with low PageRank centrality are removed. This indicates that central nodes in the co-activation graph may represent important neurons in the DNN and that our approach might be considered for pruning the DNN. This is of interest, for example, in applications where dimensionality reduction is crucial.

It was also shown that graph visualisation techniques can be used to clarify the result of each analysis performed over co-activation graphs adding a level of transparency. Unlike black box models we showed that by visualising co-activation graphs it is possible to better understand why a certain algorithm produced the resulting community structure and what is expected when changing certain parameters in the algorithm. Graph visualisations can also help interpret why central nodes in co-activation graphs tend to be important nodes in DNNs since they are visually connected to dense regions and thus can impact many other neurons.

An interesting result from the experiments in this chapter was that a CNN tend to learn hierarchical relationships between classes even though it was not trained specifically for this purpose. This observation motivated the study of a more systematic way to capture the learned relationships between classes, which was done in Chapter 4 and is summarised in next section.

However, some of the findings in this chapter require a more detailed investigation. For example, nodes with high centrality may be useful in revealing possible biases within the model, given that they have impact over the decision making process. Identifying the role of such nodes could enable fixing undesired behaviours from a CNN either by improving the model itself or by a more careful selection and curation of training data.

On the other hand, less central nodes may be good candidates for model pruning, since they cause only a small decrease in accuracy when removed from the model. Model pruning is desirable not only for improving computational performance but also for achieving more interpretable models, since analysing smaller models tends to be easier than looking into bigger ones. However, providing a competitive pruning algorithm is beyond the scope of this thesis and should be investigated in future work.

6.4 Taxonomy Extraction

In Chapter 4 we studied the semantic relationships between classes from trained CNNs in a more systematic way using taxonomy extraction. In this chapter, we first developed a method for extracting taxonomies from CNNs based on class vector embeddings that can be calculated from their corresponding nodes in a co-activation graph. Then we introduced a concept called semantic adequacy that measures how well semantic concepts and their taxonomic relationships are captured by the model's internal representation when compared to a ground truth taxonomy.

We tested our approach using state-of-the-art CNNs, namely ResNet-152, ResNet-18 and VGG-16, trained on CIFAR-100 and ImageNet datasets. We also performed two additional experiments in which we modify the ResNet variations by introducing class hierarchy information during the training phase, as proposed by [7].

The experiments have proven that our method is capable of extracting taxonomies and that the produced taxonomies captured a relevant number of correct axioms. The extracted taxonomies achieved a good level of semantic adequacy, especially considering that most models were not explicitly trained for recognising hierarchical relationships. Our results indicate that architecture depth is a relevant factor for the semantic adequacy, which was also observed by [6]. An encouraging finding was that models trained with class hierarchy information were the ones with higher semantic adequacy for both datasets. This motivates the search for more innovative ways to leverage external knowledge when designing more effective architectures instead of relying on the addition of more layers and consequently more parameters.

The proposed taxonomy extraction method and semantic adequacy together can help in comparing and choosing among different CNNs by exposing how well each model learned the semantic relationships from a given dataset instead of relying purely on performance metrics. The next steps for this part of our research include adapting the semantic adequacy in order to provide more fine-grained information, since the value is currently associated to the extracted taxonomy as a whole. In this case we expect the metric to inform which specific parts of the taxonomy are more or less adequate according to the ground truth. For example, the extracted taxonomy may be more adequate for a specific subtree (e.g. *dogs*) but less adequate for another (e.g. *primates*). This information can be useful not only for deciding when to trust a given model but also for transfer learning, where a model may not be suitable for a given task even though it may provide high accuracy.

6.5 Generating Semantic Local Textual Explanations

In Chapter 5 we extended the co-activation graph representation by connecting it with the input layer and external knowledge bases. The extended co-activation graph can be seen as an intermediate representation between the knowledge acquired by the CNN during the training phase and human knowledge described in external sources.

After describing how to extend the graph representation, we focused on providing local explanations that are human comprehensible and, for this, we developed a method that generates factual and counterfactual explanations in a textual form. A central piece in the proposed method is the use of a link prediction algorithm to discover the semantic attributes from unseen images, which was tested in an experiment using VGG-16 trained for the CUB-200 dataset.

Our results indicated that, for different types of semantic attributes and pairs of attribute values, our method is capable of predicting such attributes in a very accurate way. This is especially encouraging if we consider that the CNN was trained using only visual features and was not aware about any annotation during the training phase.

After predicting the semantic attributes, our method generates textual explanations that are easy to understand. Our evaluation considered both the accuracy of the link prediction algorithm and a visual inspection on the factual and counterfactual explanations. One limitation however is that, due to the textual nature of our explanations, it is not possible to automatically verify whether the produced counterfactual can actually invert the classification from the wrong class to the correct one. Because of this, we have considered exploring effective ways to transform our textual explanations into visual images that can be injected back into the CNN for a systematic validation of the generated counterfactuals [61]. It is important to point out that, although visual counterfactuals are important for having a more robust automatic validation, the textual explanations produced by our approach are more human friendly and less dependant on human interpretation. A related but much broader area worth investigating is the role human-in-the-loop approaches can have in validating our explanations and improve their quality, but we will better discuss this in the next section.

6.6 Challenges and Opportunities ahead

In this thesis we have developed a novel approach to explain deep neural networks through the use of knowledge graphs and graph analysis algorithms. Our main contribution is the development of the co-activation graph, which serves as an intermediate representation between the knowledge contained within deep representation and human knowledge represented in external sources.

We have systematically evaluated the role of co-activation graphs in the explainability process through a set of research questions from different facets: (i) global explanations; (ii) taxonomy extraction and semantic adequacy and; (iii) local explanations. Our research paves the way for several interesting lines of investigation we believe are worth exploring which are listed below.

Bias detection: A known challenge in Machine Learning is that models trained over unbalanced datasets tend to inherit certain biases and identifying such biases within the deep representation is not easy. We have already shown that our approaches for taxonomy extraction and link prediction can reveal existing biases on a class level and instance level, respectively. However, we believe that much is yet to be done in terms of bias detection using co-activation graphs. One possible way to characterise such bias would be to study information flow and frequent paths in the graph, which may indicate how undesired decision paths are formed in the graph. In this sense, analysing paths from co-activation graphs is more feasible than doing so in the model itself, since co-activation graphs are much less dense than DNN architectures. Another interesting direction is to analyse vector representations computed using node embeddings for nodes representing biased input data. This is especially interesting when considering annotated datasets, since semantic attributes can also be represented as nodes in the graph and thus analysed in the form of vectors, which may reveal biases from a semantic perspective.

Human-in-the-loop: Considering the local explanations provided by our method, one possibility for improving the validation process would be to apply human-in-the-loop approaches. Human-in-the-loop techniques have been widely used in Machine Learning to improve model training, validation and testing, for example by designing adaptive approaches to annotation and labelling. The use of such approaches in Explainable AI through validation and adaptive generation of explanations has yet to be fully explored and a more general view of human-centered AI is seen as critical in the European roadmap for trustworthy AI[1]. In this context, one idea would be to validate not only that our explanations are trustworthy but that they can also be helpful and actionable for the end-user. For this we first intend to count on human expertise to help in measuring the quality of our explanations. In addition, uncertainty sampling methods should be considered for when our approach is uncertain about possible causes for a given mistake, which could be supported by the output of the link prediction algorithm. Besides that, human feedback could also be leveraged and integrated with our approach in order to improve the generated explanations. As an example, human input can be helpful to distinguish what semantic features are more critical in the explainability process and this type of information combined with our hypothesis selection phase could lead to more fine-grained and

[1]https://www.humane-ai.eu

user-centered explanations. We also expect that the link prediction outcome can be considerably improved as we gain access to better data annotations around features that are known to be important for the user in making an explanation more trustworthy.

Enhanced knowledge extraction: Chapter 5 showed how co-activation graphs combined with external knowledge enabled the generation of textual factual and counterfactual explanations. The proposed representation can be reconciled with other knowledge extraction approaches in order to improve the outcome of the algorithms and graph methods used in this thesis. For example, integrating our current representation with information about the role of specific neurons, as discussed in [6], can lead to better vector representations or more accurate link prediction which would improve taxonomy extraction and local explanations respectively. Another possibility is to integrate rules extracted from CNNs that could help exposing possible biases or wrong taxonomic axioms.

Knowledge injection: In this thesis we discussed the use of co-activation graphs mainly in the context of Explainable AI. However, one interesting line of investigation would be to explore how to distill information from the graph with the purpose of injecting knowledge back into the model (to learn better representations). In Chapter 4, we observed that architectures that leverage class hierarchy information during the training phase tend to learn better taxonomic relationships. This encourages the search for other ways to inject knowledge during the training phase in order to achieve better knoweldge-informed models. Graph analysis may assist this process in two different ways. The first is to help to decide what type of information is misrepresented in the training data, which may be achieved by analysing the extracted taxonomies or by looking at how different semantic attributes are affecting the decisions. The second way is to help in deciding where to add such information and it can be supported either by analysing the community structure in the graph or important neurons revealed by centrality analysis.

We believe the research in this thesis brings a novel perspective to the field of

Explainable AI, by providing a mechanism to integrate knowledge graphs with deep representations and by enabling the use of network science and graph methods to this field.

Appendices

Appendix A

Publications on Work from this Thesis

A.1 Journal Publications

1. Vitor A. C. Horta, Ilaria Tiddi, Suzanne Little, Alessandra Mileo: Extracting knowledge from Deep Neural Networks through graph analysis. *Future Gener. Comput. Syst. 120: 109-118 (2021)*

A.2 Conference Publications

2. Vitor A. C. Horta, Robin Sobczyk, Maarten Stol and Alessandra Mileo. (2023). Semantic Interpretability of Convolutional Neural Networks by Taxonomy Extraction. *In NeSy 2023 – 17th International Workshop on Neural-Symbolic Learning and Reasoning*

3. Vitor A. C. Horta, Alessandra Mileo. (2022). Generating Local Textual Explanations for CNNs: A Semantic Approach Based on Knowledge Graphs. *In AIxIA 2021 – Advances in Artificial Intelligence (pp. 532–549). Springer International Publishing. https://doi.org/10.1007/978-3-031-08421-8_37*

4. Vitor A. C. Horta, Alessandra Mileo. (2020). Interpreting Deep Neural Net-

works through Knowledge Extraction and Graph Analysis. *In 17th International Conference on Principles of Knowledge Representation and Reasoning (Doctoral Consortium, 2020)*

5. Vitor A. C. Horta, Alessandra Mileo: Towards Explaining Deep Neural Networks Through Graph Analysis. *DEXA Workshops 2019: 155-165*

A.3 Demos

6. Vitor A. C. Horta, Alessandra Mileo. (2020). Interpreting Deep Neural Networks through Knowledge Extraction and Graph Analysis (Lightning Talk). *In 17th International Conference on Principles of Knowledge Representation and Reasoning (Doctoral Consortium, 2020)*

Appendix B

Experimental details

B.1 Datasets

B.1.1 CIFAR-100

The dataset contains colour 32×32 images categorized in 100 classes, with 600 images per class : 500 for training and 100 for test. We split randomly split the training set in 90% for the training and 10% for validation. The randomness was seeded (seed : 123456789) for reproducibility purposes. In order to be able to evaluate the semantic adequacy of taxonomies extracted from this dataset, we manually linked the class labels from CIFAR-100 to WordNet synsets.

The training hierarchy was computed with depth 4, leading to hierarchy levels of size 2, 6, 8 and 100. Labels of each level can be found in Table 1.

Level of hierarchy	Number of labels	Contained labels
Level 3	2	*abstraction* *physical entity*
Level 2	6	*causal agent* *group* *matter* *object* *process* *thing*
Level 1	8	*body of water* *community* *geological formation* *land* *person* *phenomenon* *solid* *whole*
Level 0	100	Base labels of CIFAR-100

Table B.1: Training hierarchy extracted from CIFAR-100

B.1.2 ImageNet (ILSVRC-2012)

The dataset contains colour images of varying size categorized in 1000 classes, 1 million images for the training set and 150000 images for the testing set. We split randomly split the training set in 90% for the training and 10% for validation. The randomness was seeded (seed : 123456789) for reproducibility purposes. The labels are naturally linked to WordNet.

We used the same import parameters as *torchvision* for testing set : we resized imported pictures to 256×256, center-cropped them to 224×224, then normalized with RGB mean set to [0.485, 0.456, 0.406] and RGB standard deviation set to [0.229, 0.224, 0.225].
Training pictures were resized to 256×256, random-cropped to 224×224, randomly flipped horizontally then normalized with RGB mean set to [0.485, 0.456, 0.406] and RGB standard deviation set to [0.229, 0.224, 0.225]. Here again, the random process was seeded, with seed 123456789.

The training hierarchy was computed with depth 5, leading to hierarchy levels of size 2, 5, 8, 16, 1000. Labels of each level can be found in Table 2.

Level of hierarchy	Number of labels	Contained labels
Level 4	2	*abstraction* *physical entity*
Level 3	5	*attribute* *causal agent* *communication* *matter* *object*
Level 2	8	*geological formation* *person* *shape* *sign* *signal* *solid* *substance* *whole*
Level 1	16	*artifact* *cliff* *contestant* *explorer* *food* *living thing* *material* *natural depression* *natural elevation* *natural object* *peer* $round_shape$ *shore* *spring* *street sign* *visual signal*
Level 0	1000	Base labels of ImageNet

Table B.2: Training hierarchy extracted from ImageNet

B.2 Networks hyper-parameters and training

B.2.1 VGG16 (CIFAR-100)

For CIFAR-100, we trained VGG-16 from scratch for 180 epochs. We used a Stochastic Gradient Descent (SGD) with learning rate at 0.1, momentum at 0.9 and weight decay at 10^{-4}. Learning rate was divided by 10 every 60 epochs. We used a cross-entropy loss. The batch size was set to 128.

B.2.2 ResNet-18 (CIFAR-100)

We trained ResNet-18 from scratch for 180 epochs. We used a SGD with learning rate at 0.1, momentum at 0.9 and weight decay at 10^{-4}. Learning rate was divided by 10 every 60 epochs. We used a cross-entropy loss. The batch size was set to 128.

B.2.3 HA-ResNet-18 (CIFAR-100)

HA-ResNet-18 was constructed following the method proposed by [7]. The hidden dimensions were set to 200, 300 and 400. It was trained from scratch on the hierarchy mentioned above for 180 epochs, with SGD, learning rate at 0.01, momentum at 0.9 and weight decay at 10^{-4}. Learning rate was divided by 10 every 60 epochs. We used cross-entropy losses on every output level, all equally weighted by a factor 1. The batch size was set to 128.

B.2.4 VGG16 (ImageNet)

We used default pretrained weights provided with *torchvision* python package without performing any further fine-tuning.

B.2.5 ResNet-152 (ImageNet)

We used default pretrained weights provided with *torchvision* python package without performing any further fine-tuning.

B.2.6 HA-ResNet-152 (ImageNet)

HA-ResNet-152 was constructed following the method proposed by [7]. The base weights for the layers in common with ResNet-152 were the one provided by the pretrained version of ResNet-152. The hidden dimensions were set to 200, 200, 300 and 400. It was trained on the hierarchy mentioned above for 1 epochs, with SGD, learning rate at 0.01, momentum at 0.9 and weight decay at 10^{-4}. We used cross-entropy losses on every output level, all equally weighted by a factor 1. The batch size was set to 64.

B.3 Embedding parameters

B.3.1 VGG16 (CIFAR-100)

For VGG16 on CIFAR-100, we ran FastRP with 128 embedding dimensions, for 3 iterations and with iteration weights set to 0, 1 and 1. The random seed was set to 42.

We also ran Node2vec with 128 embedding dimensions, for 1 iteration. Again, the random seed was set to 42.

B.3.2 ResNet-18 (CIFAR-100)

For ResNet-18 on CIFAR-100, we ran FastRP with 128 embedding dimensions, for 3 iterations and with iteration weights set to 0, 1 and 1. The random seed was set to 42.

We also ran Node2vec with 128 embedding dimensions, for 1 iteration. Again, the random seed was set to 42.

All the convolutionnal and linear layers were used to construct the co-activation graph.

B.3.3 HA-ResNet-18 (CIFAR-100)

For HA-ResNet-18 on CIFAR-100, we ran FastRP with 128 embedding dimensions, for 3 iterations and with iteration weights set to 0, 1 and 1. The random seed was set to 42.

We also ran Node2vec with 128 embedding dimensions, for 1 iteration. Again, the random seed was set to 42.

All the convolutionnal and linear layers were used to construct the co-activation graph.

B.3.4 VGG16 (ImageNet)

For VGG16 on ImageNet, we ran FastRP with 128 embedding dimensions, for 3 iterations and with iteration weights set to 0, 1 and 1. The random seed was set to 42.

We also ran Node2vec with 128 embedding dimensions, for 1 iteration. Again, the random seed was set to 42.

B.3.5 ResNet-152 (ImageNet)

For ResNet-152 on ImageNet, we ran FastRP with 256 embedding dimensions, for 10 iterations. Iteration weights were set to [0, 0.2, 0.2, 0.4, 0.4, 0.6, 0.6, 0.8, 0.8, 1]. The random seed was set to 42.

We also ran Node2vec with 256 embedding dimensions, for 10 iteration. Again, the random seed was set to 42.

B.3.6 HA-ResNet-152 (ImageNet)

For HA-ResNet-152 on ImageNet, we ran FastRP with 256 embedding dimensions, for 10 iterations. Iteration weights were set to [0, 0.2, 0.2, 0.4, 0.4, 0.6, 0.6, 0.8, 0.8, 1]. The random seed was set to 42.

We also ran Node2vec with 256 embedding dimensions, for 10 iteration. Again, the random seed was set to 42.

B.4 Layer used for each model

In order to enhance reproducibility, Table B.3 lists all the layer names that were used in the process of building the co-activation graph for each model in Chapter 4.

VGG16	ResNet18	HAResNet18	ResNet152	HAResNet152
features_0	conv1	conv1	layer3_31_conv3	layer3_31_conv3
features_3	layer1_0_conv1	layer1_0_conv1	layer3_32_conv3	layer3_32_conv3
features_7	layer1_0_conv2	layer1_0_conv2	layer3_33_conv3	layer3_33_conv3
features_10	layer1_1_conv1	layer1_1_conv1	layer3_34_conv3	layer3_34_conv3
features_14	layer1_1_conv2	layer1_1_conv2	layer3_35_conv3	layer3_35_conv3
features_17	layer2_0_conv1	layer2_0_conv1	layer4_0_conv3	layer4_0_conv3
features_20	layer2_0_conv2	layer2_0_conv2	layer4_1_conv3	layer4_1_conv3
features_24	layer2_0_downsample_0	layer2_0_downsample_0	layer4_2_conv3	layer4_2_conv3
features_27	layer2_1_conv1	layer2_1_conv1	fc	fc
features_30	layer2_1_conv2	layer2_1_conv2	-	-
classifier_6	layer3_0_conv1	layer3_0_conv1	-	-
-	layer3_0_conv2	layer3_0_conv2	-	-
-	layer3_0_downsample_0	layer3_0_downsample_0	-	-
-	layer3_1_conv1	layer3_1_conv1	-	-
-	layer3_1_conv2	layer3_1_conv2	-	-
-	layer4_0_conv1	layer4_0_conv1	-	-
-	layer4_0_conv2	layer4_0_conv2	-	-
-	layer4_0_downsample_0	layer4_0_downsample_0	-	-
-	layer4_1_conv1	layer4_1_conv1	-	-
-	layer4_1_conv2	layer4_1_conv2	-	-
-	fc	fc	-	-

Table B.3: The layers that were used when constructing the co-activation for each model.

B.5 Mistakes

Chapter 5 proposed a method for generating local explanations based on the prediction of semantic attributes through the use of a link prediction algorithm over

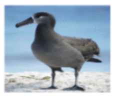
Misclassified input image	Expected class	Predicted class	Factual explanation	Counterfactual explanation
	Fish crow	Brandt Cormorant	The input was misclassified as Brandt Cormorant instead of Fish Crow because the model has identified all-purpose as value for bill_shape instead of hooked_seabird.	If the attribute bill_shape assumed the value hooked_seabird, this image would more likely be classified as a Fish Crow instead of Brandt Cormorant.
	Fish crow	Brandt Cormorant	The input was misclassified as Red Winged Blackbird instead of Fish Crow because the model has identified all-purpose as value for bill_shape instead of dagger.	If the attribute bill_shape assumed the value dagger, this image would more likely be classified as a Fish Crow instead of Red Winged Blackbird.
	Caspian Tern	Black Tern	The input was misclassified as Caspian Tern instead of Black Tern because the model has identified white as the value for primary_color instead of grey and white as the value for underparts_color instead of black and white as the value for nape_color instead of black and white as the value for upperparts_color instead of grey and dagger as the value for bill_shape instead of all_purpose.	If the attribute primary_color assumed the value grey and underparts_color assumed the value black and nape_color assumed the value black and upperparts_color assumed the value grey and bill_shape assumed the value all_purpose, this image would more likely be classified as a Black Tern instead of Caspian Tern.
	Long tailed jaeger	 Black footed albatross	The input was misclassified as Black_footed_albatross instead of Long_tailed_Jaeger because the model has identified brown as the value for primary_color instead of grey and brown as the value for underparts_color instead of grey and hooked_seabird as the value for bill_shape instead of all_purpose.	If the attribute primary_color assumed the value grey and underparts_color assumed the value grey and bill_shape assumed the value all_purpose, this image would more likely be classified as a Long_tailed_Jaeger instead of Black_footed_albatross.
	Ivory gull	Ring billed gull	The input was misclassified as Ring_billed_Gull instead of Ivory_gull because the model has identified hooked_seabird as value for bill_shape instead of all_purpose.	If the attribute bill_shape assumed the value all_purpose, this image would more likely be classified as a Ivory_gull instead of Ring_billed_Gull.
	Long tailed jaeger	Pomarine Jarger	The input was misclassified as Pomarine_Jaeger instead of Long_Tailed_Jaeger because the model has identified hooked_seabird as value for bill_shape instead of all_purpose.	If the attribute bill_shape assumed the value all_purpose, this image would more likely be classified as a Long_Tailed_Jaeger instead of Pomarine_Jaeger.
	Vesper Sparrow	Blue grosbeak	The input was misclassified as Blue_Grosbeak instead of Vesper_Sparrow because the model has identified blue as value for primary_color instead of brown and blue as the value for upperparts_color instead of brown.	If the attribute primary_color assumed the value brown and upperparts_color assumed the value brown, this image would more likely be classified as a Vesper_Sparrow instead of Blue_Grosbeak.
	Rock Wren	Blue Grosbeak	The input was misclassified as Blue_Grosbeak instead of Rock_Wren because the model has identified blue as value for primary_color instead of buff and blue as the value for upperparts_color instead of brown and blue as the value for nape_color instead of buff and blue as the color of underparts_color instead of buff.	If the attribute primary_color assumed the value buff and upperparts_color assumed the value brown and nape_color assumed the value buff and underparts_color assumed the value buff this image would more likely be classified as a Rock_Wren instead of Blue_Grosbeak.

Figure B.1: Factual and counterfactual explanations.

co-activation graphs. In order to make the visual analysis more robust, Figure B.1 shows eight additional factual and counterfactual explanations that help in assessing the capabilities of the method proposed in Chapter 5.

In Figure B.1, some of the explanations are easier to analyse than others because they are based around attributes that are visually easier to distinct. For instance, a hooked_seabird and an allpurpose bill_shape are visually very distinct, which makes it easier to analyse the first, fourth, fifth and sixth explanations, given that the misclassified images seem to have a hooked_seabird bill_shape instead of allpurpose. A similar behaviour can be observed in the third explanation, where our model identified the presence of white colors instead of grey and black, and the seventh and eigth explanations, where our model identified identified the presence of a blue color instead of buff and brown. These cases are also easy to distinguish considering that the involved colors are very distinct and they can also be visually inspected from the misclassified image.

However, not all explanations can be easily analysed, especially when the attribute values are hard to be visually distinguished. For instance, in the second mistake our method has identified an allpurpose bill_shape instead of dagger, but it is not easy to distinguish between these two values of bill_shape. This may be the case where, even though the explanation may be accurate with respect to the deep model, it may not be useful for the end user. Therefore, future work should consider adapting the proposed method to prioritise semantic attributes that are relevant for the end user, which could be achieved by considering human feedback during the process of generating the explanations.

Appendix C

Presentations on Work from this Thesis

Appendix D

Awards

- Best Presentation at the Intern Awards at Nokia Bell Labs Ireland (2019)

Bibliography

[1] Lada A Adamic **and** Eytan Adar. “Friends and neighbors on the Web”. **in***Social Networks*: 25.3 (2003), **pages** 211–230. ISSN: 0378-8733. DOI: https://doi.org/10.1016/S0378-8733(03)00009-1. URL: https://www.sciencedirect.com/science/article/pii/S0378873303000091.

[2] Marjan Alirezaie **andothers**. “A symbolic approach for explaining errors in image classification tasks”. **in** *Working Papers and Documents of the IJCAI-ECAI-2018 Workshop on*: 2018.

[3] Laith Alzubaidi **andothers**. “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions”. **in***Journal of Big Data*: 8.1 (**march** 2021). DOI: 10.1186/s40537-021-00444-8. URL: https://doi.org/10.1186/s40537-021-00444-8.

[4] Eric B. Bartlett. “Self determination of input variable importance using neural networks”. **in***Neural, Parallel Scientific Computations*: 2 (**march** 1994), **pages** 103–114.

[5] Alejandro Barredo Arrieta **andothers**. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. **in***Information Fusion*: 58 (2020), **pages** 82–115. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2019.12.012. URL: https://www.sciencedirect.com/science/article/pii/S1566253519308103.

[6] David Bau **andothers**. “Network Dissection: Quantifying Interpretability of Deep Visual Representations”. **in***Computer Vision and Pattern Recognition*: 2017.

[7] Alsallakh Bilal **andothers**. "Do Convolutional Neural Networks Learn Class Hierarchy?" **in***IEEE Transactions on Visualization and Computer Graphics*: 24.1 (**january** 2018), **pages** 152–162. DOI: 10.1109/tvcg.2017.2744683. URL: https://doi.org/10.1109%2Ftvcg.2017.2744683.

[8] Vincent Blondel **andothers**. "Fast Unfolding of Communities in Large Networks". **in***Journal of Statistical Mechanics Theory and Experiment*: 2008 (**april** 2008). DOI: 10.1088/1742-5468/2008/10/P10008.

[9] Guido Bologna. "A Simple Convolutional Neural Network with Rule Extraction". **in***Applied Sciences*: 9.12 (**june** 2019), **page** 2411. DOI: 10.3390/app9122411. URL: https://doi.org/10.3390/app9122411.

[10] Ed Bullmore **and** Olaf Sporns. "Complex brain networks: graph theoretical analysis of structural and functional systems". **in***Nature Reviews Neuroscience*: 10.3 (**february** 2009), **pages** 186–198. DOI: 10.1038/nrn2575. URL: https://doi.org/10.1038/nrn2575.

[11] Ruth M. J. Byrne. "Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning". **in***Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*: International Joint Conferences on Artificial Intelligence Organization, **july** 2019, **pages** 6276–6282. DOI: 10.24963/ijcai.2019/876. URL: https://doi.org/10.24963/ijcai.2019/876.

[12] Shijie Cao **andothers**. "SeerNet: Predicting Convolutional Neural Network Feature-Map Sparsity Through Low-Bit Quantization". **in***2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*: 2019, **pages** 11208–11217. DOI: 10.1109/CVPR.2019.01147.

[13] Shan Carter **andothers**. "Activation Atlas". **in***Distill*: (2019). https://distill.pub/2019/activati
atlas. DOI: 10.23915/distill.00015.

[14] V. Chan **and** C. W. Chan. "Development and application of an algorithm for extracting multiple linear regression equations from artificial neural networks

for nonlinear regression problems”. **in***2016 IEEE 15th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC)*: **august** 2016, **pages** 479–488.

[15] Angelos Chatzimparmpas **andothers**. “A survey of surveys on the use of visualization for interpreting machine learning models”. **in***Information Visualization*: (**march** 2020), **page** 147387162090467. DOI: 10.1177/1473871620904671. URL: https://doi.org/10.1177/1473871620904671.

[16] Haochen Chen **andothers**. *Fast and Accurate Network Embeddings via Very Sparse Random Projection.* 2019. DOI: 10.48550/ARXIV.1908.11512. URL: https://arxiv.org/abs/1908.11512.

[17] Yunjey Choi **andothers**. “StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation”. **in***CoRR*: abs/1711.09020 (2017). arXiv: 1711.09020. URL: http://arxiv.org/abs/1711.09020.

[18] Philipp Cimiano **and** Heiko Paulheim. “Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods”. **in***Semant. Web*: 8.3 (**january** 2017), **pages** 489–508. ISSN: 1570-0844. DOI: 10.3233/SW-160218. URL: https://doi.org/10.3233/SW-160218.

[19] Guillaume Cleuziou **and** Jose G. Moreno. “QASSIT at SemEval-2016 Task 13: On the integration of Semantic Vectors in Pretopological Spaces for Lexical Taxonomy Acquisition”. **in***Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*: San Diego, California: Association for Computational Linguistics, **june** 2016, **pages** 1315–1319. DOI: 10.18653/v1/S16-1205. URL: https://aclanthology.org/S16-1205.

[20] J. Deng **andothers**. “ImageNet: A Large-Scale Hierarchical Image Database”. **in***CVPR09*: 2009.

[21] Jia Deng **andothers**. “Imagenet: A large-scale hierarchical image database”. **in***2009 IEEE conference on computer vision and pattern recognition*: Ieee. 2009, **pages** 248–255.

[22] Zhuanlian Ding **andothers**. "Overlapping Community Detection based on Network Decomposition". **in***Scientific Reports*: 6.1 (**april** 2016). DOI: 10.1038/srep24115. URL: https://doi.org/10.1038/srep24115.

[23] Jonathan Dodge **andothers**. "Explaining Models: An Empirical Study of How Explanations Impact Fairness Judgment". **in***Proceedings of the 24th International Conference on Intelligent User Interfaces*: IUI '19. Marina del Ray, California: Association for Computing Machinery, 2019, **pages** 275–285. ISBN: 9781450362726. DOI: 10.1145/3301275.3302310. URL: https://doi.org/10.1145/3301275.3302310.

[24] Alexey Dosovitskiy **andothers**. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". **in***CoRR*: abs/2010.11929 (2020). arXiv: 2010.11929. URL: https://arxiv.org/abs/2010.11929.

[25] Dumitru Erhan **andothers**. "Visualizing Higher-Layer Features of a Deep Network". **in***Technical Report, Univeristé de Montréal*: (**january** 2009).

[26] Giorgos Filandrianos **andothers**. "Conceptual Edits as Counterfactual Explanations". **in***Proceedings of the AAAI 2022 Spring Symposium on Machine Learning and Knowledge Engineering for Hybrid Intelligence (AAAI-MAKE 2022), Stanford University, Palo Alto, California, USA, March 21-23, 2022*: **byeditor**Andreas Martin **andothers**. **volume** 3121. CEUR Workshop Proceedings. CEUR-WS.org, 2022. URL: http://ceur-ws.org/Vol-3121/paper6.pdf.

[27] Ruth Fong **and** Andrea Vedaldi. "Net2Vec: Quantifying and Explaining how Concepts are Encoded by Filters in Deep Neural Networks". **in***CoRR*: abs/1801.03454 (2018). arXiv: 1801.03454. URL: http://arxiv.org/abs/1801.03454.

[28] Giuseppe Futia **and** Antonio Vetrò. "On the Integration of Knowledge Graphs into Deep Learning Models for a More Comprehensible AI—Three Challenges for Future Research". **in***Information*: 11.2 (2020). ISSN: 2078-2489. DOI: 10.3390/info11020122. URL: https://www.mdpi.com/2078-2489/11/2/122.

[29] Wei Gao **andothers**. "Study of biological networks using graph theory". **in***Saudi Journal of Biological Sciences*: 25.6 (2018), **pages** 1212–1219. ISSN: 1319-562X. DOI: https://doi.org/10.1016/j.sjbs.2017.11.022. URL: https://www.sciencedirect.com/science/article/pii/S1319562X17302966.

[30] J. O. Garcia **andothers**. "Applications of Community Detection Techniques to Brain Graphs: Algorithmic Considerations and Implications for Neural Function". **in***Proceedings of the IEEE*: 106.5 (2018), **pages** 846–867.

[31] Dario Garcia-Gasulla **andothers**. "An Out-of-the-box Full-Network Embedding for Convolutional Neural Networks". **in***2018 IEEE International Conference on Big Knowledge (ICBK)*: (2018), **pages** 168–175.

[32] Dario Garcia-Gasulla **andothers**. "An Out-of-the-box Full-Network Embedding for Convolutional Neural Networks". **in***2018 IEEE International Conference on Big Knowledge (ICBK)*: 2018, **pages** 168–175. DOI: 10.1109/ICBK.2018.00030.

[33] Dario Garcia-Gasulla **andothers**. "Building Graph Representations of Deep Vector Embeddings". **in***Proceedings of the 2nd Workshop on Semantic Deep Learning, SemDeep@IWCS 2017, Montpellier, France, September 19, 2017*: **byeditor**Dagmar Gromann, Thierry Declerck **and** Georg Heigl. Association for Computational Linguistics, 2017, **pages** 9–15. URL: https://aclanthology.org/W17-7302/.

[34] Dario Garcia-Gasulla **andothers**. "On the Behavior of Convolutional Nets for Feature Extraction". **in***CoRR*: abs/1703.01127 (2017). arXiv: 1703.01127. URL: http://arxiv.org/abs/1703.01127.

[35] Amirata Ghorbani **andothers**. "Towards automatic concept-based explanations". **in***Advances in Neural Information Processing Systems*: 2019, **pages** 9273–9282.

[36] Leilani H. Gilpin **andothers**. *Explaining Explanations: An Overview of Interpretability of Machine Learning*. 2018. arXiv: 1806.00069 [cs.AI].

[37] Yash Goyal **andothers**. *Explaining Classifiers with Causal Concept Effect (CaCE)*. 2019. DOI: 10.48550/ARXIV.1907.07165. URL: https://arxiv.org/abs/1907.07165.

[38] Riccardo La Grassa, Ignazio Gallo **and** Nicola Landro. "Learn Class Hierarchy using Convolutional Neural Networks". **in***CoRR*: abs/2005.08622 (2020). arXiv: 2005.08622. URL: https://arxiv.org/abs/2005.08622.

[39] Aditya Grover **and** Jure Leskovec. *node2vec: Scalable Feature Learning for Networks*. 2016. DOI: 10.48550/ARXIV.1607.00653. URL: https://arxiv.org/abs/1607.00653.

[40] Felix Grün **andothers**. "A Taxonomy and Library for Visualizing Learned Features in Convolutional Neural Networks". **in***arXiv preprint arXiv:1606.07757*: (2016).

[41] D. Gunning **andothers**. "XAI-Explainable artificial intelligence". **in***Science Robotics*: 4.37 (**december** 2019). This is the author?s version of the work. It is posted here by permission of the AAAS for personal use, not for redistribution. The definitive version was published in Science Robotics on 4 (37) 18 December 2019, DOI: 10.1126/scirobotics.aay7120., eaay7120. DOI: 10.1126/scirobotics.aay7120. URL: https://openaccess.city.ac.uk/id/eprint/23405/.

[42] Isma Hadji **and** Richard P. Wildes. "What Do We Understand About Convolutional Networks?" **in***CoRR*: abs/1803.08834 (2018). arXiv: 1803.08834. URL: http://arxiv.org/abs/1803.08834.

[43] Tameru Hailesilassie. *Rule Extraction Algorithm for Deep Neural Networks: A Review*. 2016. arXiv: 1610.05267 [cs.CV].

[44] Zhenliang He **andothers**. "AttGAN: Facial Attribute Editing by Only Changing What You Want". **in***IEEE Transactions on Image Processing*: 28.11 (2019), **pages** 5464–5478. DOI: 10.1109/TIP.2019.2916751.

[45] Lisa Anne Hendricks **andothers**. *Grounding Visual Explanations.* 2018. arXiv: 1807.09685 [cs.CV].

[46] High-Level Expert Group on AI. *Ethics guidelines for trustworthy AI.* eng. Report. Brussels: European Commission, **april** 2019. URL: https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai.

[47] Vitor Horta **andothers**. "Analyzing scientific context of researchers and communities by using complex network and semantic technologies". **in***Future Gener. Comput. Syst.*: 89 (2018), **pages** 584–605. DOI: 10.1016/j.future.2018.07.012. URL: https://doi.org/10.1016/j.future.2018.07.012.

[48] Vitor A. C. Horta **and** Alessandra Mileo. "Generating Local Textual Explanations for CNNs: A Semantic Approach Based on Knowledge Graphs". **in***AIxIA 2021 – Advances in Artificial Intelligence*: **byeditor**Stefania Bandini **andothers**. Cham: Springer International Publishing, 2022, **pages** 532–549. ISBN: 978-3-031-08421-8.

[49] Vitor A. C. Horta **and** Alessandra Mileo. "Towards Explaining Deep Neural Networks Through Graph Analysis". **in***Database and Expert Systems Applications*: **byeditor**Gabriele Anderst-Kotsis **andothers**. Cham: Springer International Publishing, 2019, **pages** 155–165. ISBN: 978-3-030-27684-3.

[50] Vitor A.C. Horta **andothers**. "Extracting knowledge from Deep Neural Networks through graph analysis". **in***Future Generation Computer Systems*: 120 (2021), **pages** 109–118. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2021.02.009. URL: https://www.sciencedirect.com/science/article/pii/S0167739X21000613.

[51] Mir Riyanul Islam **andothers**. "A Systematic Review of Explainable Artificial Intelligence in Terms of Different Application Domains and Tasks". **in***Applied Sciences*: 12.3 (2022). ISSN: 2076-3417. DOI: 10.3390/app12031353. URL: https://www.mdpi.com/2076-3417/12/3/1353.

[52] Mathieu Jacomy **andothers**. “ForceAtlas2, a Continuous Graph Layout Algorithm for Handy Network Visualization Designed for the Gephi Software”. **in***PLoS ONE*: 9.6 (**june** 2014). **byeditor**Mark R. Muldoon, e98679. DOI: 10.1371/journal.pone.0098679. URL: https://doi.org/10.1371/journal.pone.0098679.

[53] D. Jin **andothers**. “A Survey of Community Detection Approaches: From Statistical Modeling to Deep Learning”. **in***IEEE Transactions on Knowledge amp; Data Engineering*: 01 (**august** 5555), **pages** 1–1. ISSN: 1558-2191. DOI: 10.1109/TKDE.2021.3104155.

[54] R. Jonker **and** A. Volgenant. “A shortest augmenting path algorithm for dense and sparse linear assignment problems”. **in***Computing*: 38.4 (**december** 1987), **pages** 325–340. DOI: 10.1007/bf02278710. URL: https://doi.org/10.1007/bf02278710.

[55] Enric Junque de Fortuny **and** David Martens. “Active Learning-Based Pedagogical Rule Extraction”. **in***IEEE transactions on neural networks and learning systems*: 26 (**january** 2015).

[56] Eoin M. Kenny **and** Mark T. Keane. “On Generating Plausible Counterfactual and Semi-Factual Explanations for Deep Learning”. **in***Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*: AAAI Press, 2021, **pages** 11575–11585. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17377.

[57] Asifullah Khan **andothers**. “A survey of the recent architectures of deep convolutional neural networks”. **in***Artificial Intelligence Review*: 53.8 (**april** 2020), **pages** 5455–5516. DOI: 10.1007/s10462-020-09825-6. URL: https://doi.org/10.1007/s10462-020-09825-6.

[58] Salman Khan **andothers**. “Transformers in Vision: A Survey”. **in***ACM Comput. Surv.*: 54.10s (**september** 2022). ISSN: 0360-0300. DOI: 10.1145/3505244. URL: https://doi.org/10.1145/3505244.

[59] Been Kim **andothers**. “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)”. **in**(2017): DOI: 10.48550/ARXIV.1711.11279. URL: https://arxiv.org/abs/1711.11279.

[60] Daeeun Kim **and** Jaeho Lee. “Handling Continuous-Valued Attributes in Decision Tree with Neural Network Modelling”. **in***ECML*: 2000.

[61] Itisha Kothiyal **andothers**. “Utilization of GAN for Automatic Evaluation of Counterfactuals: Challenges and Opportunities”. **in***30th Irish Conference on Artificial Intelligence and Cognitive Science*: (2022). URL: https://arxiv.org/abs/2102.12092.

[62] R. Krishnan, G. Sivakumar **and** P. Bhattacharya. “A search technique for rule extraction from trained neural networks”. **in***Pattern Recognition Letters*: 20.3 (1999), **pages** 273–280. ISSN: 0167-8655.

[63] Alex Krizhevsky. *Learning multiple layers of features from tiny images*. techreport. 2009.

[64] Alex Krizhevsky, Vinod Nair **and** Geoffrey Hinton. “CIFAR-100 (Canadian Institute for Advanced Research)”. **in**(): URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[65] Alex Krizhevsky, Ilya Sutskever **and** Geoffrey E. Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. **in***Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*: NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, **pages** 1097–1105.

[66] Freddy Lecue. “On the role of knowledge graphs in explainable AI”. **in***Semantic Web*: 11 (**december** 2019), **pages** 1–11. DOI: 10.3233/SW-190374.

[67] Y. Lecun **andothers**. “Gradient-based learning applied to document recognition”. **in***Proceedings of the IEEE*: 86.11 (1998), **pages** 2278–2324. DOI: 10.1109/5.726791.

[68] Yann LeCun **and** Corinna Cortes. “MNIST handwritten digit database”. **in**(2010): URL: http://yann.lecun.com/exdb/mnist/.

[69] Yu Liang **andothers**. “Explaining the black-box model: A survey of local interpretation methods for deep neural networks”. **in***Neurocomputing*: 419 (2021), **pages** 168–182. ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0925231220312716.

[70] Tianyang Lin **andothers**. “A survey of transformers”. **in***AI Open*: 3 (2022), **pages** 111–132. ISSN: 2666-6510. DOI: https://doi.org/10.1016/j.aiopen.2022.10.001. URL: https://www.sciencedirect.com/science/article/pii/S2666651022000146.

[71] Tsung-Yi Lin **andothers**. *Microsoft COCO: Common Objects in Context*. 2014. arXiv: 1405.0312 [cs.CV].

[72] Jin Liu **andothers**. “Complex Brain Network Analysis and Its Applications to Brain Disorders: A Survey”. **in***Complexity*: 2017 (2017), **pages** 1–27. DOI: 10.1155/2017/8362741. URL: https://doi.org/10.1155/2017/8362741.

[73] B. Mak **and** R. W. Blanning. “An empirical measure of element contribution in neural networks”. **in***IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*: 28.4 (**november** 1998), **pages** 561–564. ISSN: 1094-6977.

[74] Félix Martel **and** Amal Zouaq. “Taxonomy Extraction Using Knowledge Graph Embeddings and Hierarchical Clustering”. **in***Proceedings of the 36th Annual ACM Symposium on Applied Computing*: SAC ’21. Virtual Event, Republic of Korea: Association for Computing Machinery, 2021, **pages** 836–844.

ISBN: 9781450381048. DOI: 10.1145/3412841.3441959. URL: https://doi.org/10.1145/3412841.3441959.

[75] Marghny H. Mohamed. "Rules Extraction from Constructively Trained Neural Networks Based on Genetic Algorithms". **in***Neurocomput.*: 74.17 (**october** 2011), **pages** 3180–3192. ISSN: 0925-2312.

[76] Basudeb Mondal **and** Kajal De. "An overview applications of graph theory in real field". **in***International Journal of Scientific Research in Computer Science, Engineering and Information Technology*: 2.5 (2017), **pages** 751–759.

[77] Mohamed EL-MOUSSAOUI **andothers**. "A comprehensive literature review on community detection: Approaches and applications". **in***Procedia Computer Science*: 151 (2019). The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops, **pages** 295–302. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.04.042. URL: https://www.sciencedirect.com/science/article/pii/S1877050919305046.

[78] Chris Olah **andothers**. "The Building Blocks of Interpretability". **in***Distill*: (2018). https://distill.pub/2018/building-blocks. DOI: 10.23915/distill.00010.

[79] Lawrence Page **andothers**. *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab, **november** 1999. URL: http://ilpubs.stanford.edu:8090/422/.

[80] Zhuwei Qin **andothers**. *How convolutional neural network see the world - A survey of convolutional neural network visualization methods.* 2018. arXiv: 1804.11191 [cs.CV].

[81] Alec Radford, Luke Metz **and** Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks".

in*4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*: **byeditor**Yoshua Bengio **and** Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.06434.

[82] Aditya Ramesh **andothers**. "Zero-Shot Text-to-Image Generation". **in***CoRR*: abs/2102.12092 (2021). arXiv: 2102.12092. URL: https://arxiv.org/abs/2102.12092.

[83] 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*: KDD '16. San Francisco, California, USA: Association for Computing Machinery, 2016, **pages** 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

[84] Petar Ristoski **andothers**. "Large-Scale Taxonomy Induction Using Entity and Word Embeddings". **in***Proceedings of the International Conference on Web Intelligence*: WI '17. Leipzig, Germany: Association for Computing Machinery, 2017, **pages** 81–87. ISBN: 9781450349512. DOI: 10.1145/3106426.3106465. URL: https://doi.org/10.1145/3106426.3106465.

[85] Robin Rombach **andothers**. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV].

[86] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". **in***Nature Machine Intelligence*: 1.5 (**may** 2019), **pages** 206–215. DOI: 10.1038/s42256-019-0048-x. URL: https://doi.org/10.1038/s42256-019-0048-x.

[87] Mark Sandler **andothers**. "MobileNetV2: Inverted Residuals and Linear Bottlenecks." **in***CVPR*: IEEE Computer Society, 2018, **pages** 4510–4520. URL: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2018.html#SandlerHZZC18.

[88] Md. Kamruzzaman Sarker **andothers**. “Explaining Trained Neural Networks with Semantic Web Technologies: First Steps”. **in** *CoRR*: abs/1710.04324 (2017). arXiv: 1710.04324. URL: http://arxiv.org/abs/1710.04324.

[89] Karen Simonyan **and** Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. **in** *CoRR*: abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.

[90] Barry Smyth **and** Mark T. Keane. “A Few Good Counterfactuals: Generating Interpretable, Plausible and Diverse Counterfactual Explanations”. **in** *CoRR*: abs/2101.09056 (2021). arXiv: 2101.09056. URL: https://arxiv.org/abs/2101.09056.

[91] Rion Snow, Daniel Jurafsky **and** Andrew Ng. “Learning Syntactic Patterns for Automatic Hypernym Discovery”. **in** *Advances in Neural Information Processing Systems*: **byeditor** L. Saul, Y. Weiss **and** L. Bottou. **volume** 17. MIT Press, 2004. URL: https://proceedings.neurips.cc/paper/2004/file/358aee4cc897452c00244351e4d91f69-Paper.pdf.

[92] Ilaria Tiddi, Freddy Lécué **and** Pascal Hitzler, **editors**. *Knowledge Graphs for eXplainable Artificial Intelligence: Foundations, Applications and Challenges*. **volume** 47. Studies on the Semantic Web. IOS Press, 2020. ISBN: 978-1-64368-080-4. URL: http://ebooks.iospress.nl/volume/knowledge-graphs-for-explainable-artificial-intelligence-foundations-applications-and-challenges.

[93] Ilaria Tiddi **and** Stefan Schlobach. “Knowledge graphs as tools for explainable machine learning: A survey”. **in** *Artificial Intelligence*: 302 (2022), **page** 103627. ISSN: 0004-3702. URL: https://www.sciencedirect.com/science/article/pii/S0004370221001788.

[94] Erico Tjoa **and** Cuntai Guan. “A Survey on Explainable Artificial Intelligence (XAI): Towards Medical XAI”. **in** *CoRR*: abs/1907.07374 (2019). arXiv: 1907.07374. URL: http://arxiv.org/abs/1907.07374.

[95] Geoffrey G. Towell **and** Jude W. Shavlik. "Extracting Refined Rules from Knowledge-Based Neural Networks". **in***Mach. Learn.*: 13.1 (**october** 1993), **pages** 71–101. ISSN: 0885-6125.

[96] Nenad Trinajstic. *Chemical graph theory*. CRC press, 2018.

[97] Nicole Van Hoeck, Patrick D. Watson **and** Aron K. Barbey. "Cognitive neuroscience of human counterfactual reasoning". **in***Frontiers in Human Neuroscience*: 9 (2015), **page** 420. ISSN: 1662-5161. DOI: 10.3389/fnhum.2015.00420. URL: https://www.frontiersin.org/article/10.3389/fnhum.2015.00420.

[98] Giulia Vilone **and** Luca Longo. "Classification of Explainable Artificial Intelligence Methods through Their Output Formats". **in***Machine Learning and Knowledge Extraction*: 3.3 (2021), **pages** 615–661. ISSN: 2504-4990. DOI: 10.3390/make3030032. URL: https://www.mdpi.com/2504-4990/3/3/32.

[99] Giulia Vilone **and** Luca Longo. *Explainable Artificial Intelligence: a Systematic Review*. 2020. DOI: 10.48550/ARXIV.2006.00093. URL: https://arxiv.org/abs/2006.00093.

[100] Alvin Wan **andothers**. "NBDT: Neural-Backed Decision Trees". **in***CoRR*: abs/2004.00221 (2020). arXiv: 2004.00221. URL: https://arxiv.org/abs/2004.00221.

[101] Chengyu Wang, Xiaofeng He **and** Aoying Zhou. "A Short Survey on Taxonomy Learning from Text Corpora: Issues, Resources and Recent Advances". **injanuary** 2017: **pages** 1190–1203. DOI: 10.18653/v1/D17-1123.

[102] P. Welinder **andothers**. *Caltech-UCSD Birds 200*. techreport CNS-TR-2010-001. California Institute of Technology, 2010.

[103] Han Xiao, Kashif Rasul **and** Roland Vollgraf. "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms". **in**(28 **august** 2017): arXiv: cs.LG/1708.07747 [cs.LG].

[104] Chih-Kuan Yeh **andothers**. *On Completeness-aware Concept-Based Explanations in Deep Neural Networks*. 2019. DOI: 10.48550/ARXIV.1910.07969. URL: https://arxiv.org/abs/1910.07969.

[105] Jiaxuan You **andothers**. “Graph Structure of Neural Networks”. **in***Proceedings of the 37th International Conference on Machine Learning*: **byeditor**Hal Daumé III **and** Aarti Singh. **volume** 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, **pages** 10881–10891. URL: https://proceedings.mlr.press/v119/you20b.html.

[106] Quanshi Zhang **andothers**. “Extraction of an Explanatory Graph to Interpret a CNN”. **in***IEEE Transactions on Pattern Analysis and Machine Intelligence*: 43.11 (2021), **pages** 3863–3877. DOI: 10.1109/TPAMI.2020.2992207.

[107] Jichen Zhu **andothers**. “Explainable AI for Designers: A Human-Centered Perspective on Mixed-Initiative Co-Creation”. **in***2018 IEEE Conference on Computational Intelligence and Games (CIG)*: 2018, **pages** 1–8. DOI: 10.1109/CIG.2018.8490433.

[108] Xiaojin Zhu **and** Zoubin Ghahramani. “Learning from Labeled and Unlabeled Data with Label Propagation”. **in**(**july** 2003).