KU LEUVEN

ARENBERG DOCTORAL SCHOOL
Faculty of Engineering Science

Harnessing Knowledge Graphs for Explainable AI

Youmna Ismaeil

Supervisor:
Prof. dr. ir. Hendrik Blockeel

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

July, 2025

Harnessing Knowledge Graphs for Explainable AI

Youmna ISMAEIL

Examination committee:
Prof. Emeritus Dirk Van Compernolle, chair
Prof. dr. ir. Hendrik Blockeel, supervisor
Dr. Daria Stepanova[a], co-supervisor
Msc. Trung-Kien Tran[b], co-supervisor
Prof. Emerita Marie-Francine Moens
Prof. Sebastijan Dumančić[b]
Prof. dr. Anastasia Dimou
Prof. dr. Alessandra Mileo[c]

Dissertation presented in partial fulfillment of the requirements for the degree of Doctor of Engineering Science (PhD): Computer Science

[a]Bosch Center for Artificial Intelligence
[b]TU Delft
[c]Dublin City University

July, 2025

© 2024 KU Leuven – Faculty of Engineering Science
Uitgegeven in eigen beheer, Youmna Ismaeil, Celestijnenlaan 200A box 2402, B-3001 Leuven (Belgium)

Alle rechten voorbehouden. Niets uit deze uitgave mag worden vermenigvuldigd en/of openbaar gemaakt worden door middel van druk, fotokopie, microfilm, elektronisch of op welke andere wijze ook zonder voorafgaande schriftelijke toestemming van de uitgever.

All rights reserved. No part of the publication may be reproduced in any form by print, photoprint, microfilm, electronic or any other means without written permission from the publisher.

Acknowledgments

Reflecting on the years spent pursuing my PhD, I am filled with gratitude for the remarkable individuals who have shaped and enriched this chapter of my life, both professionally and personally. First and foremost, I extend my sincere gratitude to my supervisors, whose guidance and support have been invaluable throughout my doctoral journey. I have been immensely privileged to undertake my PhD as part of a collaboration between Bosch and KU Leuven, benefiting from exceptional mentorship at both institutions.

I wish to first extend my heartfelt appreciation to my supervisor at KU Leuven, Prof. Hendrik Blockeel. His insightful guidance and constructive feedback have profoundly influenced my development as a researcher. Hendrik's emphasis on clarity in presentation and precision in writing continually encouraged me to elevate my work, refining my ability to convey complex ideas effectively. His mentorship provided the right balance between autonomy and structured support, allowing me to explore my research interests while ensuring I remained firmly on track.

At Bosch, I was fortunate to have two co-supervisors, each contributing uniquely to my research experience. I would like to deeply thank my co-supervisor, Dr. Daria Stepanova, for her daily support, unparalleled dedication, and unwavering support. Her constant presence, even during her maternity leave, exemplified extraordinary commitment. Whether it was submissions on Christmas Eve, navigating challenging phases, or providing guidance during conferences, Daria continually inspired me through her professionalism and genuine care. Her thoughtful introductions to professional networks significantly enriched my experiences and skills. I also appreciate my co-supervisor, Dr. Trung-Kien Tran, for consistently challenging my perspectives and ideas, prompting deeper insights and rigorous thinking.

I extend sincere appreciation to my PhD committee members, Prof. Marie-Francine Moens and Prof. Sebastijan Dumančić, for engaging deeply with my

research and offering valuable feedback during our annual reviews. Additionally, I am grateful to my examiners, Prof. Anastasia Dimou and Prof. Alessandra Mileo, and to the chair, Prof. Emeritus Dirk Van Compernolle, for generously agreeing to provide their time, insights, and expertise to this final stage of my journey.

I warmly acknowledge Jannik, my previous manager at Bosch, whose support during the initial years of my PhD allowed me the opportunity to participate in enriching summer schools and international conferences. Gad's invaluable tips and Evgeny's thoughtful introductions during conferences significantly expanded my professional network.

One of the greatest joys of this PhD has been the colleagues and friends I gained along the way. Sophie, transitioning from a fellow PhD researcher to a dear friend, made the demanding journey significantly lighter. My heartfelt appreciation goes to my Bosch colleagues, including Hendrik, Stephan, Subhash, and my fellow PhD researchers, for their camaraderie, motivation, and shared experiences. I also warmly thank Simiao, Adam, Aras, and Jonnas at KU Leuven for enriching my six-month stay there through their welcoming spirit and engaging company.

This journey would have been unimaginable without the profound emotional support from my family. My parents' unwavering encouragement, love, and guidance shaped the foundation of who I am today. Dad, your constant support—emotionally, financially, and morally—alongside your lessons in resilience and determination, has empowered me since the early days of my bachelor's degree abroad. Mom, as a professor yourself and my closest confidante, your compassionate understanding and insightful counsel were invaluable, especially during periods of stress and uncertainty. To my brother, Amr, whose presence transcended distance, thank you for your constant reassurance that I was never alone.

My deepest gratitude and love go to my husband, Karim, whose presence has profoundly shaped both my academic and personal life. Meeting you during my six-month stay at KU Leuven was serendipitous; your companionship, guidance regarding every challenge of the PhD, and patient listening to my countless frustrations have been immeasurable sources of strength.

Special appreciation is extended to my dear friends whose steadfast companionship sustained me through these years. Myriame, your friendship over the past decade has provided stability, joy, and profound comfort through countless ups and downs. Yara and Nourhan, your loyalty and consistent emotional support have deeply enriched my life over the past five years. Karim Medhat, your support during my move to Stuttgart amidst the pandemic

was indispensable—your friendship over the past fifteen years has been a true blessing. To Yara, Nourhan, Karim Medhat, Taha, Bassam, and Ismail—our regular gatherings have provided warmth, laughter, and a sense of family, always brightening even the hardest of weeks.

To all the others—administrative staff, colleagues whose names I may have missed here, and those whose interactions, however brief, have positively impacted my path—I extend my heartfelt appreciation.

Finally, and most importantly, my profound thanks to God for providing me with strength and resilience, guiding me through moments of doubt and darkness, and shaping me into the person I have become today.

Thank you all for being part of this unforgettable chapter of my life.

Youmna Ismaeil
August 19, 2026
Stuttgart, Germany

Abstract

Artificial intelligence (AI) has become an integral part of our daily lives, influencing decisions in diverse domains such as healthcare, finance, and autonomous systems. Despite its widespread adoption, a major limitation of modern machine learning (ML) models—especially deep neural networks—is their lack of transparency. These models often function as black boxes, making it difficult to understand the reasoning behind their decisions. This opacity raises concerns about trust, bias, and accountability, particularly in high-stakes applications. Furthermore, neural networks are known to learn spurious correlations from training data, leading to unreliable and biased predictions. Addressing these challenges requires explainability methods that offer meaningful insights into the model's decision-making process without relying on labor-intensive manual annotations or subjective interpretations.

To overcome these limitations, this thesis investigates the use of knowledge graphs as structured, information-rich resources that can mitigate the need for manual annotations and facilitate structured explanations. Specifically, it introduces novel methods integrating knowledge graphs to enable unsupervised explainability in machine learning models, focusing particularly on image classifiers and knowledge graph embedding models.

First, we introduced a framework to explain neuron activations in convolutional neural networks by mapping them to semantic attributes extracted from commonsense knowledge graphs. This method enables a more intuitive understanding of what features influence a model's predictions. Next, we tackled the interpretability of knowledge graph embedding models by introducing a feature selection-based framework, FeaBI, to interpret knowledge graph embeddings. By identifying key features that influence learned embeddings, FeaBI provides insights into how knowledge graph representations are formed, improving interpretability for downstream tasks. To make these explanations more accessible, an interactive tool has been developed, allowing users to compute entity similarities and visualize their explanations dynamically.

Finally, we combined the structure of knowledge graphs with the power of foundation models to introduce a novel framework for discovering learned patterns in ML models. This framework extracts key patterns that influence classifier decisions. The extracted patterns are used for error analysis and bias detection, providing deeper insights into dataset composition and model behavior. In the process of refining explainability methods, a new approach is also developed for generating comprehensive image descriptions, capturing contextual elements such as background, object relationships, and environmental factors. This enhances interpretability beyond simple object detection, revealing subtle patterns that might otherwise go unnoticed.

By integrating structured knowledge representations with advanced foundation models, this thesis offers an automated approach to explainability, addressing critical gaps in AI transparency. The proposed methods enhance model interpretability, facilitate bias detection, and improve debugging, contributing to the development of more trustworthy and accountable AI systems.

Beknopte samenvatting

Kunstmatige intelligentie (AI) is uitgegroeid tot een integraal onderdeel van ons dagelijks leven en beïnvloedt beslissingen in uiteenlopende domeinen, zoals gezondheidszorg, financiën en autonome systemen. Ondanks de brede toepassing is een belangrijke beperking van moderne machine learning (ML)-modellen—vooral in geval van diepe neurale netwerken—het gebrek aan transparantie. Deze modellen functioneren vaak als zwarte doos, waardoor het lastig is om de redenering achter hun beslissingen te begrijpen. Deze ondoorzichtigheid roept vragen op over betrouwbaarheid, bias en verantwoordelijkheid, met name in toepassingen waarbij veel op het spel staat. Daarnaast staan neurale netwerken erom bekend dat zij schijncorrelaties uit trainingsdata aanleren, wat kan leiden tot onbetrouwbare en bevooroordeelde voorspellingen. Om deze uitdagingen aan te pakken zijn verklaarbaarheidsmethoden nodig die inzicht geven in het besluitvormingsproces van modellen, zonder te steunen op arbeidsintensieve handmatige annotaties of subjectieve interpretaties.

Om deze beperkingen te overwinnen onderzoekt dit proefschrift hoe kennisgrafen, als gestructureerde en informatierijke bronnen, kunnen bijdragen aan het verminderen van de noodzaak voor handmatige annotaties en het faciliteren van gestructureerde verklaringen. Specifiek worden nieuwe methoden geïntroduceerd die kennisgrafen integreren om onbegeleide verklaarbaarheid in machine learning-modellen mogelijk te maken, met een bijzondere nadruk op beeldclassificatiemodellen en kennisgraaf-inbeddingsmodellen.

Allereerst introduceren we een raamwerk dat neuronactivaties in convolutionele neurale netwerken verklaart door ze te koppelen aan semantische attributen uit gezond verstand-kennisgrafen. Deze aanpak biedt een intuïtiever begrip van de kenmerken die voorspellingen van een model beïnvloeden. Vervolgens richten we ons op de interpretatie van kennisgraaf-inbeddingsmodellen door FeaBI te introduceren, een op kenmerkselectie gebaseerd raamwerk om deze inbeddingen te interpreteren. Door sleutelkenmerken te identificeren die de geleerde inbeddingen beïnvloeden, biedt FeaBI inzicht in hoe kennisgraaf-

representaties worden gevormd, wat de interpretatie voor vervolgtoepassingen verbetert. Om deze verklaringen toegankelijker te maken is bovendien een interactieve tool ontwikkeld, waarmee gebruikers entiteitssimilariteiten kunnen berekenen en verklaringen dynamisch kunnen visualiseren.

Tot slot combineren we de structuur van kennisgrafen met de kracht van foundation-modellen om een innovatief raamwerk te ontwikkelen voor het ontdekken van patronen die door ML-modellen zijn aangeleerd. Dit raamwerk extraheert sleutelpatronen die beslissingen van classificatiemodellen beïnvloeden. De geëxtraheerde patronen worden ingezet voor foutanalyse en het detecteren van bias, waardoor een diepgaander inzicht ontstaat in datasets en modelgedrag. Tijdens het verfijnen van verklaarbaarheidsmethoden is ook een nieuwe aanpak ontwikkeld voor het genereren van uitgebreide afbeeldingsbeschrijvingen die contextuele elementen zoals achtergrond, relaties tussen objecten en omgevingsfactoren bevatten. Dit verhoogt de interpreteerbaarheid ten opzichte van eenvoudige objectdetectie en onthult subtiele patronen die anders wellicht onopgemerkt zouden blijven.

Door gestructureerde kennisrepresentaties te integreren met geavanceerde foundation-modellen biedt dit proefschrift een geautomatiseerde aanpak voor verklaarbaarheid en adresseert daarmee kritieke tekortkomingen in AI-transparantie. De voorgestelde methoden verbeteren de interpreteerbaarheid van modellen, vergemakkelijken het detecteren van bias en ondersteunen foutopsporing, en dragen zo bij aan de ontwikkeling van betrouwbaardere en meer verantwoordelijke AI-systemen.

List of Abbreviations

AI Artificial Intelligence

KG Knowledge Graph

EKG Enterprise Knowledge Graph

OKG Open Knowledge Graph

ML Machine Learning

NN Neural Network

GNN Graph Neural Network

CNN Convolutional Neural Network

FM Foundation Model

SG Scene Graph

VG Visual Genome

FeaBI Feature-Based Interpretation

CompGCN Composition Graph Convolutional Networks

INK Interpretable Knowledge Graph Embeddings

TransE Translation Embedding

ZSL Zero-Shot Learning

LIME Local Interpretable Model-agnostic Explanations

SHAP SHapley Additive exPlanations

SNoRE Symbolic Node Representations

LRP Layer-wise Relevance Propagation

Grad-CAM Gradient-weighted Class Activation Mapping

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Artificial Intelligence (AI): Trust and Transparency challenges

AI is no longer the future–it's now here in our living rooms and cars and, often, our pockets

Susan Arial Aaronson [1]

The dramatic success of machine learning has led to an explosion of new AI capabilities, advancing towards autonomous systems that perceive, learn, decide, and act independently becoming an integral parts of our daily lives. These AI systems span various domains, including our homes, vehicles, and personal devices, and promise tremendous benefits. However, they also face significant limitations due to their inability to explain their decisions and actions to human users, which impacts trust and usability.

A major consequence of lack of explanations with neural networks is their susceptibility to learning spurious correlations in the training data. These models, driven by performance optimization, often identify non-essential features—such as background elements or textures—as influential to a given label, even when these features are irrelevant to the actual task [48]. For instance, a model might associate the presence of a cat with grass in the background, although the grass is not an intrinsic feature of a cat. In sensitive applications, such shortcuts can have severe consequences. For example, in

AI-based recruitment, a model might inadvertently learn biases linked to gender or ethnicity, leading to discriminatory outcomes. Similarly, in medical contexts, an AI cancer detection model could rely on hospital-specific watermarks in scans, yielding misleading and potentially harmful diagnoses.

These examples illustrate the critical risks posed by the opaque nature of neural networks and their reliance on spurious correlations, which highlights the need for transparent and explainable AI models. Providing explanations is fundamental to building trust in AI systems, particularly in domains where human rights or lives are impacted directly. Insight into a model's decision-making helps users validate the reliability of predictions, prevents unintended biases, and enhances the safety of deploying AI in real-world scenarios. In regulated sectors, such as healthcare, finance, and law, explanations are not merely desirable—they are increasingly mandated to ensure accountability and fairness.

Furthermore, explainability is essential for practitioners to identify and correct flaws in the models, ultimately making AI systems more robust and reliable. Interpretability empowers end-users to make informed decisions based on AI recommendations, bridging the gap between complex machine learning models and human understanding. As AI continues to expand its influence, ensuring explainability is crucial for managing these systems effectively and responsibly, particularly as they become trusted partners in decision-making processes across diverse applications.

1.2 Explainability for AI

If science was a journey, then its destination would be the discovery of simple explanations to complex phenomena.

Robert Geirhos, et al.[48]

Numerous definitions of explainability have emerged over time (see [18] for an overview). The one most aligned with the perspective adopted in this work is proposed by Ciatto et al. [28], who define an explanation as "the additional meta information, generated by an external algorithm or by the machine learning model itself, to describe the feature importance or relevance of an input instance towards a particular output classification.". The idea of AI explainability first appeared in the literature over 47 years ago [129, 152], with human experts justifying AI system decisions. Since the inception of AI,

researchers have stressed the need for systems that can explain their decisions in human-understandable terms.

In the early days, AI predominantly relied on logical and symbolic reasoning methods, where decisions were made through logical inference on human-readable symbols. These systems could generate a trace of their inference steps, which served as a basis for explanations [71, 153, 136]. However, the rigidity of these rule-based systems made them unsuitable for complex real-world scenarios. This encouraged the growth of more sophisticated AI models with less emphasis on explainability. The increasing complexity of models, particularly deep neural networks (DNNs), made it difficult to maintain transparency. Explainability was often seen as a hindrance that negatively impacted model performance [51], leading to the widespread acceptance of black-box models.

Since 2014, as models began surpassing human capabilities in certain tasks, there has been renewed interest in understanding these opaque models [128, 82]. However, despite this resurgence, current approaches to explainability exhibit the following limitations that affect their practical applicability.

Restrictive Applicability and Reliability

Early interpretability methods like deconvolutional networks [180] offered insights into CNN feature representations by visualizing internal activations, but were limited to convolutional architectures and required significant manual effort. Later approaches, such as guided backpropagation [146] and saliency maps [138], sought to identify important features yet remain sensitive to noise, lack robustness, and heavily depend on model architecture [2, 142], often yielding inconsistent and artifact-prone results [124, 132].

To address these limitations, more robust and faithful interpretability techniques have been proposed. Integrated Gradients [151] and SmoothGrad [142] were introduced to enhance the reliability of saliency maps by reducing noise and improving attribution quality. Score-CAM [165] and FullGrad [147] aimed to address architectural dependencies and improve the faithfulness of the extracted saliency maps.

Relevance-CAM [21] extended interpretability to transformer-based models. Further innovations include Adaptive-CAM [38], which combines Grad-CAM and Layer-wise Relevance Propagation (LRP) for improved visual explanations, and Feature-CAM [29], offering fine-grained, class-discriminative visualizations. AS-XAI [150] introduces a self-supervised framework for automatic semantic interpretation in CNNs.

All these methods provide visual representations that highlight regions of the input data deemed most influential in the model's decision-making process. However, they typically do not assign explicit labels or meanings to these important regions. An exception is AS-XAI, which employs concept whitening introduced in [25] —a technique that maps internal representations to a predefined limited set of concepts, thereby providing semantic interpretations of the highlighted regions [150].

Subjective Interpretations

Providing visual explanations in the form of highlighting relevant regions in the input is a widely adopted approach for explaining machine learning model decisions. These techniques, often referred to as attribution methods, aim to identify which parts of the input most influence the model's output. Notable examples include methods like [43, 61, 138, 2, 142, 124, 132, 151, 165, 147, 21, 38, 29, 25, 12].

While these methods provide valuable insights, they have certain limitations. Typically, they indicate areas of importance without offering semantic labels for the highlighted regions, leaving interpretation up to the user. This can be challenging, especially when trying to understand the model's reasoning across multiple instances. Users may need to examine several examples to discern consistent patterns or features that the model relies upon for its predictions.

Labor-Intensive Annotations

Data-driven methods have been proposed to avoid subjective interpretations, utilizing pixel-wise manual annotations to reveal patterns and relationships within the data [8, 103, 43, 61]. When annotations are linked to the important regions identified by an explainability method, they can provide a human-understandable explanation of which parts of an image contribute to a model's prediction. However, obtaining these annotations is labor-intensive and only practical for datasets that already have pixel-level labels readily available. For example, NetDissect [8] maps neurons in a network's convolutional layers to these attributes, helping identify which neurons respond to specific objects or regions. While effective, the reliance on annotated data limits the generalization of these methods.

Interpreting Embeddings: An Overlooked Challenge

Embedding models face significant challenges with interpretability, as they learn dense numerical representations of data for tasks such as knowledge representation and natural language processing. While these embeddings are effective in downstream tasks like recommendation systems and language modeling [24, 33, 9], they remain inherently opaque, offering little insight into which specific aspects of the input data are represented within these dense vectors. Despite the importance of this issue, interpretability for embedding models is frequently overlooked.

This gap has spurred research aimed at developing frameworks to explain embedding behavior within specific tasks [120, 20, 10, 104, 52, 45]. However, these approaches tend to focus on explaining the results of embedding-based models used in downstream tasks, rather than on providing insight into the embeddings themselves. Other methods have introduced representations based on interpretable features [46, 100, 148], yet these too encode features as numerical vectors, making it challenging to determine precisely what information the model retains or discards during encoding.

This lack of transparency in embedding models leaves a critical gap in understanding the information embedded within learned representations, as well as in identifying how to utilize these embeddings effectively. Addressing this gap is essential to advancing interpretability for models that rely on embeddings, ensuring that users can better trust and validate the representations learned by these models.

Lack of Standardized Evaluation Frameworks

Despite significant progress in developing explainability methods for various AI models, evaluating these explanations continues to pose a major challenge. The process is similar to deciphering an Enigma code without a comprehensive key—attempting to make sense of only partial clues. Due to the inherently black-box nature of neural networks, there is no definitive ground truth for explanations, making it difficult to verify the accuracy or relevance of extracted interpretations. Explanations are often provided in diverse formats, hindering the development of a standardized framework or consistent metrics for assessing their quality [39]. This absence of evaluation standards has led many studies to rely on selective examples that align with human intuition [103, 99], and in some cases, to omit quantitative evaluation entirely [8]. Such reliance on qualitative illustrations complicates statistical validation, underscoring the need

for robust quantitative evaluation methods and metrics to ensure reliability and generalizability in explainable AI.

An early effort to address these evaluation challenges was proposed by Doshi-Velez and Kim [39], who introduced three abstract evaluation strategies:

1. **Application-grounded evaluation**: Domain experts assess the model's explanations in real-world scenarios, comparing their decisions against a baseline.
2. **Human-grounded evaluation**: Non-experts perform simplified tasks, such as selecting the best classifier based on explanations provided [117].
3. **Functionally-grounded evaluation**: Explanation quality is evaluated through proxy measures without human input, such as testing model accuracy when key features are removed [137].

However, despite the introduction of the aforementioned evaluation strategies, their adoption in practice has been limited, leaving a substantial gap in the standardized evaluation of explainability.

1.3 Problem Statement and Contributions

The gaps introduced earlier can be filled by addressing the following research questions:

- How can we design explainability frameworks that are not subjective and applicable across diverse AI models?
- How can we develop frameworks to explain AI models without relying on extensive additional labeled data beyond the model's training data?
- How can we evaluate the extracted explanations both qualitatively and quantitatively without golden labels?
- How can we extend explainability frameworks to effectively interpret embedding models, addressing the currently overlooked challenges in understanding their representations?

This thesis makes five key contributions that address these research questions.

A framework for explaining neuron activity in a neural network layer using commonsense knowledge graphs.

In this work, we introduce a framework that minimizes the reliance on extensive semantic labeling for explaining image classifiers as an example of an AI model. Our approach maps hidden neurons from convolutional neural network's layer to semantic attributes of classes extracted from commonsense knowledge graphs. We evaluated the explanations on copy-paste adversarial image classification and provided a use-case for the explanations as an intermediate medium to merge knowledge learnt by different classifiers that can be used for zero-shot learning tasks.

A Feature Selection-Based Framework for Interpreting KG Embeddings

Knowledge Graph (KG) embedding methods represent KG nodes as vectors in an embedding space, and they have been successfully used for a variety of tasks, including link prediction and entity classification. While some of the recent embedding methods outperform traditional approaches on these tasks, their main drawback is the lack of interpretability. Prior to our work, none of the existing explainability methods targeted the problem of constructing model explanations for embeddings, i.e., interpretable KG representations that behave similarly to embeddings on certain tasks. We address this problem and propose a novel method for generating interpretable vectors for entity embeddings. To achieve this, we propose FeaBI [64] that uses embedded feature selection techniques to extract from the KG, on which the embedding model was trained, propositional features that are important for a given KG embedding model. Our approach sheds light on the information in the KG captured by embeddings and provides valuable insights that can be used to further enhance the embedding models. Additionally, we proposed an evaluation of our explanations on two downstream tasks. We also demonstrate the usefulness of our method for explaining embedding-based entity similarity.
We also developed an interactive demo for FeaBI, enabling users to easily utilize the framework for computing embedding-based similarities between KG entities, as well as for generating and visualizing explanations for these similarities.

A Framework for Discovering Patterns Learned by Image Classifiers using Foundation Models and Knowledge Graphs

We introduced a dynamic, unsupervised framework for extracting learned patterns by image classifiers from its training data. Our framework leverages commonsense knowledge and open-source foundation models to describe training images. The extracted patterns show how combinations of semantic dimensions influence classifier decisions, providing clear deep insights. Evaluations confirm that our image descriptions are of high quality, incorporating details from manual annotations along with additional information not covered in the annotations. Moreover, our framework demonstrates high predictive accuracy, effectively identifying patterns responsible for both correct and incorrect classifications, with F1 scores ranging from 0.72 to 0.96. Our framework is also utilized for error analysis and proactive bias detection in datasets.

A Method for Generating Comprehensive Image Descriptions

In our exploration of neural network explanations, we found that existing methods for image understanding primarily focus on detecting objects and their relationships, often overlooking the background, overall scene setup, and image attributes like contrast and brightness. These elements, however, are essential for a comprehensive understanding of the image and play a critical role in identifying potential biases in AI models. To overcome this, we introduced an innovative approach that leverages foundation models and knowledge graphs to produce more comprehensive and detailed image descriptions in the form of scene graphs.

1.4 Outline of the Thesis

The remainder of this thesis is organized as follows. Chapter 3 provides an overview of the existing literature and highlights the key gaps that motivate the contributions of this thesis. In Chapter 2, we explore the relationship between knowledge graphs and explainability, illustrating how knowledge graphs can serve as powerful tools for enhancing the interpretability of AI models. Chapter 4 presents a framework for explaining neuron activity in a neural network layer using commonsense knowledge graphs and its evaluation using copy-paste adversarial image classification as well as the use of explanations for zero-shot image classification. This chapter covers the work from the following paper:

Youmna Ismaeil *, Daria Stepanova, Trung-Kien Tran, Piyapat Saranrittichai, Csaba Domokos, and Hendrik Blockeel*(2022).

Towards Neural Network Interpretability Using Commonsense Knowledge Graphs. In: The 21^{st} *International Semantic Web Conference – ISWC 2022. Lecture Notes in Computer Science*, vol 13489, pp. 74-90. Springer, Cham.

Chapter 5 introduces a framework for interpreting KG embeddings using features-selection techniques, with evaluations conducted on entity and relation classification tasks. The explanations are further utilized for similarity explanation. This chapter covers the work from the following paper:

Youmna Ismaeil *, Daria Stepanova, Trung-Kien Tran, and Hendrik Blockeel* (2023). FeaBI: A Feature Selection-Based Framework for Interpreting KG Embeddings. In: *The* 22^{nd} *International Semantic Web Conference – ISWC 2023. Lecture Notes in Computer Science*, vol 14265. pp. 599-617. Springer, Cham.

Chapter 6 introduces a demo for FeaBI that allows users to conveniently exploit the respective framework for computing embedding-based similarity between KG entities as well as generating and visualizing explanations for the respective similarity. This chapter covers the work from the following paper:

Huu Tan Mai, ***Youmna Ismaeil****, Trung-Kien Tran, Hendrik Blockeel, and Daria Stepanova.* Look beyond the Surface: A Demo for Explaining Knowledge Graph Embeddings and Entity Similarity. In *ISWC (Posters/Demos/Industry). 2023.*

Chapter 7 introduces a framework for discovering patterns learned by image classifiers using foundation models and knowledge graphs. These patterns show how combinations of image features influence classifier decisions, providing clear, deep insights. We propose suitable evaluation method for the extracted patterns and used them for error analysis and proactive bias detection in datasets. This chapter covers the work from the following paper:

Youmna Ismaeil*, Jan Hednrik Metzen, Trung-Kien Tran, Daria Stepanova and Hendrik Blockeel.* Beyond Manual Labels: Unsupervised Graph-Based Explanations for Error Analysis in Image Classifiers. Under review *The* 24^{th} *International Semantic Web Conference (ISWC). 2025.*

Chapter 8 serves as the final chapter, summarizing the key contributions of this thesis and reflecting on the lessons learned during the development of

explainability frameworks for neural networks. It highlights the major challenges faced, examines the limitations of existing approaches, and offers actionable recommendations for future research to further advance the field of model interpretability.

Chapter 2

The link between explainability and knowledge graphs

Every machine learning journey begins with data, whether it's images, text, graphs, or signals [87]. This data is the raw material for training models to capture essential features that drive their predictions. Yet while training is data-driven, understanding and explaining a model's decisions often requires associating this data with the model's behavior. Traditional explainability methods focus on analyzing how a model responds to specific input features, which vary depending on the data type. For text, these may be words; for graphs, nodes and edges; and for images, pixels, segments, or recognized objects. The more intuitive these features are to humans, the more meaningful and accessible the explanations become. For instance, associating an object like a "chair" with an "office" setting is far clearer to users than associating a class, e.g., "office", with individual pixels [124]. However, achieving this level of interpretability is not always straightforward. When dealing with complex data, such as images, it requires more than isolated feature detection; it calls for scene understanding. An initial step in scene understanding involves extracting and labeling objects within a scene, a process known as semantic segmentation and labeling, as shown in Fig. 2.1. This task, however, is traditionally performed manually, making it labor-intensive and time-consuming. For datasets without domain-specific requirements, a human annotator with general knowledge can classify images of common environments—such as rooms (e.g., "classroom", "office", "kitchen") or objects (e.g., "cars", "buses"). In other words, these human annotators

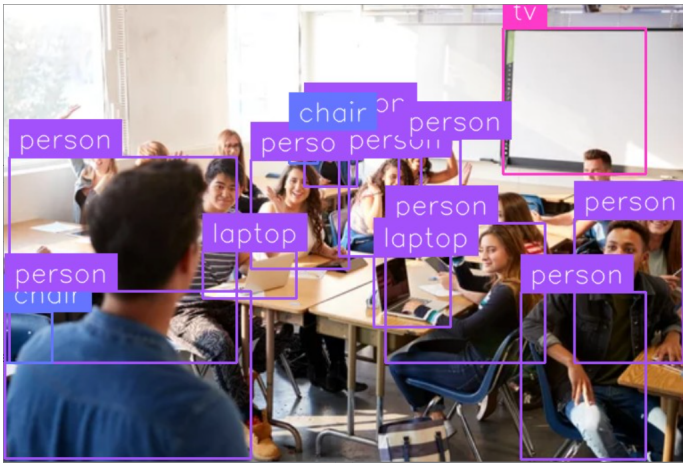


Figure 2.1: A classroom image with semantic segmentation, featuring bounding boxes and labels that highlight various objects such as people, laptops, chairs, and a TV.

incorporate their commonsense knowledge to tell that the rectangular screen in the image is that of a "computer" and that the "computer" is an indication that the image is more likely to be of an "office" rather than a "bedroom". A known source of a subset of human commonsense knowledge is found in what is known as commonsense knowledge graphs. In this chapter, I discuss what the different types of knowledge graphs are and how they can be leveraged to create explanations that resonate with human understanding.

2.1 Knowledge Graphs

Knowledge graphs (KGs) are powerful tools for organizing and linking diverse data in a structured and meaningful way. A knowledge graph represents data as a network of nodes and labeled edges, where nodes correspond to entities—such as people, places, events, or concepts—and edges are labeled with the relationships between these entities. More formally, knowledge graphs (KGs) encode data as a set of $\langle subject\ predicate\ object \rangle$ triples, e.g., $\langle tree\ has\ branches \rangle$. This graph-based structure not only facilitates the storage and retrieval of data [116] but also enables machines to infer new knowledge in the form of links between entities by analyzing the connections within the graph [80].

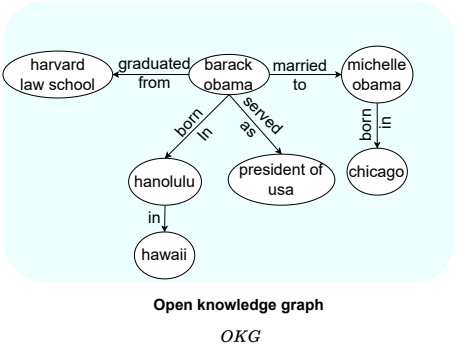


Figure 2.2: A snippet of an open knowledge graph from Freebase KG.

Representing data as knowledge graphs (KGs) offers a structured and context-rich format that enhances interpretability. This structure enables more meaningful reasoning over data, making KGs particularly valuable in tasks such as semantic search, information retrieval, and the development of advanced AI and ML systems, as demonstrated in recent works [62, 95].

2.1.1 Types of Knowledge Graphs

Knowledge graphs (KGs) can be broadly classified into several types [60], each tailored to specific use cases:

Open Knowledge Graphs are publicly accessible and aggregate information from open data sources, such as Wikipedia. Examples include DBpedia [5] and ConceptNet [145], which serve as general-purpose knowledge bases for a wide range of applications [161, 188, 68]. These graphs are designed to be used by the broader public, yet they can serve different purposes. For example, DBpedia has structured information extracted from Wikipedia. Its primary aim is to create a large-scale, open-source knowledge graph that supports semantic web applications. It focuses on general knowledge, encompassing a wide variety of topics ranging from geography and people to history and science. It serves as a foundational component for many applications that require general-purpose knowledge, same for Freebase [13]. A visual example for that is in Fig. 2.2. Similarly, ConceptNet is an open-source knowledge graph that captures common-sense knowledge and everyday semantic relationships—such as "ice is cold" or "a cat is a type of animal"—which humans typically understand implicitly. Knowledge graphs that have common-sense knowledge are often referred to

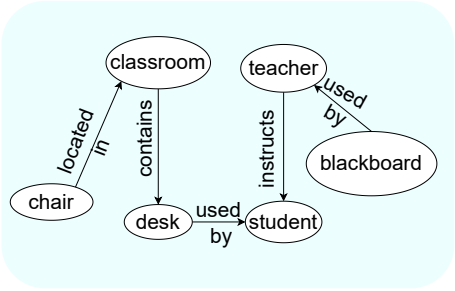


Figure 2.3: A snippet of commonsense knowledge graph from conceptNet.

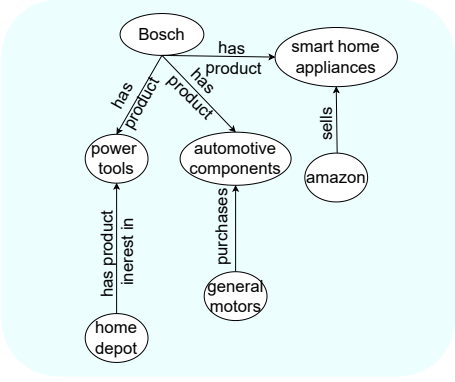


Figure 2.4: A hypothetical example for an enterprise knowledge graph.

as common-sense knowledge graphs, as shown in Fig.2.3. In chapters 4 and 7, we show how one can utilize this commonsense knowledge to enhance the interpretability of a given machine learning model.

Enterprise Knowledge Graphs (EKGs) are proprietary knowledge graphs developed within organizations to integrate internal data sources. EKGs break down data silos, enhance data governance, and enable more informed decision-

making by providing a unified view of enterprise data. For instance, a company might use an EKG to create a 360-degree view of its customers, consolidating data from different departments such as marketing, sales, and customer support [171]. In this thesis, we do not specifically address this type of graph, but the proposed methods are versatile and can be applied to any graph structure, including enterprise knowledge graphs (EKGs). An example of that is in Fig. 2.4.

2.1.2 Open-Source Knowledge Graphs for Interpretability

The key applications of knowledge graphs are in data integration, analytics, and search. They are commonly known to unify data from diverse sources, enabling tasks like question answering, recommendation systems, and search optimization, as demonstrated by platforms like Wikidata, Google, and LinkedIn [164, 141]. Additionally, they address incompleteness through knowledge graph completion tasks, such as link prediction, which estimates connections between entities to enhance their utility [120, 10].

While knowledge graphs have proven invaluable in numerous domains, their application in explaining machine learning models, such as image classifiers, remains underexplored. Existing work, such as [35], has utilized semantic knowledge in the form of ontologies to interpret convolutional neural networks (CNNs), but these approaches rely heavily on segmented and labeled data. This dependency not only limits scalability but also overlooks the broader potential of knowledge graphs for enabling more accessible and generalizable explainability.

This thesis builds upon this hidden link between knowledge graphs and explainability by introducing methods that leverage knowledge graphs for interpreting machine learning models without the need for pre-segmented or labeled data. In chapters 4, 5 and 7, we explore two distinct types of open-source knowledge graphs: commonsense knowledge graphs, to provide explanations for image classifiers, and general-purpose knowledge graphs, for the interpretation of embedding models. Notably, chapter 5 presents a novel method for interpreting embedding models by finding the building blocks in the knowledge graphs that contribute to the learning of the embeddings, uncovering patterns and relationships that enhance interpretability. chapters 4 and 7 introduces new ways to utilize commonsense knowledge graphs as an alternative to semantic labels. Through these contributions, the thesis shows how knowledge graphs can serve as a bridge between raw data and human understanding, offering a scalable and versatile approach to explainability. By uncovering this link, we not only advance the field of interpretable machine learning but also open new pathways for applying knowledge graphs to enhance transparency and trust in AI systems. This chapter has laid the groundwork for these explorations, paving

the way for the detailed methods and applications presented in the following chapters.

Chapter 3

Foundations and Gaps in Explainable AI

This chapter provides a high-level overview of the existing literature in the areas most relevant to this thesis: explainability of image classifiers and interpretation of knowledge graph (KG) embedding models. It outlines the gaps that motivate the contributions of this work.

3.1 Explaining Image Classifiers

Neural networks (NNs) have achieved impressive performance across a range of tasks, including image classification. However, their decision-making processes remain largely opaque, posing a serious concern in high-stakes applications where a lack of transparency can lead to critical errors. As a response, early explainability research focused on visual attribution methods [138, 146, 132]which highlight regions in the input deemed important to the model's prediction. Despite their intuitive appeal, these methods are often noisy and sensitive to model architecture [2].

To address some of these limitations, subsequent approaches have improved the robustness and fidelity of attribution techniques. For instance, Integrated Gradients [151], SmoothGrad [142], and Score-CAM [165] aim to generate clearer and more stable explanations. Architecture-specific frameworks, such as those designed for transformer models [21], have also expanded the reach of attribution methods. Nevertheless, these techniques primarily produce spatial

explanations, highlighting influential input regions without revealing what these regions semantically represent. Consequently, the burden of interpretation is shifted to users, making the process subjective, labor-intensive, and limited to individual instances.

To introduce semantic meaning into models' visual explanations, concept alignment methods such as Concept Whitening [25], Network Dissection [8], and Compositional Explanations [103] have attempted to align neuron activations with human-understandable concepts (e.g., "window", "fur"). These methods demonstrate that individual neurons may correspond to interpretable visual concepts; however, they rely heavily on pixel-level annotations or predefined concept vocabularies. This dependence on manual effort hinders their usability and limits their ability to generalize across datasets with different concept distributions.

Beyond interpreting individual neurons, increasing attention has been directed toward understanding model predictions on downstream tasks, such as classification, to diagnose systematic failures arising from data bias or limited generalization. Notable efforts [19, 74, 99] have examined model behavior across fixed semantic dimensions such as race, gender, or lighting conditions. While these analyses yield valuable insights, they rely on manually defined attributes and often fail to capture the complex interdependencies between semantic dimensions. This limits their generalizability in real-world scenarios, where relevant factors may be unlabeled or not explicitly known.

To address these limitations, Chapters 4 and 7 introduce unsupervised frameworks that generate class-level explanations for image classifiers by leveraging external knowledge graphs (KGs). These methods utilize structured semantic information from knowledge graphs (KGs) to extract interpretable and generalized patterns from the training data, eliminating the need for manual annotations or predefined concept sets. By incorporating commonsense and contextual knowledge, these KG-driven approaches enable robust explanations across diverse datasets, thereby overcoming the requirement of labor-intensive labeling or fixed semantic dimensions of prior work in both neuron-level and input-level interpretability.

3.2 Interpreting Knowledge Graph Embedding Models

While explaining classifier decisions has received much attention, the interpretability of learned representations, particularly knowledge graph (KG)

embeddings, remains underexplored. Knowledge graphs (KGs) encode real-world entities and their relationships and have become a core resource in a wide range of applications, from search and recommendation systems to question answering. To enable scalable computation over large KGs, various KG embedding models have been proposed, such as TransE [14], DistMult [173], and ComplEx [159], which learn low-dimensional vector representations of entities and relations. These representations capture both structural and semantic information implicitly and are widely used in downstream tasks such as link prediction and entity classification.

However, these embeddings are inherently opaque: while they encode useful relational patterns, their dimensions do not correspond to human-interpretable concepts. Unlike symbolic representations, embeddings lack transparency and offer no direct clues about what information is preserved or discarded during training. This makes it difficult to diagnose model behavior or build trust in embedding-based systems, especially when errors occur.

Graph Neural Networks (GNNs) are a class of neural network models specifically designed to operate on graph-structured data. They learn representations of nodes, edges, or entire graphs by capturing both relational and topological information. Node representations are learned through iterative message passing across the graph structure, a process shown to be effective for tasks such as node classification [127, 126].

Although GNNs operate on inherently interpretable structured data, the embeddings they produce are not inherently interpretable. This is because the message aggregation and transformation functions used in GNN layers yield dense, abstract representations that obscure the contributions of individual features or neighbor messages. Moreover, the composition of multiple nonlinear layers further complicates understanding how local graph structure influences the final embedding or prediction.

Several methods have been proposed to enhance the interpretability of Graph Neural Networks (GNNs) by uncovering the structural patterns and feature-level contributions that influence their behavior on downstream tasks such as classification. GNNExplainer [176] is one of the earliest model-agnostic approaches designed to provide post-hoc, instance-level explanations. It operates by optimizing a mask over the input graph's edges and node features to reveal the most influential subgraph for a given prediction. While effective, it requires re-optimization for each instance, which limits its scalability and generalization to unseen inputs.

To overcome these limitations, PGExplainer [94] introduces a parameterized explanation model that learns to predict edge importance scores across multiple

instances. This enables efficient and inductive explanations, allowing the explainer to generalize to new input graphs without retraining. The method achieves high explanation fidelity while significantly reducing computational overhead compared to instance-specific optimizers like GNNExplainer.

In contrast to both methods, which provide *instance-level/local* explanations for individual predictions, XGNN [178] offers a *model-level/global* explanation strategy. Rather than highlighting important substructures in real input graphs, XGNN uses reinforcement learning to generate graphs from the input graph that maximize the model's confidence in a particular class label. These generated graphs represent the patterns that the GNN has learned to associate with specific classes, offering insight into the global decision logic of the model.

Together, these methods address different levels of explainability in GNNs: GNNExplainer and PGExplainer provide *instance-level* interpretability with varying levels of generalization, while XGNN enables *class-level* understanding by synthesizing prototypical graph examples.

While these approaches mark important advances in building explainable models, they are not designed to interpret embeddings learned by arbitrary knowledge graph embedding models, irrespective of their architectural design or associated downstream tasks. Consequently, the development of post-hoc methods that are both model- and task-agnostic, capable of explaining pretrained KG embeddings without access to internal weights or reliance on downstream predictions, remains largely unexplored. This constitutes a gap, especially given the extensive deployment of pretrained embeddings in real-world systems. We targeted this gap in chapter 5 by developing a model- and task-agnostic framework for providing instance- and model-level explanations.

Note that while this chapter offers a unified overview, each of the core chapters in this thesis includes a more detailed and topic-specific discussion of related work pertinent to its contributions.

Chapter 4

Towards Neural Network Interpretability Using Commonsense Knowledge Graphs

In this chapter we laid the foundations of our studies; we introduce a novel approach that leverages commonsense knowledge graphs to help understand image classifiers. By mapping the features and outputs of neural networks to structured knowledge graphs, we aim to provide a more intuitive and transparent explanation of the model's behavior. This chapter delves into the methodologies employed, the experiments conducted, and the findings that underscore the potential of knowledge graphs to demystify neural network predictions. This chapter is based on the following publication:

> *Ismaeil, Y., Stepanova, D., Tran, TK., Saranrittichai, P., Domokos, C., Blockeel, H.* (2022). Towards Neural Network Interpretability Using Commonsense Knowledge Graphs. In: *International Semantic Web Conference – ISWC 2022. Lecture Notes in Computer Science*, vol 13489, pp. 74-90. Springer, Cham.

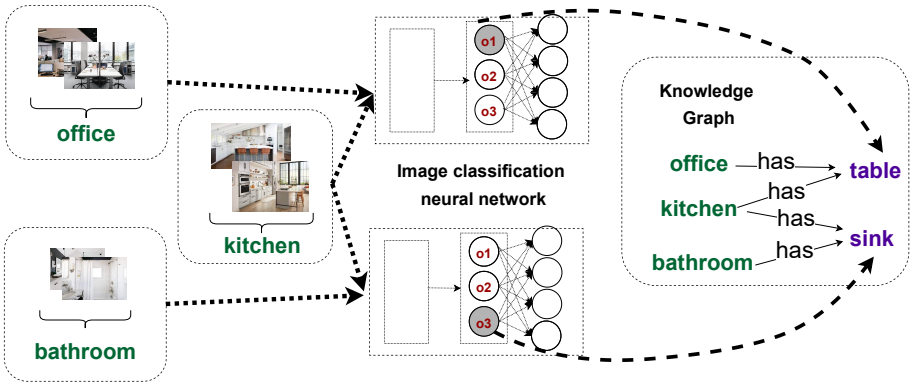


Figure 4.1: The goal of this chapter is to assign meaning to neurons using semantic properties of classes from a knowledge graph.

4.1 Introduction

Convolutional neural networks (CNNs) are well-known for their capacity to learn different powerful representations of the input in their successive layers. It is also well-known that they process images in a way that is not always intuitive to humans [50]. The lack of interpretability of CNNs is clearly undesirable, especially in safety-critical applications such as medical diagnosis or autonomous driving. This has led to an increased interest in methods that make the behavior of a trained CNN more interpretable by trying to assign human-understandable concepts (*e.g.*, face) to the neurons in the intermediate layers, often without explicit supervision [8, 103, 34, 106, 138, 185].

An important class of methods [8, 103, 35] proposes to align neurons with class attributes by using images in which segments are labeled. More specifically, one tries to find out which neurons are activated by particular image segments and, in that manner, associate these neurons with the label of the segment. For instance, if, over multiple images, a particular neuron tends to be active for the image segments labeled with *table*, one could argue that this neuron recognizes tables, and neurons in later layers essentially use this high-level information to decide whether the image shows, *e.g.*, an office. However, an important limitation of such methods is that during training they require fine-grained semantic labels of the images that are often not readily available and can be expensive to construct.

In this chapter, we provide an alternative. Instead of using semantically labeled images, we use external knowledge extracted from a knowledge graph (KG) is

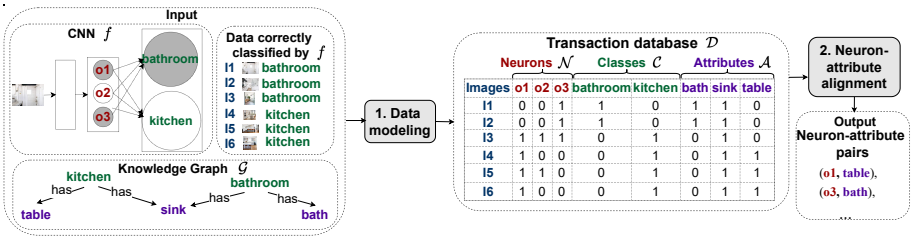


Figure 4.2: Workflow of the proposed framework.

available that contains symbolic descriptions of objects as a substitute. Extensive KGs of this kind exist. For example, ConceptNet [145] or WebChild [154] store semantic (including visual) information about concepts (*e.g.*, *offices contain tables*, *kitchens contain ovens*, *etc.*) acquired using crowd sourcing or information extraction from the Web. We develop methods that exploit such KGs by linking the class label of images to typical visible attributes of this class, and then trying to correlate neuron activation with these attributes. For example, if a particular neuron tends to be active for pictures of offices and kitchens, but not for pictures of bathrooms, and the KG states that offices and kitchens tend to contain tables while bathrooms do not, then this may be an indication that the neuron reflects the presence of a table (see Figure 4.1).

We demonstrate experimentally that the methods we propose successfully interpret neurons to the extent that they enable zero-shot learning of new classes with comparable performance to existing methods, but with the advantage of interpreting neurons in human-understandable terms. In the zero-shot learning setting, a model is expected to classify images from classes it has never encountered during training. Earlier works in this direction that likewise make use of external knowledge about classes [121, 105] propose to exploit such knowledge during training. While natural, these methods typically assume that prior to training, the knowledge of both seen and unseen classes is available. This implies that whenever the source of knowledge (e.g., KG) is updated with information about new unseen classes, training needs to be done completely from scratch, which might be undesirable. Removing knowledge about unseen classes during training makes the respective methods less effective (see Section 4.5). In contrast, our approach has the advantage that it requires knowledge about unseen classes from the knowledge graph only during inference, without needing it during training or earlier processing stages.

The main contributions of this chapter are summarized as follows:

- We propose a framework for mapping neurons in a fully connected layer of a CNN to attributes of classes from an external knowledge graph.
- We show that KGs indeed contain important semantic attributes of classes, which are helpful for CNNs.
- We experimentally demonstrate the usefulness of our framework for zero-shot learning and show how it can be effectively exploited for retrieving class predictions using reasoning over multiple networks.

4.2 Preliminaries

Image Classification CNN. Assuming a set of object classes $\mathcal{C}$, for a given (RGB) image $I\colon \Omega \to \mathbb{R}^3$, where Ω denotes the pixel space, we consider a function $f\colon (\Omega \to \mathbb{R}^3) \to \mathcal{C}$ for a parameter vector w. The function f, providing image classification, is defined as an L-layer *convolutional neural network* (CNN), namely $f(I) = \arg\max_{k \in \mathcal{C}} (\text{softmax}(f_L \circ \cdots \circ f_2 \circ f_1(I))_k)$, where f_l defines the l^{th} layer of the network.

Knowledge Graphs. In this chapter, we denote $\mathcal{V}$ the set of entities (also known as constants) and $\mathcal{P}$ the set of predicates in a given *knowledge graph* (KG). The knowledge graph (KG) in this chapter is represented as $\mathcal{G} \subseteq \mathcal{V} \times \mathcal{P} \times \mathcal{V}$, consisting of collections of factual information encoded as triplets of the form $\langle subject, predicate, object \rangle$. More formally, $\mathcal{G} = \{\langle s, p, o \rangle \mid s, o \in \mathcal{V} \text{ and } p \in \mathcal{P}\}$[1]. In this work, we focus on *commonsense KGs*, that is, KGs that describe visual and physical properties of object classes (*e.g.*, $\langle bathroom, has, bath \rangle$, $\langle kitchen, has, table \rangle$). Examples of such knowledge graphs include, e.g., ConceptNet [145] or WebChild [154].

Association Rules. Association rules are widely used in the context of data mining. We will use association rules to model relations between neurons and attributes. First, we define a *transaction database* $\mathcal{D} = \{(i, \mathcal{X}) \mid i \in \mathcal{U} \text{ and } \mathcal{X} \subseteq \mathcal{I}\}$, where $\mathcal{I}$ stands for a set of items, and $\mathcal{U}$ is a set of IDs. A *transaction* $(i, \mathcal{X})$ has a (unique) ID i and $\mathcal{X} \subseteq \mathcal{I}$ denoting an itemset. Furthermore, we introduce the *support* of an itemset $\mathcal{X}$ in a given transaction database $\mathcal{D}$, which is the frequency of transactions in $\mathcal{D}$ containing the itemset $\mathcal{X}$:

$$\text{supp}(\mathcal{X}) = |\{(i, \mathcal{X}') \in \mathcal{D} \mid \mathcal{X} \subseteq \mathcal{X}')\}| \; / \; |\mathcal{D}| \; .$$

[1] Alternatively, a KG $\mathcal{G} = (\mathcal{V}, \{\mathcal{E}_p \subseteq \mathcal{V} \times \mathcal{V}\}_{p \in \mathcal{P}})$ can be viewed as a directed super-graph (*i.e.* a composition of directed graphs $\mathcal{G}_p = (\mathcal{V}, \mathcal{E}_p), \forall p \in \mathcal{P}$, where the edges are labeled by the predicates p.

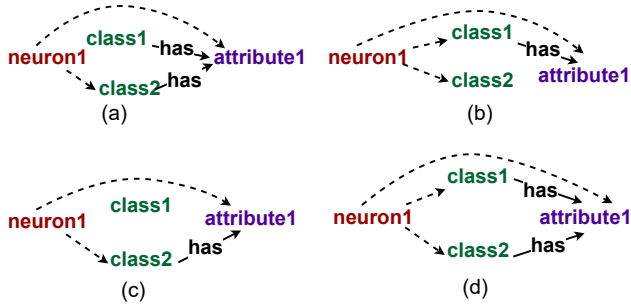


Figure 4.3: Illustration of neuron-attribute alignments, where dashed lines show correlations between items in $\mathcal{D}$. (a), (b) present invalid alignments; (c), (d) reflect desirable alignments.

For example, if $\mathcal{I} = \{\text{Neuron\_A}, \text{Neuron\_B}, \text{Attribute\_X}, \text{Attribute\_Y}\}$, then a possible transaction is $(1, \{\text{Neuron\_A}, \text{Attribute\_X}\})$, indicating that transaction 1 contains both `Neuron_A` and `Attribute_X`. Thus, $\mathcal{X}$ denotes a set of co-occurring items and is not a single item.

We consider bi-directional *association rules*, which are expressions of the form $\mathcal{X} \Leftrightarrow \mathcal{Y}$, where $\mathcal{X}, \mathcal{Y} \subseteq \mathcal{I}$. Association rules can be ranked relying on certain *interestingness metrics* [3]. In this work, we focus on Jaccard index (*a.k.a.* intersection over union), which for a given association rule $\mathcal{X} \Leftrightarrow \mathcal{Y}$ computes the ratio of co-occurrences to all occurrences of $\mathcal{X}$ and $\mathcal{Y}$ in the transaction dataset:[2]

$$J(\mathcal{X} \Leftrightarrow \mathcal{Y}) = \frac{\text{supp}(\mathcal{X} \cap \mathcal{Y})}{\text{supp}(\mathcal{X}) + \text{supp}(\mathcal{Y}) - \text{supp}(\mathcal{X} \cup \mathcal{Y})} \,. \tag{4.1}$$

4.3 Generating Neuron-Attribute Alignments

Our goal is to interpret neurons' behavior[3] in human-interpretable terms. In this chapter, we consider image classification as an application. In contrast to existing methods [8, 103], we aim at developing a framework that supports both qualitative and quantitative evaluation, as defined by the functionality-grounded evaluation in [39], without relying on semantically labeled data.

We propose to interpret neurons' behavior in terms of the semantic properties (*a.k.a.* attributes, encoded by a set $\mathcal{A}$) from pre-constructed KGs, which store

[2] We also write $J(\mathcal{X}, \mathcal{Y})$ for conciseness.

[3] A neuron can also be understood as an element of the vector of activation output for a given layer.

human knowledge about classes from $\mathcal{C}$. To this end, we aim at answering the following key questions:

Q1. Do the extracted alignments comply with the behavior of the CNN from which they were extracted?

Q2. Do networks learn in terms of attributes defined by the knowledge graph?

Q3. Beyond explainability, how can we utilize the extracted explanations?

Framework Overview. First, we present our framework, depicted in Figure 4.2, for aligning individual neurons with high-level attributes from a KG.

Let us consider a neural network f trained for image classification on a labeled image dataset $\mathcal{T} = \{(I, c) \mid I \colon \Omega \to \mathbb{R}^3 \text{ and } c \in \mathcal{C}\}$. Moreover, assume we are given a knowledge graph $\mathcal{G} = \{\langle c, p, a\rangle \mid c \in \mathcal{C} \text{ and } a \in \mathcal{A}\}$ storing semantic properties (*a.k.a.* attributes) $\mathcal{A}$ of classes from $\mathcal{C}$, where the predicate p reflects a visual property, e.g., *hasColor*, *hasShape*, *hasPart*. While any kind of relation can be used in this context, we select those that likely provide visual attributes because naturally they are more effective for vision tasks than other relations such as *capableOf*, *isA*, etc.

We aim at aligning the neurons with the nodes $a \in \mathcal{A}$ from the KG $\mathcal{G}$. While, in principle, any set of neurons can be interpreted by our framework, we propose to specifically focus on the neurons from fully-connected layers, since these are known to reflect high-level abstract visual features [131]. Therefore, we consider a layer $f_l \colon \mathbb{R}^m \to \mathbb{R}^n$, $x \mapsto \sigma(Wx + b)$, where m and n stand for the fan-in and fan-out, respectively, for the given layer indexed by l and $\sigma \colon \mathbb{R}^n \to \mathbb{R}^n$ is an activation function. For $f_l(x) = \begin{bmatrix} o_1 & o_2 & \dots & o_n \end{bmatrix}^\top$, we will use the notation $o_1, o_2, \dots, o_n$ for the individual neurons.

In the first step of our framework, we model the input data as a transaction database (Section 4.3.1). We then compute the neuron-attribute alignments using data mining-based methods (Section 4.3.2). The output of our framework is a set of neuron-attribute pairs ρ of the form (o, a), where o is an individual neuron of f, and $a \in \mathcal{A}$ is an entity in $\mathcal{G}$ corresponding to an attribute of a class in $\mathcal{C}$. In Section 4.4 and Section 4.5, we empirically provide answers to **Q1-Q3**.

4.3.1 Data Modeling

Suppose we are given a dataset $\mathcal{S} = \{(I, c) \mid f(I) = c\} \subseteq \mathcal{T}$ of images that are correctly classified by f. Given the neural network f, and the set $\mathcal{N} =$

$\{o_1, \ldots, o_n\}$ of individual neurons from the target layer f_l we proceed with constructing the transaction dataset $\mathcal{D}$ with items $\mathcal{I} = \mathcal{C} \cup \mathcal{N} \cup \mathcal{A}$. For every $(I, c) \in \mathcal{S}$, $\mathcal{D}$ stores a transaction $(i, \mathcal{X}_i)$, where i is the unique ID of the image I, and $\mathcal{X}_i \subseteq \{c\} \cup \mathcal{N} \cup \mathcal{A}$. For $a \in \mathcal{A}$, we have that $a \in \mathcal{X}_i$ iff $\langle c, p, a \rangle \in \mathcal{G}$, that is, the class of the image c and all of its attributes from the KG are in $\mathcal{X}_i$.

Intuitively, for every neuron $o \in \mathcal{N}$, it holds that $o \in \mathcal{X}_i$ iff o has high value before softmax when I is passed through f. To detect neurons from $\mathcal{N}$ with high value, the continuous values of the neurons are simply thresholded to a binary value $v_o \in \{0, 1\}$. This can be done *a priori* (*e.g.*, for post-ReLU activations) or by dynamically thresholding above neuron-specific percentile [139, 103]. This way, for each image $I_i \in \mathcal{S}$ we identify a set of neurons from $\mathcal{N}$ with high activation value for the given image, and collect them into $\mathcal{X}_i$.

4.3.2 Neuron-Attribute Alignment

We then rely on the constructed transaction data $\mathcal{D}$ to compute the alignments between neurons and attributes, *i.e.*, items in $\mathcal{N}$ and those in $\mathcal{A}$, respectively. We propose methods for computing such alignments that we describe next.

Direct Method. Intuitively, a neuron o is *correlated* with an attribute a if the following two conditions hold: 1) It is highly probable that the attribute a is visually present in an image, given that the neuron o is active for it; 2) it is highly probable that the neuron o is active, given that the attribute a is visually present in the image. Note that we cannot directly compute the respective probabilities since the images are not explicitly labeled with the attributes. Therefore, instead, we estimate such probabilities by relying on the assumption that an attribute a is likely visible in an image I belonging to the class c if $\langle c, p, a \rangle \in \mathcal{G}$.

Example 1: Given $\mathcal{D}$ from Figure 4.2 and $\theta = 0.7$, we have $J(o_1 \Leftrightarrow \mathit{sink}) = 4/6$, $J(o_1 \Leftrightarrow \mathit{table}) = 3/4$ and $J(o_3 \Leftrightarrow \mathit{bath}) = 2/3$, $J(o_3 \Leftrightarrow \mathit{sink}) = 3/6$. Thus, we obtain only (o_1, table) as the resulting alignment.

Constrained Method. While natural, the main drawback of the above *direct* method is that it only considers correlated pairs of neurons and attributes but ignores the knowledge about classes, like both bathrooms and kitchens have sinks, while offices and bedrooms do not. This information is important, especially when the dataset is unbalanced, to ensure that all meaningful alignments are computed.

Example 2: Reconsider Example 1. Looking closer at $\mathcal{D}$, one can observe that o_1 is highly correlated with the class *kitchen*, as it is active for all images of this class. Similarly, o_3 is highly correlated with the class *bathroom*. Since *bath* is the attribute which is relevant only for the class *bathroom* but not for *kitchen*, it would be expected that $(o_3, bath)$ is also included in the resulting set of alignments along with $(o_1, table)$. Decreasing the threshold θ to a lower value (e.g., 0.65) would resolve this, but would also lead to $(o_1, sink)$ being in the result, which is counter-intuitive, since *sink* is an attribute which is relevant both for *bathroom* and *kitchen*, but *o1* is active only for images of the latter class.

Intuitively, an alignment is deemed meaningless if it fits into the cases (a) or (b) depicted in Figure 4.3 (more formally defined via constrains below). An alignment in Figure 4.3 (a) is undesirable, because we consider the neuron's activity as a sign of the existence or absence of the attribute in the input image. Consequently, if a neuron is active frequently for images of only one particular class, it might be a sign that this neuron is triggered by some attribute that only belongs to this class but not shared with other classes. Similarly, following Figure 4.3 (b), if a neuron is active frequently for images of a set of classes, then it might be an indication that it is triggered by attributes that are shared among these classes.

For example, if o is aligned with the sink, then we expect o with high probability to be active for images of classes that typically contain sink (*e.g.*, bathrooms and kitchens), but not those that do not have sink (*e.g.*, bedrooms and offices). To alleviate the threshold's rigidity in the *direct* method, we establish formal constraints that allow us to filter out those and only those alignments that become completely meaningless when the class information is taken into account. The respective constraints are presented below:

(1) if $|\{c_i \in \mathcal{C} \mid \langle c_i, p, a\rangle \in \mathcal{G}\}| \geq 2$, and $|\{c_j \in \mathcal{C} \mid \langle c_j, p, a\rangle \in \mathcal{G} \text{ and } J(o, c_j) \geq \beta\}| < 2$, then (o, a) is an *invalid* alignment (see Figure 4.3 (a)).

(2) if $\langle c_i, p, a\rangle \in \mathcal{G}$, for all $c_j \in \mathcal{C} \setminus \{c_i\}$, $\langle c_j, p, a\rangle \notin \mathcal{G}$ and $|\{c_j \in \mathcal{C} \setminus \{c_i\} \mid J(o, c_j) \geq \beta\}| \geq k - 1$, then (o, a) is an *invalid* alignment, where $2 \leq k \leq |\mathcal{C}|$ is a parameter (see Figure 4.3 (b) for $k = 2$).

Intuitively, the first constraint (1) states that if at least two classes have an attribute a, but only less than two out of them are correlated with the neuron o, then the alignment (o, a) is invalid. The second constraint (2) reflects that if a is relevant for a single class only, and the number of other classes correlated with o is larger than k, then (o, a) is invalid.

In the *constrained-k* method, after computing the alignments relying on the Jaccard similarity for a given threshold θ, we post-process the results by removing alignments that violate the above constraints. The *constrained-k* method is illustrated by the following example.

Example 3: We have $\langle c_i, has, sink\rangle \in \mathcal{G}$ for $c_i \in \{kitchen, bathroom\}$ in Example 1, *i.e.*, *sink* is an attribute that is relevant for at least two classes. Moreover, for $\beta = \theta$, we have $J(o_1, kitchen) > \beta$, but $J(o_1, bathroom) < \beta$, namely, the neuron o_1 is only frequently active for images of kitchen, but rarely for those of bathroom. Hence, based on the constraint (1), we remove $(o_1, sink)$ from the list of alignments computed by the *direct* method. Analogously, $(o_3, sink)$ is removed.

4.4 Evaluation Strategies and Applications of Neuron-Attribute Alignments

We now explore strategies for evaluating neuron-attribute alignments using functionality-grounded evaluation [39] and discuss their potential applications.

Copy-paste Adversarial Examples. Adversarial images are images that are intentionally perturbed to confuse and deceive visual models. Changes to the image can be made in different ways, one of which is by copying patches from images of one class and pasting them into images of another class. Images with this type of perturbation are known as copy-paste adversarial images (as defined in [17]). Our hypothesis is that, if the network learns in terms of the attributes used in our work for interpreting neurons, then copying an image patch of an attribute relevant only for single class and pasting it into an image from another class should result in confusing the CNN to predict the class of the given image as the one to which the patch belongs. An example of a copy-paste adversarial image can be obtained by copying the image patch representing the bed from a bedroom image and pasting it into a bathroom image. Intuitively, if passing the bathroom image with a bed through a CNN trained on images of bathrooms and bedrooms results in confusing the network to classify the input image as a bedroom, then one can confirm that the network learns in terms of the attributes associated with the classes. We exploit the copy-paste adversarial images to in Section 4.5 to address **(Q1)** and **Q2**.

Towards addressing **(Q3)**, next we propose applications, in which the computed neuron-attribute alignments could be useful.

Zero-shot Learning. The first application concerns zero-shot learning, *i.e.*, a popular task in image classification, in which images of new (*i.e.*, unseen) classes that do not exist in the training set need to be classified. For that, we develop an image classifier from the obtained neuron-attribute alignments. More specifically, given an image I of an unseen class, the trained network f, the pre-computed neuron-attribute pairs $\rho = \{(o, a) \mid o \in \mathcal{N}, a \in \mathcal{A}\}$, as well as the KG $\mathcal{G}$ storing the semantic information about the (un)seen classes, our goal is to derive the most likely class to which the target image I belongs. First, we pass I through f, and collect the set of activated neurons among $o \in \mathcal{N}$. We then exploit the neuron-attribute pairs ρ to retrieve the list of attributes, with which the active neurons are aligned, *i.e.*, $\mathcal{A}_I = \{a \in \mathcal{A}_c \mid (o, a) \in \rho$, o is active by I in $f\}$ and $\mathcal{A}_c$is a set of the attributes for class c. Finally, relying on $\mathcal{G}$, the classes $c \in \mathcal{C}$ are ranked based on the following scoring function:

$$\text{Score}(c) = \sum_{a \in \mathcal{A}_I} w_a \; / \; |\{a : \langle c, p, a\rangle \in \mathcal{G}\}| \tag{4.2}$$

where w_a is the ratio of neurons activated by the given image that are aligned with the attribute a to the total number of neurons activated by the image. The more active neurons are aligned with an attribute a, the more certain we are that a exists in the image and subsequently the greater is w_a. The candidate classes are ranked based on the formula from Equation 4.2 and the top-ranked class is selected as the final prediction. Intuitively, the proposed technique referred to as *attribute-based classifier* allows one to reduce the image classification task to reasoning over the KG. The *attribute-based classifier* acts as an evaluator of the extracted alignments; if it gives high classification accuracy, then the extracted alignments reflect what the network learns.

Reasoning over Multiple Networks. The described *attribute-based classifier* can also be exploited in the setting, where multiple neural networks trained on non-intersecting sets of classes are used to solve tasks that are outside their initial scope (i.e., classify images into classes unseen by either of the networks). The systematic way of combining knowledge extracted from multiple networks is beneficial, as it allows one to avoid massive retraining on a larger number of classes while preserving high accuracy of predictions.

The procedure for attribute-based zero-shot classification can be naturally extended to handle several networks. First, we collect activated neurons for a given image from a number of networks, and then use the neuron-attribute alignments pre-computed for each network separately to detect attributes in the image. Finally, we merge the acquired knowledge using KG to make a decision regarding the most likely class of the image exactly in the same way as described above.

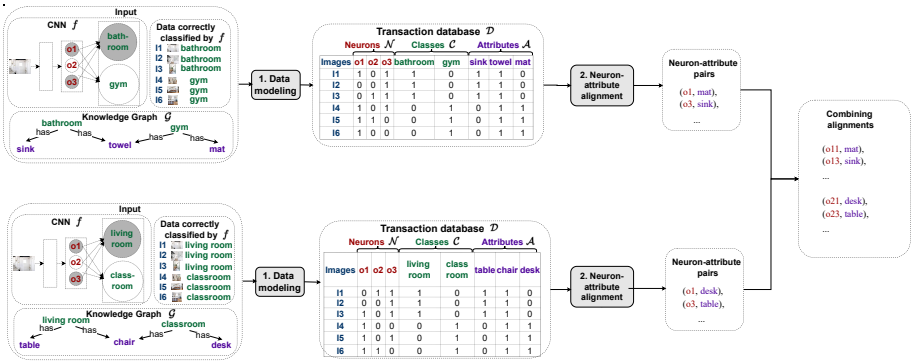


Figure 4.4: The figure illustrates how to combine knowledge acquired by two networks. The aggregate alignments are later employed to reason about images from unseen classes.

Example 4: Consider two convolutional neural networks f_1 and f_2 trained on the classes $C_1 = \{livingRoom, classroom\}$ and $C_2 = \{gym, bathroom\}$, respectively. Assume that the classes from C_1 have the following attributes $A_1 = \{table, chair, desk\}$, while those from C_2 contain the set of attributes $A_2 = \{mat, sink, towel\}$. Combining the knowledge acquired by the two networks would allow us to classify images into a new class, e.g., *kitchen*, which contains the set of attributes $A_3 = \{table, sink\}$ that were seen separately by the respective networks (see Figure 4.4).

4.5 Experiments

We evaluate the proposed method for aligning neurons of a network with attributes of classes from a knowledge graph by empirically analyzing **(Q1)-(Q3)** from Section 4.3.

4.5.1 Experimental Setup

Datasets. We consider the popular *MITScenes* [114] and *AwA2* [169] datasets, and use *ConceptNet* [145] knowledge graph as a knowledge source.

MITScenes: We have selected images from the MIT scene dataset [114] belonging to classes whose visual properties are well covered in ConceptNet, which resulted in 10,475 images labelled with 15 classes. We use 3,840 images from 10 classes for *training*. For testing the performance on seen classes, we have 2,320 images belonging to the same 10 classes. For testing on the unseen classes, we have 4,675 images from the remaining 5 classes. On average, for each class, we get 645 images.

AwA2: We also consider a subset of the *AwA2* [169] dataset, in which the semantic information about image classes is well covered in the KG. We get 17,746 images spread across 20 classes. For training, we have 7,842 images from 15 classes. For testing on seen classes, we have 5,229 images belonging to the same 15 classes. For testing on unseen classes, we have 4,675 images from the remaining 5 classes.

ConceptNet KG: The *AwA2* and *MITScenes* datasets come with attributes already; however, their coverage is rather low (7.2 attributes per class at most). Our goal is to demonstrate the usefulness of commonsense KGs as sources for acquiring further class knowledge. For that, we have extracted attributes from a popular commonsense KG, ConceptNet [145]. For *MITScenes*, we collect the attributes connected to the classes via the inverse of *atLocation* relation (e.g., $\langle table, atLocation, kitchen \rangle$). In total, we get 1,680 attributes which on average amounts to 112 attributes per class. For *AwA2*, we use the predicate *has* to get 3,352 attributes, which yields 167 attributes per class on average.

CNN training. We adopt ResNet50 [56] pre-trained on ImageNet [36] as the backbone, which in some experiments is fine-tuned on the considered datasets, while in others trained from scratch as described separately in each subsection. We replaced the fully connected layer before the last one in ResNet50 with a fully connected layer that has 2,048 neurons, which we aim at aligning with the attributes from ConceptNet.

It is important to note that the values for the parameters θ and β used in this chapter were determined empirically through experimentation and may require adjustment when applied to different datasets. **Baselines.** We compare our direct (*dir*) and constrained (*con*) methods (with fine-tuned parameters θ and β) for the attribute-based classification against the state-of-the-art methods that likewise make use of KGs (but do not map neurons to the KG entities), namely Dense Graph Propagation method (*DGP*) [73], Attentive Zero-Shot Learning method (*AZSL-D*) [49], and *ZSL-KG* [105]. *AZYL-D* and *ZSL-KG* rely on *DGP*, which is a framework that proposes a dense connection scheme of a knowledge graph to optimize the knowledge propagation between distant nodes in shallow networks such as graph convolution networks.


Figure 4.5: Copy-paste adversarial examples: *art studio* with an *island* attribute relevant only for *kitchen* (left) and *office* with a *rug* relevant to classes *bedroom* and *office*.

Table 4.1: Class samples of adversarial copy-paste examples.

Original class	**Inserted attribute**	**Resulting class**
Dining room	Wall with doors	Corridor
Classroom	Shower	Bathroom
Office	Bed	Bedroom
Kitchen	Furniture	Office
Living room	Painting	Art Studio

We run the methods proposed in [49] and [105], ensuring that no semantic information about unseen classes in the KG is used during training.

Evaluation metrics. We use the standard *hit@1* metric, which reflects the percentage of test images for which the method returned the correct class prediction in the top-1.

4.5.2 Copy-paste Adversarial Examples

To answer **(Q1)-(Q2)**, following [103] we first generate the copy-paste adversarial examples using the *MITScenes* dataset, which comes with labeled semantic segments, as follows. Out of all labels, we select those (set $\mathcal{A}$) which are present in ConceptNet. Then, for each class $c \in \mathcal{C}$, we construct the set $\mathcal{A}_c = \{a \mid \langle c, p, a\rangle \in \mathcal{A} \text{ and } \forall c' \in \mathcal{C}, \langle c', p, a\rangle \notin \mathcal{G}\}$. For every pair of images I, I' from the test set belonging to different classes c and c' respectively, we insert

Table 4.2: Results for copy-paste adversarial classification on the *MITScenes* dataset. The alignment score is the percentage of images on which the attribute-based classifier (i.e., *con-all* (direct), *con-2* (using constraints)) made the same prediction as *ResNet50*.

Method	**hit@1**(%)	**Alignment**(%)
ResNet50	78.2	-
con-all	**82.3**	96.14
con-2	82.1	96.64

the visualization of a randomly selected attribute $a \in \mathcal{A}_{c'}$ from the image I' into the image I, and label it with c'. For example, given an image I of an art studio and I' of a kitchen, as a copy-paste adversarial example, we generate an image I with the kitchen island from I' inserted into I (see Figure 4.5 for illustration). The resulting image is labeled as a kitchen. The attributes to be inserted for every class are chosen randomly while making sure that they exist visually in at least one image (see Table 4.1 for class and attribute examples).

This way, we obtain 2,320 adversarial copy-paste examples. We then analyze whether the considered CNN ResNet50 and our attribute-based classifier described in Section 4.4 misclassify the adversarial images relying on the inserted attribute. For instance, in the above example, we expect the network to misclassify the art studio as a kitchen. To perform such an evaluation, we pass every copy-paste adversarial example through the network and compute the *hit@1* score.

The results are presented in Table 4.2. High misclassification and alignment scores demonstrate that the KG attributes that distinguish classes from each other are indeed important for classification [27].

We have also repeated the same experiment on examples, constructed by inserting attributes relevant for multiple classes into the image (e.g., inserting a kitchen door into the art studio). We observe that in this case, the misclassification score in *hit@1* drops to around 2% for *ResNet50*. This witnesses that not all parts of an image are equally important for the network to make decisions, but only those that are distinguishing a given class from others based on the KG.

4.5.3 Zero-shot Learning Task

To answer **(Q1)** and **(Q3)**, we compare the introduced methods for attribute-based classification to the baselines *AZSL-D* [49], *ZSL-KG* [121], and *DGP* [73]

with respect to their performance on the zero-shot learning task.

For this task, we trained *ResNet50* on 10 classes of the *MITScene* dataset from scratch. We then used the trained network to compute the neuron-attribute alignment pairs for the classes from the training set. The other 5 classes are used for testing.

Since the knowledge graph stores the semantic information about both seen and unseen classes, we effectively exploit this knowledge along with the neuron-attribute alignments computed by our methods to classify the images from the unseen classes as described in Section 4.4. Importantly, the attributes of unseen classes are only used at the inference phase, but not during training.

Table 4.3 presents the results for the zero-shot learning tasks for *MITScenes* and *AwA2* datasets, respectively. Importantly, we report the performance both when using the attributes based on the semantic labels that accompany the datasets, as well as those from the ConceptNet KG.[4] For the *MITScenes* dataset, the attribute-based classifier that exploits attributes from *ConceptNet* outperforms all baselines including the attribute-based classifier that makes use of the labels coming with the dataset. This is due to the fact that, among visual attributes, the KG also provides a set of non-visual attributes that help in linking semantically similar classes via alignments. Moreover, the attribute labels coming from the dataset are shared among different classes, which leaves only a few attributes discriminatively describing each class.

Example Alignments. We present the alignments computed by our method for the considered datasets in Figure 4.7. One can observe that the alignments indeed contain attributes visually relevant to the respective images.

4.5.4 Reasoning over Multiple Networks

We analyze the usefulness of the neuron-attribute alignments for the task of joint reasoning over multiple networks without having to retrain or fine-tune them.

In this experiment we trained from scratch two ResNet50 networks net_1 and net_2, on *AwA2* as follows. net_1 was trained for 15 epochs on 8 classes, and net_2 for 15 epochs on the other 7 classes. Moreover, we trained another network net on all 15 classes for 16 epochs. We get for net_1 a test accuracy on seen classes of 74.2%, for net_2 of 86.4%, and for net 80.5%. For the test set, we have images from 5 unseen classes.

[4]We also experimented with the WebChild [154] KG, but the results for ConceptNet are more promising.

Table 4.3: Zero-shot learning results on *MITScenes* and *AwA2*.

	Attribute source	**Method**	**Unseen hit@1**	**Seen hit@1**
MITScenes	-	AZSL-D	8.26	8.33
	-	DGP	19.87	9.73
	-	ZSL-KG	9.31	11.14
	Labels	*direct*	20.0	9.4
		con-2	20.0	19.1
		con-all	23.8	30.4
	Concept Net	*direct*	31.7	10.2
		con-2	38.4	50.0
		con-all	37.6	50.8
AwA2	-	AZSL-D	23.9	6.5
	-	DGP	24.1	6.6
	-	ZSL-KG	9.38	14.18
	Labels	*direct*	29.2	12.0
		con-2	38.6	41.5
		con-all	24.6	23.4
	Concept Net	*direct*	32.7	9.5
		con-2	40.2	47.6
		con-all	23.1	19.1

Figure 4.6 shows the results of the presented methods for reasoning over knowledge learned by multiple networks. For *AwA2* dataset, the attribute-based classifier that jointly considers neuron-attribute alignments from both net_1 and net_2 outperforms the attribute-based classifiers constructed separately for net_1, net_2 and net respectively. For the *MITScenes* dataset, a similar trend is observed with the exception that the attribute-based classifier constructed relying on net_2 significantly outperforms the one based on net_1 and has comparable performance to other classifiers. This is due to the fact that net_1 (resp. net_2) was trained on classes with many (resp. few) attributes in common with the unseen classes.

4.6 Related work

Recently, there has been an increasing interest in understanding what deep learning models learn. Our work extends the earlier proposals on explaining individual neurons in deep representations [8, 103, 34, 44, 35]. However, in contrast to existing work, we do not rely on large amounts of training data, but instead exploit external knowledge graphs. In [35], semantic knowledge (in the

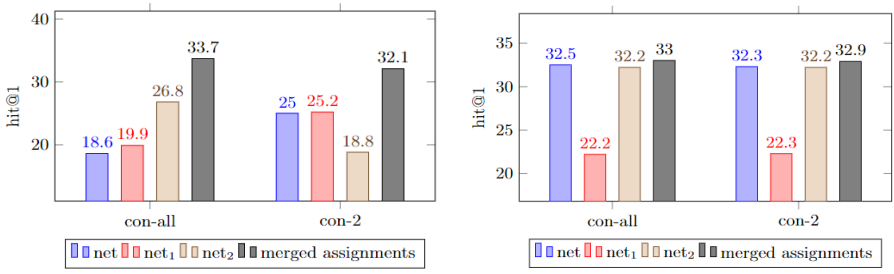


Figure 4.6: Zero-shot learning using neuron-attribute alignments over multiple networks on *AwA2* (left) and *MITScenes* (right).

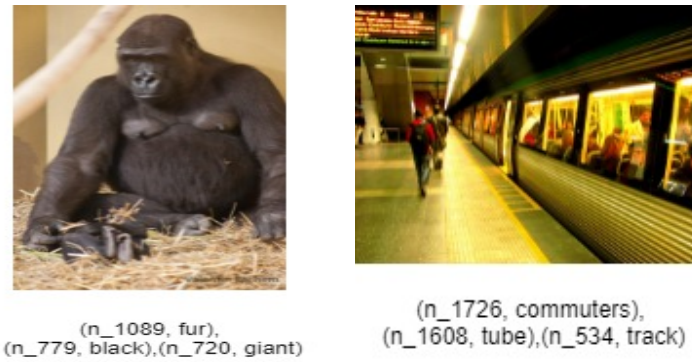


Figure 4.7: Examples of neuron-attribute pairs computed by our method.

form of ontologies) has also been considered for interpreting CNNs by generating explanations, yet this method again makes use of labeled and segmented images, unlike we do.

Our approach for aligning neurons and KG attributes is in the spirit of [43, 61], where data mining has been exploited for interpreting neural networks or forming concepts based solely on neurons, but KGs have not been considered in this context. Recently, many works have motivated the exploitation of KGs for enhancing the performance of image classifiers, e.g., [49, 121, 105, 73] (see [22] for an overview). The direction of explaining the behavior of CNNs with semantic technologies has been also discussed as a valuable research stream in several works [125, 86]. However, to the best of our knowledge, no concrete proposals for aligning individual neurons with KG entities exist to date.

Several works have considered graph neural networks (GNNs) for zero-shot learning [49, 105, 121, 73]. These methods make use of graph-structured external

knowledge, in which each class is represented by a single node and each inter-class link is represented by an edge. Given the external knowledge graph, its embedding representation is first computed using a GNN, and then exploited for the zero-shot learning task. Some techniques [73, 49] utilize convolutional layers with an additional dense connection layer to propagate features to distant nodes within the same network. The work [49] further introduced weighted aggregation as a method for emphasizing more significant neighboring nodes for class nodes. A key difference between these approaches and our work is that they do not aim at aligning neurons with KG entities like we do. Additionally, they explicitly include KG-based information about unseen classes during training, whereas we only exploit this knowledge at the inference phase. We observed that when knowledge about unseen classes is omitted from the information used for training, the performance of AZSL-D [49], DGP [105], and ZSL-KG [121] drops significantly (see Table 4.3).

4.7 Conclusion

This chapter proposes a framework for aligning neurons of a neural network to attributes defined by external commonsense knowledge graphs. These alignments not only make NNs more interpretable (see Figure 4.7 for examples), but are also useful in various applications, such as zero-shot classification (using multiple networks). Our framework does not require the knowledge about unseen classes to be used during training, but rather exploits it at inference stages. Our results demonstrate that commonsense KGs contain distinctive attributes relying on which CNNs tend to perform classification. This demonstrates the importance and usefulness of commonsense KGs for computer vision tasks.

Although we relied on ConceptNet KG in the experiments, our work is certainly not bound to it, and other KGs (or combinations of them) can likewise be exploited. We believe that our method has a broader impact, as it offers an interesting perspective for reducing machine learning tasks to those of reasoning over KGs.

Chapter 5

FeaBI: A Feature Selection-Based Framework for Interpreting KG Embeddings

This chapter addresses some of the shortcomings identified in the previous chapter regarding model interpretation. Rather than using potentially irrelevant information from knowledge graphs, we focus on incorporating only relevant knowledge graph information that aligns with the training data used by the models. Knowledge graph embedding models are particularly well-suited for this purpose, as they directly utilize knowledge graphs as their training data. This chapter introduces a framework that offers interpretable versions of the dense, non-transparent representations learned by KG embedding models, as discussed in the following paper:

Ismaeil, Y., Stepanova, D., Tran, TK., and Blockeel, H. (2023).
FeaBI: A Feature Selection-Based Framework for Interpreting KG Embeddings.
In: *International Semantic Web (pp. 599-617) Cham: Springer Nature Switzerland.*

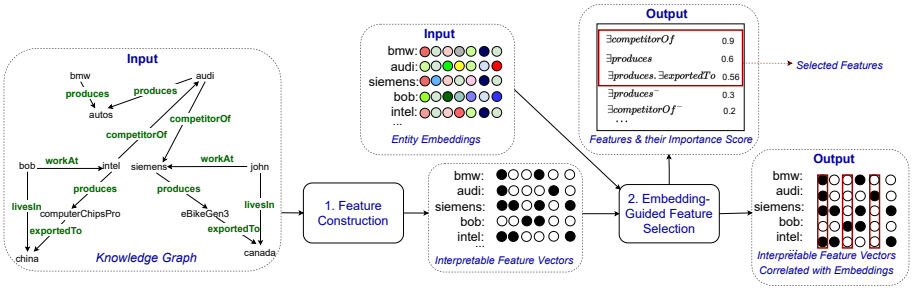


Figure 5.1: Embedding-guided feature vector generation for KG entities

5.1 Introduction

Motivation. Knowledge Graphs (KG) describe facts about a certain domain of interest by representing them using entities interconnected via relations. Existing KGs such as YAGO [149], Freebase [13], and DBpedia [135] contain millions of facts about people, places, organizations, etc. over hundreds of relations. For instance, an example of a KG presenting information about companies, people, products, and relations among them is presented on the left side of fig. 5.1.

Deep learning-based embedding techniques, particularly Knowledge Graph (KG) embeddings (see [166] for a comprehensive overview), have gained significant traction in addressing a range of machine learning tasks. These include, among others, link prediction [14, 160, 173] and entity classification [119, 75], where KGs serve as structured input data. However, these techniques generally produce latent representations for entities that lack direct human interpretability and are often regarded as black-box models, unless specifically designed or augmented with mechanisms to enhance transparency.

State-of-the-Art and its Limitations. There has been a surge of interest in understanding how deep learning models work. Two distinct types of explanations have emerged: prediction explanations and model explanations [31]. While prediction explanations justify a particular prediction generated by a given black-box model, different definitions for model explanations exist in the literature (see, e.g., [85] for an extensive discussion). In this chapter, with model explanations we refer to the identification of parts of the input data that contributed most to the construction of the model, also known as feature attribution [11]. Several works have focused on constructing prediction explanations for embeddings [120, 20, 10, 53]. E.g., [120] generates explanations for the link prediction task by perturbing the training data, while

[20] constructs prediction explanations for link prediction and triple classification using entity co-occurrence statistics. However, none of the above methods target the problem of computing model explanations, which are complementary to prediction explanations and can be exploited for model analysis. While some methods construct interpretable embeddings [148], and others develop inherently interpretable embedding models [178, 6], these approaches focus on building interpretable embeddings or models from scratch. In contrast, this chapter aims to explain and understand existing, trained embedding models.

Our Approach and Contributions. We present a novel method (see fig. 5.1 for overview), which computes explainable vectors for entity embeddings and generates KG embedding model explanations in the form of KG features important for the given embedding model.

Our method proceeds as follows. First, we extract propositional features from the KG and express them in Description Logic [7]. Then, we use these features to construct Boolean vectors (a.k.a. feature vectors) for each entity relying on its neighborhood. For example, in the KG from fig. 5.1, the Boolean feature vector representation of the entity *siemens* contains the information that *siemens* produces some products that are exported to *canada* and *audi* is among its competitors. Next, given a pre-computed embedding model that we want to explain (i.e., a function mapping entities to vectors in some d-dimensional space), we train a regression random forest on the task of reconstructing embedding-based entity representations using features defined in the first step of our method. The regression random forest model ranks features based on their importance for the reconstruction task. We consider the highest-scoring features as the KG embedding model explanations.

Intuitively, the obtained feature vector and embedding entity representations are two ways of representing entities in the KG. Thus, if there is a regression model that can accurately reconstruct KG embeddings from feature vector-based representations, then the feature vectors and the embeddings contain comparable information. We use the list of the computed most important features as an explanation for the KG embedding model. Finally, we reduce the Boolean feature vector of each entity such that it contains only the features that exist in the list of most important features obtained in the previous step. For instance, in the KG from fig. 5.1, the resulting Boolean feature vector representation for *siemens* will contain only selected features.

Our main contributions are summarized as follows:

- We present an automated method for generating model explanations for KG embeddings in the form of KG features most relevant for the given embedding. These identified important KG features are used to construct

DL concept	Informal description
$\exists r.\top$	Existence of some relation r
$\exists r.\{e\}$	Existence of some relation r to the entity e
$\exists R.\{e\}$	Existence of any relation to the entity e
$\exists r_1.\exists r_2...\exists r_k.\top$	Existence of a chain of relations $r_1, r_2, \ldots, r_k$
$\leq kR.\top$, $\geq kR.\top$	Less (greater) than or equal to k overall number of relations

Table 5.1: Description Logic concepts where r is an (atomic) relation or its inverse, R is any relation or its inverse, e is an entity, $\top$ is the universal (top) concept.

interpretable feature vector representations (entity-level explanations) for each KG entity, approximating their respective embedding vectors.

- We empirically demonstrate that the generated interpretable feature vectors behave similarly to the respective entity embeddings on the entity and relation classification tasks.
- We show how the interpretable entity representations generated by our method can facilitate the analysis of the behavior of embedding models.
- We provide evidence that our interpretable entity representations can be leveraged to explain similarities between entity embeddings.

5.2 Preliminaries

Knowledge Graphs. In this chapter, we define Knowledge Graphs (KGs) as a set of $\langle subject, predicate, object \rangle$ triples, such as $\langle bmw, produces, autos \rangle$, which can also be expressed as ground binary predicates in predicate logic, e.g., $produces(bmw, autos)$. A signature of a KG $\mathcal{G}$ is $\Sigma_{\mathcal{G}} = \langle \mathbf{R}, \mathbf{E} \rangle$, where $\mathbf{R}$ is a set of binary predicates, *i.e.*, relations and $\mathbf{E}$ is a set of constants, *i.e.*, entities, in the knowledge graph $\mathcal{G}$. Concepts can be built over KGs following the Description Logic (DL) [7] syntax. In this chapter we mainly rely on concepts described in table 5.1.

KG Embeddings. Deep learning methods (in particular, KG embeddings) have been proposed to perform different machine learning tasks on top of KGs, such as link prediction or entity classification. KG embeddings aim at representing all entities and relations in a continuous vector space, usually as vectors or matrices called *embeddings*. Embeddings can be used to estimate the likelihood of a triple being true via a scoring function: $f : \mathbf{E} \times \mathbf{R} \times \mathbf{E} \rightarrow \mathbb{R}$, or to classify entities [119, 75, 184, 177]. In this chapter, we only make use

of entity-based embedding representations leaving the exploitation of relation embeddings for future work. Thus, without loss of generality, we assume that an embedding model is given a function: $\mathcal{E} : \mathbf{E} \mapsto \mathbb{R}^d$, which maps entities from $\mathbf{E}$ to d-dimensional real vectors.

One of the earliest and most popular embeddings is *e.g.*, TransE [14], which embeds entities and relations as vectors and assumes $\mathbf{v_s} + \mathbf{v_r} \approx \mathbf{v_o}$ for true triples, where $\mathbf{v_s}, \mathbf{v_r}, \mathbf{v_o}$ are vector embeddings for subject s, relation r and object o, resp. The likelihood that the above assumption holds should be higher for triples in the KG than for those outside. The learning process is done by minimizing the error induced by the respective assumption given the considered loss function.

Several KG embeddings that are based on graph neural networks have also been proposed in the literature (e.g., [162]) and, to the best of our knowledge, currently, they demonstrate the state-of-the-art performance on link prediction and entity classification tasks [187, 46]. One of the prominent representatives of this group of embeddings is CompGCN [162], which leverages a variety of entity-relation composition operations from KG embedding techniques as well as several existing multi-relational graph convolutional neural network methods.

Feature Selection for Explainability of Machine Learning Models.

Feature selection is introduced as a critical step in building machine learning models, especially when dealing with high-dimensional datasets. It aims to reduce the number of features while maintaining the model's performance, allowing for faster computation. Feature selection methods are generally categorized into three types: filter methods, wrapper methods, and embedded methods [72].

1. **Filter methods** assess the importance of features using statistical measures that do not rely on a specific model, as depicted in Fig.5.2. While these methods are computationally efficient, they may overlook interactions between features [76]. An example of such methods is the Information Gain metric [79], which quantifies the amount of information a feature provides in reducing uncertainty in the prediction task.

2. **Wrapper methods** involve training a model multiple times with different subsets of features, using model performance as a metric for selecting the most relevant features, as shown in Fig.5.3. Though accurate, these methods are computationally expensive [76]. An example for a wrapper method algorithm is the sequential Forward selection methods [111] that starts with an empty feature set and iteratively adds features that improve model performance the most.

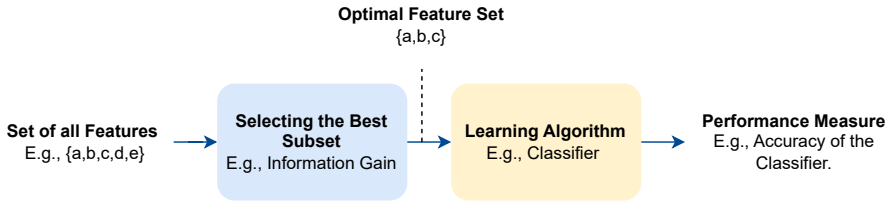


Figure 5.2: An example of a pipeline using filter-based feature selection method. The pipeline starts with a set of input features, where each feature is assigned a score based on the chosen selection method, such as information gain. Features with scores meeting or exceeding a predefined threshold are then selected as the most important. These selected features are subsequently used in downstream tasks, such as classification. The effectiveness of the model on these downstream tasks serves as an indicator of the quality of the selected features.

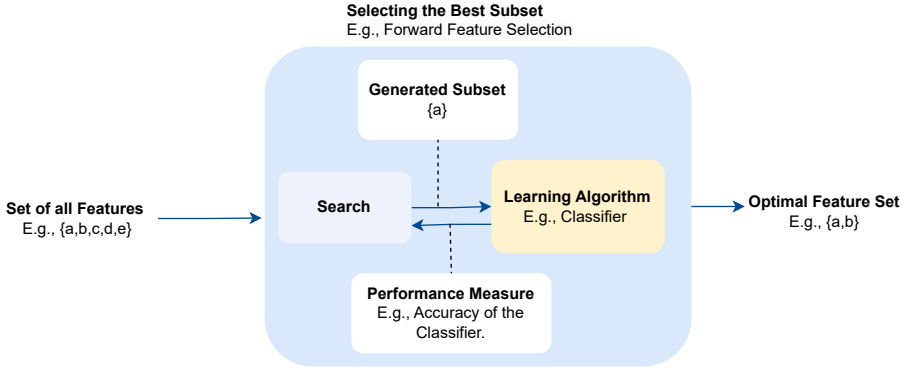


Figure 5.3: Wrapper method pipeline. In this approach, various subsets of features are evaluated by training a learning algorithm, and the subset that yields the best performance is selected as the optimal feature set.

3. **Embedded methods** perform feature selection as part of the model training process, offering a computationally efficient alternative to wrapper methods while still interacting with the learning algorithm [76], as illustrated in Fig.5.4. An example for that is random forest classifier [16] that compute feature importance score to measure the importance of each feature in the input based on how well the feature helps in splitting the data to different classes during the model training.

In models like decision trees and random forests [102, 16], which have built-in

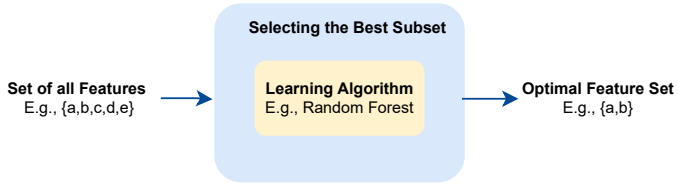


Figure 5.4: A representation of the embedded feature selection pipeline. In this approach, the learning model is designed to perform feature selection concurrently while optimizing for a specific task, such as image classification.

feature selection methods, the model is considered self explainable because the features used to make predictions are accessible. However, for most neural network-based models, this transparency is lacking, as the features influencing predictions are not directly available. Additional efforts are required to infer which features are involved in the decision-making process [8, 103, 43, 61]. In this chapter, we employed embedded feature selection technique to interpret KG embeddings for their computational efficiency.

5.3 Embedding-Guided Feature Construction

KG embedding models achieve promising results on different popular tasks; however, they are not interpretable. To address this issue, our work aims at providing insights into the effectiveness of pre-computed KG embedding models in capturing information in the KG by identifying the most important features in the training data for the embedding models and generating interpretable feature vector representations for KG entities. This can help in improving KG embedding models traditionally considered as black boxes.

More formally, given a KG $\mathcal{G}$ and an embedding model $\mathcal{E} : \mathbf{E} \mapsto \mathbb{R}^d$, which maps KG entities to numerical vectors, we aim at investigating the following questions: (a) Can we find a set of features F and an encoding $\mathcal{F}$ of entities using these features such that a function $\mathcal{W}$ exists that maps feature-based encoding $\mathcal{F}(e)$ of any entity $e \in \mathbf{E}$ to $\mathcal{E}(e)$? (b) If we use $\mathcal{F}(\mathbf{E})$ instead of $\mathcal{E}(\mathbf{E})$ as the input for downstream tasks, do we get similar results for these tasks?

In this section, we present a method for constructing $\mathcal{F}$ based on criterion (a). In Section 5.4, we assess its practical utility according to criterion (b), using entity and relation classification as downstream tasks.

The overview of our method for constructing an interpretable model based on a

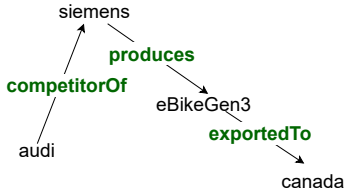


Figure 5.5: Fragment of a KG from fig. 5.1

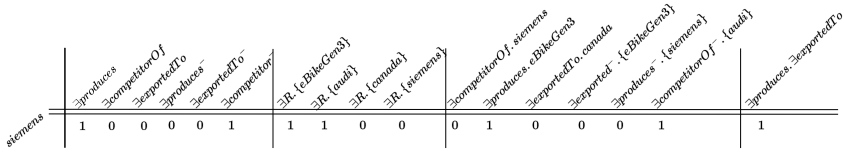


Figure 5.6: Feature vector for the entity *siemens* for KG from fig. 5.5

given KG embedding is presented in fig. 5.1. We take as input a knowledge graph and a pre-computed embedding model (e.g., TransE [14], CompGCN [162], or any other model). We then construct interpretable feature vectors for entities as described in Section 5.3.1. The obtained features are used for the embedding-guided feature selection to identify only those features that are important for approximating the given embedding model (see Section 5.3.2). The resulting features are then used to encode each KG entity and serve as embedding explanations, which are further evaluated in Section 5.4.

5.3.1 Feature Construction

In the first step of our method, we construct interpretable feature vector representations for KG entities relevant to a given embedding model. To achieve this, several proposionalization techniques [78, 84, 118, 26, 148] can be invoked, including methods that rely on rule learning [83, 98]. In this chapter, we incorporate four types of features. In addition to relations and relations with entities features proposed in INK [148], we introduce two new features: graph structural statistics (capturing patterns important for graph-based embedding models) and surrounding entities (necessary for embedding models lacking relations and paths with labeled edges, such as Snore [100]). To ensure that the features used in our analysis originated from the triples, on which the embedding models were initially trained, we focus on the above simple feature types. While more complex features extracted from rules or ontologies could be utilized, we choose features (represented in DL) that are more likely to be

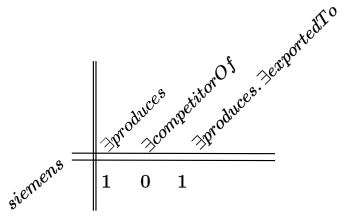


Figure 5.7: Feature vector for the entity *siemens* from fig. 5.6 after feature selection based on the important features in fig. 5.1.

learned by the embedding models, as they are used explicitly as part of the input during training.

Relations. The first feature type indicates whether an entity takes part in a given relation or not. For each relation r, there are features $\exists r.\top$ and $\exists r^-.\top$ that describe for each entity whether it has an outgoing or incoming relation with the respective label r. For example, for the entity *siemens* in fig. 5.5 we have the following set of *relation* features: $\exists produces.\top$, $\exists competitorOf^-.\top$.

Entities. The second group of features is formed by collecting for each entity e, the entities e' to which e is connected via relations with some label. In the Description Logic syntax, this amounts to $\exists R.\{e'\}$ and $\exists R^-.\{e'\}$. E.g., *siemens* in fig. 5.5 has $\exists R.\{eBikeGen3\}$ and $\exists R^-.\{audi\}$ among its *entity* features.

Relations with Entities. The third type of features indicates whether a given entity is connected to another entity via a certain relation. For each entity e we consider its outgoing or incoming relations along with the entities to which the given entity is connected. In the Description Logic syntax, this results in the features of the following form: $\exists r.\{e'\}$ and $\exists r^-.\{e'\}$. E.g., for *siemens* in fig. 5.5 we have the following features $\exists produces.\{eBikeGen3\}$, $\exists competitorOf^-.\{audi\}$.

Paths. Another type of features corresponds to paths with a predefined length k $\exists r_1.\exists r_2.\ldots.\exists r_k$. E.g., when considering the entity *siemens* and $k = 2$, a single path feature is generated: $\exists produces.\exists exportedTo$.

Statistics. Finally, we consider the number of outgoing and incoming relations r and r^- for each entity e. In the DL syntax, this amounts to the following constructors $\geq k.R \sqcap \leq k.R$ and $\geq k.R^- \sqcap \leq k.R^-$ respectively, for which (with some abuse of notation) we use the shortcuts $= k.R$ and $= k.R^-$ to save space. For instance, for *siemens* in fig. 5.5 we have $= 1.R$, and $= 1.R^-$, as there is 1

outgoing and 1 incoming relation for this entity. Similar to Statistics, additional information about the entities (e.g., textual or numerical attributes) can be considered as an extra entry in the feature vector.

Building on the work in [148], we compute all features including paths for a predefined depth k. We then collect all features of the described types into the set $F = \{f_1, \ldots, f_p\}$, where p is the total number of features extracted from the KG $\mathcal{G}$. Based on the knowledge graph and the respective set of features F, we form an interpretable Boolean vector $\mathbf{fv}_e = [f_0^e, f_1^e, \ldots, f_p^e]$ for each entity e, such that $|\mathbf{fv}_e| = |F| = p$ and $f_i^e = 1$ iff the feature f_i holds for the entity e in the KG $\mathcal{G}$, and $f_i^e = 0$ otherwise. E.g., the resulting representation for the entity *siemens* for a small KG fragment from fig. 5.5 is presented in fig. 5.6.

5.3.2 Feature Selection

The pre-constructed set of features generated as described in Section 5.3.1 encodes information about the entities based on their neighborhood. However, it is unclear which of these features are relevant to the embedding model. To identify the most relevant features in the KG for embedding reconstruction, we adopt an embedded feature selection technique inspired by [155]. This approach allows us to use the embeddings to guide us in identifying and scoring the crucial features in the KG needed to reconstruct the embeddings. For the step of embedded feature selection, various models (e.g., decision trees, MLP) providing feature scores can be in principle utilized. In our case, we use random regression forest due to its simplicity, interpretability and resistance to overfitting [130]. The model, denoted as M, is trained to reconstruct the corresponding embeddings from their feature-based entity representations.

The random regression forest constructs a separate regression tree for predicting each embedding dimension using a subset of the features in the input. The respective trees are then combined to make predictions for all the embedding dimensions at once by averaging the predictions made by each regression tree for each dimension. During training, the random forest regression model scores the input features based on their importance (a.k.a. feature importance score) for the reconstruction task. The features with the highest importance score are used to explain the info captured by the KG embedding.

Formally, let $F = \{f_1, \ldots, f_p\}$ be a set of features computed in the previous step, e.g., $\exists competitorOf, \exists produces$, etc. As mentioned in Section 5.3.1, each entity $e \in \mathbf{E}$ is represented as a Boolean vector of the form $\mathbf{fv}_e = [f_0^e, \ldots, f_p^e]$, where f_i^e reflects the presence or absence of the respective feature f_i for the entity e. For the training input to the random forest regression model for each entity $e \in \mathbf{E}$, we have its Boolean feature vector $\mathbf{fv}_e = [f_0^e, f_1^e, \ldots, f_p^e]$ with the label being the

KG elements	FB15K-237	DBpedia50K
Entities	14,541	49,900
Relations	237	654
Train triples	272,115	32,388
Validation triples	17,535	399
Test triples	20,466	10,969

Table 5.2: Knowledge graph data statistics. The train and test splits are used for training the respective embedding models

embedding of e, i.e., $\mathbf{v}_e \in \mathbb{R}^d$ computed by the target embedding model $\mathcal{E}$ which we aim at explaining. The random forest regression model [130] is used to find the mapping between the feature vectors and embeddings. Additionally, the model outputs the importance score for each feature, which is computed relying on the mean squared error on the respective predictive task of reconstructing entity embeddings from feature vector-based entity representations. Once the model M is trained, the scores assigned to the features are extracted, and only the features with scores that are greater than a certain threshold θ are deemed essential. In our experiments, we set the threshold for the feature selection parameter to be the mean of the feature importance scores, as commonly done in practice.

The features selected by our feature selection algorithm reflect the parts of the KG that the embedding model focuses on most. Therefore, we refer to them as model explanations. Next, we utilize the selected features to obtain instance-level explanations for each entity embedding. This is achieved by keeping only the selected features within each entity representation. For example, if we have a set of selected features for a KG embedding model (fig. 5.1), we obtain the corresponding feature vector-based representation for entity *siemens* (fig. 5.7).

5.4 Experiments

We evaluate the quality and usefulness of the interpretable feature-based entity representations computed by our method. To evaluate the quality, we compare the behavior of the original embedding-based entity representations and our interpretable entity representations on the two tasks: 1) the entity classification following [66], and 2) the relation classification, i.e., given a pair of entities, predict a relation that holds between the entities. Additionally, to demonstrate the usefulness of our method, we consider the entity similarity task and show

how our feature-based entity representations can be exploited to compute explanations for entities being close to each other in the embedding space.

5.4.1 Experimental Setup

Datasets. We experiment with two widely used knowledge graphs namely, Freebase15k-237 (a.k.a. FB15k-237) [158] and DBpedia50K [135]. table 5.2 shows the statistics of the used datasets.

Embedding Models. For embeddings, we consider (1) TransE [14] as one of the first and most popular models; (2) CompGCN [162] as one of the latest GNN-based models which achieves state-of-the-art results on the entity classification tasks; (3) NodePiece [46] as a shallow path-based embedding model; and (4) Snore [100] and INK [148] as embeddings that are interpretable by construction. Table 5.2 shows the statistics of the respective datasets.
Evaluation Tasks. We consider entity classification and relation classification as the tasks on which we evaluate whether our feature vectors encompass the same information as the corresponding embedding-based entity representations.

• **Feature Selection Evaluation.** To evaluate our feature selection module, we used our framework to reconstruct interpretable embeddings for entities. We chose two types of interpretable embedding models for this evaluation: 1) manually created interpretable embeddings, and 2) interpretable Boolean INK [148] embeddings.

Specifically, we first extracted the interpretable embedding for each node using INK. Then, we applied our method to generate the corresponding approximate representation. We computed the F1 score to measure the similarity between the most important features identified by our method and those used for constructing INK embeddings.

We conducted the same experiment with manually created interpretable embeddings. For the manually created interpretable embeddings, we represented each entity based solely on the entities to which it is connected.

• **Entity Classification.** Entity classification is concerned with the prediction of a label from a given set of labels for KG entities. The goal of this evaluation is to verify whether different classifiers when trained on feature vector-based entity representations and embedding-based entity representations, align with each other by making the same correct and incorrect predictions for the entity classification task. For the FB15K-237 KG, we consider the labels for entities from [66], and for the DBPedia50K KG, we extract the entity types as labels. Similar to [109], we have selected the following classifiers: (1) Random

forest [130]; (2) K-nearest neighbor classifier [32]; (3) Multi-layer perceptron (MLP) [54]. For all the models used in this chapter, we used the default parameter setup defined by the scikit-learn library [108] for the respective model.

We train the respective classifiers on the train set of labeled KG entity embeddings and the train set of our feature vector representations approximating the target embedding. We then use the respective classifiers to predict the labels for entities in the test set, pick the classifier that shows the best performance for the embedding-based entity classification, and use its predictions to compute the alignment score (weighted average F1 score) between the predictions made by the embedding-based and feature-vector based classifiers. The alignment score evaluates the consistency between predictions using embeddings and those using feature-vectors. Accounting for false positives and negatives, the F1 score quantifies the overall agreement. A perfect alignment score is 1, reflecting that the behavior of both embeddings and feature vectors is identical on a specific task.

Since, to the best of our knowledge, no previous works have targeted the problem of computing model explanations for KG embedding models, as a baseline, we use the feature vectors constructed by our method but without the feature selection step. If the feature selection step removes a large percentage of features, but this does not impact the weighted average F1 score, one can conclude that the removed features are indeed not important for the embedding model on the respective task.

• **Relation Classification.** In the relation classification task, we randomly select a set $\mathbf{R}'$ of 50 relations from the KG, resulting in 39,273 (resp. 24,832) entity pairs $E = \{(e_1, e_2) \,|\, \langle e_1, r, e_2\rangle, r \in R\}$ for FB15k-237 (resp. for DBpedia50k). We split the set E into the train (70 % of entity pairs) and the test (30 %) sets.

We proceed with evaluating the alignment between the embedding-based entity representations and the feature-vector-based entity representations in the same way as for the entity classification task. The only difference is that the task considered in this experiment is rather concerned with classifying pairs of entities from the test set into a class r from the set of 50 classes in $\mathbf{R}'$. Intuitively, the meaning of a pair being classified to a specific class r is that the relation r holds between the respective entities.

Applications. Our feature vector-based entity representations can be exploited for analyzing and comparing different embedding models. To this end, we report and examine the types of features in the model explanations computed by our method for different embeddings and datasets as well as present examples

	Mean squared error (MSE)	
Embedding model	DBPedia50K	FB15K-237
Random	55.70	54.76
INK	0.0001	0.0003
TransE	0.026	0.0257
CompGCN	0.007	0.005
NodePiece	0.016	0.035
Snore	0.138	0.076

Table 5.3: Mean squared error of the random forest regression model used to learn important features for the regeneration of the embeddings.

of selected features along with computed entity representations. Another application of our method is explainable entity similarity. Given the embeddings learned by an embedding model and their corresponding interpretable feature vectors **fv** computed by our method, one can generate explanations for the similarity between any pair of entities e_i and its neighbor e_j in the embedding space as the intersection between their respective feature vectors, i.e., $\mathbf{fv}_{e_i} \cap \mathbf{fv}_{e_j}$.

5.4.2 Experimental Results

Evaluation of Feature Selection. We report the mean squared error (MSE) of the random forest regression model, trained to regenerate the embeddings from feature vectors in table 5.3. As a baseline, we consider a simple model that outputs a vector of random numbers of size 50 (average size of the embeddings used) when given a feature vector as input. The respective baseline is referred to as Random in table 5.3. The low MSE values for all models apart from the baseline witness that the random forest regression model is capable of identifying a meaningful relationship between the input feature vectors and the corresponding KG embeddings. The model failed to find a connection between the input feature vectors and the randomly generated embeddings, leading to high MSE. Additionally, we also computed the alignment F1 score between the features selected by our feature selection algorithm and the original features present in the INK embeddings. For FB15k-237, the F1 score is 0.77 and for DBpedia50K it is 0.82, indicating that the majority of the selected features were indeed in the input embeddings. These results further confirm the effectiveness of the feature selection method in accurately retrieving the content of embeddings. In Figure 5.9 we present the top 10 most important features computed over DBpedia50K. We observe that all of the selected features are

Interpretable embedding model	Dataset	Alignment F1 score
Entity based embedding	FB15K-237	0.99
	DBpedia50K	0.99
INK	FB15K-237	0.77
	DBpedia50K	0.82

Table 5.4: The F1 score showing the similarity between the used features to create INK and the actual features selected by our framework.

indeed of type *entities*, which demonstrates that our feature selection mechanism has managed to capture well the information encoded in the input.

Additionally, we computed the F1 score for the features selected by the feature selection algorithm and the original features present in the embeddings. These results in Table. 5.4 further affirm the findings in the main paper regarding the effectiveness of the feature selection method in accurately retrieving the original embeddings' content.

Interpreted model	Dataset	Selected feature %	Entity-classification (F1)		Relation classification (F1)	
			Before selection	After selection	Before selection	After selection
INK [148]	FB15K-237	75.5	0.93	0.80	0.90	0.90
	DBpedia50k	50.5	0.73	0.78	0.97	0.96
TransE [14]	FB15k-237	19.7	0.84	0.84	0.73	0.78
	DBpedia50k	13.9	0.61	0.64	0.53	0.54
CompGCN [162]	FB15k-237	24.4	0.64	0.64	0.57	0.61
	DBpedia50k	31.8	0.52	0.69	0.63	0.65
NodePiece [46]	FB15k-237	20.0	0.68	0.68	0.47	0.51
	DBpedia50k	22.7	0.73	0.73	0.75	0.75
Snore [100]	FB15k-237	11.9	0.66	0.69	0.23	0.23
	DBpedia50k	2.5	0.59	0.59	0.50	0.51

Table 5.5: The alignment F1 score between embedding-based classifiers and our feature-vector-based classifiers (before and after the feature selection step) on the entity and relation classification tasks. The third column represents the percentage of the selected features out of the total number of features.

Entity and Relation Classification. In table 5.5, we report the weighted average F1 score for the alignment between the classification results computed by the respective embedding models and the feature vector-based entity representations before (columns 4 and 6) and after (columns 5 and 7) the feature selection step. We also report the percentage of all features left after the feature selection step (column 3). One can observe that the alignment scores reach 0.84 for entity classification and 0.78 for relation classification tasks, respectively.The alignment score for Snore on the relation classification task was low, which is attributed to the fact that its embeddings do not consider relations as mentioned in [100]. As a consequence, the performance of Snore on the relation classification task is rather poor, with the F1 score of 0.16. Our feature selection-based method also manages to reproduce this, which is witnessed by the results in fig. 5.9, where the top 10 features selected as the most important ones for Snore on DBPedia50K correspond to entities. Moreover, we found that this result is invariant to the value of θ, as the performance was poor even before feature selection. Furthermore, we observed that the alignment with INK is the highest, which can be attributed to the interpretable nature of the INK embeddings and the intersection of features used in our method with those used in INK. Based on the percentage of selected features, the results show that NodePiece uses less features than CompGCN but more than TransE. This is due to the facts that NodePiece retains small amount of information about explicit nodes without sacrificing its performance on down stream tasks.

table 5.5 shows that the feature selection discards up to 97.5% of the features without negative impact on the alignment F1 score. This indicates that the embeddings do not utilize all of the information in the neighborhood of entities.

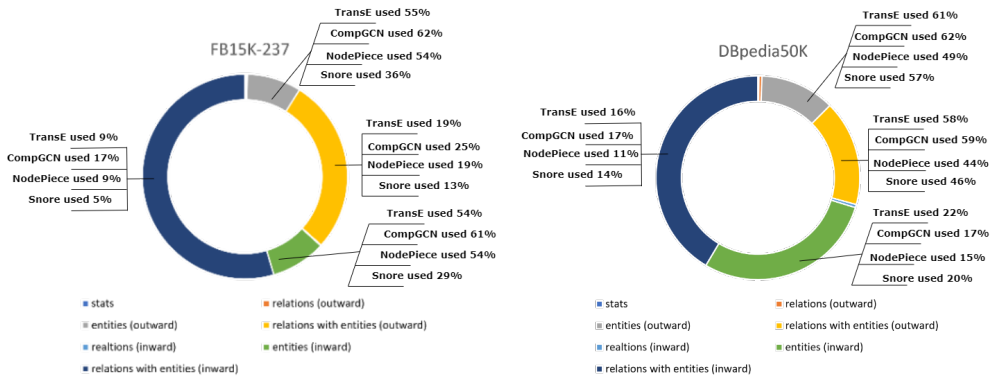


Figure 5.8: The percentage of each feature type out of the total number of features for FB15k-237 (left) and DBPedia50k (right). Labels reflect the percentage of features per feature type selected by our method for the considered models.

Model	Feature vector representation
Original (no feature selection)	$\exists team^-, \exists team^-.\{\text{Andrés Andrade}\}, \exists team^-.\{\text{Christian Montaño}\},$ $\exists team^-.\{\text{Daniel Tílger}\}, \exists team^-.\{\text{David Ferreira}\}, \exists team^-.\{\text{Diego Gómez}\},$ $\exists team^-.\{\text{Eduardo Ferreira}\}, \exists team^-.\{\text{Ernesto Farías}\},$ $\exists team^-.\{\text{Giovanni Hernández}\}, \exists team^-.\{\text{Hugo Sosa}\}, \exists team^-.\{\text{Jair Reinoso}\},$ $\exists team^-.\{\text{Javier Arizala}\}, \exists team^-.\{\text{John Córdoba}\},$ $\exists team^-.\{\text{John Harold Lozano}\}, \exists team^-.\{\text{Juan Camilo Angulo}\},$ $\exists team^-.\{\text{Leandro Castellanos}\},$ $\exists team^-.\{\text{Luis Barbat}\}, \exists team^-.\{\text{Luis Eduardo Zapata}\},$ $\exists team^-.\{\text{Luis Marcoleta}\}, \exists team^-.\{\text{Roberto Cabañas}\},$ $\exists team^-.\{\text{Wilson Morelo}\}, = 20.R^-, = 0.R,$ $\exists R^-.\{\text{Andrés Andrade}\}, \exists R^-.\{\text{Christian Montaño}\},$ $\exists R^-.\{\text{Daniel Tílger}\}, \exists R^-.\{\text{David Ferreira}\}, \exists R^-.\{\text{Diego Gómez}\},$ $\exists R^-.\{\text{Eduardo Ferreira}\}, \exists R^-.\{\text{Ernesto Farías}\}, \exists R^-.\{\text{Giovanni Hernández}\},$ $\exists R^-.\{\text{Hugo Sosa}\}, \exists R^-.\{\text{Jair Reinoso}\}, \exists R^-.\{\text{Javier Arizala}\},$ $\exists R^-.\{\text{John Córdoba}\}, \exists R^-.\{\text{John Harold Lozano}\},$ $\exists R^-.\{\text{Juan Camilo Angulo}\}, \exists R^-.\{\text{Leandro Castellanos}\}, \exists R^-.\{\text{Luis Barbat}\},$ $\exists R^-.\{\text{Luis Eduardo Zapata}\}, \exists R^-.\{\text{Luis Marcoleta}\},$ $\exists R^-.\{\text{Roberto Cabañas}\}, \exists R^-.\{\text{Wilson Morelo}\}$
TransE	$\exists team^-, = 0.R, \exists team^-.\{\text{Daniel Tílger}\}, \exists R^-.\{\text{John Córdoba}\}$
CompGCN	$\exists team^-.\{\text{Javier Arizala}\}, \exists team^-.\{\text{Luis Eduardo Zapata}\},$ $= 20.R, = 0.R^-,$ $\exists R^-.\{\text{Javier Arizala}\}, \exists R^-.\{\text{Luis Eduardo Zapata}\}, \exists R^-.\{\text{Luis Marcoleta}\}$
NodePiece	$\exists team^-.\{\text{Eduardo Ferreira}\}, \exists team^-, = 20.R^-, = 0.R,$ $\exists R^-.\{\text{Eduardo Ferreira}\}$
Snore	$\exists team^-, = 20.R^-, = 0.R$

Table 5.6: Feature vector of entity "*América de Cali Club*" from DBpedia50k KG computed by our method for different models.

KG Embedding Analysis. In fig. 5.8, we present detailed information regarding the contribution of each feature type to the total number of features before and after the feature selection step for the models TransE, CompGCN, NodePiece, and Snore on FB15k-237 and DBpedia50k. fig. 5.8 shows that even though inward relations with entities constitute more than 50% of all features, they are mostly ignored by our selection mechanism (at most 16% were selected for TransE and 17% for CompGCN across all considered datasets). This might

Model	Feature vector representation
Original (no model)	$\exists class, \exists class.\{gastropoda\}, \exists class^-, \exists class^-.\{dotidae\}, \exists class^-.\{favorinusTsuruganus\}, \exists family^-,$ $\exists family^-.\{capeSilvertipNudibranch\}, \exists order^-, \exists order^-.\{aeolidiella\}, \exists order^-.\{amandaGastropod\},$ $\exists order^-.\{anetarcaPiutaensis\}, \exists order^-.\{anteaecolidiellaCacaotica\}, \exists order^-.\{anteaecolidiellaOliviae\},$ $\exists order^-.\{berghiaStephanieae\}, \exists order^-.\{candelabraNudibranch\}, \exists order^-.\{eubranchusGlacialis\},$ $\exists order^-.\{eubranchusOccidentalis\}, \exists order^-.\{eubranchusRubeolus\}, \exists order^-.\{eubranchusRubropunctatus\},$ $\exists order^-.\{eubranchusVittatus\}, \exists order^-.\{eubranchusYolandae\}, \exists order^-.\{godivaGastropod\},$ $\exists order^-.\{phyllodesmiumCrypticum\}, \exists order^-.\{phyllodesmiumKarenae\}, \exists order^-.\{spurillaFaustina\},$ $= 1.R, = 20.R^-, \exists R.\{gastropoda\}, \exists R^-.\{aeolidiella\}, \exists R^-.\{amandaGastropod\}, \exists R^-.\{anetarcaPiutaensis\},$ $\exists R^-.\{anteaecolidiellaCacaotica\}, \exists R^-.\{anteaecolidiellaOliviae\}, \exists R^-.\{berghiaStephanieae\}, \exists R^-.\{candelabraNudibranch\},$ $\exists R^-.\{capeSilvertipNudibranch\}, \exists R^-.\{dotidae\}, \exists R^-.\{eubranchusGlacialis\}, \exists R^-.\{eubranchusOccidentalis\},$ $\exists R^-.\{eubranchusRubeolus\}, \exists R^-.\{eubranchusRubropunctatus\}, \exists R^-.\{eubranchusVittatus\},$ $\exists R^-.\{eubranchusYolandae\}, \exists R^-.\{favorinusTsuruganu\}, \exists R^-.\{godivaGastropod\},$ $\exists R^-.\{phyllodesmiumCrypticum\}, \exists R^-.\{phyllodesmiumKarenae\}, \exists R^-.\{spurillaFaustina\}$
TransE	$\exists class, \exists class.\{gastropoda\}, \exists family^-, \exists order^-, = 1.R, \exists order^-.\{spurillaFaustina\}, \exists R.\{gastropoda\}, \exists R^-.\{favorinusTsuruganu\}$
CompGCN	$\exists class, \exists class.\{gastropoda\}, \exists class^-, \exists family^-, \exists order^-, = 1.R, = 20.R^-, \exists R.\{gastropoda\}, \exists R^-.\{favorinusTsuruganu\}$
NodePiece	$\exists class, \exists class.\{gastropoda\}, \exists class^-, \exists family^-, \exists order^-, = 1.R, = 20.R^-, \exists R.\{gastropoda\}$
Snore	$\exists class, \exists class.\{gastropoda\}, = 1.R, \exists R.\{gastropoda\}, \exists R^-.\{eubranchusGlacialis\}$

Table 5.7: Feature vector-based representations of the biological entity '*Dexiarchia*' from DBpedia50k KG before and after feature selection for all of the considered models.

Model	Feature vector representation
Original (no model)	$\exists class, \exists class.\{insect\}, \exists family^-, \exists family^-.\{drepaninae\}, \exists family^-.\{habrona\},$ $\exists family^-.\{microblepsis\}, \exists family^-.\{negra\}, \exists family^-.\{urogoentities\}, \exists order,$ $\exists order.\{lepidoptera\}, = 2.R, = 5.R-, \exists R.\{insect\}, \exists R.\{lepidoptera\}, \exists R^-.\{habrona\},$ $\exists R^-.\{drepaninae\}, \exists R^-.\{microblepsis\}, \exists R^-.\{negra\}, \exists R^-.\{urogoentities\}$
TransE	$\exists class, \exists class\{insect\}, \exists family^-, \exists family^-.\{microblepsis\}, \exists family^-.\{urogoentities\},$ $\exists order, \exists order.\{lepidoptera\}, = 2.R, = 5.R-, \exists R.\{lepidoptera\}, \exists R^-.\{urogoentities\}$
CompGCN	$\exists class, \exists class\{insect\}, \exists family^-, \exists family^-.\{microblepsis\}, \exists family^-.\{urogoentities\}, \exists family^-.\{negra\},$ $\exists order, \exists order.\{lepidoptera\}, = 2.R, = 5.R^-, \exists R.\{insect\}, \exists R.\{lepidoptera\}, \exists R^-.\{urogoentities\}$
NodePiece	$\exists class, \exists class\{insect\}, \exists family^-, \exists order, \exists order.\{lepidoptera\}, = 2.R, = 5.R^-, \exists R.\{insect\}, \exists R.\{lepidoptera\}$
Snore	$\exists class, \exists class\{insect\}, \exists order, \exists order.\{lepidoptera\}, = 2.R, = 5.R^-, \exists R.\{insect\}, \exists R.\{lepidoptera\}$

Table 5.8: Feature vector-based representation of the biological entity "*Drepanidae*" from DBpedia50k KG before and after feature selection for all of the considered models.

Model	Feature vector representation
Original (no model)	$\exists broadcastArea, \exists broadcastArea.\{SouthKorea)\}, \exists language, \exists language.\{EnglishLanguage\}, \exists sisterStation, \exists sisterStation.\{FoxFilipino\}, \exists sisterStation.\{NatGeoPeople\}, \exists sisterStation.\{NatGeoWild\}, \exists sisterStation.\{StarSports\}, \exists sisterStation.\{StarWorld\}, = 7.R, = 0.R^{-}, \exists R.\{SouthKorea)\}, \exists R.\{EnglishLanguage\}, \exists R.\{FoxFilipino\}, \exists R.\{NatGeoPeople\}, \exists R.\{NatGeoWild\}, \exists R.\{StarSports\}, \exists R.\{StarWorld\}$
TransE	$\exists broadcastArea, \exists broadcastArea.\{SouthKorea)\}, \exists language, \exists language.\{EnglishLanguage\}, \exists sisterStation, = 0.R^{-}, \exists R.\{SouthKorea)\}, \exists R.\{EnglishLanguage\}, \exists R.\{NatGeoPeople\}$
CompGCN	$\exists broadcastArea, \exists language, \exists language.\{EnglishLanguage\}, \exists sisterStation, \exists sisterStation.\{FoxFilipino\}, \exists sisterStation.\{NatGeoPeople\}, \exists sisterStation.\{NatGeoWild\}, = 7.R, = 0.R^{-}, \exists R.\{SouthKorea)\}, \exists R.\{EnglishLanguage\}, \exists R.\{FoxFilipino\}, \exists R.\{NatGeoPeople\}, \exists R.\{NatGeoWild\}$
NodePiece	$\exists broadcastArea, \exists broadcastArea.\{SouthKorea)\}, \exists language, \exists language.\{EnglishLanguage\}, \exists sisterStation, \exists sisterStation.\{NatGeoPeople\}, \exists R.\{EnglishLanguage\}, \exists R.\{FoxFilipino\}, \exists R.\{NatGeoPeople\}, \exists R.\{NatGeoWild\}$
Snore	$\exists broadcastArea, \exists language, \exists language.\{EnglishLanguage\}, \exists sisterStation, \exists sisterStation.\{NatGeoPeople\}, \exists sisterStation.\{NatGeoWild\}, = 7.R, = 0.R^{-}, \exists R.\{SouthKorea)\}, \exists R.\{EnglishLanguage\}, \exists R.\{FoxFilipino\}, \exists R.\{NatGeoPeople\}, \exists R.\{NatGeoWild\},$

Table 5.9: Feature vector of the entity '*Fox Channel (Asia)*' from DBpedia50k KG before feature selection (Original) and after feature selection computed by our method TransE, CompGCN, NodePiece, and Snore models.

Model	Feature vector representation
Original (no model)	$\exists currency, \exists currancy\{usDollar\}, \exists location, \exists location.\{fortWorth\}, \exists placeOfBirth, \exists placeOfBirth.\{alexandria\}, \exists profession, \exists profession.\{tvProducer\}, \exists program, \exists program.\{mightyMorphinPowerRangers\}, \exists programCreator^-, \exists programCreator^-.\{mightyMorphinPowerRangers\}, \exists programCreator^-.\{powerRangers\}, = 6.R, = 2.R^-, \exists R.\{alexandria\}, \exists R.\{fortWorth\}, \exists R.\{mightyMorphinPowerRangers\}, \exists R.\{usDollar\}, \exists R.\{marriage\}, \exists R.\{tvProducer\}, \exists R^-.\{mightyMorphinPowerRangers\}, \exists R^-.\{powerRangers\}, \exists typeOfUnion, \exists typeOfUnion.\{marriage\}$
TransE	$\exists currency, \exists currancy\{usDollar\}, \exists location, \exists location.\{fortWorth\}, \exists placeOfBirth, \exists profession, \exists profession.\{tvProducer\}, = 6.R, = 2.R^-, \exists R.\{alexandria\}, \exists R.\{fortWorth\}, \exists R.\{usDollar\}, \exists R.\{marriage\}, \exists R.\{tvProducer\}, , \exists R^-.\{mightyMorphinPowerRangers\}, \exists typeOfUnion, \exists typeOfUnion.\{marriage\}$
CompGCN	$\exists currency, \exists currancy\{usDollar\}, \exists location, \exists location.\{fortWorth\}, \exists placeOfBirth, \exists placeOfBirth.\{alexandria\}, \exists profession, \exists profession.\{tvProducer\}, \exists program, \exists programCreator^-, \exists programCreator^-.\{mightyMorphinPowerRangers\}, \exists programCreator^-.\{powerRangers\}, = 6.R, = 2.R^-, \exists R.\{fortWorth\}, \exists R.\{usDollar\}, \exists R.\{marriage\}, \exists R.\{tvProducer\} , \exists R^-.\{mightyMorphinPowerRangers\}, \exists typeOfUnion, \exists typeOfUnion.\{marriage\}$
NodePiece	$\exists currency, \exists currancy\{usDollar\}, \exists location, \exists location.\{fortWorth\}, \exists placeOfBirth, \exists profession, \exists profession.\{tvProducer\}, \exists program, \exists programCreator^-, = 6.R, = 2.R^-, \exists R.\{fortWorth\}, \exists R.\{usDollar\}, \exists R.\{marriage\}, \exists R.\{tvProducer\}, \exists R^-.\{mightyMorphinPowerRangers\}, \exists typeOfUnion, \exists typeOfUnion.\{marriage\}$
Snore	$\exists currency, \exists currancy\{usDollar\}, \exists location, \exists location.\{fortWorth\}, \exists placeOfBirth, \exists profession, \exists profession.\{tvProducer\}, \exists program, \exists programCreator^-, = 6.R, = 2.R^-, \exists R.\{fortWorth\}, \exists R.\{mightyMorphinPowerRangers\}, \exists R.\{usDollar\}, \exists R.\{marriage\}, \exists R.\{tvProducer\}, \exists R^-.\{mightyMorphinPowerRangers\}, \exists typeOfUnion, \exists typeOfUnion.\{marriage\}$

Table 5.10: Feature vector of the entity "*haimSaban*" from FB15K-237 KG before and after feature selection for all models.

FB15K-237		**DBpedia50K**	
TransE	**CompGCN**	**Snore**	**NodePiece**
$\exists profession$	$= 0.R^-$	$\exists genre$	$= 1.R$
$\exists film^-$	$= 1.R^-$	$\exists R.\{germany\}$	$\exists team$
$\exists award^-$	$= 2.R^-$	$\exists R.\{london\}$	$\exists musicalArtist^-$
$\exists genre$	$= 3.R^-$	$= 0.R$	$\exists birthplace$
$\exists artists^-$	$= 2.R$	$\exists R.\{drumKit\}$	$= 1.R^-$
$\exists institution^-$	$= 1.R$	$\exists recordLabel$	$\exists staring$
$\exists role^-$	$= 4.R$	$\exists R.\{iran\}$	$\exists genre$
$= 0.R^-$	$= 0.R$	$\exists R.\{insect\}$	$\exists team^-$
$\exists team^-$	$\exists placeOfBirth$	$\exists R.\{southKorea\}$	$\exists producer$
$= 1.R^-$	$= 3.R$	$\exists kingdom.\{plant\}$	$\exists writer$

Figure 5.9: The top 10 most important features identified by our method for FB15K-237 and DBpedia50k, sorted in descending order of their importance scores.

indicate that the embedding models learn to represent each entity in terms of outward (paths of) relations and entities rather than inward ones. This behavior was consistent over all four knowledge graph embedding models used.

n^{th} Neighbor	Neighbor	Similarity Explanation
1^{st}	Katy Perry	$\exists artists^{-}.\{dancepop\}, \exists artists^{-}.\{popmusic\}$, $\exists award.\{GrammyAwardforBestFemalePopVocalPerformance\}$, $\exists award.\{MTVVideoMusicAwardoftheYear\}$ $\exists award.\{MTVVideoMusicAwardforBestNewArtist\}$, $\exists award.\{(MTVVideoMusicAwardForBestFemaleVideo\}$, $\exists netWorthCurrency.\{USdollar\}, \exists gender.\{femaleorganism\}$, $\exists profession.\{actor\}, \exists vacationer^{-}, \exists participant^{-}$, $\exists awardWinner^{-}, \exists person^{-}, \exists artist^{-}, \exists award$, $\exists specialPerformanceType, \exists awardNominee, \exists profession$, $\exists origin, \exists film, \exists participant, \exists netWorthCurrency$, $\exists religion, \exists nationality, \exists R^{-}.\{poprock\}$, $\exists R^{-}.\{popmusic\}, \exists R^{-}.\{dancepop\}, = 43.R^{-}$, $\exists R.\{MTVVideoMusicAwardforVideooftheYear\}$, $\exists R.\{MTVVideoMusicAwardforBestFemaleVideo\}$, $\exists R.\{MTVVideoMusicAwardforBestNewArtist\}$, $\exists R.\{MTVVideoMusicAwardforBestPopVideo\}$, $\exists R.\{MTVVideoMusicAwardforBestArtDirection\}$, $\exists R.\{GrammyAwardforBestFemalePopVocalPerformance\}$, $\exists R.\{femaleorganism\}, \exists R.\{actor\}, \exists R.\{USdollar\}, \exists R.\{friend\}$,
100^{th}	Ice Cube	$\exists profession.\{recordproducer\}, \exists languages.\{English\}$, $\exists netWorthCurrency.\{USdollar\}, \exists artists^{-}$, $\exists awardNominee^{-}, \exists awardWinner^{-}, \exists profession$, $\exists film, \exists gender, \exists languages, \exists award, \exists netWorthCurrency$, $\exists nationality, \exists religion, \exists location, \exists awardNominee$, $\exists awardWinner, \exists R.\{USdollar\}, \exists R.\{recordproducer\}, \exists R.\{English\}$

Table 5.11: The list of explanations for the similarity of "*Christina Aguilera*" to its nearest neighbours. The entity is from FB15K-237 and its 1^{st} and 100^{th} nearest neighbors are based on TransE embeddings.

n^{th} Neighbor	Neighbor	Similarity Explanation
1^{st}	Pseudocossus	$\exists order.\{Lepidoptera\}, \exists class.\{Insect\}, \exists class, \exists order,$ $\exists R.\{Insect\}, \exists R.\{Lepidoptera\}, = 2.R^{-}, = 1.R$
100^{th}	Loxostegopsis	$\exists R.\{Insect\}, = 2.R^{-}, = 1.R, \exists class, \exists class.\{Insect\}$

Table 5.12: The explanations of the similarity between entity "*Noduliferola*" from DBpedia50K and its 1^{st} and 100^{th} nearest neighbors based on NodePiece embeddings.

We can observe that the portion of features filtered out by our selection method within each feature type is significant, reaching more than 60% for some feature types. Our feature selection technique filtered out more features for Snore and NodePiece models than for other models. The selected feature types are consistent with the methods for constructing the respective embeddings [100, 46]. For the CompGCN model the majority of the features were retained. In fig. 5.9, we also present the top 10 features selected by our framework for FB15k-237 on TransE and CompGCN. One can notice that CompGCN focuses more on the information about the number of inward and outward relations and entities with large numbers of incoming and outgoing relations, which aligns with the limitations of GNNs in [170]. A similar behavior has been observed for DBpedia50k.
The model explanations can also be used to analyze the behavior of the embedding models throughout training phases. For example, Figures 5.10, 5.11 shows how the attention of the TransE shifts from focusing on entities and relations with entities to relations only. This information can help in determining when the training should be stopped, e.g., based on a given use case for embeddings.

Example Feature Vectors. In table 5.6 and 5.9, we present example feature-based representations computed by our method for entities "*América de Cali Club*" and "*Fox Channel (Asia)*" from DBPedia50K KG. For these entities the CompGCN embedding model tends to capture more inward relations with entities than other models. For TransE the majority of the selected features are relations with entities, which aligns well with its limitation to capture one-to-many relations [166].

Although CompGCN better grasps inward relations with entities compared to other models, it fails to capture all the entities with dense neighborhood, i.e., entities having numerous inward relations with the same label. We observed this behavior for the entity "*América de Cali Club*", which has 20 inward relations with entities labeled with the same relation type (see table 5.6). The same holds for the outward relations with entities for "*Fox Channel (Asia)*" in table 5.9.

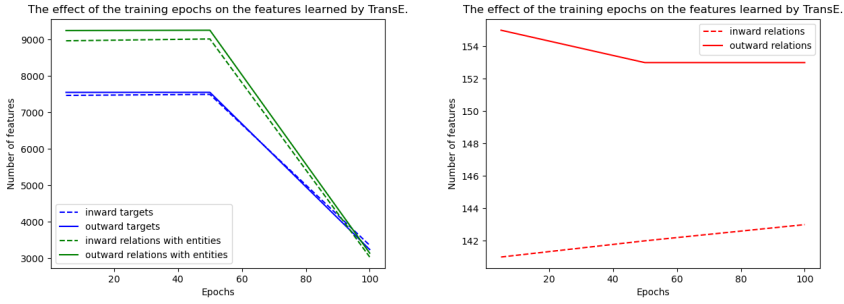


Figure 5.10: Number of features of each feature type after feature selection for TransE during training after 5, 50, and 100 epochs. The plots show that TransE shifts its attention during learning from entities to relations.

$\exists awardWinner^-$	$\exists nationality$	$\exists profession$
$\exists gender.\{maleOrganism\}$	$\exists award$	$\exists film^-$
$\exists nationality$	$= 3.R$	$\exists award^-$
$\exists language.\{English\}$	$\exists gender$	$\exists genre$
$\exists nominatedFor^-$	$\exists R.\{USA\}$	$\exists artists^-$
$\exists gender$	$\exists country$	$\exists institution^-$
$= 3.R$	$\exists R.\{actor\}$	$\exists role^-$
$\exists award$	$= 2.R$	$= 0.R^-$
$\exists typeOfUnion.\{marriage\}$	$\exists location$	$\exists team^-$
$= 2.R$	$\exists profession.\{actor\}$	$= 1.R^-$

Figure 5.11: Top 10 features sorted in descending order based on their importance score for FB15K-237 for TransE after training for 5, 50, and 100 epochs.

More examples in tables 5.7, 5.8 and 5.10.

After conducting the feature selection process, we observed an average of around 29K-19K selected features for FB15K-236 and DBPedia50k, respectively. Despite the high average, individual entities typically possess a limited number of features in their representations, 6-54 features per entity on average for FB15K-236 and DBPedia50k, respectively. Hence, feature representations after feature selection are easily comprehensible by humans, making them suitable for manual examination as shown in tables 5.13 and 5.14

Explainable Entity Similarity. Our explanations for embedding models can be utilized to explain similarity between entities. To exemplify this application,

Model	FB15K-237	DBpedia50K
Original	156,599	70,738
TransE	30,898	21,842
CompGCN	38,196	22,276
NodePiece	31,327	16,053
Snore	18,693	18,693

Table 5.13: The overall number of features before (Original) and after the feature selection per dataset and per model.

Model	FB15K-237	DBpedia50K
Original	81	26
TransE	56	7
CompGCN	59	7
NodePiece	58	7
Snore	46	6

Table 5.14: The average number of relevant features per entity before (original) and after feature selection for each embedding model and dataset

in table 5.11 we report the closest and the 100^{th} closest entity to "*Christina Aguilera*" based on the cosine similarity of the respective vectors for TransE model on the FB15K-237 dataset along with the features computed by our method that the respective entities share. The results reveal that "*Christina Aguilera*" shares more features with its closest neighbor "*Katy Perry*" than with "*Ice Cube*", which explains why the respective entities are closer in the embedding space. We provide further results for the entity similarity application in table 5.12 These results show insights into the embedding space learned by the embedding models, allowing for a better understanding of possible reasons for the respective positions of entities in the embedding space.

Analysis. In Table 5.13 we present the absolute number of features before (original) and after the feature selection step for each model. For all models the number of features has been reduced by more than four times for FB15K-237 and by more than three times for DBpedia50K. While the number of features after the feature selection is still rather high for each model, in reality, every entity has only a small fraction of the respective features. We present the average length of vectors for all models and datasets in Table 5.14.

The results in Table. 5.13 are similar to the observed trend in Figure 5.8 Where the percentage of selected features per feature type in the interpretable vectors computed by our method for the considered models is represented.

Among all models, Snore has captured the least number of features per feature type for FB15K-237. On the DBpedia50K dataset, NodePiece has the lowest number of selected features. By construction, NodePiece and Snore exploit the information in the entity neighborhood only partially. While Snore samples random walks, NodePiece encodes each entity using a set of selected entities only. This might explain the observed behavior. For the GNN-based model, CompGCN our method has selected the highest percentage of features per feature type compared to other models.

5.5 Related Work

Explanations of KG Embedding Models. In recent years, a variety of KG embeddings have been proposed, e.g., [107, 14, 46, 119, 162] (see [166] for an overview). While some of the methods achieve state-of-the-art performance on certain tasks [187, 46, 162], the models often remain to be black boxes. This has raised interest in explaining KG embeddings and led to works targeting outcome explanations for embeddings on the link prediction task [120, 20, 10, 104, 52, 45]. In [120] a Kelpie framework has been introduced, which explains predictions made by a given embedding by identifying the combinations of training facts that have enabled the respective predictions. The works [10, 52] exploit rule learning techniques for identifying triples which are logical explanations for a particular prediction. Generation of post-hoc explanation for triples inferred by a (factorization-based) embedding model has been considered in [104]. This method first augments the underlying KG by introducing weighted edges between entities relying on their embedding-based similarity and then computes human-understandable explanations in the form of paths. All of the above methods focus on explaining the *outcome* of an embedding model rather than generating interpretable feature-based representations of entities that would mimic the behavior of embeddings on downstream tasks as we do. The recent work [46] is aligned with our idea of representing entities using a fixed-size entity vocabulary, but the respective model targets a different task of making embedding models space efficient, and in contrast to our entity representations, the representations generated by NodePiece are not interpretable.

In contrast, the works [100, 148] generate KG embeddings that are interpretable by construction. Their feature-driven entity encodings are similar to ours, but these methods do not approximate existing black box models as we do.

Interpretable Propositionalization-based Embeddings. Propositionalization, the task of constructing table-based representations, has been proposed for relational data [78]. It has also been studied in the context of knowledge

graphs [148, 119, 110, 118, 84]. While these existing propositionalization methods are interpretable by nature, they were developed and used as an alternative to black box KG embedding models, rather than for explaining existing KGE embedding models. Thus, these methods complement our work, and any interpretable propositionalization method can be incorporated during the feature construction step of our approach. To the best of our knowledge, none of the existing propositionalization methods targeted the problem of approximating the behavior of a given embedding model, which is our main focus.

5.6 Conclusion

Knowledge Graph embedding models are widely used for various tasks, but their numeric vector representations are not interpretable. Even with KG embedding models that create embeddings from interpretable features like NodePiece [46] and Snore [100], an extra embedding layer that is not reversible is used to turn the feature vectors into numerical vectors that are not interpretable. To address this issue, we have presented a feature selection-based approach to explain the behavior of knowledge graph embedding models. The results demonstrate that the proposed approach is effective in identifying the most important features for a given embedding model by finding the alignment between feature-based entity representations and the embedding-based entity representations. The findings reveal that KG embeddings do not capture all the information in the neighborhood of entities and that the feature selection process can discard up to 86% of the features without sacrificing the F1 alignment score. Furthermore, our method can be used to explain the similarities of entities in the embedding space. We believe that our work offers interesting perspectives for debugging embedding models and makes a further step towards revealing relations between embeddings and propositionalization methods following [84].

While the presented framework provides insights into the behavior of embeddings, it still could be further extended by accounting for embeddings of relations along with entity embeddings, more complex features like longer paths or trees as well as textual and numerical attributes. Despite the advantages of propositionalization for interpretable feature vector construction, it also has limitations. The size of the feature vector naturally scales with the knowledge graph size. Additionally, for larger KGs, the search space for the feature selection algorithm increases, which may lead to scalability issues. Thus, for ongoing and future work, we plan to integrate embeddings of relations into our method and exploit the information coming from the embeddings already at the feature generation stage. This can be achieved by relying on frequent

pattern mining methods or rule learning approaches [83, 98], especially those that already account for embeddings during rule construction [59, 181]. We also plan to identify ways for concise representation of feature vectors to adapt for large KGs.

In chapter 7 we will show how to utilize relevant information from knowledge graphs when the training dataset is not the knowledge graph.

Chapter 6

Look beyond the Surface: A Demo for Explaining Knowledge Graph Embeddings and Entity Similarity

This chapter introduces a demo for FeaBI that allows users to conveniently exploit the respective framework for computing embedding-based similarity between KG entities as well as generating and visualizing explanations for the respective similarity. This chapter is based on a demo developed by Huu Tan Mai during his internship at Bosch, under my supervision, and presented in the following paper:

Huu Tan Mai, ***Youmna Ismaeil****, Trung-Kien Tran, Hendrik Blockeel, and Daria Stepanova.* Look beyond the Surface: A Demo for Explaining Knowledge Graph Embeddings and Entity Similarity. In *ISWC (Posters/Demos/Industry). 2023.*

6.1 Introduction

Knowledge Graph embeddings (KGEs) (see, e.g., [166]) represent entities and relations in a low-dimensional vector space. They have been useful in a range

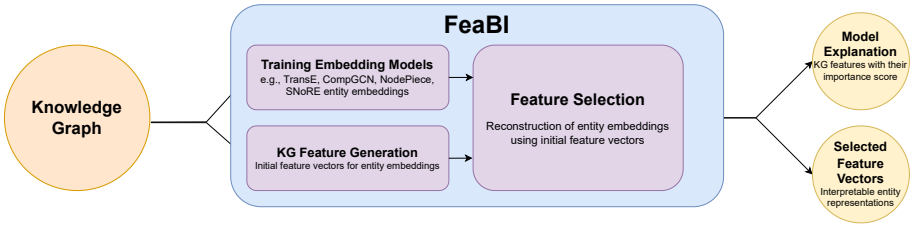


Figure 6.1: Overview of the Feature Selection-Based Framework

of tasks, including link prediction (e.g., [14, 160, 173]), entity classification (e.g., [119, 75]) or entity similarity. However, despite their success, KG embeddings are often regarded as black boxes. Lack of transparency and interpretability of KGEs limits users' understanding of their inner mechanisms, and undermines the trust in these models. E.g., given an entity, embedding-based suggestions regarding other entities similar to it might be less convincing if the user cannot examine the reasons behind the similarities.

Recently, a framework named FeaBI [64] has been proposed for explaining pre-computed entity embeddings. More specifically, given a KG and its embedding model FeaBI employs embedded feature selection techniques to extract from the KG propositional features in the form of relations and entities that are important for a given KG embedding model. These features are treated as KG embedding model explanations. FeaBI can be conveniently used for explaining similarities between entities.

In this chapter, we present a demo for FeaBI, which offers a user-friendly graphical interface to facilitate user experience in exploring the computed explanations. This is achieved through the visualizations of the relevant graph-based entity surroundings and textual descriptions of explanations. The demo can be used for the analysis and comparison of different embedding models in terms of KG features that they capture, thus supporting the users in deciding which embedding model would suit their purpose best.

The recorded video of our demo is available at `https://figshare.com/s/b941c7e0c800c23f5d82`.

6.2 Demo Overview

Figure 6.2 presents an overview of our demo which is designed as a web application. In what follows, we describe the server and the frontend components

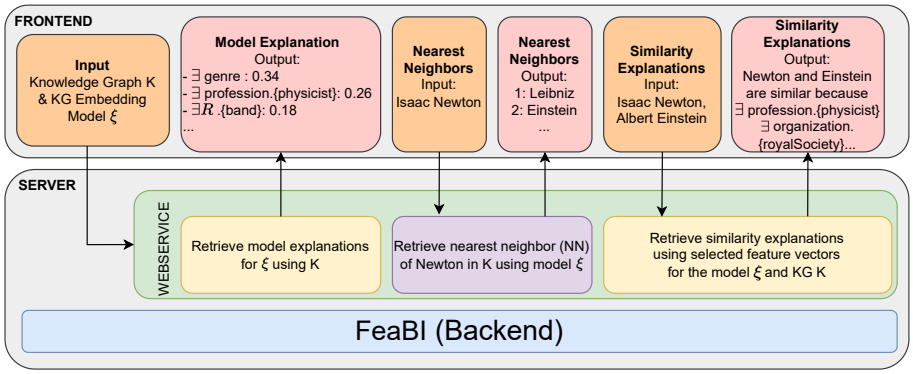


Figure 6.2: Structure of the Demo for the Framework (local setup)

KG	Model	FeaBI total runtime (s)
DBpedia50	TransE	10.39 ± 0.37s
	CompGCN	9.28 ± 0.31s
	NodePiece	9.96 ± 0.19s
	SNoRe	16.94 ± 0.20s
FB15k237	TransE	30.89 ± 2.02s
	CompGCN	26.45 ± 1.0s
	NodePiece	30.53 ± 0.82s
	SNoRe	33.01 ± 0.68s

Table 6.1: The runtime of feature construction and feature selection steps of FeaBI

of the demo in details.

The sever side is divided into two parts: 1) the backend (based on FeaBI [64]), which given a KG and its embedding model computes the explanations, and 2) the webservice, which communicates the KG and the embedding model chosen by the user to FeaBI and presents the received results to the user via the frontend.

Feabi (Backend). For a given KG and its embedding model, FeaBI computes KG embedding explanations defined as a list of KG features ranked based on their importance for the generation of the KG embedding. The top most important features are then used to build interpretable representations of the KG entity embeddings. The main components of FeaBI are KG embedding training, feature construction and feature selection (see [64] for details). The training of the KG

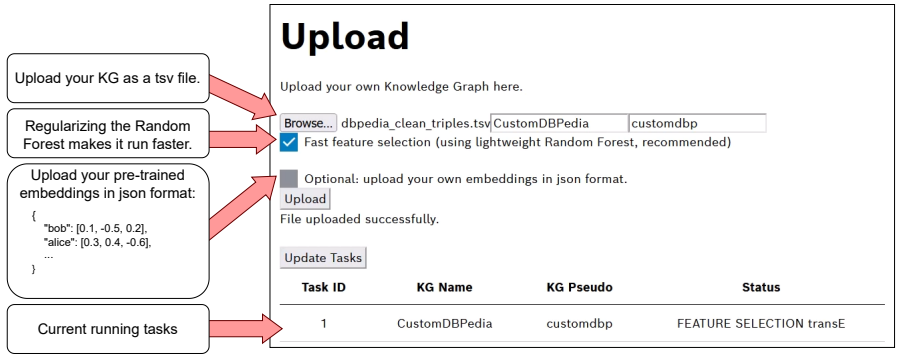


Figure 6.3: Illustration for different customizations, e.g., uploading a custom KG to the demo

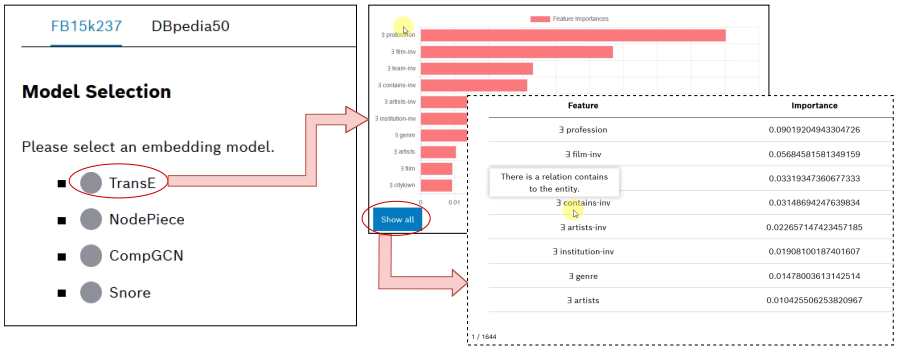


Figure 6.4: Model explanations as a ranked list of selected KG features

embedding model is naturally the most time-consuming step, which typically takes up to 5 hours (e.g., for CompGCN on FB15K237 dataset). Therefore, in our demo we provide a number of pre-trained embedding models. At the moment we support 4 popular embedding models: TransE [14], CompGCN [162], NodePiece [46] and SNoRe [100], but other pretrained embeddings can also be provided by users as illustrated in Figure 6.3.

Table 6.1 shows the running time of the feature construction and feature selection steps of FeaBI for two popular KGs and embedding models available in the demo.

Webservice. The webservice handles the communication of FeaBI with the frontend. In the frontend, the KG and KG embedding models are first selected

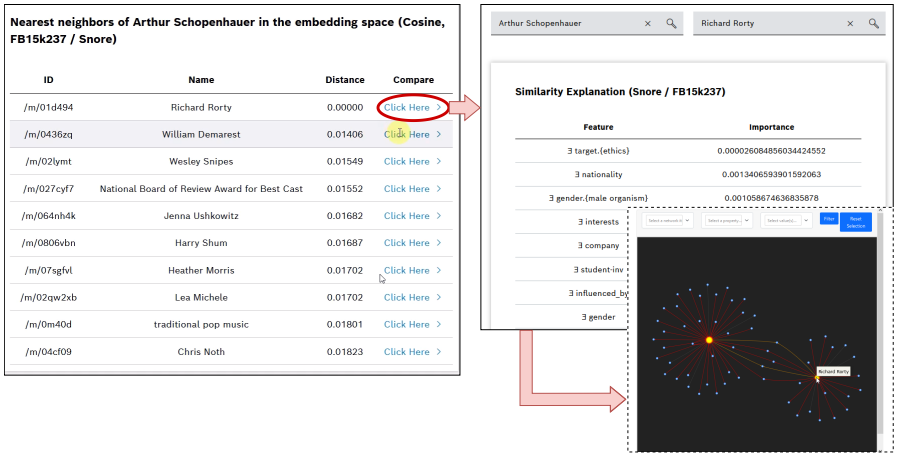


Figure 6.5: Similarity explanations and their visualization

by the user, and then passed to FeaBI via the webservice. Subsequently, FeaBI computes the results, which are then sent to the webservice and presented to the user via the frontend.

Frontend. The frontend allows users to conveniently explore the model explanations for a given embedding model, entity embedding explanations, as well as explanations for similarities between a pair of selected entities retrieved by the webservice.

The workflow of the demo proceeds as follows. First, the user selects a KG and an embedding model from the provided list (or uploads custom ones) via the visual interface. Then, a model explanation (i.e., a list of symbolic features ranked by their importance) is automatically generated and presented to the user (see Figure 6.4).

Additionally, the demo offers a possibility to compare entities in the KG in terms of their similarity relying on the given embedding model. As shown in Figure 6.5, for a given entity provided by the user, similar entities can be retrieved based on the distance metric in the embedding space (cosine similarity and Euclidean distance are currently supported). The user can select any pair of entities and use the system to generate explanations for their similarity, i.e., a list of selected KG features that the entities share along with their graph-based visualizations.

6.3 Conclusion

We presented a demo for FeaBI [64], which is a recently proposed framework for explaining KG embedding models. While the work in [64] focuses on technical details of the method, our demo system allows the users to easily analyse KG features captured by an embedding model as well as reasons behind embedding-based entity similarities. Open directions include the analysis of explanations for relation embeddings as well as the consideration of ontologies and KG schemes within the studied framework.

Chapter 7

Beyond Manual Labels: Unsupervised Graph-Based Explanations for Error Analysis in Image Classifiers

In this chapter, we introduce a dynamic, unsupervised framework that leverages commonsense knowledge and open-source foundation models to uncover and interpret patterns learned by image classifiers. To achieve this, we developed a method to represent images as graphs and extract distinct sub-graph patterns for both correctly and incorrectly classified images. These patterns reveal how combinations of image features influence classifier decisions, offering deep insights into the model's behavior. We evaluate our framework's explanations by constructing an image classifier using the extracted patterns and measuring how closely its predictions align with those of the original image classifier. Additionally, we provided use cases for the extracted patterns for error analysis and proactive bias detection in datasets. This chapter is based on the following publication:

Youmna Ismaeil, *Jan Hednrik Metzen, Trung-Kien Tran, Daria Stepanova, and Hendrik Blockeel*. Breaking the Mold: Dynamic Pattern Discovery in Image Classifiers with Foundation Models and Knowledge Graphs. Under review *The* 24^{th} *International Semantic Web Conference (ISWC). 2025.*

7.1 Introduction

Neural networks achieve impressive classification performance through high-quality training data, but their ability to generalize weakens when the data is biased. For instance, a classifier trained on classrooms with green desks may fail to generalize to other desk colors. Such biases are prevalent in datasets like CelebA [93], where blond females dominate certain labels. Exploring and understanding failures of image classifiers is challenging and has been addressed in several works [89, 140, 19, 74, 99, 57]. These studies typically rely on manually predefined dimensions (e.g., age or gender), which are labor-intensive and restrict bias analysis to fixed dimensions and failure cases only that may not encompass the full spectrum of possible bias causes.

To address these shortcomings, we propose an unsupervised framework that combines commonsense knowledge graphs (common-sense KGs), foundation models, and graph mining techniques to extract patterns from automatically extracted dimensions, uncovering learned bias. This approach enables the discovery of complex patterns that go beyond limited predefined labels by generating detailed, task-agnostic scene descriptions. These descriptions include object attributes (e.g., shape, color), environmental settings (e.g., dark, cloudy, rainy), and visual styles (e.g., blurry, high or low contrast), addressing gaps left by previous scene description methods [101, 182]. The descriptions are structured as scene graphs and analyzed using graph mining techniques to uncover patterns that emulate classifier behavior, revealing how input components influence predictions. E.g., a pattern can state that images correctly classified as classrooms often feature green desks and wooden chairs.

Additionally, we utilize insights from the common-sense KG and extracted patterns to predict potential failures, enhancing the robustness of image classifiers and helping avoid failures due to low or no coverage of specific scenarios in the training data. For example, if a pattern shows that classrooms with blue desks and metal chairs are often misclassified, collecting more such images for retraining could improve classifier performance. We also utilized the extracted patterns to help in understanding and justifying failures made by an image classifier. The main contributions of this work are:

- An unsupervised framework that extracts graph-based patterns from automatically extracted semantic dimensions, enabling the discovery of complex success and failure patterns learned by an image classifier.
- An unsupervised scene descriptor that uniquely utilizes common-sense KGs and foundation models to create detailed scene graphs for images.

- A novel use of sub-graph mining to extract patterns for analyzing failure cases in image classifiers, providing structured insights into their behavior.
- A novel application of the framework to predict and explain failure cases using structured graph explanations for systematic error analysis.

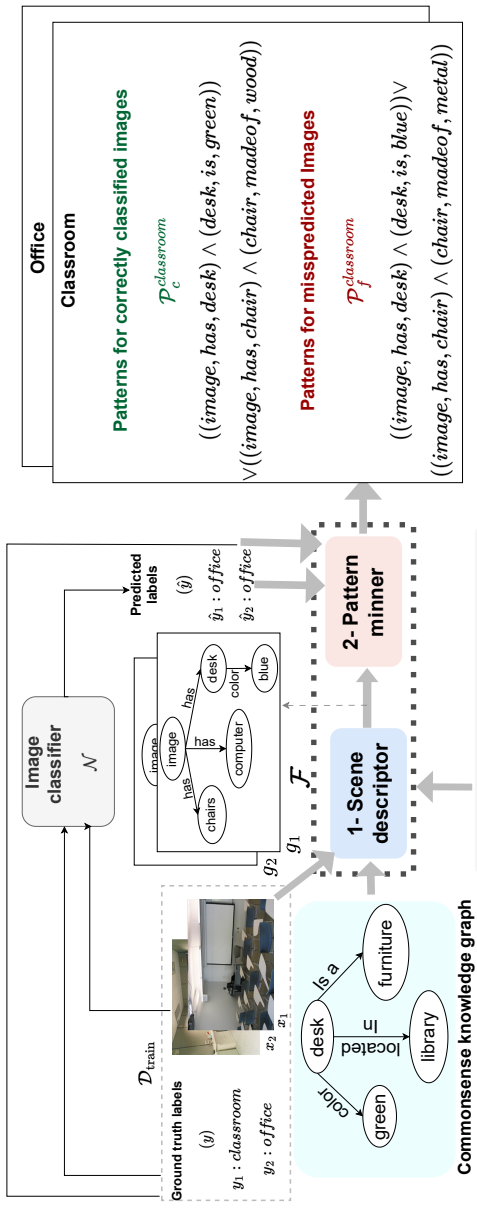


Figure 7.1: Illustration of the proposed framework for pattern extraction, featuring an unsupervised scene descriptor for generating scene graphs and a pattern miner that identifies common graph patterns among scene graphs for correctly and misclassified images per class.

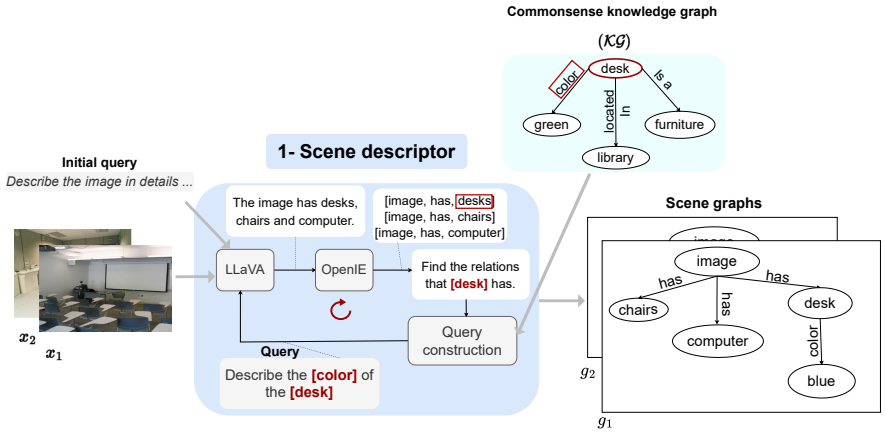


Figure 7.2: The scene descriptor leverages $\mathcal{KG}$ and foundation models to generate a scene graph g_1 for image x_1 unsupervisedly.

7.2 Preliminary

In this section, we provide preliminary information on several topics relevant to this chapter.

Scene Graphs. Scene graphs are structured representations of visual scenes that encode objects, their attributes, and relationships as a graph, enabling contextual understanding and reasoning in computer vision tasks [70]. Similar to knowledge graphs, scene graphs depict objects as nodes and relationships as edges, offering a more nuanced understanding than mere object detection by encoding interactions and attributes among objects. For instance, a scene graph might capture detailed relationships such as "chairs are behind the board" or "desk in front of the chair". This detailed representation is vital for tasks like image retrieval, captioning, and visual question answering.

Graph Mining. Graph mining focuses on extracting patterns and insights from graph-structured data. These methods identify frequent sub-graph structures either within a single graph or across multiple graphs. They are widely applied in fields such as social networks, biology, and the semantic web, supporting tasks like community detection, link prediction, and graph classification.

Foundation Models. Foundation models are a generalized breed of machine learning models trained on massive amounts of generalized and unlabeled data and can be used as is or fine-tuned to a wide range of downstream tasks like text and image generation, question answering, image classification, natural language processing tasks, and more.

7.3 Framework

Before dwelling on the details of our framework, we introduce necessary notations used throughout this chapter.

- Knowledge Graphs A knowledge graph (KG) comprises triples in the form (s, r, t), where: s (subject) and t (object) are entities or concepts and r denotes the relationship between them.

  For example, the triple *(desk, is, blue)* indicates that the desk is blue. In this framework, we utilize ConceptNet 5.5 [144], a common sense knowledge graph known for its high coverage and open access, though other common-sense KGs can also be used.

- $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{m}$ represents the dataset composed of image-label pairs, where x_i is the i^{th} image, y_i its corresponding class label and m denotes the total number of images. The dataset is divided into 3 subsets: training ($\mathcal{D}_{\text{train}}$), validation ($\mathcal{D}_{\text{valid}}$), and testing ($\mathcal{D}_{\text{test}}$).

- $\mathcal{N}$ denotes the image classifier trained on $\mathcal{D}_{\text{train}}$ to predict the label $\hat{y}$ from the input image x.

Our framework $\mathcal{F}$, shown in Fig. 7.1, takes $\mathcal{KG}$ and $\mathcal{D}_{\text{train}}$ as input and output patterns reflecting distinguishing commonalities among correctly classified images (w.r.t. falsely classified ones) and, separately, patterns for falsely classified images (w.r.t. correctly classified images) for each class k. To extract these patterns for each class, our framework includes two main modules: a scene descriptor that gathers detailed information from images in $\mathcal{D}_{\text{train}}$, and a pattern extractor that identifies recurring patterns in the scene descriptions. In the following sections, we explain our framework in detail.

7.3.1 Scene Descriptor

The scene descriptor takes images $\{x_1, \ldots, x_q\}$ from $\mathcal{D}_{\text{train}}$ and a commonsense knowledge graph (common-sense KG) $\mathcal{KG}$ as input, and outputs a set of graphs $\{g_1, \ldots, g_q\}$, with each graph thoroughly describing a single image. To generate these graphs, we use the open-source LLaVA v1.5 model [91] and ask it a series of customized questions based on the image content, as shown in Fig. 7.2. First, we ask LLaVA an initial query to describe the image in detail, using a carefully crafted set of instructions to ensure consistency and clarity in the responses. The query is designed to elicit a description detailed enough for someone to redraw the image without seeing it, as shown in Fig. 7.3. We then

extract triplets from its responses using OpenIE [4], as it is an unsupervised method, unlike foundation models, which often require fine-tuning for effective information extraction [123]. Each triplet consists of a source, a relation, and a target, as shown in Fig. 7.2.

Next, we pass the triplets extracted from the initial query, along with the $\mathcal{KG}$, to the query construction module, which identifies the relevant semantic dimensions for each node to generate follow-up questions. The relation labels for nodes in the common-sense KG act as semantic dimensions to form queries, which are passed to LLaVA to extract additional visual insights. More precisely, we use the template *"Describe the [relation] of the [node]"*, as shown in Fig. 7.4, to guide the model in providing structured and comprehensive answers, helping to capture the entire scene rather than isolated objects [96, 58, 40]. If a node is not found in ConceptNet, we use the most similar concept based on embeddings from MiniLM [167].

For example, if *writing desk* is one of the nodes in the triplets extracted from the initial query's response, we search for a similar node in the common-sense KG. Based on the embedding of *writing desk*, the closest node in the common-sense KG is *desk*, which has relations like *color*, representing a possible semantic dimension. Leveraging this, we inquire LLaVA about the *color* of the *desk* in the image. We repeat this process for each node in the triplets extracted from the initial query's response for image x_i and extract triplets from the follow-up answers.

These triplets form a level in the output scene graph g_i for image x_i. The depth of the scene graph can be increased by adding more iterations, denoted by n.

The scene descriptor takes the images $\{x_1, \ldots, x_q\}$ from $\mathcal{D}_{\text{train}}$), and a common-sense KG $\mathcal{KG}$ as input, and outputs a set of graphs $\{g_1, \ldots, g_q\}$, with each graph thoroughly describing a single image. To generate these graphs, we use the open-source LLaVA v1.5 model [91] and ask it a series of customized questions based on the image content, as shown in Fig. 7.2. First, we ask LLaVA an initial query to describe the image in detail, as shown in Fig. 7.3. We then extract triplets from its responses using OpenIE [4], as it is an unsupervised method, unlike foundation models which often require fine-tuning for effective information extraction [123]. Each triplet consists of a source, a relation, and a target, as shown in Fig. 7.2.

Then we pass the triplets extracted from the answer of the initial query along $\mathcal{KG}$ to the query construction module that identifies the proper semantic dimensions that each concept might have to ask relevant questions about them. If a concept is not in ConceptNet, we extract the relations of its most similar concept based on embeddings computed using MiniLM [167], a task-agnostic, lightweight

You are given an image and your task is to write a paragraph describing the image. The description should be detailed, enough to draw and paint the image again without seeing it. Provide reasoning why do you think that the image has specific features. Answer in short simple full sentences. Do not use pronouns. Stick to the image, do not mention information that is not in the image.

Figure 7.3: Details of the initial query (*"Describe the image in detail .."*) given to LLaVA for each image x_i.

model. We consider the relation labels for the node in the common-sense KG as the semantic dimensions of the node. We then employ a structured approach, using the template *Describe the [relation] of the [node]* as a query that is passed to LLaVA along with the image, as shown in Fig. 7.4. These queries help us obtain additional visual insights about the node and its attributes in the image.

For example, if *writing desk* is one of the nodes in the triplets extracted from the answer to the initial query, we search for a similar node to *writing desk* in the common-sense KG. Based on the embedding of *writing desk*, *desk* node in the common-sense KG is the closest node and it has several relations, such as *color*, which are used as possible semantic dimension. Leveraging this insight, we can inquire LLaVA about the *color* of the *desk* present in the image. We repeat this process for each node mentioned in the triplets extracted from the answer to the initial query w.r.t. a given image x_i, and extract triplets from their answers.

These triplets form a level in the output scene graph g_i for image x_i. The depth of the scene graph can be further increased by increasing the number of iterations denoted by n. This iterative approach allows us to control the level of granularity of the scene graph describing each image.

Formulating prompts to LLaVA model affects the result provided by LLaVA. For that, we provided a detailed set of instructions to make sure that the LLaVA model provides the expected answer.

In Fig. 7.2 we ask the LLaVA model to describe the image with the instructions shown in Figs. 7.3,7.4. We mentioned that *"the description should be detailed, enough to draw and paint the image again without seeing it"* to force the model to describe the entire setting and avoid the shortcomings of foundation models of focusing only on the objects in the image [96, 58, 40].

Answer this question with respect to the image. If the object not in the image or is not clear skip the question. Answer in complete sentences. Do not answer in single word. Do not answer with yes/no. Provide reasoning why do you think that the image has specific features. Answer in short simple full sentences. Do not use pronouns (e.g., He, She, His, Her, ..).

Figure 7.4: Instructions given to LLaVA to along the question *"Describe the [relation] of the [node]."*

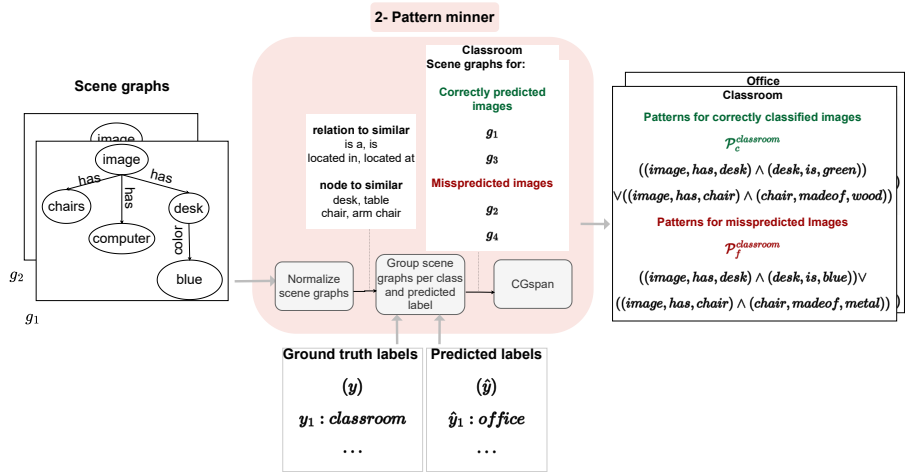


Figure 7.5: The pattern miner extracts the common patterns between the scene graphs created by the scene descriptor.

7.3.2 Pattern Miner

Once we obtain the scene graphs for each image in the form of triplets, we extract the common sub-graph patterns among images that are correctly classified for each class as well as the falsely classified images. To achieve this, we use cgSpan [134], which extracts frequent closed sub-graph patterns from a set of graphs and a node-to-label mapping. This allows it to identify sub-graph patterns among nodes that have different names but belong to the same class. We use a node-to-label mapping to help cgSpan find patterns among semantically similar but differently named nodes, such as *desk* and *writing desk*.

To construct this mapping, we compute node embeddings using MiniLM [167], a task-agnostic, lightweight embedding model, and consider nodes with a cosine

similarity of at least 0.8 to be similar. To prepare the input for cgSpan, we group the scene graphs by class and split them into graphs for correct and incorrect classifications based on the ground truth labels vs predicted labels. As shown in Fig. 7.5 for $k = \textit{classroom}$, g_1 and g_3 belong to images that were correctly classified by $\mathcal{N}$ as classroom.

For each class, we pass the group of graphs for correctly classified images, along with the node-to-similar list, to cgSpan to extract the most frequent closed sub-graph patterns, denoted as $\mathcal{P}_c^k$. We repeat this process for the graphs of images with the ground truth label k which were misclassified to obtain $\mathcal{P}_f^k$. Each set of patterns $\mathcal{P}_c^k$ is composed of a disjunction of patterns, (i.e., $\mathcal{P}_c^k = p_1 \vee \ldots \vee p_j$), where each pattern p_j is a conjunction of triplets, as all the triplets in p_j are crucial for forming the pattern. The pattern can be composed of only one triplet. The same applies to $\mathcal{P}_f^k$. We then exclude the common sub-graph patterns between $\mathcal{P}_f^k$ and $\mathcal{P}_c^k$ from both sets, as patterns that exist in both sets are uninformative.

CgSpan also returns a frequency score f_j for each extracted pattern p_j that can be used to filter patterns. We only consider patterns that appear in at least 50% of the input image graphs, as it provides an optimal balance between the quantity and quality of the extracted patterns.

7.4 Experiments and Results

In this section, we provide an overview of the experiments used to evaluate the scene graphs generated by the scene descriptor as well as the patterns extracted by the pattern miner. In all the experiments below, we use ResNet18 [55] as the image classifier. We also introduce new methods that adopt the functionality-grounded evaluation strategy outlined in [39].

7.4.1 Scene Descriptor Evaluation

In this section, we evaluate the quality of the scene graphs generated using our method by comparing them to scene graphs obtained relying on manual annotations. In what follows, we set the parameter n of our framework, responsible for the number of iterations of the scene descriptor, to 2.

Dataset. We then evaluate the scene descriptor using the *Visual Genome (VG)* dataset [77], which contains scene graphs created from manual annotations with

Image	VG graph ($\hat{g}$)	Scene graph (g)
	(boats, are on, ocean), (ship, on, water)	(fighter jets, flying in, formation), (fighter jets, flying in, coordinated pattern), (pattern, is, coordinated), (planes, synchronize, movement), (planes, are shaped like rockets), (planes, have, distances), (planes, is, military), (altitude flyby, is, low) (prowess, is, military), (primary, impressive aerial display of, military), (smoke, is coming from, planes), (smoke, is, white), (smoke, is, trailing), (smoke, left by, trails), (smoke, high into, sky), (navy ship, are, multiple), (navy ship, are on, ocean), (navy ships, is, several), (navy ships, including, battleship), (scene, is, dynamic), (sight, is, impressive), (atmosphere, is, bustling), (ocean, is in, background), (two cruise ships, are in, background), (island, is, background), (background, is near, horizon), (view, is, up-close) (event, is, outdoor), (exercises, are, military), (military, is displaying aircraft at, location), (two boats, is in, water) ⋯
	(surfer, in, wet-suit), (surfer, with, hair), (arm, of, surfer), (pattern, on, surf board), (man, wearing, wet-suit), (man, standing on, surfboard)	(ocean, is in, image), (ocean, is, blue), (ocean, is, large), (ocean, is, turbulent),(water, has, surface), (wave, crushing around, surfer), (wave, is, big), (wave, is, strong), (wave, is, challenging), (surfer, has, posture), (surfer, has, surfboard), (surfer, is in, thrilling crests of waves),(surfboard, is well balanced on, surface), (surfboard, is, red), (surfer, riding waves on, surfboard), (day, is, clear), (sea foam, is, white), (wet-suit, is in black), (surfer, is balancing on, surfboard), (man, wearing, wet-suit), (man, is standing on, surfboard) ⋯
	(boats, are on, ocean), (ship, on, water)	(fighter jets, flying in, formation), (fighter jets, flying in, coordinated pattern), (pattern, is, coordinated), (planes, synchronize, movement), (planes, are shaped like rockets), (planes, have, distances), (planes, is, military), (altitude flyby, is, low) (prowess, is, military), (primary, impressive aerial display of, military), (smoke, is coming from, planes), (smoke, is, white), (smoke, is, trailing), (smoke, left by, trails), (smoke, high into, sky), (navy ship, are, multiple), (navy ship, are on, ocean), (navy ships, is, several), (navy ships, including, battleship), (scene, is, dynamic), (sight, is, impressive), (atmosphere, is, bustling), (ocean, is in, background), (two cruise ships, are in, background), (island, is, background), (background, is near, horizon), (view, is, up-close) (event, is, outdoor), (exercises, are, military), (military, is displaying aircraft at, location), (two boats, is in, water) ⋯
	(surfer, in, wet-suit), (surfer, with, hair), (arm, of, surfer), (pattern, on, surf board), (man, wearing, wet-suit), (man, standing on, surfboard)	(ocean, is in, image), (ocean, is, blue), (ocean, is, large), (ocean, is, turbulent),(water, has, surface), (wave, crushing around, surfer), (wave, is, big), (wave, is, strong), (wave, is, challenging), (surfer, has, posture), (surfer, has, surfboard), (surfer, is in, thrilling crests of waves),(surfboard, is well balanced on, surface), (surfboard, is, red), (surfer, riding waves on, surfboard), (day, is, clear), (sea foam, is, white), (wet-suit, is in black), (surfer, is balancing on, surfboard), (man, wearing, wet-suit), (man, is standing on, surfboard) ⋯

Table 7.1: Visual Genome graph $\hat{g}$ vs a snippet of scene graph g generated by our scene descriptor for the same image. The scene graphs g generated by our scene descriptor reveal additional image details not captured by the Visual Genome graphs that are based on manual annotations.

Instructions: Provide a set of triplets describing the picture in great level of detail. The description should be detailed, enough to draw and paint the image again without seeing it. First you need to describe the image in details in paragraph then extract the triplets from the paragraph by following these instructions:

1. **Identify Key Entities**: Read through the paragraph and identify the main entities involved. These will be the subjects and objects of your triplets.
2. **Determine Relationships**: For each entity, identify the relationships that connect them to other entities or describe their state. These relationships will form the predicates of your triplets.
3. **Create Triplets**: Structure the entities and relationships into triplets in the format (subject, predicate, object).
4. **Example** : Consider the following paragraph and the resulting triplets. Example Paragraph 'John bought a new car from the dealership. 'Example Triplets(John, bought, car),(car, from, dealership)

Query: Provide a set of triplets describing the picture in great level of detail for this image.

LLaVA: (bird, is, white), (bird, has, black head and tail), (bird, is, above water), (bird, is, over body of water), (bird, is, above green field), (bird, is, in flight), (bird, is, in natural setting), (bird, is, in tranquil setting), **(bird, is, in front of, sun), (bird, is, in front of, light), (bird, is, in front of, day),** $\cdots$ **(bird, is, in front of, daytime realistic photograph)**x23
Query: Answer in paragraph instead of triplets.
LLaVA: The image depicts a large white bird with a prominent crest, soaring above a body of water. The bird's wings are fully extended, and it appears to be in mid-flight. The bird's tail feathers are also visible, adding to the sense of motion. The background is a blurred landscape, with hints of green grass and a clear sky, suggesting that the photo was taken during the day. The lighting in the image is bright, with a noticeable lens flare effect, which could indicate that the sun is either rising or setting. The bird's position in the sky and its direction of flight suggest that it is moving towards the right side of the image. The image is a realistic photograph, capturing the bird in its natural habitat.

Figure 7.6: LLaVA responses seems to have formatting issues; produce quadruples instead of triplets, repeats triplets multiple times until all tokens were used.

an average of 35 objects, 26 attributes, and 21 relationships per image. For this work, we randomly select 100 images and their corresponding scene graphs.

Baseline. As a baseline for evaluating the quality of the graphs produced by the scene descriptor, we utilized our framework with $n = 1$ and initial query *Describe the image in details*, with no extra instructions about the description.
Experimental questions. We utilize Visual Genome to answer the following questions:
• **Do scene graphs generated by the scene descriptor cover the aspects represented by human annotator?**
The aim of this experiment is to evaluate the quality of graphs produced by the scene descriptor. Quality assessment is conducted using the similarity between graphs generated by the scene descriptor and those derived from manual annotations. Given the potential for differing lexical representations of the same semantic concepts between the scene descriptor's output and the VG, we employ embeddings for each triplet to measure similarity. Specifically, cosine similarity between triplet embeddings from our scene graphs and those in the VG is calculated. For each image triplet in the VG, we identify the most similar triplet within the graph generated by our scene descriptor for the same image. The average similarity score is then computed using the formula defined in Eqn. 7.1.

$$S(\hat{g}_i, g_i) = \frac{\sum_{\hat{\tau} \in \hat{g}_i} \max_{\tau \in g_i} (\text{Sim}(\xi(\tau), \xi(\hat{\tau})))}{|\hat{g}_i|} \tag{7.1}$$

where, $\hat{g}_i$ is the list of triplets for a given image within the Visual Genome dataset, while g_i represents the set of triplets for the same image generated by the scene descriptor. $\text{Sim}(\xi(\tau), \xi(\hat{\tau}))$ is the cosine similarity. The triplet embedding ξ is computed as the concatenation of embeddings: for triple $\tau = (s, r, t)$ we have $\xi(\tau) = [\text{Embedding}(s), \text{Embedding}(r), \text{Embedding}(t)]$, where embeddings are computed using MiniLM. Same applies to $\xi(\hat{\tau})$.

While triplet embeddings could theoretically be derived from node embeddings generated by knowledge graph embedding models [14, 46, 162], these models generally focus on learning node embeddings from the graph structure and node neighborhoods, rather than the semantic meanings of the node labels.

Our findings show that the triplets in the VG dataset have comparable counterparts in the outputs generated by our scene descriptor, with an observed average similarity score of 0.79. This is noteworthy compared to the baseline similarity score of 0.48. These results validate the scene descriptor and its sub-components, LLaVA & OpenIE, in replicating the quality of manually annotated graphs and extracting high-quality triplets, showcasing its potential for detailed, unsupervised scene graph construction.

Image	VG graph ($\hat{g}$)	Scene descriptor graph (g)
	(dogs, have, shirts), (dogs, on, surfboard), (surfboard, on, water), (woman, in, water), (words, on, chest), (hat, on, head)	(dogs, are wearing, shirts), (dogs, wear, bandanas), (dogs, are, brown), (dogs, are on, surfboard), (dogs, are, labrador retrievers), (surfboard, is in, water), (surfboard, is, blue), (woman, is, old), (woman, is on, surfboard in water), (water, is, blue), (woman, wearing, pink hat), (custom, is with, reality rally), (day, is, sunny), (sky, is, clear), (apron, is, white), ...
	(towel set, hanging on, holder), (tile, on, floor)	(towel, is, black), (towels, are, hung),(floor, is, black tiled), (natural light, enter, bathroom),(toilet, is, white), (toilet, is, in the corner against the wall),(toilet, is by, sink), (tub, under, window)

Table 7.2: VG graph vs a snippet of our scene graph for the same image.

Average number of	Visual Genome	Our scene descriptor
Nodes	36	769
Relations	19	384

Table 7.3: The average number of nodes and relations per image in Visual Genome [77] vs our scene descriptor.

• Does the scene descriptor produce richer graphs than graphs from manual annotations?

To explore this, we use the count of five components as a metric to compare the level of detail captured by our scene descriptor against the Visual Genome dataset. We perform a comparative analysis focusing on five key components: triplets, relations, nodes, node clusters, and relation clusters. This analysis evaluates not only the total number of nodes and relations but also the diversity within these elements. We specifically examine cluster counts to determine whether variations in nodes (e.g., *desk* versus *writing desk*) or relation phrases (e.g., *has* versus *has a*) contribute to the differences in graph counts. For that, we use HDBSCAN [97], which determines the optimal number of clusters based on data density, with a minimum cluster size of 2 and cluster selection $\epsilon = 0.1$.

Fig. 7.7 shows the overall number for each component, proving that graphs generated by our scene descriptor are denser compared to those from the VG dataset among all five components. Further evidence is provided in Tab. 7.2,

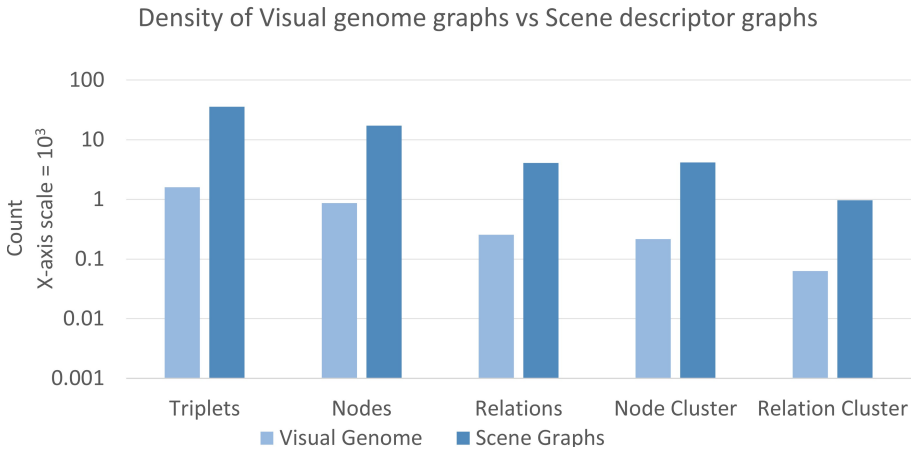


Figure 7.7: A plot in logarithmic scale of the total number of the triplets, nodes, relations, node clusters and relation clusters found in VG graphs vs our scene graphs.

Average number of	n	Waterbirds	biased CelebA	Indoor
Nodes	1	28	48	52
	2	376	63	465
Relations	1	14	24	26
	2	188	32	232

Table 7.4: The average number of nodes and relations per image for each dataset over different iteration n.

where it is apparent that the VG graphs often contain less detail about the scenes compared to the graphs generated by our scene descriptor. This is also seen in the examples in Tab. 7.2, where our scene descriptor captures details that are typically missing in VG like the dogs' color and species, the background, and the surfing board color in the 1^{st} row, and the towel color and the exact placement of the toilet and bathtub in the 2^{nd} row. The difference in graph density can be further observed in Tab. 7.3, where the average number of nodes and relations per scene graph extracted by our scene descriptors exceeds the numbers reported [77] for Visual Genome graphs. These findings underscore the superior descriptive power and comprehensiveness of our scene descriptor, highlighting its potential to enhance data richness in automated graph generation.

The number of nodes and relations found in each scene graphs changes based on

the image, dataset and the number of iterations. Tab. 7.4 shows that the Indoor dataset has on average the most dense graphs as it has the highest number of nodes and relations per scene graph, this aligns with the nature of the indoor scenes that have lots of details. While the biased CelebA has the sparsest scene graphs as the image is usually compromising of head-shot of a celebrity, with little vision of the background. The Waterbirds dataset has denser scene graphs than the CelebA but sparser than the Indoor.

Foundation models (FMs) for triplet construction

While some FMs can generate text in a specified format, they often need fine-tuning for effective information extraction [123], which conflicts with our goal of creating an unsupervised framework. Using LLaVA for triplet extraction without fine-tuning revealed two key shortcomings as shown in Fig. 7.6. First, specifying a detailed format for answers reduces the level of detail in the response compared to where it is asked to answer in a paragraph. Second, LLaVA sometimes exhausts the allowed token limit by repeating triplets, with the last triplet repeated 23 times in Fig. 7.6, and struggles to maintain the specified triplet format, often producing quadruples instead. Conversely OpenIE, an unsupervised heuristic-based algorithm, produces consistent triplet with details, as demonstrated in Tab. 7.2. It also offers greater control over the quality of the extracted triplets through parameter tuning, and it resolves ambiguities in references like "his/her/..." to form clearer triplets.

On the effect of the granularity of scene graphs on their effectiveness in mimicking the image classifier behavior

We used different values for n, which controls the number of iterations used to create the scene graphs, testing iterations of $n = \{1, 2\}$. Our experiments show that the larger n the better the alignment with the decision-making process of the image classifier $\mathcal{N}$. In particular, for the CelebA (example in Tab. 7.7) and Indoor (example in Tab. 7.6) datasets, $\mathcal{N}$ appears to rely on detailed scene information for classification. This contrasts with the Waterbirds dataset, where the alignment weighted F1 score is already 0.90 after using information about the scene extracted from the first iteration. Tabs. 7.5,7.6, 7.7 shows the new triplets added per level to highlight the effect of increasing n on the level of details added to the description of the image for different datasets.

Image	Scene graph (g)	
	$n = 1$	$n = 2$
	(bird, is with, yellow body), (environment, is, pleasant), (bird, rests with, one foot), (forest, is with, multiple trees), (trees, are, multiple), (natural habitat, is for, bird), (structure, is, bamboo like), (habitat, is, natural)	(bamboo, is, green), (bamboo, appears, to full growth), (bird, is, outdoor), (bird, has, black wing), (bird, has, long pointed beak), (bird, is, small), (black marking, is on, wings), (bird, is perched on, tree stump), (bird, is as, focal point), (bird, is in, natural setting), (bird, has, coloration), (coloration, is, vibrant), (birdhouse, is surrounded by, bamboo stalk), (bird's house, has, cylindrical shape), (beak, is, pointed), (prominent crest, is on, head), (posture, is, alert), (lighting, is, natural), (forest, is in , image), (image, is taken during, daytime), (bamboo groove, is located in, forest setting), (area, is, wooded), (area, is, forest), (forest, is, green), (setting, is, serene), (setting, is, outdoor), (environment, is, natural), (ground, is, covered), (color, is, vibrant), (hue, is, greenish) $\cdots$
	(leaf, is, large), (sky, is, cloudy), (Pelican, is, large), (Pelican, is, white), (Pelican, with, long bill), (bill, is, long), (image, features, large Pelican), (white bird, is with, long bill), (setting, is, natural), (bird, is, white)	(bird, has, bill), (bill, is, long), (bill, is, large), (feathers, are, white), (neck, is, long), (bird, has, feather), (bird, is in, natural habitat), (natural habitat, is, such lake), (lake, is, shallow), (bird, is of type, Pelican), (bird, is, predominantly white), (bird, is surrounded by, green leaves), (bird, is, facing left), (image, features, large bird), (large bird, is in, foreground), (large bird, soaring above, body of water), (head, is, bent downwards), (Pelican, has beak) (throat, is, pouch like), (lightning, is in, image), (background, being, slightly blurred), (background, is, body of water), (water, is, calm), (beak, is, slightly open), (beak, is, curved), (pinkish hue, is on, beak),(image, features, large green leaf), (leaf, is, green), (leaf, is, large), (sky, is, overcast), (day, is, cloudy), (atmosphere, is, calm), $\cdots$

Table 7.5: Snippet of scene graph g generated by our scene descriptor for images from the Waterbirds dataset for different values of n. Our scene graph g reveal image details without prior fine tuning due to its iterative nature.

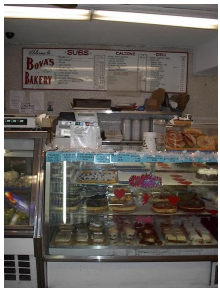
Image	Scene graphs (g)	
	$n = 1$	$n = 2$
	(selection, is, wide), (pizzas, are arranged in, neat rows), (pizzas, each with, different toppings), (rows, is, neat), (display case, is, vibrant), (case, highlighting, pizzas), (case, is, filled), (toppings, is with, different)	(goods, is, baked), (products, is in, case), (case, is in, store), (case, is, illuminated), (case, is, refrigerated display), (case, has, glass front), (lights, is by, overhead), (display case, is, appealing), (display case, being, with main source of illumination), (focus, is on, products in case), (focus, is on, food), (products, are, showcased), (angle, is, low), (lighting, is dim with, interior of case illuminated by overhead lights), (background, is, dark), (room, is with, darker background), (lights, are located above, display case), (image, shows, display case), (environment, is, commercial), (toppings, is, visible), ⋯
	(bakery, is, clean), (ink, is in, black), (baked goods, is on, display), (bakery counter, is with, baked goods on display), (bakery counter, is, clean), (image, depicts, bakery counter), (goods, is, baked), (menu board, also lists, prices), (menu board, is with, words written in red letters), (BOVA, is, bakery), (bakery, is, organized), (customers, see clearly, baked goods), (cakes, each with, different decorations), (cakes, are arranged in, row), (cakes, is near, front of counter), (cakes, is, several)	(environment, is, clean), (image, shows, bakery), (items, is, baked), (counter top, is, white), (bakery counter, is, display case), (bakery counter, is with, various pastries), (display case, has, glass front), (display case, is, filled), (display case, is with, clean), (display case, is with, metal frame), (display case, is with, blue tint), (case, is illuminated from, above), (cakes, look, fresh), (bakery, has, menu board), (menu board, is mounted on, wall above counter), (menu board, is in, background), (menu board, is with, various items listed), (lighting, creating, welcoming atmosphere), (lighting, is, bright), (customers, see inside, items), (cash register, is in, background), (cup, is in, background), (wall, is above, counter), (selection, is, diverse), (decorations, is on, counter), (papers, is on, counter), (refrigerator, is with, glass door), (beverages, is such, coffee), (bakery, look, appetizing), (wall, is, white), (pastries, is of, various), (pastries, are arranged on, shelves), ⋯

Table 7.6: Snippet of scene graph g generated by our scene descriptor for images from the Indoor dataset for different values of n. Our scene graph g reveal image details without prior fine tuning due to its iterative nature.

Image	Scene graphs (g)	
	$n = 1$	$n = 2$
	(person, appears, young adult), (image, is, close up portrait), (hair, is, blonde), (hair, falls over, shoulders), (hair, is parted in, middle), (hair, is, straight),(approach, is, natural), (image, focus is, woman's face), (woman, is, young), (lighting, creating, atmosphere), (lighting, is, soft), (atmosphere, is, gentle), (casual approach, to, portrait photography), (complexion, is, fair), (lips, is, full), (expression, is, neutral), (features, is, facial), (image, style of, photograph)	(features, is, facial), (image, style of, photograph), (hair, is, straight), (image, shows, person), (person, has, eyebrows), (eyes, looking directly at, camera), (eyes, are, open), (image, is, close up portrait)
	(man, is wearing, dark suit), (tie, is, red), (man, has, dark hair), (suit, is, dark), (hair, is, dark), (shirt, is, white), (image, is, portrait), (image, is, professional), (lighting, is in, image), (background, is, white)	(hair, is of, medium length), (hair, appears to, straight), (hair, is, styled)

Table 7.7: Snippet of scene graph g generated by our scene descriptor for images from the biased CelebA dataset for different values of n. Our scene graphs reveal image details without prior fine tuning due to its iterative nature.

7.4.2 Pattern Evaluation

In this section, we evaluate the quality of the patterns extracted using our framework. The evaluation comprises two different experimental approaches: 1) utilizing datasets with known patterns and 2) employing datasets with unknown patterns. These experiments are designed to confirm the accuracy of our framework in detecting patterns recognized by image classifiers and to evaluate the quality of these patterns in datasets where the patterns are not previously defined. In the experiments devoted to the evaluation of the pattern miner, the parameter n of our framework is set to 2, unless specified otherwise. The appendix demonstrates the impact of different values of n on the quality of the extracted patterns.

Datasets. For evaluation we utilize the following datasets:

•**Biased CelebA.** CelebA [93] is a large face attributes dataset of images, each annotated with 40 attributes. For our study, we created a biased subset similar to [122], selecting 1000 images of *female* celebrities with *blond hair* and 1000 images of *male* celebrities with *dark hair*. We refer to this as biased CelebA. The data was split 3:2 for training and validation. For testing, we selected 2000

Dataset	training F1	validation F1	test F1
CelebA	0.71	0.71	0.52
Waterbirds	0.76	0.77	0.47
Indoor	0.55	-	0.49

Table 7.8: The F1 scores of ResNet18 trained from scratch on various datasets.

additional images featuring *females* with *dark hair* and *males* with *blond hair*.

• **Waterbirds.** The Waterbirds dataset [122] is categorized into *waterbird* and *landbird* classes. *Waterbirds* have *aquatic* backgrounds, while *landbirds* have *terrestrial* backgrounds. We randomly selected 500 images of each class, splitting the 1000 images 3:2 for training and validation. For testing, we selected another 1000 images where the usual background context is altered—*landbirds* with *aquatic* backgrounds and *waterbirds* with *terrestrial* backgrounds.

• **MIT Indoor.** We chose six visually similar classes; *bakery, deli, classroom, office, kitchen, pantry* from the MIT Indoor dataset [115] to challenge the image classifier and consequently induce misclassifications that allow us to analyze and extract patterns for failures. This resulted in 2,662 images, which were split in a stratified manner with a 3:2 ratio for training and testing.

Model. For the image classifier, we trained a ResNet18 classifier with parameter tuning similar to that in [122] for the CelebA and Waterbirds datasets. For the CelebA and Waterbirds, we evaluated the model on the data validation split that exhibits different patterns compared to those in the training to ensure that the model captured the pattern in the dataset. This can be verified if the model trained on the data split with a known pattern (biased) performs better on the validation split that has the same bias than on the test split with a different bias. For the MIT Indoor dataset, we trained the model for 20 epochs with the learning rate of 0.001. As we do not know the patterns in this dataset, we report the F1 scores of the model on the train and test split.

Baseline. As a baseline, we used the same framework with $n = 1$ where we do not utilize the iterations to extract more information about the images; that also means that we do not use the common-sense KG to create scene graphs. Given the novelty and fundamental difference of our structured graph pattern explanations from prior methods in the related work section, which rely on single-label explanations or visual illustrations, direct comparisons are infeasible.

Experimental questions. We utilize the introduced datasets and models to answer the following questions:

• **Given an image classifier with known pattern bias, can our framework find this pattern?**

Dataset	Class	Patterns from correct predictions (P_c^K)
Biased CelebA	female	(fair, is, complexion) ∨ (hair, is with, blonde)
	male	(hair, is, darkness)
Waterbirds	landbird	**(**(small, is, bird) ∧ (bird, has, position in branch) ∧ (bird, is focus of, image)∧ (trees, in, image)**)** ∨ **(**(background, is in, the forest)∧ (background, is, quite natural)∧ (bird, is, in quite natural)∧ (bird, has, bright coloration), ...
	waterbird	(bird, is, seagull) ∨ **(**(bird, in, image) ∧ (bird, on, water) ∧ (large, is, bird) ∧ (bird, is, black) ...**)**

Table 7.9: Patterns extracted from the ResNet18 trained on the biased CelebA and Waterbirds datasets, sorted by frequency.

We assess the quality of the patterns extracted by our framework both qualitatively and quantitatively.

Qualitative evaluation. The performance of ResNet18 on the biased CelebA and Waterbirds datasets is indicative of its capacity to capture specific patterns. This is evidenced by high F1 scores on the training and validation splits, which share the same patterns, but lower performance on test splits that present different patterns, as documented in Tab. 7.8. Upon achieving this, our framework is used to extract patterns based on the classifier's correct and incorrect predictions. These extracted patterns are then compared to the known bias patterns within the dataset. For instance, the bias detected in the patterns extracted from the ResNet18 trained on the CelebA dataset in Tab. 7.9 aligns with both the observed lack of generalization on the CelebA test set of the ResNet18 shown in Tab. 7.8 and the known patterns in the biased dataset [122], confirming the effectiveness of our framework in identifying patterns that contribute to classifier biases. This procedure was similarly executed for the Waterbirds dataset, with the findings (Tab. 7.9) consistent with the behaviors noted in the biased CelebA dataset.

Quantitative evaluation. When dataset patterns learned by $\mathcal{N}$ are unknown, we evaluate the effectiveness of the extracted patterns in replicating $\mathcal{N}$'s predictive behavior. This assessment is conducted using a pattern-based classifier, denoted as PC. The PC operates by taking a scene graph g_i constructed by our scene descriptor and the patterns mined by the pattern miner for different classes. It then predicts a class $\dot{y}_i$ for each image x_i by evaluating the similarity between the triplets in the scene graph g_i for x_i and the extracted patterns for each class $\mathcal{P}_j^k$, where j can be c (correctly classified) or f (falsely classified), and $\mathcal{P}_c^k$ represents the set of patterns associated with images correctly classified under class k. More formally, PC relies on the scores computed by Eqn.7.2 to predict the class k of an image x_i with scene graph g_i.

$$score_{j^k}(g_i, \mathcal{P}_j^k) = \frac{\sum_{p_a \in \mathcal{P}_j^k} f_a \cdot S(g_i, p_a)}{|\mathcal{P}_j^k|} \tag{7.2}$$

where f_a is the cgSpan frequency score of p_a, $S(g_i, p_a)$ is the average similarity between scene graph g_i and sub-graph pattern p_a (see eqn. 7.1). For convenience, in Eq. 7.2 each pattern $\mathcal{P}$ is treated as a set of conjunctions and a conjunction p as a set of triplets. After computing $score_{j^k}$ for $j \in \{c, f\}$ for each image x_i and each class k, we apply the softmax function to these scores to obtain the probability of each $\mathcal{P}_j^k$ and select the pattern with the highest probability. E.g., if $\mathcal{P}_c^{office}$ has the highest probability then the PC outputs *office*. Similar to previous work [65], we compute the alignment score between the predictions made by PC and those by $\mathcal{N}$. However, in contrast to previous work, we employ a weighted F1-score instead of percentage, to more accurately reflect the varying number of images correctly and incorrectly classified by $\mathcal{N}$.

The weighted F1 alignment score evaluates the coherence between the predictions made by $\mathcal{N}$ and PC. We compute this alignment with respect to the ground truth labels. Let the prediction made by PC for image x_i be $\dot{y}_i$. The alignment score increases if $y_i = \hat{y}_i = \dot{y}_i$ or if $\hat{y}_i \neq y_i$ and $\dot{y}_i \neq y_i$.

The weighted F1 alignment score is the weighted F1 score between the predictions made by PC and $\mathcal{N}$ relative to the ground truth labels. It is computed between two lists of binary values. The first list compares $\mathcal{N}$'s predictions $(\hat{y}_1, \ldots, \hat{y}_m)$ for images $(x_1, \ldots, x_m)$ to their ground truth $(y_1, \ldots, y_m)$, where 1 indicates $\hat{y}_i = y_i$ and 0 otherwise. The second list compares PC's predicted labels $(\dot{y}_1, \ldots, \dot{y}_m)$ to the ground truth $(y_1, \ldots, y_m)$, with 1 indicating $\dot{y}_i = y_i$ and 0 otherwise. Tab. 7.10 shows the weighted F1 alignment score for patterns extracted from different frameworks. The results demonstrate that the extracted patterns using our framework outperform the baseline and the foundation model based framework used to build the PC effectively and accurately predict whether $\mathcal{N}$ will correctly classify or misclassify a given unseen image from the test set.

We performed a statistical significance test for the weighted F1 scores of the PC and $\mathcal{N}$ to determine whether the observed alignment between the predictions of the image classifier $\mathcal{N}$ and the pattern classifier PC could be due to random chance. Using bootstrapping [42], known for its flexibility and robustness, we computed the p-values for the weighted F1 scores of both models[1]. To assess whether the observed weighted F1 alignment scores could have occurred by chance, we performed a bootstrapped statistical significance test [42] comparing the weighted F1 scores of our proposed PC (using common-sense KG & FM) and $\mathcal{N}$. Bootstrapping was chosen for its flexibility and robustness in estimating p-values. Under the null hypothesis—assuming no difference between the predictions of the two models beyond what would be expected by chance—the resulting p-values were found to be less than 0.001 across all datasets. This

[1] $P - value(F1(\hat{Y}, Y), F1(\dot{Y}, Y))$

Dataset	Baseline	FM-based	(common-sense KG & FM)-based (Ours)
Biased CelebA	0.64	0.66	0.75
Waterbirds	0.90	0.93	0.96
Indoor	0.67	0.63	0.72

Table 7.10: Alignment F1 score comparing $\mathcal{N}$'s predictions on the test split to PC using different frameworks: baseline, FM-based model without common-sense KG, and our framework.

provides strong evidence that the alignment between the models is statistically significant and not attributable to random variation.

• **Can we replace the common-sense KG with a foundation model?** To address this question, we replaced the commonsense knowledge graph (common-sense KG) with a foundation model (LLaVA). Instead of using the common-sense KG to extract attributes, we instructed the LLaVA model to propose possible attributes for an object by using the following prompt "*Return the list of properties that [node] has.*" Subsequently, we inquired about the values of the returned properties one by one. The results in Tab. 7.10 indicate that using our framework that utilizes common-sense KG is more effective at extracting patterns that mirrors the behavior of $\mathcal{N}$, compared to replacing the common-sense KG with foundation model to extract semantic dimensions. These results align with findings from previous work [101, 58, 157, 179, 183], which revealed that multimodel foundation models struggle to provide deep structured visual understanding when used alone, as they often overlook objects' attributes.

Figure 7.8: Left: Top 3 images from the MIT Indoor dataset, manually labeled as *deli* but misclassified as *bakery* due to red-highlighted bakeries found in the extracted graph patterns for both the correct patterns of the *bakery* class and the false patterns of the *deli* class. Bottom 3 images are *deli* examples misclassified as *bakery*, due to the frequently occurring display cases and baked goods, identified by the graph-patterns in the false patterns of *deli* and correct patterns of *bakery*. Right: 6 *bakery* images from the same dataset, manually labeled as *bakery* but visually similar to the *deli* images on the left, further demonstrating the validity and effectiveness of the insights uncovered by our patterns.

Figure 7.9: Confusion matrix from evaluating the ResNet18 network trained on the Indoor dataset on the test split, showing that the model confuses a *Deli* with a *Bakery* more than 50% of the time.

7.4.3 Use Cases

Our framework's extracted patterns have versatile applications. We highlight two: explaining the image classifier's errors and proactively detecting bias in datasets.

Error Analysis.

Confusion matrices are useful to understand which classes often get confused with each other, but they do not explain *why* this happens. Our framework can address this limitation. When an image classifier frequently misclassifies instances of class A as class B, patterns in $(P_c^A \cap P_f^B)$ reveal likely reasons. For instance, in the Indoor dataset, the confusion matrix in Fig. 7.9 shows that *deli* is misclassified as *bakery* 56% of the time. Our framework reveals that the pattern *[goods, are, baked]* $\vee$ *[items, in, display case]* appears both in

correctly classified images of *bakeries* and in images misclassified *delis*. This suggests *delis*, like *bakeries*, have baked goods; or perhaps there are images of *bakeries* that are labeled as *delis* in the training dataset. Looking at some actual images reveals that both scenarios hold: *deli* images often have bread, as highlighted in red in the first row of deli images in Fig. 7.8 and annotation errors within the training dataset exist, as highlighted in the second row of images for *deli*, where the images feature only baked goods and are more likely to be identified by humans as *bakery* rather than *deli*. This overlap leads to noisy data, complicating the training process and contributing to the misclassification of similar images across different classes.

Proactive bias detection.

Our framework can be utilized to identify potential misclassification candidates by using patterns extracted from correctly classified images, denoted as $\mathcal{P}_c^k$. This can be done by leveraging commonsense knowledge graph (commonsense KG) to extract antonyms for each node in $\mathcal{P}_c^k$ and then generate all possible antonym pattern combinations. For instance, consider $\mathcal{P}_c^{female} = [hair, is, long] \wedge [hair, is, blond]$ from CelebA dataset. Based on the commonsense KG, while *hair* has no antonym, *long* and *blond* have antonyms *short* and *brunette*, respectively. This yields the antonym patterns: $([hair, is, short] \wedge [hair, is, blond]) \vee ([hair, is, long] \wedge [hair, is, brunette]) \vee ([hair, is, short] \wedge [hair, is, brunette])$. Images with these antonym patterns are more likely to be misclassified by the image classifier, especially if these patterns are rare or absent in the training data and, therefore, not present in the extracted patterns. This approach was applied to the CelebA and Waterbirds datasets, revealing that the derived antonym patterns align with the filtering used to create unbiased test splits. These test splits often contain images that the classifier misclassifies if it is not robust, as shown by the low test accuracies in Tab. 7.8. This highlights the potential of using patterns derived from correctly classified images to identify gaps in the classifier's knowledge.

Additionally, our framework can be used to identify spurious patterns in the dataset before training by setting predicted labels to the ground truth. The extracted patterns can then be manually inspected for biases, serving as a proactive measure to develop fairer, more robust models.

7.5 Experimental Configurations

7.5.1 Models and Parameter Setup

In this chapter we utilised several existing models. In this section we provide more details about their software and parameter setting.

LLaVA.

LLaVA is an open-source multi-modal model designed to understand and generate content based of visual inputs and textual instructions. We utilised LLaVA-v1.5-7b [91] for it light weight and fast computation. We did not fine tune it on any task. The maximum number of token generated was set to 300. As for the implementation, we used the hugging face implementation [90] with the v1.5-7b model.

OpenIE.

OpenIE is an unsupervised heuristic-based algorithm relying on linguistic rules and dependency parsing to extract information. It has several parameters that can used to control the quality of the extracted triplets. We set four key parameters to control the quality of extracted triplets:

•**Affinity Probability:** This influences the likelihood of word combinations being considered valid relations. We set this to 1/3 to reduce noise, given the long descriptive answers from LLaVA.

•**Maximum Entailment's per Clause:** This parameter controls the maximum number of entailments (i.e., inferred relationships or extractions) that OpenIE will generate per clause. We limit this to 2 to helps focus on the most relevant or confident extractions.

•**Resolve Coreference:** Set to *True*, this parameter ensures pronouns and references are linked to the correct entities, improving extraction accuracy.

•**Triple All Nominals:** By setting this to *True*, we enabled triplet extraction from nominal phrases, increasing the number of extracted relations. For example, with this setting, OpenIE would extract (leather, is, armchair) form *leather armchair*; without it, no triplet would be produced.

For the implementation, we used OpenIE module from the Stanza library [113]. Stanza (v1.7.0) is a Python natural language analysis package from Stanford. It

contains tools, which have different use cases including triplets extraction using openIE [4]. Stanza can process text in more than 70 languages so it is diverse.

Cgspan.

cgSpan is designed to extract frequent closed sub-graph patterns from a set of graphs. The algorithm is controlled by several key parameters that influence its performance and the nature of the patterns it identifies. One important parameter is the minimum support threshold, which determines the frequency a sub-graph must have across the dataset to be considered "frequent" Setting this threshold too high might cause the algorithm to miss important patterns, while setting it too low could result in a large number of trivial or irrelevant patterns. For that we choose 50% to balance between quantity and quality.
Another parameters are the minimum and maximum sub-graph size, which controls the size of the extracted sub-graphs to be between the minimum and maximum size. We set only the minimum sub-graph size to one edge with no limit on the maximum to allow both simple and complex patterns. For other parameters we used the default settings.

For the implementation of cgSpan (v0.2.2), we used the code [133] introduced by the authors of the cgspan paper [134].

7.5.2 Computing infrastructure

The experiments in this chapter were conducted using an NVIDIA Tesla V100-SXM2-32GB GPU, operating within a Red Hat Enterprise Linux 8.9 (Ootpa) environment. This setup featured a single GPU with 32 GB of memory (32768 MiB), a compute capability of 7.0, and 5110 CUDA cores. The GPU includes 640 Tensor cores and operates at a clock speed of 135 MHz. In terms of floating-point performance, the GPU delivers 31.33 TFLOPS for FP16, 15.67 TFLOPS for FP32, and 7.834 TFLOPS for FP64 operations. The system utilized NVIDIA Driver version 550.54.15 and CUDA 11.6.2 for all computations.

7.6 Related Work

Systematic Error Analysis

Identifying systematic errors in neural networks has been a focal point in recent studies [140, 19, 74, 99, 57]. Traditional methods typically use predefined

semantic dimensions, such as gender and ethnicity, or object labels to pinpoint failure cases. For example, [140] uses object labels to find correlations between objects in images and model predictions, while [19] examines biases in facial analysis algorithms using predefined phenotypes. Other approaches manipulate predefined image attributes like style or blurriness to assess classifier robustness [57], or analyze rare subgroups [99].

Unlike these methods, our framework employs dynamic semantic attributes instead of fixed ones and introduces sub-graph mining to reveal deeper relationships between attributes, providing a new layer of understanding to the analysis. We focus on both patterns responsible for correct and false predictions, as both reveal a lot about network bias.

Another study [41] used image captions to find commonalities between groups of images. While this method does not rely on fixed attributes, it uses captions that fail to capture the detailed structural relationships among objects in the image, which our scene descriptors provide. Image captions have also been used to generate failure-inducing images [163, 15]. However, these approaches struggle with generating complex scenes and often miss critical details. Additionally, several methods [37, 67, 143, 168] use clustering of misclassified images in various latent spaces, but this has been found to produce incoherent groups, complicating the task of associating captions or evaluating impacts for corrections [47].

Scene Graphs

The goal of scene graph generation is to parse an image to create a structured representation, ultimately aiming for a complete understanding of visual scenes. The following two approaches are typically used: (1) a two-stage method that first detects objects and then recognizes relationships [23, 112, 175, 174, 182, 101], which gained significant attention with the release of the VG dataset; and (2) a one-stage method that simultaneously detects and recognizes objects and relations [92, 30, 156, 88]. A comprehensive review of scene graph generation methods was conducted in [186]. Existing work focuses primarily on detecting objects and their relationships, often lacking detailed object descriptions (e.g., color, shape) and contextual information (e.g., weather, lighting, image mode, etc.) that could be the cause of model bias. Our scene descriptor, being unsupervised, addresses this gap by providing these crucial details for a better understanding of the visual scene.

Frequent Sub-Graph Mining

Frequent sub-graph mining is a field within data mining dedicated to identifying common sub-graphs across graph datasets. Numerous studies have contributed to this field, [63, 172, 134, 81], with an extensive review [69]. We chose cgSpan [134] in our study as it finds only those frequent sub-graphs that have no supergraphs with the same support, known as frequent closed sub-graphs. This approach provides a compact representation, encapsulating all frequent sub-graphs efficiently without the redundancy typically observed in other methods. Additionally, it offers enhanced processing speed compared to techniques that enumerate all frequent sub-graphs.

7.7 Conclusion

We introduced a dynamic, unsupervised framework that generates graph-based patterns, offering deeper and more comprehensive explanations for a better understanding of classifier behavior compared to the shallow, manually predefined dimensions used in previous work. The framework comprises two components: a novel scene descriptor that generates detailed and adaptable scene graphs for images, and a pattern extractor that identifies key patterns distinguishing between correct and incorrect classifications. Empirical evaluations demonstrate the robustness and effectiveness of our approach. Our scene graphs are richer and more detailed compared to manual annotations from VG. The pattern miner achieves high F1-scores in predicting classifier behavior, with extracted patterns proving valuable for error analysis and bias detection, fostering fairer and more robust models. This work presents the first approach for generating detailed graph-based explanations for image classifiers in an unsupervised fashion, surpassing prior efforts that rely on simplistic, manually annotated data.

Chapter 8

Conclusion

The explainability of neural networks remains a critical challenge in artificial intelligence (AI). Current methods face several limitations. Many rely on semantically segmented and labeled data, which are both time-consuming and labor-intensive to generate. Others provide explanations in the form of highlighted regions, which are subjective and depend heavily on user interpretation. Furthermore, many techniques are architecture-specific and not suited for interpreting models that learn embeddings. A key challenge addressed in this thesis is the absence of standardized evaluation frameworks for explanations. This stems from the lack of a fixed form for explanations—a desirable property that allows for diverse and flexible explanation formats. However, this flexibility also makes it difficult to compare or evaluate different explanations on a common scale. As a result, we are motivated to develop our evaluation methods tailored to the nature of the explanations proposed in this thesis.

This thesis addresses these challenges by introducing unsupervised frameworks that eliminate reliance on additional labeled datasets, generate structured and objective explanations, and apply across different AI architectures. Notably, these frameworks leverage training data and knowledge graphs (KGs) to provide explanations without the need for semantically segmented data, ensuring results are not influenced by user subjectivity. Furthermore, this work expands explainability to embedding models, a largely unexplored area, and introduces quantitative evaluation methods for assessing explanations.

The proposed frameworks tackle multiple dimensions of explainability, including neural activity interpretation and output-based explanations using knowledge graphs and foundational models. Three primary frameworks were presented:

two for explaining image classifiers with distinct methodologies and one for interpreting black-box knowledge graph embedding models. We also demonstrated the practical utility of the extracted explanations by applying them to tasks such as zero-shot image classification, error analysis, and proactive bias detection, showcasing their versatility and impact beyond improving model transparency.

Additionally, this thesis highlights the utility of feature selection as a method for understanding AI models. By identifying patterns in input data that influence model behavior, feature selection provides valuable insights into how models process information. This technique was a central component of the research and proved instrumental in improving the interpretability of neural networks.

Despite these advancements, certain limitations remain. The performance of the proposed frameworks is constrained by the coverage of the knowledge graphs. For example, the methods presented in Chapters 4 and 7 depend on the presence of classification classes (Chapter 4) or objects (Chapter 7) in the KG to provide semantic context. Consequently, these methods are less effective for domain-specific problems lacking corresponding knowledge graphs or sufficient semantic information. We hope that the findings in this thesis encourage domain experts to develop their own knowledge graphs.

8.1 Future Directions

While this work makes significant progress in explainability, several open challenges remain for future research:

Developing a universal evaluation framework

Existing explainability methods might target similar AI models, such as image classifiers, but produce explanations in various formats (e.g., highlighted regions, concepts). This diversity complicates evaluation and comparison. Future work could focus on designing an evaluation framework that is invariant to explanation format, enabling consistent assessment of how well explanations align with model behavior.

Benchmarking explainability methods

A universal evaluation framework would facilitate the systematic benchmarking of existing explainability methods. This would help identify their strengths and weaknesses, foster comparisons, and drive improvements across the field.

Addressing the validity of multiple explanations

Changing the explainability framework's parameters in existing frameworks, including those proposed in this thesis, can result in different explanations that still align with the AI model's behavior. This raises the question: which explanation truly reflects what the model has learned? Future research should explore ways to resolve this ambiguity and establish criteria for identifying the most accurate explanation.

Explaining pre-trained models without access to training data

The frameworks in this thesis rely on access to training data to link patterns in the data to the model's predictions. However, pre-trained models, such as transformer-based models like LLaVA, are often trained on inaccessible or massive datasets. Developing techniques to explain such models without relying on their training data presents a significant challenge for future work.

Developing metrics to measure dataset bias

Given that neural networks are driven by data, they tend to exploit any obvious features in the data to learn their task. For instance, a network might rely on the background of an image if it consistently appears with certain classes. These "shortcuts" can be as obvious as backgrounds or as subtle as low-contrast elements, logos, or watermarks. Future work should focus on developing metrics that assess the biasses of training datasets and detect spurious correlations between image features and class labels. This could be done using pattern mining techniques or further extending our framework in 7 to provide bias metric for the training datasets. While much work has been done to explain machine learning models, there is an equally important need to explore and understand the training data itself.

8.2 Closing Remarks

This thesis contributes to the growing field of explainable AI by addressing critical gaps in current methods and extending explainability to embedding models. By leveraging knowledge graphs and introducing novel evaluation frameworks, we provide structured, objective explanations that enhance the interpretability of AI models. These contributions advance the understanding of neural network behavior and lay the groundwork for future innovations in explainable AI.

Bibliography

[1] Aaronson, S. Building trust in ai, 2015.

[2] Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I. J., Hardt, M., and Kim, B. Sanity checks for saliency maps. In *Neural Information Processing Systems* (2018).

[3] Agrawal, R., Imielinski, T., and Swami, A. N. Mining association rules between sets of items in large databases. In *SIGMOD 1993* (1993), pp. 207–216.

[4] Angeli, G., Johnson, M., and Manning, C. D. Leveraging linguistic structure for open domain information extraction. In *Annual Meeting of the Association for Computational Linguistics* (2015).

[5] Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., and Ives, Z. Dbpedia: A nucleus for a web of open data. In *ISWC/ASWC* (2007), p. 722–735.

[6] Azzolin, S., Longa, A., Barbiero, P., Lio, P., Passerini, A., et al. Global explainability of gnns via logic combination of learned concepts. In *11th International Conference on Learning Representations, ICLR 2023* (2023), International Conference on Learning Representations, ICLR.

[7] Baader, F., Calvanese, D., McGuinness, D., Patel-Schneider, P., Nardi, D., et al. *The description logic handbook: Theory, implementation and applications*. Cambridge university press, 2003.

[8] Bau, D., Zhou, B., Khosla, A., Oliva, A., and Torralba, A. Network dissection: Quantifying interpretability of deep visual representations. In *CVPR* (2017), pp. 3319–3327.

[9] Bauer, L., Wang, Y., and Bansal, M. Commonsense for generative multi-hop question answering tasks. In *Conference on Empirical Methods in Natural Language Processing* (2018).

[10] Betz, P., Meilicke, C., and Stuckenschmidt, H. Adversarial explanations for knowledge graph embeddings. In *IJCAI 2022* (2022), L. D. Raedt, Ed., pp. 2820–2826.

[11] Bhatt, U., Xiang, A., and Shubham Sharma, t. Explainable machine learning in deployment. In *FAT* 2020* (2020), pp. 648–657.

[12] Binder, A., Montavon, G., Lapuschkin, S., Müller, K.-R., and Samek, W. Layer-wise relevance propagation for neural networks with local renormalization layers. In *Artificial Neural Networks and Machine Learning–ICANN 2016: 25th International Conference on Artificial Neural Networks, Barcelona, Spain, September 6-9, 2016, Proceedings, Part II 25* (2016), pp. 63–71.

[13] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. Freebase: A collaboratively created graph database for structuring human knowledge. In *SIGMOD 2008* (2008), p. 1247–1250.

[14] Bordes, A., Usunier, N., García-Durán, A., Weston, J., and Yakhnenko, O. Translating embeddings for modeling multi-relational data. In *NeurIPs* (2013), pp. 2787–2795.

[15] Boreiko, V., Hein, M., and Metzen, J. H. Identifying systematic errors in object detectors with the scrod pipeline. *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)* (2023), 4092–4101.

[16] Breiman, L. Random forests. *Machine Learning 45* (2001), 5–32.

[17] Brunner, T., Diehl, F., and Knoll, A. Copy and paste: A simple but effective initialization method for black-box adversarial attacks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019).

[18] Buijsman, S. Defining explanation and explanatory depth in xai. *Minds and Machines 32*, 3 (2022), 563–584.

[19] Buolamwini, J., and Gebru, T. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on Fairness, Accountability and Transparency, FAT 2018, 23-24 February 2018, New York, NY, USA* (2018), vol. 81 of *Proceedings of Machine Learning Research*, PMLR, pp. 77–91.

[20] Chandrahas, Sengupta, T., Pragadeesh, C., and Talukdar, P. P. Inducing interpretability in knowledge graph embeddings. In *ICON 2020* (2020), P. Bhattacharyya, D. M. Sharma, and R. Sangal, Eds., pp. 70–75.

[21] Chefer, H., Gur, S., and Wolf, L. Transformer interpretability beyond attention visualization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 782–791.

[22] Chen, J., Geng, Y., Chen, Z., Horrocks, I., Pan, J. Z., and Chen, H. Knowledge-aware zero-shot learning: Survey and perspective. In *IJCAI* (2021), ijcai.org, pp. 4366–4373.

[23] Chen, T., Yu, W., Chen, R., and Lin, L. Knowledge-embedded routing network for scene graph generation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 6156–6164.

[24] Chen, Y., Wu, L., and Zaki, M. J. Bidirectional attentive memory networks for question answering over knowledge bases. *ArXiv abs/1903.02188* (2019).

[25] Chen, Z., Bei, Y., and Rudin, C. Concept whitening for interpretable image recognition. *Nature Machine Intelligence 2*, 12 (2020), 772–782.

[26] Cheng, W., Kasneci, G., Graepel, T., Stern, D. H., and Herbrich, R. Automated feature generation from structured knowledge. In *CIKM, 2011* (2011), pp. 1395–1404.

[27] Cheng, X., Lu, J., Feng, J., Yuan, B., and Zhou, J. Scene recognition with objectness. *Pattern Recognit. 74* (2018), 474–487.

[28] Ciatto, G., Schumacher, M. I., Omicini, A., and Calvaresi, D. Agent-based explanations in ai: Towards an abstract framework. In *Explainable, Transparent Autonomous Agents and Multi-Agent Systems* (Cham, 2020), D. Calvaresi, A. Najjar, M. Winikoff, and K. Främling, Eds., Springer International Publishing, pp. 3–20.

[29] Clement, F., Yang, J., and Cheng, I. Feature cam: Interpretable ai in image classification. *arXiv e-prints* (2024), arXiv–2403.

[30] Cong, Y., Yang, M. Y., and Rosenhahn, B. Reltr: Relation transformer for scene graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 45* (2022), 11169–11183.

[31] Costabello, L., Lecue, F., Giannotti, F., Guidotti, R., Hitzler, P., Minervini, P., and Sarker, K. On explainable ai: From theory to motivation, applications and limitations. In *A tutorial at AAAI 2019* (2019).

[32] Cover, T. M., and Hart, P. E. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theory 13*, 1 (1967), 21–27.

[33] Dai, Z., Li, L., and Xu, W. Cfo: Conditional focused neural question answering with large-scale knowledge bases. *ArXiv abs/1606.01994* (2016).

[34] Dalvi, F., Nortonsmith, A., and *et al*, A. B. Neurox: A toolkit for analyzing individual neurons in neural networks. In *AAAI 2019* (2019), pp. 9851–9852.

[35] de Sousa Ribeiro, M., and Leite, J. Aligning artificial neural networks and ontologies towards explainable AI. In *AAAI 2021* (2021), pp. 4932–4940.

[36] Deng, J., Dong, W., Socher, R., and *et al*, L. L. Imagenet: A large-scale hierarchical image database. In *CVPR 2009* (2009), pp. 248–255.

[37] d'Eon, G., d'Eon, J., Wright, J. R., and Leyton-Brown, K. The spotlight: A general method for discovering systematic errors in deep learning models. *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency* (2021).

[38] Dhore, V., Bhat, A., Nerlekar, V., Chavhan, K., and Umare, A. Enhancing explainable ai: A hybrid approach combining gradcam and lrp for cnn interpretability. *arXiv preprint arXiv:2405.12175* (2024).

[39] Doshi-Velez, F., and Kim, B. Towards a rigorous science of interpretable machine learning, 2017.

[40] Doveh, S., Arbelle, A., Harary, S., Panda, R., Herzig, R., Schwartz, E., Kim, D., Giryes, R., Feris, R. S., Ullman, S., and Karlinsky, L. Teaching structured vision and language concepts to vision and language models. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 2657–2668.

[41] Dunlap, L., Zhang, Y., Wang, X., Zhong, R., Darrell, T., Steinhardt, J., Gonzalez, J. E., and Yeung-Levy, S. Describing differences in image sets with natural language. *ArXiv abs/2312.02974* (2023).

[42] Efron, B. Bootstrap methods: Another look at the jackknife. *Annals of Statistics 7* (1979), 1–26.

[43] Endres, D., and Földiák, P. Interpreting the neural code with formal concept analysis. In *NIPS* (2008), pp. 425–432.

[44] Fong, R., and Vedaldi, A. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *CVPR* (2018), pp. 8730–8738.

[45] Galárraga, L. Effects of locality and rule language on explanations for knowledge graph embeddings. *CoRR abs/2302.06967* (2023).

[46] Galkin, M., Denis, E. G., Wu, J., and Hamilton, W. L. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. In *ICLR 2022* (2022).

[47] Gao, I., Ilharco, G., Lundberg, S. M., and Ribeiro, M. T. Adaptive testing of computer vision models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2022), 3980–3991.

[48] Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R. S., Brendel, W., Bethge, M., and Wichmann, F. Shortcut learning in deep neural networks. *Nature Machine Intelligence 2* (2020), 665 – 673.

[49] Geng, Y., Chen, J., and Zhiquan Ye, e. Explainable zero-shot learning via attentive graph convolutional network and KGs. *SW 12* (2021).

[50] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR 2015* (2015).

[51] Gunning, D., Vorm, E., Wang, J. Y., and Turek, M. Darpa's explainable ai (xai) program: A retrospective. *Applied AI Letters 2*, 4 (2021), e61.

[52] Gusmão, A. C., Correia, A. H. C., Bona, G. D., and Cozman, F. G. Interpreting embedding models of knowledge bases: A pedagogical approach. *CoRR abs/1806.09504* (2018).

[53] Halliwell, N., Gandon, F., and Lécué, F. User scored evaluation of non-unique explanations for relational graph convolutional network link prediction on knowledge graphs. In *K-CAP* (2021), pp. 57–64.

[54] Haykin, S. *Neural networks: a comprehensive foundation*. Prentice Hall, 1994.

[55] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 770–778.

[56] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[57] Hendrycks, D., Basart, S., Mu, N., Kadavath, S., Wang, F., Dorundo, E., Desai, R., Zhu, T., Parajuli, S., Guo, M., Song, D., Steinhardt, J., and Gilmer, J. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021* (2021), IEEE, pp. 8320–8329.

[58] Herzig, R., Mendelson, A., Karlinsky, L., Arbelle, A., Feris, R. S., Darrell, T., and Globerson, A. Incorporating structured representations into pretrained vision and language models using scene graphs. *ArXiv abs/2305.06343* (2023).

[59] Ho, V. T., Stepanova, D., Gad-Elrab, M. H., Kharlamov, E., and Weikum, G. Rule learning from knowledge graphs guided by embedding models. In *ISWC* (2018), vol. 11136, pp. 72–90.

[60] Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutierrez, C., Gayo, J. E. L., Kirrane, S., Neumaier, S., Polleres, A., Navigli, R., Ngomo, A.-C. N., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., and Zimmermann, A. Knowledge graphs. *ACM Computing Surveys (CSUR) 54* (2020), 1 – 37.

[61] Horta, V. A. C., and Mileo, A. Towards explaining deep neural networks through graph analysis. In *DB and Exp. Sys. Appl.* (2019), pp. 155–165.

[62] Ibrahim, N., AboulEla, S., Ibrahim, A. F., and Kashef, R. A survey on augmenting knowledge graphs (kgs) with large language models (llms): models, evaluation metrics, benchmarks, and challenges. *Discov. Artif. Intell. 4* (2024), 76.

[63] Inokuchi, A., Washio, T., and Motoda, H. An apriori-based algorithm for mining frequent substructures from graph data. In *European Conference on Principles of Data Mining and Knowledge Discovery* (2000).

[64] Ismaeil, Y., Stepanova, D., Tran, T.-K., and Blockeel, H. Feabi: A feature selection-based framework for interpreting kg embeddings. In *The Semantic Web – ISWC 2023: 22nd International Semantic Web Conference, Athens, Greece, November 6–10, 2023, Proceedings, Part I* (Berlin, Heidelberg, 2023), Springer-Verlag, p. 599–617.

[65] Ismaeil, Y., Stepanova, D., Tran, T.-K., Saranrittichai, P., Domokos, C., and Blockeel, H. Towards neural network interpretability using commonsense knowledge graphs. In *International Workshop on the Semantic Web* (2022).

[66] Jain, N., Kalo, J.-C., Balke, W.-T., and Krestel, R. Do embeddings actually capture knowledge graph semantics? In *ESWC* (2021).

[67] Jain, S., Lawrence, H., Moitra, A., and Madry, A. Distilling model failures as directions in latent space. *ArXiv abs/2206.14754* (2022).

[68] Ji, S., Pan, S., Cambria, E., Marttinen, P., and Yu, P. S. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems 33* (2020), 494–514.

[69] Jiang, C., Coenen, F., and Zito, M. A. A. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review 28* (2012), 75 – 105.

[70] Johnson, J., Krishna, R., Stark, M., Li, L.-J., Shamma, D., Bernstein, M., and Fei-Fei, L. Image retrieval using scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 3668–3678.

[71] Johnson, W. L. Agents that learn to explain themselves. In *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 2* (1994), AAAI Press / The MIT Press, pp. 1257–1263.

[72] Jović, A., Brkić, K., and Bogunović, N. A review of feature selection methods with applications. In *2015 38th international convention on information and communication technology, electronics and microelectronics (MIPRO)* (2015), Ieee, pp. 1200–1205.

[73] Kampffmeyer, M., Chen, Y., Liang, X., Wang, H., Zhang, Y., and Xing, E. P. Rethinking knowledge graph propagation for zero-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2019).

[74] Kim, B., Wattenberg, M., Gilmer, J., Cai, C. J., Wexler, J., Viégas, F. B., and Sayres, R. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018* (2018), vol. 80 of *Proceedings of Machine Learning Research*, PMLR, pp. 2673–2682.

[75] Kipf, T. N., and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR 2017* (2017).

[76] K.Pavya, and B.Srinivasan. Feature selection techniques in data mining: A study. *international journal of scientific development and research 2* (2017), 594–598.

[77] Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., Chen, S., Kalantidis, Y., Li, L.-J., Shamma, D. A., Bernstein, M. S., and Fei-Fei, L. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision 123* (2016), 32 – 73.

[78] Krogel, M., Rawles, S. A., Železný, F., Flach, P. A., Lavrac, N., and Wrobel, S. ILP 2003. vol. 2835, pp. 197–214.

[79] Kullback, S., and Leibler, R. A. On information and sufficiency. *Annals of Mathematical Statistics 22* (1951), 79–86.

[80] Kumar, A., Singh, S. S., Singh, K., and Biswas, B. Link prediction techniques, applications, and performance: A survey. *Physica A-statistical Mechanics and Its Applications 553* (2020), 124–289.

[81] Kuramochi, M., and Karypis, G. Finding frequent patterns in a large sparse graph. *Data Mining and Knowledge Discovery 11* (2004), 243–271.

[82] Kyrimi, E., McLachlan, S., Wohlgemut, J. M., Perkins, Z. B., Lagnado, D. A., and Marsh, W. Explainable ai: definition and attributes of a good explanation for health ai. *AI and Ethics* (2025), 1–14.

[83] Lajus, J., Galárraga, L., and Suchanek, F. M. Fast and exact rule mining with AMIE 3. In *ESWC 2020* (2020), vol. 12123, pp. 36–52.

[84] Lavrač, N., Škrlj, B., and Robnik-Šikonja, M. Propositionalization and embeddings: Two sides of the same coin. *Mach. Learn. 109*, 7 (2020), 1465–1507.

[85] Lawler, I., and Sullivan, E. Model explanation versus model-induced explanation. *Foundations of Science 26* (2021), 1049–1074.

[86] Lécué, F. On the role of knowledge graphs in explainable AI. *Semantic Web 11*, 1 (2020), 41–51.

[87] LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *nature 521*, 7553 (2015), 436–444.

[88] Li, R., Zhang, S., and He, X. Sgtr: End-to-end scene graph generation with transformer. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 19464–19474.

[89] Li, Z., and Xu, C. Discover the unknown biased attribute of an image classifier. *2021 ICCV* (2021), 14950–14959.

[90] Liu, H. Llava v1.5: Language-image pretraining model. https://huggingface.co/liuhaotian/llava-v1.5-13b, 2023. Accessed: January 1, 2024.

[91] Liu, H., Li, C., Li, Y., and Lee, Y. J. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2024), pp. 26296–26306.

[92] Liu, H., Yan, N., Mortazavi, M. S., and Bhanu, B. Fully convolutional scene graph generation. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 11541–11551.

[93] Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. *2015 IEEE International Conference on Computer Vision (ICCV)* (2014), 3730–3738.

[94] Luo, D., Cheng, W., Xu, D., Yu, W., Zong, B., Chen, H., and Zhang, X. Parameterized explainer for graph neural network. *Advances in neural information processing systems 33* (2020), 19620–19631.

[95] Ma, C., Chen, Y., Wu, T., Khan, A., and Wang, H. Large language models meet knowledge graphs for question answering: Synthesis and opportunities. *arXiv e-prints* (2025), arXiv–2505.

[96] Ma, Z., Hong, J., Gul, M. O., Gandhi, M., Gao, I., and Krishna, R. @ crepe: Can vision-language foundation models reason compositionally? *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 10910–10921.

[97] McInnes, L., Healy, J., and Astels, S. hdbscan: Hierarchical density based clustering. *J. Open Source Softw. 2* (2017), 205.

[98] Meilicke, C., Chekol, M. W., Ruffinelli, D., and Stuckenschmidt, H. Anytime bottom-up rule learning for knowledge graph completion. In *IJCAI 2019, 2019* (2019), pp. 3137–3143.

[99] Metzen, J. H., Hutmacher, R., Hua, N. G., Boreiko, V., and Zhang, D. Identification of systematic errors of image classifiers on rare subgroups. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)* (2023), 5041–5050.

[100] Mežnar, S., Lavrač, N., and Škrlj, B. Snore: Scalable unsupervised learning of symbolic node representations. *IEEE Access 8* (2020), 212568–212588.

[101] Mitra, C., Huang, B., Darrell, T., and Herzig, R. Compositional chain-of-thought prompting for large multimodal models. *ArXiv abs/2311.17076* (2023).

[102] Morgan, J. N., and Sonquist, J. A. Problems in the analysis of survey data, and a proposal. *Journal of the American Statistical Association 58* (1963), 415–434.

[103] Mu, J., and Andreas, J. Compositional explanations of neurons. *Advances in Neural Information Processing Systems 33* (2020), 17153–17163.

[104] Nandwani, Y., Gupta, A., Agrawal, A., Chauhan, M. S., Singla, P., and Mausam. Oxkbc: Outcome explanation for factorization based knowledge base completion. In *AKBC 2020* (2020).

[105] Nayak, N. V., and Bach, S. H. Zero-shot learning with common sense knowledge graphs. *CoRR abs/2006.10713* (2020).

[106] Nguyen, A. M., Dosovitskiy, A., and Jason Yosinski, e. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Neurips 2016* (2016), pp. 3387–3395.

[107] Nguyen, D. Q., Nguyen, T. D., Nguyen, D. Q., and Phung, D. Q. A novel embedding model for knowledge base completion based on convolutional neural network. In *NAACL* (2018).

[108] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[109] Pellegrino, M. A., Altabba, A., Garofalo, M., Ristoski, P., and Cochez, M. Geval: A modular and extensible evaluation framework for graph embedding techniques. In *ESWC 2020* (2020), vol. 12123, pp. 565–582.

[110] Portisch, J., Heist, N., and Paulheim, H. Knowledge graph embedding for data mining vs. knowledge graph embedding for link prediction - two sides of the same coin? *Semantic Web 13*, 3 (2022), 399–422.

[111] Pudil, P., Novovičová, J., and Kittler, J. Floating search methods in feature selection. *Pattern Recognition Letters 15*, 11 (1994), 1119–1125.

[112] Qi, M., Li, W., Yang, Z., Wang, Y., and Luo, J. Attentive relational networks for mapping images to scene graphs. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), 3952–3961.

[113] Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (2020).

[114] Quattoni, A., and Torralba, A. Recognizing indoor scenes. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 413–420.

[115] Quattoni, A., and Torralba, A. Recognizing indoor scenes. *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009), 413–420.

[116] Reinanda, R., Meij, E., and de Rijke, M. Knowledge graphs: An information retrieval perspective. *Found. Trends Inf. Retr. 14* (2020), 289–444.

[117] Ribeiro, M. T., Singh, S., and Guestrin, C. "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).

[118] Ristoski, P., and Paulheim, H. A comparison of propositionalization strategies for creating features from linked open data. In *LOD Workshop at ECML PKDD 2014* (2014), I. Tiddi, M. d'Aquin, and N. Jay, Eds.

[119] Ristoski, P., Rosati, J., Noia, T. D., Leone, R. D., and Paulheim, H. Rdf2vec: RDF graph embeddings and their applications. *Semantic Web 10*, 4 (2019), 721–752.

[120] Rossi, A., Firmani, D., Merialdo, P., and Teofili, T. Explaining link prediction systems based on knowledge graph embeddings. In *International conference on Management of Data 2022* (2022), SIGMOD '22, Association for Computing Machinery, p. 2062–2075.

[121] Roy, A., Ghosal, D., Cambria, E., Majumder, N., Mihalcea, R., and Poria, S. Improving zero shot learning baselines with commonsense knowledge. *CoRR abs/2012.06236* (2020).

[122] Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks. In *International Conference on Learning Representations* (2020).

[123] Sainz, O., García-Ferrero, I., Agerri, R., de Lacalle, O. L., Rigau, G., and Agirre, E. GoLLIE: Annotation guidelines improve zero-shot information-extraction. In *The Twelfth International Conference on Learning Representations* (2024).

[124] Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., and Müller, K. Explainable ai: Interpreting, explaining and visualizing deep learning. *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (2019).

[125] Sarker, M. K., Xie, N., Doran, D., Raymer, M., and Hitzler, P. Explaining trained neural networks with semantic web technologies: First steps. In *NeSy* (2017).

[126] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks 20*, 1 (2008), 81–102.

[127] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE transactions on neural networks 20*, 1 (2008), 61–80.

[128] Scorzato, L. Reliability and interpretability in science and deep learning. *Minds and Machines 34*, 3 (2024), 27.

[129] Scott, A. C., Clancey, W. J., Davis, R., and Shortliffe, E. H. Explanation capabilities of production-based consultation systems. *American Journal of Computational Linguistics* (Feb. 1977), 1–50. Microfiche 62.

[130] Segal, M., and Xiao, Y. Multivariate random forests. *WIREs Data Mining and Knowledge Discovery 1* (2011).

[131] Selvaraju, R. R., Chattopadhyay, P., Elhoseiny, M., Sharma, T., Batra, D., Parikh, D., and Lee, S. Choose your neuron: Incorporating domain knowledge through neuron-importance. In *ECCV (13)* (2018), pp. 540–556.

[132] Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision 128* (2016), 336 – 359.

[133] Shaul, Z., and Naaz, S. cgspan. https://github.com/NaazS03/cgSpan, 2021. Accessed: January 1, 2024.

[134] Shaul, Z., and Naaz, S. cgspan: Closed graph-based substructure pattern mining. In *2021 IEEE International Conference on Big Data (Big Data)* (2021), pp. 4989–4998.

[135] Shi, B., and Weninger, T. Open-world knowledge graph completion. *ArXiv abs/1711.03438* (2018).

[136] Shortliffe, E. H., and Buchanan, B. G. *A model of inexact reasoning in medicine*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990, p. 259–275.

[137] Shrikumar, A., Greenside, P., and Kundaje, A. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (2017), ICML'17, JMLR.org, p. 3145–3153.

[138] Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *ICLR 2014* (2014).

[139] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *ICLR 2015* (2015).

[140] Singh, K. K., Mahajan, D., Grauman, K., Lee, Y. J., Feiszli, M., and Ghadiyaram, D. Don't judge an object by its context: Learning to overcome contextual bias. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020* (2020), Computer Vision Foundation / IEEE, pp. 11067–11075.

[141] Singhal, A. Introducing the knowledge graph: things, not strings. https://googleblog.blogspot.com/2012/05/introducing-knowledge-graph-things-not.html, 2012.

[142] Smilkov, D., Thorat, N., Kim, B., Viégas, F., and Wattenberg, M. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825* (2017).

[143] Sohoni, N. S., Dunnmon, J. A., Angus, G., Gu, A., and Ré, C. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. *ArXiv abs/2011.12945* (2020).

[144] Speer, R., Chin, J., and Havasi, C. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI Conference on Artificial Intelligence* (2016).

[145] Speer, R., Chin, J., and Havasi, C. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI 2017* (2017), pp. 4444–4451.

[146] Springenberg, J. T., Dosovitskiy, A., Brox, T., and Riedmiller, M. A. Striving for simplicity: The all convolutional net. In *ICLR (workshop track)* (2015).

[147] Srinivas, S., and Fleuret, F. Full-gradient representation for neural network visualization. *Advances in neural information processing systems 32* (2019).

[148] Steenwinckel, B., Vandewiele, G., Weyns, M., Agozzino, T., Turck, F. D., and Ongenae, F. INK: knowledge graph embeddings for node classification. *Data Min. Knowl. Discov. 36*, 2 (2022), 620–667.

[149] Suchanek, F. M., Kasneci, G., and Weikum, G. Yago: A core of semantic knowledge. In *WWW 2007* (2007).

[150] Sun, C., Xu, H., Chen, Y., and Zhang, D. As-xai: Self-supervised automatic semantic interpretation for cnn. *Advanced Intelligent Systems 6*, 12 (2024), 2400359.

[151] Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International conference on machine learning* (2017), PMLR, pp. 3319–3328.

[152] Swartout, W. R. *Explaining and Justifying Expert Consulting Programs.* Springer New York, New York, NY, 1985, pp. 254–271.

[153] Swartout, W. R., and Moore, J. D. Explanations in knowledge systems: design for explainable expert systems. *IEEE Expert 6*, 3 (1991), 58–64.

[154] Tandon, N., de Melo, G., Suchanek, F. M., and Weikum, G. Webchild: harvesting and organizing commonsense knowledge from the web. In *WSDM* (2014), ACM.

[155] Tang, J., Wang, Y., Luo, Y., Fu, J., Zhang, Y., Li, Y., Xiao, Z., Lou, Y., Qiu, Y., and Zhu, F. Computational advances of tumor marker selection and sample classification in cancer proteomics. *Computational and Structural Biotechnology Journal 18* (2020), 2012–2025.

[156] Teng, Y., and Wang, L. Structured sparse r-cnn for direct scene graph generation. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2021), 19415–19424.

[157] Thrush, T., Jiang, R., Bartolo, M., Singh, A., Williams, A., Kiela, D., and Ross, C. Winoground: Probing vision and language models for visio-linguistic compositionality. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 5228–5238.

[158] Toutanova, K., and Chen, D. Observed versus latent features for knowledge base and text inference. *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality* (2015).

[159] Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., and Bouchard, G. Complex embeddings for simple link prediction. In *Proceedings of The 33rd International Conference on Machine Learning* (New York, New York, USA, 20–22 Jun 2016), M. F. Balcan and K. Q. Weinberger, Eds., vol. 48 of *Proceedings of Machine Learning Research*, PMLR, pp. 2071–2080.

[160] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. Complex embeddings for simple link prediction. In *ICML* (2016), pp. 2071–2080.

[161] Turki, H., Owodunni, A. T., Taieb, M. A. H., Bile, R. F., Aouicha, M. B., and Zouhar, V. A decade of scholarly research on open knowledge graphs. In *International Conference on Language Resources and Evaluation* (2023).

[162] Vashishth, S., Sanyal, S., Nitin, V., and Talukdar, P. P. Composition-based multi-relational graph convolutional networks. In *ICLR 2020* (2020).

[163] Vendrow, J., Jain, S., Engstrom, L., and Madry, A. Dataset interfaces: Diagnosing model failures using controllable counterfactual generation. *ArXiv abs/2302.07865* (2023).

[164] Vrandečić, D., and Krötzsch, M. Wikidata. *Communications of the ACM 57* (2014), 78 – 85.

[165] Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., and Hu, X. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops* (2020), pp. 24–25.

[166] Wang, Q., Mao, Z., Wang, B., and Guo, L. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering 29*, 12 (2017), 2724–2743.

[167] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *ArXiv abs/2002.10957* (2020).

[168] Wiles, O., Albuquerque, I., and Gowal, S. Discovering bugs in vision models using off-the-shelf image generation and captioning. *ArXiv abs/2208.08831* (2022).

[169] Xian, Y., Lampert, C. H., Schiele, B., and Akata, Z. Zero-shot learning—a comprehensive evaluation. *IEEE T. on Pat. Anal. and M. I. 41*, 9 (2019), 2251–2265.

[170] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *CoRR abs/1810.00826* (2018).

[171] Xuan, C. C., Gad-Elrab, M. H., Tran, T.-K., Schiller, M., Kharlamov, E., and Stepanova, D. Supplier optimization at bosch with knowledge graphs and answer set programming. In *Extended Semantic Web Conference* (2023).

[172] Yan, X., and Han, J. gspan: graph-based substructure pattern mining. *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* (2002), 721–724.

[173] Yang, B., Yih, W., He, X., Gao, J., and Deng, L. Embedding entities and relations for learning and inference in knowledge bases. In *ICLR* (2015).

[174] Yang, X., Liu, Y., and Wang, X. Reformer: The relational transformer for image captioning. *Proceedings of the 30th ACM International Conference on Multimedia* (2021).

[175] Yao, Y., Zhang, A., Han, X., Li, M., Weber, C., Liu, Z., Wermter, S., and Sun, M. Visual distant supervision for scene graph generation. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), 15796–15806.

[176] Ying, Z., Bourgeois, D., You, J., Zitnik, M., and Leskovec, J. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems 32* (2019).

[177] Yogatama, D., Gillick, D., and Lazic, N. Embedding methods for fine grained entity type classification. In *ACL 2015* (2015), pp. 291–296.

[178] Yuan, H., Tang, J., Hu, X., and Ji, S. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM*

SIGKDD international conference on knowledge discovery & data mining (2020), pp. 430–438.

[179] Yuksekgonul, M., Bianchi, F., Kalluri, P., Jurafsky, D., and Zou, J. Y. When and why vision-language models behave like bags-of-words, and what to do about it? *ArXiv abs/2210.01936* (2022).

[180] Zeiler, M. D., and Fergus, R. Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014* (Cham, 2014), Springer International Publishing, pp. 818–833.

[181] Zhang, W., Paudel, B., Wang, L., Chen, J., Zhu, H., Zhang, W., Bernstein, A., and Chen, H. Iteratively learning embeddings and rules for knowledge graph reasoning. In *WWW 2019* (2019), pp. 2366–2377.

[182] Zhao, S., and Xu, H. Less is more: Toward zero-shot local scene graph generation via foundation models, 2023.

[183] Zhao, T., Zhang, T., Zhu, M., Shen, H., Lee, K., Lu, X., and Yin, J. Vl-checklist: Evaluating pre-trained vision-language models with objects, attributes and relations. *ArXiv abs/2207.00221* (2022).

[184] Zhao, Y., Zhang, A., Xie, R., Liu, K., and Wang, X. Connecting embeddings for knowledge graph entity typing. In *ACL 2020* (2020), pp. 6419–6428.

[185] Zhou, B., Khosla, A., Lapedriza, À., Oliva, A., and Torralba, A. Object detectors emerge in deep scene cnns. In *ICLR 2015* (2015).

[186] Zhu, G., Zhang, L., Jiang, Y., Dang, Y., Hou, H., Shen, P., Feng, M., Zhao, X., Miao, Q., Shah, S. A. A., and Bennamoun. Scene graph generation: A comprehensive survey. *Neurocomputing 566* (2022), 127052.

[187] Zhu, Z., Zhang, Z., Xhonneux, L. A. C., and Tang, J. Neural bellman-ford networks: A general graph neural network framework for link prediction. In *NeurIPS 2021* (2021), pp. 29476–29490.

[188] Zou, X. A survey on application of knowledge graph. *Journal of Physics: Conference Series 1487* (2020).

List of Publications

Youmna Ismaeil , Daria Stepanova, Trung-Kien Tran, Piyapat Saranrittichai, Csaba Domokos, and Hendrik Blockeel(2022). Towards Neural Network Interpretability Using Commonsense Knowledge Graphs. In: *The* 21^{st} *International Semantic Web Conference – ISWC 2022. Lecture Notes in Computer Science*, vol 13489, pp. 74-90. Springer, Cham.

Youmna Ismaeil , Daria Stepanova, Trung-Kien Tran, and Hendrik Blockeel (2023). FeaBI: A Feature Selection-Based Framework for Interpreting KG Embeddings. In: *The* 22^{nd} *International Semantic Web Conference – ISWC 2023. Lecture Notes in Computer Science*, vol 14265. pp. 599-617. Springer, Cham.

Huu Tan Mai, Youmna Ismaeil, Daria Stepanova, Trung-Kien Tran, and Hendrik Blockeel. Look beyond the Surface: A Demo for Explaining Knowledge Graph Embeddings and Entity Similarity. In *The* 22^{nd} *International Semantic Web Conference - ISWC 2023 (Posters/Demos/Industry).*

Youmna Ismaeil, Jan Hednrik Metzen, Trung-Kien Tran, Daria Stepanova and Hendrik Blockeel. Breaking the Mold: Dynamic Pattern Discovery in Image Classifiers with Foundation Models and Knowledge Graphs. Under review *The* 24^{th} *International Semantic Web Conference. 2025.*

Declaration

I, Youmna Ismaeil, hereby declare that no generative artificial intelligence (AI) tools (such as ChatGPT or comparable language models) were used for writing, generating, developing, or formulating any part of the content in this doctoral thesis. This includes the creation of scientific text, development of research questions or hypotheses, construction of arguments, literature reviews, analyses, and conclusions.

All ideas, interpretations, and scholarly contributions presented in this thesis are the result of my independent academic work. I take full responsibility for the accuracy, originality, and integrity of the entire content.

Youmna Ismaeil
July, 2025

FACULTY OF ENGINEERING SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
DECLARATIVE LANGUAGES AND ARTIFICIAL INTELLIGENCE
Celestijnenlaan 200A box 2402
B-3001 Leuven
youmna.salahmahmoudismaeil@kuleuven.be