DATA-CENTRIC PREDICTION EXPLANATION AND MODEL EDITING FOR DEEP NEURAL NETWORKS

by

Mahtab Sarvmaili

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

at

Dalhousie University
Halifax, Nova Scotia
August 2024

Table of Contents

List of Tables

List of Figures

Abstract

Over the past decade, complex black-box models have excelled in various tasks, but their lack of transparency undermines trust in their predictions. This study contributes to Explainable AI (XAI) by introducing data-centric post-hoc explainers. We present two frameworks, FEHAN and DICTA, for locally explaining text classifiers through interpretable surrogate models. Experimental evaluations on four datasets demonstrate their effectiveness, with a focus on simplifying the explanation process. Additionally, we explore the explainability of Graph Convolutional Networks (GCNs) applied to molecular structures, offering multiple perspectives on their predictions. We also introduce HD-Explain, a post-hoc, model-aware, example-based explanation method for neural classifiers. HD-Explain uses Kernelized Stein Discrepancy (KSD) to identify influential training data points and potential distribution mismatches. This research advances the understanding of data contributions to machine learning models and addresses the emerging challenge of Machine Unlearning (MU) by leveraging insights into data-model interactions.

Chapter 1

Introduction

Automated decision-making systems frequently employ sophisticated with enhanced predictive performance. These models are characterized by their opaque, intricate internal mechanisms and substantial size, leading to their classification as "black box" models [34]. Post training, it is common to prioritize models that exhibit superior predictive performance. Nevertheless, this preference for performance necessitates addressing the "complex internal processing" and "lack of transparency" in these systems [116]. The complexity of data often mandates the use of advanced machine learning models, which are capable of discerning non-linear relationships and are effective in predictive tasks. The obscurity inherent in black box models poses significant concerns, particularly when these models are tasked with making critical decisions [5, 72, 145].

From a legislative standpoint, the European Union has implemented new regulations to ensure the right to comprehend the reasoning behind all software outputs including black-box models [36]. Consequently, the widespread utilization of machine learning algorithms has legally and ethically heightened the imperative to demystify and understand the operations of these enigmatic models. This regulatory framework underscores the necessity of transparency in automated decision-making, aligning legal requirements with ethical considerations for technology deployment. Understanding the processing of models not only facilitates compliance with regulatory standards but also aids in the development of new models, thereby reducing bias and enhancing overall model robustness.

To elucidate the inner workings of black-box models, various algorithms have been developed to explore the underlying relationships that govern the predictions made by these complex models. Some approaches specifically target explaining the behaviours of certain families of machine learning models.

One common starting point for developing interpretability techniques is the usage of white-box or transparent models. These models, such as decision trees [66], linear and logistic regression models [117], and Bayesian networks and naive Bayes models [18], are characterized by their simple structures and limited complexity. Due to their straightforward nature, they are productive for simple tasks and are considered transparent and understandable to users as a whole.

These white-box models are not only useful in their own right but also serve as an interpreter for developing methods to explain more complex ML models. For instance, the clear decision paths in decision trees, the explicit relationships in linear and logistic regression models, and the probabilistic reasoning in Bayesian networks all provide a foundation for understanding how inputs relate to outputs.

Other approaches specifically target explaining the behaviours of certain families of machine learning models. These methods, known as Model-aware explainers, scrutinize the internal structure of machine learning models to understand their predictions. They are primarily applied to parameterized models, especially deep neural networks, and rely on backpropagated gradients or the internal architecture of the model.

Model-aware explainers often aim to elucidate the inner representations of visual models by leveraging the internal structure of Convolutional Neural Networks (CNNs). For example, NetDissect [13] assigns convolutional filters to predefined semantic concepts, while IIN [39] translates hidden representations onto semantic concepts by analyzing the complete layer output space. These methods provide insights into how CNNs process and represent visual information.

Other Model-aware techniques focus on the backpropagated gradient to explain input features. Sensitivity analysis is a gradient-based method that identifies the vector representing the steepest ascent, highlighting the most influential features. Layer-wise Relevance Propagation (LRP) [9] iteratively redistributes relevance scores from the output layer back to the input layer, making it possible to trace the contribution of each input feature. Grad-CAM [144] identifies important regions of the input by averaging gradients to feature maps in CNN layers, providing a visual explanation of which parts of an image are most relevant to the model's decision.

Additionally, methods like Integrated Gradients [156] and SmoothGrad [151] measure feature contributions by averaging gradients with the help of a baseline sample. Integrated Gradients compute the average gradient along the path from a baseline to the input, providing a comprehensive attribution of feature importance. SmoothGrad improves gradient-based explanations by addition of noise to the input and averaging the resulting gradients, to highlight the most salient features.

In addition to model-specific methods, some researchers have explored using surrogate models that approximate a model's predictions at individual data points [89]. These surrogate models serve as interpretable proxies that mimic the behaviour of complex black-box models, providing insights into their decision-making processes.

RISE [127] and D-RISE [128] are specialized model-agnostic local explainers designed for image classification and object detection, respectively. These methods identify influential superpixels, offering a clear visualization of which parts of an image are most critical for the model's predictions.

Other model-agnostic local explainers utilize interpretable surrogate models to explain predictions. SHAP [104] (SHapley Additive exPlanations) measures feature importance using the Shapley value concept from game theory. SHAP creates a dataset where features are perturbed to form different coalitions and then trains a linear or weighted linear surrogate model on this perturbed dataset to determine the contribution of each feature. LIME [136] (Local Interpretable Model-agnostic Explanations) is a simple, local, model-agnostic explainer that can be efficiently applied to various classifiers. It uses a linear surrogate model to explain the predictions of the black-box model locally. By fitting the surrogate model to a locally generated neighbourhood, it approximates the model's behaviour in the vicinity of the data point.

These techniques often employ data augmentation strategies, such as random perturbation [136] or adversarial autoencoders [109], to generate a local neighbourhood around the data point. This local data is then used to train the surrogate model, enabling it to mimic the behaviour of the original black-box model. By doing so, these methods provide interpretable explanations that help users understand and trust complex machine-learning models.

The aforementioned methods are primarily applicable to parameterized models, specifically Convolutional Neural Networks (CNNs) and image datasets. However, generalizing these methods for text data presents several limitations.

Local gradient-based explainers face significant challenges when applied to text classifiers. As highlighted by DeepLift [148], selecting an appropriate reference example requires domain-specific knowledge. This necessity poses a significant challenge for text classifiers, where identifying a suitable reference example that is devoid of informative content proves particularly difficult. Beyond the correct selection of a baseline example, the choice of data interpolation technique is crucial when applying gradient methods to text data. For instance, InteriorGrads [155] employs a data interpolation technique to generate a series of samples for approximating integrated gradients. Even with a well-chosen, unbiased baseline, applying interpolation techniques requires meticulous consideration to ensure accuracy and relevance. Both LIME and SHAP approach the audit of the black-box model using potentially implausible, meaningless, or even adversarial texts. These synthetic texts might represent outliers relative to the original training dataset of the machine learning model [51]. This factor necessitates careful consideration, as it can influence the reliability and validity of the explanations generated by these methods.

Considering the advantages of local model-agnostic explainers, we intend to design a data-centric text explainer. This algorithm will be capable of producing local explanations for various text classifiers. It will consider the data's structure in the explainer's design and utilize white-box methods or lexical resources to generate a local neighbourhood. This approach aims to ensure that the explanations are both relevant and understandable, reflecting the unique attributes of textual data. By integrating domain-specific knowledge and leveraging appropriate interpolation techniques, the proposed data-centric text explainer will provide more reliable and valid explanations for text classifiers, thereby enhancing the transparency and trustworthiness of these models.

We initially introduced FEHAN (Framework for Explaining Hierarchical Attention Network). FEHAN is designed to locally explain the behaviours of the Hierarchical Attention Network (HAN) [173] by generating a collection of semantically similar documents. These documents serve as the training set for an interpretable model that locally mimics the behaviour of the black-box HAN model. This approach aims to provide insights into HAN's decision-making processes by examining how input document changes affect the model's outputs.

Expanding on this foundation, we propose DICTA (modularizeD model-agnostic framework for the explanatIon of black box Classifiers for Text dAta), which broadens the scope to accommodate a wider range of text classifiers. DICTA represents a novel approach in the interpretability literature by integrating a widely recognized lexical database to facilitate explanations of text classifiers. When provided with a black box classifier, a text document, and the classifier's decision on that document, DICTA generates an explanation by highlighting words in the document based on their importance. This importance is quantified by their contribution to the classification outcome.

DICTA leverages the concept of "influential sentences" to produce a corpus of similar documents. This is achieved through the use of WordNet [115], which helps identify semantically related words and phrases to construct new, yet contextually similar, document variations. This method allows DICTA to provide a detailed and context-sensitive explanation of the classifier's decision-making process, enhancing the transparency and accessibility of text classifiers.

The DICTA framework offers several significant advantages that enhance its utility in explaining black-box text classifiers. Firstly, it facilitates fast, straightforward, and highly customizable neighbourhood generation, allowing for tailored analysis specific to varying data sets and objectives. Secondly, DICTA provides clear and comprehensible explanations, both factual and counterfactual (CF), that users can easily understand [54, 55]. This dual explanatory approach offers insights into both the actual processing of data and hypothetical scenarios that could alter model outcomes. Lastly, the modular design of DICTA ensures flexibility and adaptability. Each component of this framework can be modified or updated independently, allowing for continuous improvement and customization without compromising overall system integrity. These features collectively make DICTA a robust and versatile tool for enhancing transparency in AI-driven text classification. The main advantages of DICTA are (i) fast, simple and easily customizable neighbourhood generation, (ii) simple and understandable factual and CF explanations [54, 55], (iii) and modular design.

Having discussed explainers for text classifiers, it is also important to consider the unique challenges and methods associated with graph-structured models. In recent years, the booming prevalence of social networks and practical applications of graph-structured data [69, 124] highlights the effectiveness of Graph Convolutional Networks (GCNs). These networks have demonstrated impressive performance in complex predictive tasks such as predicting molecular properties [40, 135, 166], toxicity [130], and protein interfaces [42]. However, the attainment of such high performance with GCNs often comes at the cost of reduced interpretability. GCN models inherently possess the opaque qualities of neural networks, and this lack of transparency is compounded by the intricate nature of molecular property modalities.

The critical need for trust in these applications [169] has spurred a wealth of research into Explainable Artificial Intelligence (XAI) methods [71, 93, 178]. These efforts aim to demystify the decision-making processes of GCNs, ensuring that these advanced models can be trusted and their predictions understood, particularly in high-stakes domains such as healthcare and environmental science.

Recent research frequently centers on post-hoc explanation methods that analyze the prediction patterns of a trained Graph Convolutional Network (GCN) by attributing the model's prediction outcomes to specific elements of its training data, such as known nodes or subgraphs within the graph structure. A notable approach within this realm is the prototype explanation (PT) method, which identifies a highly relevant subgraph associated with a node of interest. This subgraph, which closely aligns with the model's prediction, is then presented as the explanation for that prediction [52]. This method facilitates a deeper understanding of the decision-making process by highlighting the structural features within the graph that are most influential in the model's computations.

GNNExplainer [175], PGExplainer [105], and GraphMask [142] are representative examples of methods that provide local explanations for model predictions. These approaches attempt to clarify the reasoning behind specific predictions by isolating and interpreting the relevant portions of the graph. In this domain, XGNN [176] and GLGExplainer [7] aim to offer global explanations through the generation of a subgraph that represents the overall behaviour of the model across various predictions.

The algorithms discussed thus far explain the behavior of a model and why it makes certain predictions. However, to understand how a model's output changes when specific alterations are made to the input, we need counterfactual explanations. A counterfactual explanation identifies the minimal changes needed in the input features of a model to alter its prediction to a desired outcome.

This concept is particularly intriguing when applied to graph data. For instance, in molecular structures, counterfactual explanations can help identify modifications that turn a poisonous compound into a non-poisonous one, or adjustments that optimize a molecule for drug discovery. In these cases, a counterfactual explanation for a graph is obtained by perturbing the graph's structure and observing changes in the model's predictions.

Counterfactual explanations (CF) aim to determine the minimal perturbations required on a graph to change the model's prediction for specific nodes of interest. Prominent methods in this field include CF-GNNExplainer [103], MEG [121], and CLEAR [107]. These approaches enhance our understanding by identifying and modifying critical components of the graph that, when altered, significantly impact the model's output. This helps illustrate potential weaknesses or areas of sensitivity within the model's decision-making process and provides valuable insights for practical applications, such as improving molecular structures in drug discovery.

Despite the significant contributions of earlier works, several challenges remain that limit their practical application and necessitate further research. As highlighted in [74], many studies have focused predominantly on one type of explanation—either prototype (PT) or counterfactual (CF). This narrow focus can lead to misunderstandings about the model's reliability, potentially resulting in unexpected behaviours that compromise the trustworthiness of the system, particularly in critical research areas like molecular property prediction.

Moreover, in both existing types of explanations—PT and CF—it is crucial to manage the extent of edge perturbation to prevent excessive pruning of connections within the original input graph. However, previous studies have not adequately explored the guidelines for controlling this degree of perturbation. This oversight can significantly affect the fidelity and usefulness of the explanations generated, as excessive or insufficient modifications can distort the insights provided by the explanation methods, leading to less effective or misleading interpretations.

Considering the mentioned points, we intend to design a local graph explainer to offer multiple perspectives of explanations, allowing for a more comprehensive understanding of the model's decision-making process. It will control the magnitude of perturbations to ensure that the explanations remain meaningful, and it will strive to keep the generated explanations as close as possible to the original data manifold. This method seeks to balance the alteration of the graph structure with the preservation of its fundamental properties, thereby providing reliable and insightful explanations.

We introduce the Multi-Perspective GCN Explainer (MPGE), a framework designed to bridge research gaps by integrating and enhancing existing explanation strategies. MPGE produces prototype (PT) and counterfactual (CF) explanations and introduces a novel approach termed EXemplar Explanation (EXE). EXE is particularly valuable in scenarios where the input graph is sparse or when a comprehensive view of all influential connections affecting the prediction is necessary [44]. EXE functions by presenting a subgraph of the original GCN model, eliminates those connections whose removal doesn't change the prediction of model. [57]. This approach enables a detailed and intuitive understanding of the model's reasoning process by visually distinguishing influential links within the graph structure.

The MPGE framework encompasses three types of explanations:

1. CF Explanation: This involves a subgraph where the minimum set of edges, whose removal changes the model's prediction, are identified and eliminated. This "minimal change" approach helps to pinpoint the edges that are crucial for the specific outcome.

2. PT Explanation: This is presented as a subgraph of the input, maintaining the same label as the graph being explained but including only the most significant connections that are responsible for the prediction. This method focuses on the core elements that drive the model's decision-making process.

3. EXE Subgraph: This mirrors the original input graph in terms of label and eliminates those connections that can be removed without damaging models prediction. The EXE subgraph serves as a near-replica of the input, providing a clear and focused illustration of the influential factors in the model's classification.

Together, these strategies enable a comprehensive, multi-faceted exploration of how GCN models derive their predictions, thereby increasing the transparency and interpretability of graph-based machine learning models. Following our exploration of machine learning model explainability at the dataset level, we delved deeper into a statistical analysis of model behaviour and data dependencies.

The primary challenges associated with example-based prediction explanations stem from the need to retrieve relevant data points from a large pool of training samples and to substantiate the rationale behind these explanations [94, 183]. This phase of our research focuses on understanding the complexities of how models interact with specific data instances and the justifications for the explanatory outcomes derived from them.

The example-based prediction explanation methods address challenges by establishing an influence chain between training and test data points. Notable methods include the Influence Function, which measures a sample's influence by the shift in model parameters due to up-weighting the sample, and Representer Point Selection (RPS). While RPS, in contrast to IF, is computationally efficient, it often produces coarse-grained, class-level explanations rather than detailed instance-level explanations. Despite several later variants attempting to overcome these limitations, their improvements are constrained by inherent theoretical scalability bounds. We present Highly-precise and Data-centric Explanation (HD-Explain), a post-hoc, model-aware, example-based explanation solution for neural classifiers. Instead of relying on data co-influence on model parameters or feature representation similarity, HD-Explain retains the influence chain between training and test data points by exploiting the underrated properties of Kernelized Stein Discrepancy (KSD) [100] between the trained predictive model and its training dataset. Specifically, we note that the Stein operator augmented kernel uniquely defines a pairwise data correlation (in the context of a trained model) whose expectation on the training dataset results in the minimum KSD (as a discrete approximation) compared to that of the dataset sampled from different distributions. By exploiting this property, we can 1) reveal a subset of training data points that provides the best predictive support to the test point and 2) identify the potential distribution mismatch among training data points. Jointly leveraging these advantages, HD-Explain can produce explanations that are faithful to the original trained model.

Our design and approach for the HD-Explain model advance our understanding of data contributions to machine learning (ML) models. This understanding is crucial for addressing one of the most pressing challenges in the ML community: Machine Unlearning (MU). MU involves a process whereby ML models can selectively eliminate specific training samples, effectively reversing their influence on the model while maintaining performance on the remaining data. This concept has gained considerable interest due to the increasing global emphasis on data privacy rights. Numerous countries and territories have enacted "Right to be Forgotten" regulations—such as the CCPA in California, GDPR in Europe, PIPEDA in Canada, LGPD in Brazil, and NDBS in Australia—which grant individuals the right to withdraw their consent for the use of their data in ML training. These developments have spurred significant research into various machine-learning approaches, reflecting the growing need to align ML practices with legal and ethical standards.

Data contribution models, can play a significant role in the field of MU by identifying which data points most influence the model's predictions. Despite the development of numerous new MU techniques, there remains a gap in understanding the fundamental feasibility of the unlearning process itself. While researchers have been proactive in devising methods to eliminate data from models, there is insufficient consideration of whether it is genuinely possible to selectively remove specific data points from a model without detrimentally affecting its overall performance. This oversight suggests a need for more rigorous investigation into the practical limits and capabilities of MU techniques, ensuring that they not only adhere to theoretical expectations but are also viable in real-world applications.

Our research seeks to address these gaps by examining the factors that impact the feasibility of MU. We hypothesize that the distribution of the data used in training the model, the method by which the model encodes information, and the model's confidence level in its predictions are critical determinants of the ease with which information can be unlearned. To investigate this, we have integrated our HD-Explain tool with unlearning feasibility analysis, employing the Kernel Stein Discrepancy (KSD) as the base to find feasible data points.

We conducted a thorough experimental analysis using various unlearning approximation algorithms. By evaluating their performance across different scenarios, defined by the aforementioned factors, we were able to assess the real-world efficacy of these algorithms. This comprehensive approach allows us to not only understand the limitations and capabilities of current unlearning techniques but also to contribute valuable insights into the optimization of MU processes under practical conditions.

1.1 Summary of Contribution

Data Centric Text Explainer: We present two local, modular, data centric text classifier explainer: FEHAN and DICTA. FEHAN works by replacing informative sentences in a document with artificial ones to create similar documents, which are then used to train a decision tree. This interpretable model helps identify important words, creating a saliency map that explains the class label assigned by the original HAN, while preserving the document's original meaning and enriching it with semantically similar examples. DICTA is a model-agnostic explainer that locally interprets the predictions of black-box models by generating a set of similar documents around a given input. It uses backward elimination to identify the most influential sentences in a document and then replaces words within these sentences using semantic replacements obtained from WordNet. The generated documents are then used to train a surrogate model that locally mimics the behavior of the black-box model, providing insights into its decision-making process.

Data Centric Graph Explanation: Later, we present a local, data-centric, perturbation-based explainer for GCNs. Our Multi-Perspective GCN Explainer (MPGE) is a framework designed to enhance existing explanation strategies by producing three types of explanations: Prototype (PT), Counterfactual (CF), and a novel Exemplar Explanation (EXE). MPGE provides detailed insights into the reasoning process of Graph Convolutional Networks by perturbing the input graph and tracing how these changes affect the model's predictions. By adjusting the perturbation matrix, the explainer produces the desired explanations, offering insights into the different factors influencing the GCN's decision-making process.

Data Centric Prediction Explanation: We introduce HD-Explain, a Kernel Stein Discrepancy-driven example-based prediction explanation method. We performed comprehensive qualitative and quantitative evaluation comparing three baseline explanation methods using three datasets. The results demonstrated the efficacy of HD-Explain in generating explanations that are accurate and effective in terms of their granularity level. In addition, compared to other methods, HD-Explain is flexible to apply on any layer of interest and can be used to analyze the evolution of a prediction across layers. HD-Explain serves as an important contribution towards improving the transparency of machine learning models.

Data-centric Assessment of Machine Unlearning Feasibility: We explore the overall feasibility and challenges of machine unlearning for individual samples in the training set, regardless of the specific unlearning methods used, and identify the factors that influence this process. Rather than relying on predictive confidence-based methods, we propose that the difficulty of unlearning is related to the data distribution augmented by the model. We demonstrate that trained machine learning models can define parameterized kernel functions over training data points, which represent the distribution of training samples conditioned on the model. Analyzing this conditional data distribution provides a more precise estimation of the unlearning difficulty.

1.2 Conference Publication

This research has contributed to a number of conference and journal publications.

Conference Publications directly contributes to PhD thesis:

- Sarvmaili, Mahtab, Amilcar Soares, Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, and Stan Matwin. "A modularized framework for explaining hierarchical attention networks on text classifiers." In Canadian AI. 2021.
- Sarvmaili, Mahtab, Riccardo Guidotti, Anna Monreale, Amilcar Soares, Zahra Sadeghi, Fosca Giannotti, Dino Pedreschi, and Stan Matwin. "A Modularized Framework for Explaining Black Box Classifiers for Text Data." In Canadian AI. 2022.

Conference Publications as side research projects:

- Etemad, Mohammad, Nader Zare, Mahtab Sarvmaili, Amílcar Soares, Bruno Brandoli Machado, and Stan Matwin. "Using deep reinforcement learning methods for autonomous vessels in 2d environments." In Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, May 13–15, 2020, Proceedings 33, pp. 220-231. Springer International Publishing, 2020.
- Zare, Nader, Bruno Brandoli, Mahtab Sarvmaili, Amilcar Soares, and Stan Matwin. "Continuous control with deep reinforcement learning for autonomous vessels." arXiv preprint arXiv:2106.14130 (2021).

Submissions (Under-review) directly contributes to PhD thesis:

- Sarvmaili, Mahtab, Hassan Sajjad, and Ga Wu. "Data-centric Prediction Explanation via Kernelized Stein Discrepancy." arXiv preprint arXiv:2403.15576 (2024).
- Sarvmaili, Mahtab, Ga Wu, Anna Monreale, Riccardo Guidotti, and Stan Matwin "Contrastive Explanations for Graph Convolutional Networks in Molecular Compound Analysis."
- Sarvmaili, Mahtab, Hassan Sajjad, and Ga Wu. "Investigating the Feasibility of Machine Unlearning."

1.3 Outline

The remainder of this proposal is organized as follows:

Chapter 2 reviews existing works on explainable artificial intelligence (XAI), focusing on XAI for text models and graph model explainers.

Chapter 3 is dedicated to XAI for text data. We introduce our proposed frameworks: FEHAN (Framework for Explaining Hierarchical Attention Networks) and DICTA (a model-agnostic modular framework for explaining text classifiers). We also explore techniques and criteria for evaluating these frameworks.

Chapter 4 covers XAI for graph data. We start with the problem formulation and background on the explainability of Graph Neural Network (GNN) models, present our algorithm for obtaining contrastive explanations, and then introduce the experimental setting and discuss the empirical results.

Chapter 5 delves into highly precise explainers by formulating the problem and introducing the KSD algorithm. We evaluate the effectiveness of HD-Explain through comprehensive experiments, analyzing its ability to provide faithful and insightful explanations for various deep learning models.

Chapter 6 examines the feasibility of Model Unlearning (MU). This chapter provides a comprehensive background on current MU approaches and explores techniques for removing information from trained models. We outline their strengths and limitations, offering context for the challenges associated with unlearning and paving the way for our novel approach to assessing feasibility.

Finally, **Chapter 7** summarizes our findings, discusses possible limitations of this research, and outlines the future plan for this research.

Chapter 2

Literature review

The widespread deployment of intricate and sophisticated ML models across various domains [2] has heightened the necessity for comprehending the rationale behind the decisions made by systems that utilize these models, particularly when employed in critical decision-making processes [116]. Consequently, there has been a surge in research focused on the explainability of these complex and opaque models [33, 34, 38, 180]. This growing interest aims to enhance transparency and reliability in automated systems by elucidating how these models arrive at their conclusions.

Various types of black box explanation algorithms have been developed to enhance the interpretability of machine learning models. These algorithms can be broadly categorized from two perspectives: *model-specific* versus *model-agnostic*, and *local* versus *global*, as outlined in [56]. Global interpretation algorithms seek to analyze the entire training process of deep neural network (DNN) models, focusing on elements like weights and overall model structure [3]. However, the complexity of these models often makes understanding their internal processes challenging. Consequently, local explanation models, which provide insights into individual predictions, are gaining popularity due to their lower computational and implementation complexity.

Local explanation methods can further be subdivided based on the type of data or the specific black box model they are designed to elucidate. The taxonomy proposed in [92] differentiates these methods into data-centric versus model-aware approaches. In the subsequent sections, we will delve into these techniques, exploring their respective advantages and disadvantages to provide a comprehensive understanding of their applications and limitations.

Although the major contribution of XAI is intepreting and explaining complex black-box models, yet the white-box family of models exist which the explanation process can be carried out with simple procedures. Understanding and explaining them requires minimum effort such as Linear models, Decision Tree, and Generalized Additive Models (GAMs).

2.1 Model-aware local explainer

Model-aware local explainer methods scrutinize the sensitivity of the internal components (weights and neurons) of deep neural networks (DNNs) to discern the influence and importance of the model's inputs. These methods are typically classified into gradient-based, correlation-score interpretation, and class activation mapping categories. For instance, activation mapping methods like GradCAM [144] and earlier studies [129, 182] employ heatmaps to visualize the activity of neurons and weights, elucidating the information learned by the DNN model during training.

Gradient-based methods, such as INTGRAD, ascertain feature importance by analyzing the gradients of an example as the input scales uniformly to zero. Another prominent technique, DeepLift [155], evaluates the importance of input features by comparing the difference in neuron outputs between a given example and a reference example. Meanwhile, correlation-based algorithms [149] compute the correlation scores for each input feature to assess their significance, avoiding the direct calculation of importance signals.

Despite their robust mathematical foundation, these methods are constrained to parameterized functions and often require a meticulously chosen reference example that aligns with specific domain knowledge to produce meaningful results. This dependency on the problem context and the necessity for domain expertise renders these techniques as Model-aware and Data-Specific explainers.

2.2 Data-centric Local Explainers

Data-centric local explainer methods provide insights into various black-box models by focusing exclusively on the input data, disregarding the internal workings of the black-box model. These methods are often regarded as model-agnostic post-hoc explainers.

In practice, data-centric methods assess how modifications to input data affect the predictions of a black box. The concept of "data-centric" is described as "transitioning from model focus to the underlying data for evaluation of the AI models". This transition aligns with the principles of abductive reasoning, which centers on identifying the most plausible explanations based on observable data [62, 132]. In data-centric Explainable AI, abductive reasoning guides the process of hypothesizing which aspects of the data are most influential in driving a model's predictions [112]. By focusing on the data rather than the model's internal mechanisms, these approaches use abductive reasoning to create explanations that are intuitive and grounded in the actual inputs, ensuring that the resulting insights are both relevant and understandable.

This connection between data-centric explanation and abductive reasoning is vital for creating robust, generalizable AI models [73]. As data-centric methods use iteratively refine explanations and identify key data influences, they not only clarify individual predictions but also enhance the model's overall performance by revealing potential biases or data quality issues. Thus, the synergy between data-centric methods and abductive reasoning forms a comprehensive framework that ensures AI systems are explainable, reliable, and aligned with real-world data, meeting the demands of transparency and trustworthiness in practical applications.As the strong connection between data-centric Explainable AI and abductive reasoning validates the importance of this topic, it becomes evident that we should prioritize the study of data-centric explanation approaches.

To begin with, perturbation methods are prime examples of data-centric Explainable AI approaches. Perturbation-based methods are quintessential examples, where perturbations are applied to the given input to observe changes in predictions, thereby identifying critical input features. Initially conceptualized by approaches like LIME [136], these methods examine the impact of various input alterations on model predictions. LIME, for instance, addresses the challenge of approximating the decision boundary of complex models like DNNs by learning about the model's local behaviour. It generates sample instances around a specific input, creating a dataset that serves as a training set for an interpretable surrogate model designed to emulate the complex model's predictions.

Similarly, SHAP [104] employs a distinct approach by using Shapley value estimation [95] to calculate the contribution of each feature to the model's prediction, offering a different perspective on feature importance.

While these methods are effective in highlighting features that support the prediction of a black-box model, providing logical reasons for decision-making, they do not necessarily reveal the underlying factors that might lead to different outcomes [102]. The concept of counterfactual (CF) explanations addresses this gap by proposing a way to understand the causal relationships within the model's decision-making process [52]. Exploring CF explanations could thus represent a significant advancement in elucidating the behaviours of black-box models, offering a deeper dive into the rationale behind different possible outcomes.

2.3 Text Explainers

LIME and SHAP are two of the most prominent local, model-agnostic, and data-agnostic explanation methods applicable to natural language processing classifiers.

LIME is particularly suited for text classifiers, where it creates synthetic neighbourhood documents by randomly omitting tokens from the existing data. It then trains a linear model on these modified instances to serve as an interpretable local surrogate. The weights assigned by this linear model indicate the importance of each word in the text for explaining the classifier's decision. Guidotti and et. al. [51] investigated, a notable limitation of using LIME for text data is that the neighbourhood texts are generated through random word removal, potentially resulting in nonsensical sentences.

Conversely, SHAP utilizes Shapley value estimation [95] to assess the contribution of each word by testing various combinations of words in the text. Like LIME, SHAP also involves randomly removing words to analyze their significance in decision-making. Consequently, both methods may audit the black box model with texts that are implausible, meaningless, or could be perceived as adversarial, as they may represent outliers relative to the original training set of the machine learning model. This raises concerns about the plausibility and relevance of the explanations generated by these approaches.

A descendant of LIME is X-SPELLS [86] a model-agnostic explainer specifically designed for text classifiers that provides explanations in the form of exemplar and counter-exemplar texts. X-SPELLS addresses the limitations observed in LIME and SHAP by employing a Variational Auto-Encoder (VAE) to generate neighbourhood texts. This method moves the generation process into a latent space dimension, in alignment with approaches recommended by ABELE [55]. However, this technique introduces its own challenges, notably the requirement for specific training and fine-tuning of the VAE. This process can be labour-intensive and may introduce additional complexity and ambiguity, complicating the overall interpretability alongside the original black-box model.

The strategy of generating data in latent space, while effective for continuous domains such as images or time series, poses challenges for text data. Even minor perturbations in text feature space can significantly alter the meaning of the input, leading to the generation of inaccurate examples [88]. Furthermore, the use of attention and attribution mechanisms to explain classifier decisions is often insufficient for two reasons, as discussed by Grimsley and et al. [50]: (i) The attention mechanism assigns a real value from the interval [0,1] to each sentence in a document, where a higher value indicates greater importance. However, even less significant sentences receive values close to, but not exactly, zero, which can obscure their true irrelevance. (ii) The mechanism does not adequately distinguish the importance of each sentence relative to different class labels. Since all sentences are assigned a real value, it is unclear how each sentence specifically influences the distribution across class labels, leading to ambiguous interpretations of the model's reasoning based on these values.

In response to these challenges, this work introduces two local, model-agnostic methods for explaining decisions made by text classifiers. Our approaches aim to clarify the reasoning behind a classifier's decision on a specific document by generating semantically similar samples in its vicinity. By focusing on informative sentences and strategically modifying them, we create neighbourhood data that is realistically and semantically aligned with the original text, enhancing the clarity and relevance of our explanations.

We begin by introducing FEHAN, a framework designed to locally explain the behavior of the Hierarchical Attention Network (HAN) [173]. This modular framework focuses on identifying Informative Sentences (IS) within a given document. FEHAN generates a set of semantically similar documents by replacing these informative sentences with artificially constructed sentences sampled from the original dataset. This approach not only preserves the structural integrity of the document in the neighbourhood data but also enriches the semantic context by integrating closely related sentences.

Subsequently, we present DICTA, a method that elucidates the decision-making process of a text classifier by generating similar samples in close proximity to the original document. DICTA's approach to neighbourhood generation leverages influential sentences and semantic replacements to ensure that the black box is probed with plausible text instances, enhancing the reliability and relevance of the explanations provided.

Both FEHAN and DICTA operate without the need for additional data or training of supplementary models. This is particularly advantageous given the complexity of modern language models, which entail managing billions of hyperparameters. Introducing another complex language model for explanation purposes [78, 168] could unnecessarily complicate the user's experience by adding another layer of complexity.

Moreover, a significant advantage of DICTA is its flexibility; it does not impose constraints on the length of documents or restrict its application to specific problem domains, unlike some existing methods [97, 122]. This broad applicability makes DICTA a versatile tool for explaining text classifiers across various contexts.

2.4 Graph Neural Network Explainer

Graph Neural Networks (GNNs) have recently emerged as a prominent technology in various graph mining applications, including node classification [63, 123, 138], graph classification [8, 87], and link prediction [70, 77]. Despite the exceptional performance of GNN models through information propagation and aggregation, the complex and non-linear nature makes them difficult to interpret. Understanding and explaining the outputs of GNN models is crucial, as it allows for the identification of potential flaws in their predictions, enhancing the trustworthiness and transparency of these systems.

Recent efforts to improve the interpretability of GNNs have adapted techniques originally developed for more traditional neural network architectures. For instance, GraphLIME [64], an adaptation of LIME, employs a nonlinear feature selection method to achieve local explainability of GNNs. This approach is particularly useful for understanding the specific contributions of subgraphs or nodes within the overall network structure. Similarly, GradCAM has been utilized within the context of Graph Convolutional Networks (GCNs) to produce heatmaps that highlight significant regions of the input graph. These heatmaps provide visual representations of the areas within a graph that most influence the model's predictions, offering intuitive insights into the decision-making processes of GNNs. Such techniques bridge the gap between complex GNN operations and actionable insights, making them indispensable tools for researchers and practitioners seeking to harness the full potential of graph-based learning.

From a graph-specific perspective, most contemporary GNN-XAI (Graph Neural Network - Explainable Artificial Intelligence) methods focus on elucidating the model's decision-making by highlighting a subgraph from the original input graph that significantly influences the GNN's prediction. GNNExplainer [175] and PGExplainer [105] both operate by perturbing the graph's structure to obscure unrelated nodes and edges, thereby isolating the most influential subgraph that exhibits the highest mutual information with the GNN's prediction. While GNNExplainer identifies influential subgraphs directly, PGExplainer employs a parameterized mask to systematically extract significant subgraphs, facilitating a deeper understanding of the black box model's predictive behaviour. Alternatively, XGNN introduces a different approach by using a graph generator to create graphs that are most likely to belong to a particular class. However, this method is predicated on the assumption that each class can be represented by a single, definitive graph—an assumption that may not hold in the face of complex phenomena, casting doubts on the realism and generalizability of this approach [162].

Highlighting the most influential subgraph within an input graph provides critical insights for predicting a given data point, yet the inclusion of counterfactual explanations (CF) offers a deeper understanding of the model's intrinsic biases. CF explanations not only illuminate the importance of specific features but also clarify how changes to parts of the input could alter the model's predictions.

CF-GNNExplainer [103] advances the use of CF explanations for Graph Neural Networks (GNNs). This approach employs a perturbation-based algorithm that seeks to derive CF explanations by modifying the graph structure. The perturbation matrix used in this method is trained based on the negative log-likelihood of the prediction loss, aiming to identify changes that would significantly impact the model's output. However, the effectiveness of this algorithm in providing CF explanations can diminish as the complexity of the graph increases, a limitation influenced by the chosen loss function. This constraint highlights the challenges in ensuring that CF explanations are feasible and reliable across all nodes within complex graph structures.

To effectively integrate the concepts of extracting both significant and influential subgraphs for the prediction of Graph Convolutional Networks (GCN), we have adopted the principles of Prototype (PT) and Counterfactual (CF) explanations as proposed in the study by Dhurandhar et al. [31]. This research initially focused on image data, identifying minimally sufficient and necessarily absent features crucial for the model's final classification. The perturbation matrix in this framework is crafted using a specialized loss function that is designed to elicit both PT and CF explanations.

Building on this foundational work, we have adapted the design of the loss function to develop our model for explaining predictions in GNNs. Additionally, we introduced a new dimension to our explanation approach by incorporating Exemplar (EXE) explanations. EXE explanations align with the input's label and encapsulate features instrumental to the model's prediction, providing a comprehensive view of the factors driving the decision-making process.

We have developed a multi-component GNN explainer that utilizes the loss function proposed by Dhurandhar and et al. [31]. This explainer is designed to delve into various facets of our model, which will be extensively discussed in Chapter 4. This chapter will explore the intricate details of our approach, highlighting how each component contributes to a robust understanding of GNN predictions.

2.5 Machine Unlearning

XAI focuses on making the decision-making processes of complex AI models understandable to humans, thereby enhancing trust, accountability, and compliance with regulatory standards. Previous studies have shown that an effective explanation algorithm should reveal the influence of training data points on the predictions for test data points, as well as capture the correlation between training data conditioned on the model. If this is accurate, then removing the training data points identified as influential explanations should result in a shift in the model's predictions. If the training data points highlighted as good explanations are highly correlated with a given example, their removal is likely to perturb the model's predictions. This raises the question of how trustworthy our explanations are. To explore this relationship, we delve into the topic of machine unlearning. We seek to understand whether there is a causal link between the removal of influential training data points and the performance of the model that can be reflected as the feasibility of unlearning.

Machine Unlearning (MU) is a process designed to forget specific subsets of training data [14,21]. The earliest method proposed for this involved retraining the machine learning model from scratch using the "remaining data" after removing the "forget set." However, even with the introduction of SISA [14], which aimed to reduce the computational burden, retraining remained resource-intensive. To address this challenge, "unlearning approximation" techniques have been developed. One such approach is "Fine Tuning" [47,164], where the model is fine-tuned on the remaining data to facilitate forgetting. Alternatively, Gradient Ascent methods [49, 159] adjust the model's weights in the direction of the gradient to increase the model's error on the data intended for forgetting.

Other techniques have frequently utilized the Newton update as a fundamental step for removing data influence [47,59,126,143]. These methods typically leverage the Fisher Information Matrix (FIM) to gauge the sensitivity of the model's output to perturbations in its parameters. For example, Fisher Forgetting [47] employs a scrubbing approach where noise is added to parameters based on their relative importance in distinguishing the forget set from the remaining data set. Mehta et al. [113] employs conditional independence coefficient to identify sufficient sets of parameters for targeted unlearning.

Several methods have incorporated the principles of differential privacy (DP) [1] to ensure the MU does not inadvertently reveal information about the data that has been removed. Izzo [67,179] adhere to the DP framework to ensure a high probabilistic similarity between models before and after unlearning. Guo et al. [59] introduce the concept of certified unlearning, grounded in information theory and specifically tailored to the Fisher Information Matrix. Certified Minimax Unlearning [98] has developed an algorithm specifically for minimax models. This method removes data influences through a total Hessian update and incorporates the Gaussian Mechanism to achieve (ϵ, δ)-minimax unlearning certification, ensuring a balance between data removal and model integrity. Chourasia et al. [27] propose a data deletion technique that ensures the privacy of deleted records. Ullah [161] investigated the MU in the context of SGD and streaming removal requests, but ensured that their method is differentially private. DP algorithms provide the upper bound for the unlearning scheme, but they don't guarantee the full unlearning of requested data [120].

Other research efforts extend into various domains of unlearning: knowledge distillation [28], selective forgetting for lifelong learning [147], federated unlearning [22], online MU [25, 91], and exploring adversarial attacks using MU methods [32, 165], reducing the vulnerability of models to privacy attacks by forgetting training examples [68]. These studies underscore the breadth of approaches being developed to manage data deletion, privacy, and the continuous adaptation of machine learning models in secure and efficient ways.

The majority of the literature on MU primarily concentrates on the development of unlearning algorithms or unlearning approximation techniques for selectively forgetting data pertaining to single classes, multiple classes, or random subsets of multiple classes. A common assumption underlying much of this research is that MU is universally feasible for all data points within a dataset and that MU techniques will behave consistently across different datasets. This assumption often overlooks the potential variability in MU efficacy due to differences in data characteristics or model dependencies, suggesting a need for more nuanced studies that evaluate the specific conditions under which MU can be effectively implemented.

2.6 Summary

This chapter focuses extensively on the evolution and methodologies of XAI, examining their application to text classifiers and graph neural networks. Additionally, this chapter introduces MU, and provides a comprehensive description of preliminary works.

XAI Techniques: The discussion begins with an overview of XAI techniques, which are essential for making the decision-making processes of AI models transparent and understandable. These techniques are crucial for enhancing trust, accountability, and compliance with regulatory standards. Two broad categories of XAI approaches are explored: model-aware and data-centric local explainers. Model-aware explainers are techniques designed to interpret and clarify the decision-making processes of specific machine learning models by leveraging knowledge of the model's internal structure and parameters. These explainers provide insights tailored to the particular architecture of the model, offering more accurate and relevant explanations compared to general-purpose methods. Data-centric explainers focus on interpreting machine learning models by analyzing the input data's influence on the model's predictions. They emphasize understanding how variations in data impact outcomes, offering explanations that highlight the relationship between specific data features and the model's decisions.

Graph Neural Network (GNN) Explainers: The chapter also delves into GNN explainers, which are increasingly used in various graph-related tasks such as node classification, graph classification, and link prediction. Techniques like GraphLIME and GradCAM adapt traditional neural network explainers to the specific needs of GNNs, focusing on local and global importance to enhance model interpretability.

Machine Unlearning: An emerging field that addresses the need for data deletion without compromising the integrity or performance of the model. This is particularly relevant in light of regulations such as the GDPR's Right to be Forgotten. This work explores the feasibility and difficulty of unlearning individual samples, proposing that these challenges are best understood through the model-augmented data distribution. By analyzing the distribution of training data points using parameterized kernel functions, particularly Kernelized Stein Discrepancy (KSD), the study introduces a more reliable metric for assessing unlearning difficulty. This approach aims to reduce unnecessary unlearning operations by identifying data points that are inherently challenging to unlearn.

Challenges and Assumptions: The chapter identifies key challenges in both XAI and MU, such as the assumption that unlearning is universally feasible across all data points and behaves similarly across different datasets. It also addresses the potential limitations of current explanations models, which may not adequately distinguish the importance of features or could generate misleading results due to random perturbations in data.

In conclusion, the related works chapter synthesizes a range of sophisticated methodologies and emerging challenges within the fields of XAI and MU, pointing towards the need for further research to address the nuances of these technologies in practical applications. The exploration of these topics highlights the current capabilities and limitations of AI systems and sets the stage for future advancements that could lead to more robust, transparent, and compliant AI technologies.

Chapter 3

Data-centric Text Explanation

In this work, we propose two modularized frameworks for explaining text classifiers: the Framework for Explaining Hierarchical Attention Networks (FEHAN) and the model-agnostic framework for the Explanation of Black Box Classifiers for Text Data (DICTA). We refer to them as a "modularized frameworks" as these methods are designed to allow different components and modules to works together for producing the final explanation. Later modules developed for different steps of explanation can be replaced with similar characteristics, making the framework adaptable and efficient for various applications.

This chapter is divided into two sections. First, we explore FEHAN, a modular framework that provides local explanations for an attention-based deep text classifier. Following this, we introduce DICTA, a model-agnostic approach aimed at understanding the predictions of various text classifiers. The content of this chapter is based on the research presented the two published papers [139, 140].

3.1 FEHAN

FEHAN is a modularized local explainer framework designed to elucidate the workings of the Hierarchical Attention Network (HAN) [173], an attention-based recurrent neural network for text documents.

FEHAN explains HAN's local behaviour for a specific data point by analyzing its surrounding neighbourhood. The underlying idea is based on the intuition that while the decision boundary for a black-box model may be highly complex across the entire data space, it is often possible to learn an interpretable model in the local neighbourhood of a data point [56, 136]. Thus, FEHAN generates a set of semantically similar instances near a given document to explore the predictions of text classifiers. The attention layer in HAN identifies Informative Sentences (IS) within a document, which significantly impacts class label assignment. FEHAN creates data for the document's vicinity by replacing these IS with artificial sentences.

FEHAN produces a set of semantically similar documents for a given instance classified by HAN. These synthetic documents are then used to train an interpretable model, specifically a decision tree, from which important words can be extracted to construct a saliency map that explains the class label for the document.

Training the interpretable classifier (i.e., decision tree) on the neighbourhood documents reveals the *important words* that locally explain the focus of the black-box classifier when classifying a document. This approach preserves the original essence of a given document while enriching it with semantically similar examples. The explanation produced by FEHAN is a saliency map highlighting the crucial words in the document that contribute to the black-box model's decision.

An overview of the general structure of FEHAN is provided in Figure 3.1. This process occurs during the inference phase. FEHAN[1] is presented as a modularized framework for understanding the behaviour of attention-based document classifiers. The modularity of FEHAN facilitates its adaptation to similar scenarios or the incorporation of other components with similar characteristics.

The main components of FEHAN are: *(i)* the HAN model, *(ii)* the informative sentences extraction, *(iii)* the neighbourhood generator module, and *(iv)* the interpretable model. The explanation procedure is detailed as follows:

3.1.1 Hierarchical Attention Network

The Hierarchical Attention Network (HAN) [173] is a document classification algorithm that leverages Recurrent Neural Networks (RNNs) along with an attention mechanism. HAN constructs the latent representation of documents by aggregating the latent representations of sentences within those documents. The model employs two levels of attention mechanisms [106] to adjust the significance of individual words and sentences during the document classification process.

HAN consists of four main components: a word sequence encoder, word-level attention, a sentence encoder, and sentence-level attention. This structure is depicted in Figure 3.2. A notable feature of HAN is its ability to extract importance coefficients at both the word and sentence levels, providing a clearer understanding of HAN's decision-making process. This feature is particularly valuable for the neighbourhood generator module in FEHAN.

3.1.2 Informative Sentences Extraction

The key element for the neighbourhood generation module is the availability of informative sentences provided by HAN. In the first step, HAN receives a selected document d, processed using the word2vec embedding model [114], and returns the predicted class label y along with the most

[1]https://github.com/MahtabSarvmaili/FEHAN

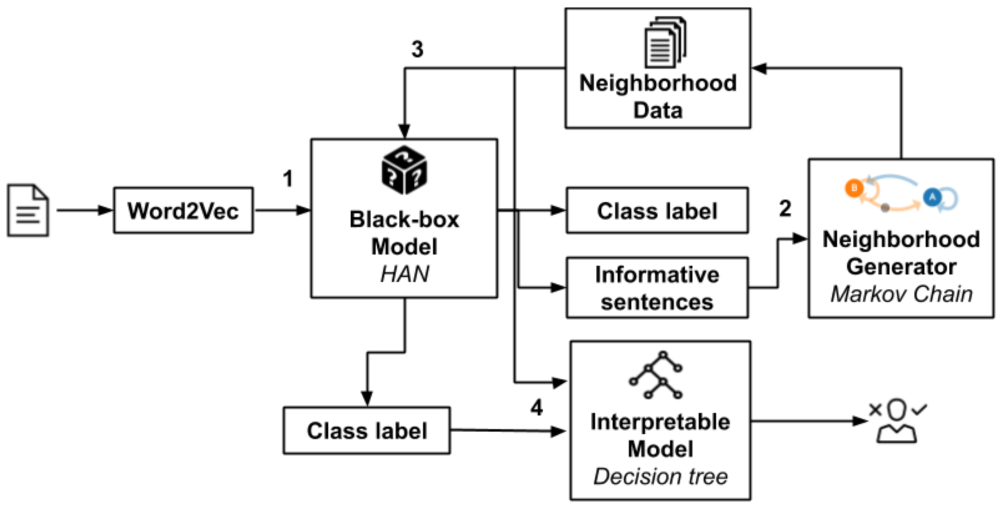


Figure 3.1: FEHAN: A modularized local explainer framework elucidating the workings of the Hierarchical Attention Network (HAN) by analyzing the neighbourhood of a document. It generates semantically similar instances near a given document and uses them to train an interpretable model, such as a decision tree, to produce a saliency map highlighting crucial words influencing HAN's predictions. Step 1) HAN process each document in two levels, word level and sentence level. It gives us the advantage to use the attention value to find the most influential sentences for neighborhood generation. Step 2) After obtaining the influential sentences from the attention value FEHAN generates a set of new sentences to replace the IF sentences and generates the neighborhood. Step 3) Then it passes the generated documents to the HAN for obtaining the classification labels. Step 4) The generated examples along with their class label will be employed to train a surrogate model for mimicking the behavior of black box model. At the end we features' importances from this surrogate model

Informative Sentence (IS) identified in d. The sentence attention layer in HAN assigns a score to each sentence, indicating its importance for the classification of the document.

After extracting the importance scores from the sentence attention layer, the IS and the original document are passed to the neighbourhood generator module. The number of IS is a data-dependent hyper-parameter that varies from one dataset to another. For instance, if the average number of sentences in each document is around ten, the top three sentences would be selected as IS. In cases where the document length is shorter than the predetermined length, the attention layer assigns high scores to empty sentences.

Our explanation method leverages the attention scores to identify the IS. However, if attention scores are unavailable, alternative methods such as backward elimination [83] can be used to extract the IS.

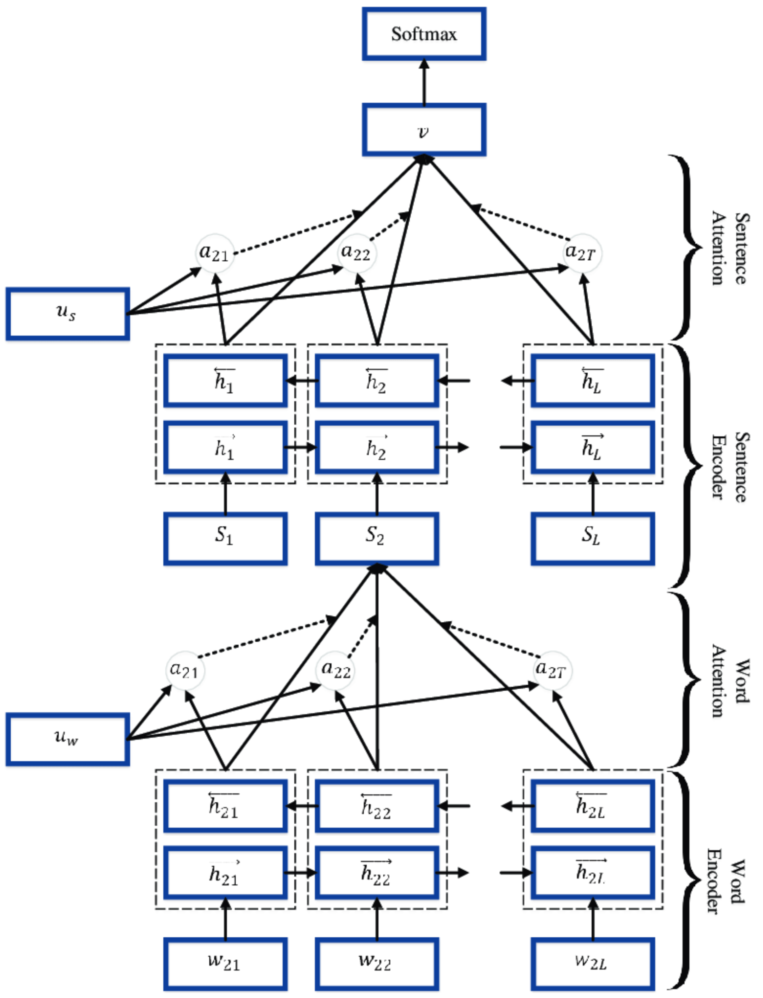


Figure 3.2: The Hierarchical Attention Network (HAN) for document classification. HAN leverages Recurrent Neural Networks (RNNs) and an attention mechanism to construct latent representations of documents by aggregating the latent representations of sentences. It comprises four main components: a word sequence encoder, word-level attention, a sentence encoder, and sentence-level attention. This structure allows HAN to adjust the significance of individual words and sentences during classification and extract importance coefficients at both levels, aiding in the understanding of HAN's decision-making process. This feature is particularly valuable for the neighbourhood generator module in FEHAN.

3.1.3 Neighbourhood generator

We propose using a Markov Chain model to generate synthetic sentences S. Markov Chain Models are based on a sound statistical foundation and aim to model the probability of observing a series of events, where each event is a token of the corpus. This ensures that the synthetic sentences will be semantically similar to the original sentence, following the distributed semantic principle that "a word is characterized by the company it keeps" [41].

The neighborhood generator module receives the selected document along with the IS index. It examines the first element of IS to generate the synthetic sentences S. If the first element is a word, the Markov Chain uses it as the initial state; otherwise, if the index of IS refers to the end of the document, it starts from a random word.

The general structure of the Markov Chain text generator resembles a transition matrix. However, for implementation purposes, we utilize a dictionary of dictionaries to preserve all possible states (words) and all potential subsequent items in the chain. This setup allows the creation of neighbourhood data H by replacing the IS of the document to be explained with those in S.

Given a document d with m IS, the module generates $m \times |S|$ synthetic documents by sequentially exchanging one of the m sentences with one from S. Finally, each synthetic document in H is labelled using the HAN model.

3.1.4 Interpretable Classifier and Explanation

FEHAN constructs an interpretable decision tree c trained on locally generated documents. First, each document in the neighbourhood is transformed into a frequency vector representation using the bag of unigrams. Then, an interpretable classifier is trained on this data representation. If the instances of different class labels in the neighbourhood are imbalanced, we apply a heuristic proposed by King and Zeng [80], which assigns higher weights to the minority class and lower weights to the majority class. This model extracts the important features (words) for any class label, which are then used to produce an explanation. The explanation provided by our method is a saliency map of important words identified by the interpretable model.

We chose the decision tree as the interpretable model because as it is highly interpretable due to its intuitive and transparent structure, which closely mirrors human decision-making processes. Its tree-like visualization allows users to easily follow the decision path from the root to the leaf nodes, with each node representing a simple "if-then" rule based on a specific feature. This clear hierarchical structure ensures that the decision-making process is transparent, with no hidden layers or complex transformations, making it easy to understand how a particular prediction is reached. The most influential features on the prediction of class labels are structured in a top-down format, indicating the relative importance of features in the prediction [37].

An example of a returned saliency map based on the outputs of FEHAN and LIME is presented in Figure 3.3. In this figure, words relevant to the identified class label are highlighted in green, with shades of green indicating the importance of words that support the current class. Words relevant to the opposite class (in this example, the negative class) are highlighted in red. The intensity of the colours indicates the importance of each word for a class label. Words in green are essential for assigning the *positive* label to the document, while words in red support its assignment to the negative class. The interpretable text classifier is trained on neighbourhood documents generated with artificial sentences created by a Markov Chain text generator.

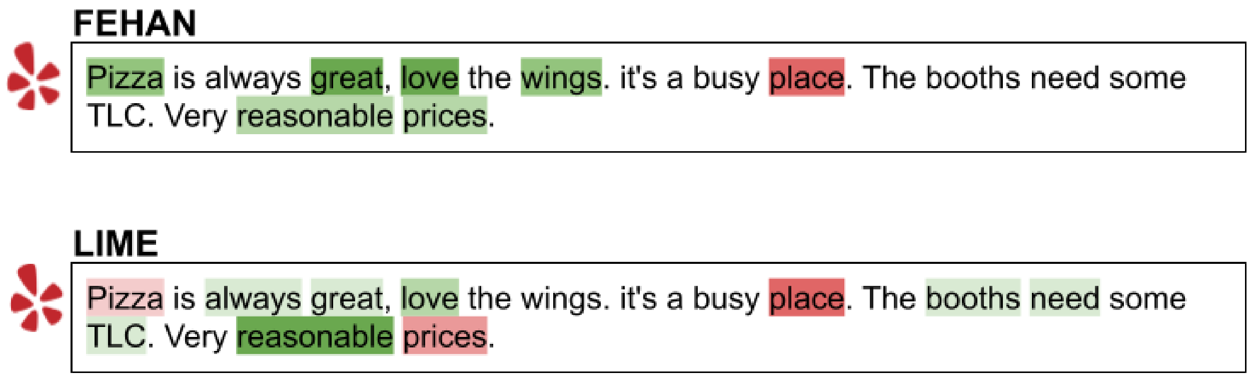

FEHAN
Pizza is always great, love the wings. it's a busy place. The booths need some TLC. Very reasonable prices.
LIME
Pizza is always great, love the wings. it's a busy place. The booths need some TLC. Very reasonable prices.

Figure 3.3: An original document from the Yelp data classified by HAN as a *"5 star"* place (on a scale of 1 - negative - to 5 - positive). The green shades represent the important features for assigning the class label "5 star" to the data, while the red shades represent features for assigning the "1 star" label. Stronger colours highlight more important features for classification. The important words are extracted from the decision tree.

3.2 DICTA

This section presents DICTA, a modularized model-agnostic framework for explaining black-box classifiers for text data. Let $d = \langle S_1, \ldots, S_n \rangle$ be a document represented as a sequence of sentences $S_i = \langle w_1, \ldots, w_m \rangle$, where $1 \leq i \leq n$ and w_j with $1 \leq j \leq m$ is a word. Explaining the decision of a black-box model f on a given document d (i.e., $f(d) = y$) means presenting an explanation e that belongs to a human-understandable domain *E* [55].

The proposed explanation method builds on the line of research into local model-agnostic methods initiated by Ribeiro et al. (2016) and Guidotti et al. (2019) [55, 136, 140]. DICTA aims to elucidate the reasons behind the classification prediction of a trained text classifier by studying its behaviour on the synthetic neighbourhood of a given document. Essentially, DICTA locally estimates the decision boundary of a complex decision function f for every classified document d. Specifically, the explanation e produced by DICTA approximates the decision boundary of f around d by highlighting the words most responsible for the decision $f(d) = y$ using the concept of

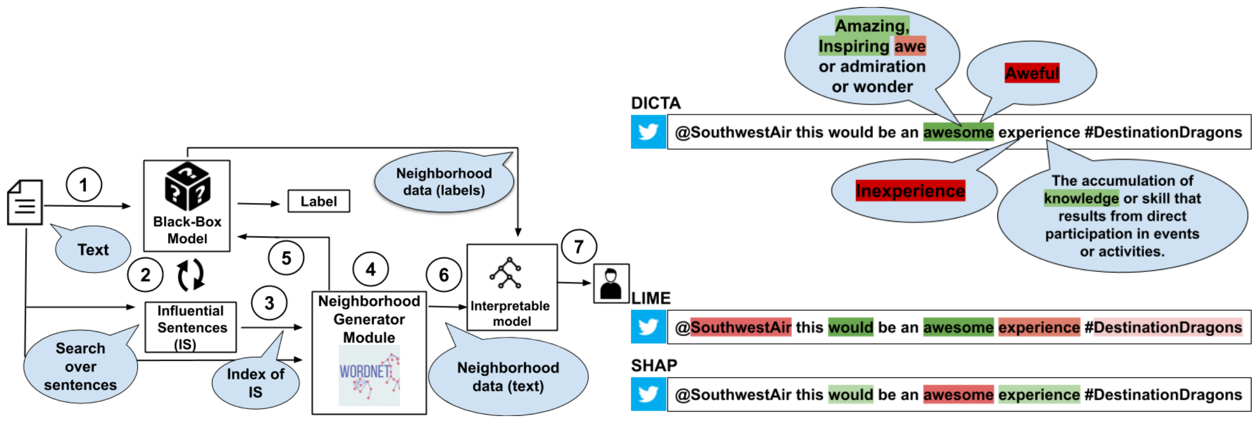

1
Text
Black-Box Model
Label
Neighborhood data (labels)
2
Influential Sentences (IS)
Search over sentences
3
Index of IS
5
4
Neighborhood Generator Module
WORDNET
6
Interpretable model
Neighborhood data (text)
7
DICTA
@SouthwestAir this would be an awesome experience #DestinationDragons
Amazing, Inspiring awe or admiration or wonder
Aweful
Inexperience
The accumulation of knowledge or skill that results from direct participation in events or activities.
LIME
@SouthwestAir this would be an awesome experience #DestinationDragons
SHAP
@SouthwestAir this would be an awesome experience #DestinationDragons
a
b

Figure 3.4: (a) Overview of DICTA framework. DICTA takes as input a textual document (step 1), classifies it with a black box, and extracts the most influential sentences impacting the probability label (step 2). It matches words in the influential sentences with possible semantic replacements using an ontology (e.g., WordNet) and generates a synthetic dataset (steps 3–5). Finally, it trains a local decision tree on the synthetic neighbourhood, exploits the tree to retrieve the importance of the words used for classification, and returns them to the final user (steps 6–7). (b) Explanations of a document labelled as "Positive" by a Bidirectional-GRU on the Airlines tweets dataset with DICTA, LIME, and SHAP. Positive and negative impacts are highlighted with green and red shades, respectively.

semantic replacement. Thus, the syntax of d remains relatively unchanged while its semantics are modified.

The core idea of DICTA (illustrated in Figure 3.4(a) and detailed in Algorithm 1) is to analyze how the semantic replacement of specific words affects the classification. DICTA focuses on words within *influential sentences* with the highest impact on document classification. The three main steps of DICTA for explaining the behaviour of black-box models are *(i)* identification of influential sentences, *(ii)* neighbourhood generation through semantic replacement, and *(iii)* local interpretable surrogate training and explanation extraction. DICTA operates by first identifying the most influential sentences in the document that significantly impact the model's prediction. The algorithm 1 does this by iteratively removing each sentence from the document and observing the change in the prediction probability. The top k sentences, those whose removal most alters the prediction, are selected as influential. Next, the algorithm performs semantic replacements on words within these influential sentences, using a predefined words ontology T to generate alternatives. This step helps create variations of the original document, forming a "neighborhood" of similar documents. The neighborhood is then classified by the black-box model to observe how these variations affect the prediction. Finally, the algorithm trains a decision tree on the neighborhood

Algorithm 1: DICTA(d, f, R, T)

```
Input  : $d$: document to explain, $f$: black box function, $k$: nbr. influential sentences, $T$:
         words ontology, $n$: neighbourhood size
Output : $e$ - explanation
$y_p \leftarrow f_p(d)$;                                   // get probability of prediction
$A \leftarrow \emptyset$;                                  // init. infl. sent. scores
for $S_i \in d$ do
    $d' \leftarrow remove(S_i, d)$;                        // remove sentence
    $A_i \leftarrow |y_p - f_p(d')|$;                      // store cand score.
$I \leftarrow select(\mathcal{S}_c, k)$;                   // get indexes top k sentences
$\mathcal{S} \leftarrow \{S_i | i \in I\}$;                // select top k sentences
$\mathcal{R} \leftarrow \emptyset$;                        // init. semantic replacement
for $S_i \in \mathcal{S}$ do
    for $w_j \in S_i$ do
        $\mathcal{R}_{w_j} \leftarrow repl(w_j, T)$;       // semantic replacement
$N \leftarrow \emptyset$;                                  // init. neighbourhood
for $i \in [1, n]$ do
    $d' \leftarrow copy(d)$;                               // copy the document
    $S_i \leftarrow rndSelection(\mathcal{S})$;            // select sentence
    $W \leftarrow rndSelection(S_i)$;                      // select words
    for $w_j \in W$ do
        $d' \leftarrow repalce(w_j, \mathcal{R}_{w_j}, S_i, d')$;   // replace word j in sentence i
    $N \leftarrow N \cup \{d'\}$;                          // add to neighbourhood
$Y \leftarrow f(N)$;                                       // classify neighbourhood
$dt \leftarrow train(N, Y)$;                               // train decision tree
$e \leftarrow extractExpl(dt, f(d))$;                      // get explanation
return $e$;
```

data, using the variations and their corresponding predictions to model the decision-making process of the black-box model. The decision tree, being interpretable, is used to extract a human-readable explanation e that explains the model's prediction for the original document. The output is a clear and understandable explanation of why the model made its specific prediction.

3.2.1 Influential Sentences Extraction

A key component of DICTA is the identification of influential sentences that have a high impact on the document's class label. These influential sentences contain the most descriptive words essential for distinguishing the document's class label.

DICTA identifies influential sentences as follows (lines 1–7 in Algorithm 1, steps 1–2 in Figure

3.4(a)). First, DICTA queries the black-box model f and stores the *probability* of obtaining the label $y = f(d)$ for the document under analysis d ($y_p = f_p(d)$ in Algorithm 1, line 1). Then, for each sentence $S_i \in d$ (lines 3–5), DICTA creates a synthetic document d' as a copy of d but without the sentence S_i (line 4). It then stores in the influential sentences score candidate set A the absolute deviation between y_p and $f(d')$.

Finally, DICTA identifies the indexes of the k sentences with the most significant influence and stores them in the set $\mathcal{S}$. The value of k is a data-dependent hyper-parameter determined based on the dataset's average or maximum number of sentences. After this step, the influential sentences $\mathcal{S}$ and the document d are passed to the neighbourhood generator process (step 3 in Figure 3.4(a)).

3.2.2 Neighbourhood Generator

The neighbourhood generator process is responsible for creating synthetic documents Z that are similar to d. These documents are used to query the black-box function to understand the reasons behind the label $y = f(d)$. The process starts with the identification of the set of words $\mathcal{R}$ to be used for semantic replacement (lines 8–11 in Algorithm 1, steps 4–5 in Figure 3.4(a)). For each influential sentence $S_i \in \mathcal{S}$ and for each word $w_j \in S_i$, DICTA identifies the set of words $\mathcal{R}_{w_j}$ to be used as semantic replacements for w_j with respect to a given ontology T, ensuring that the meaning of the sentence remains unchanged upon substitution (line 11).

Given the number n of neighbours to generate, DICTA creates a copy d' of the document under analysis d (line 14). It randomly selects an influential sentence S_i (line 15) and then randomly chooses a set of words W from S_i (line 16). These selected words W are replaced with random words from their semantic replacements $\mathcal{R}_{w_j}$ (lines 17–18). Finally, the synthetic document created through this procedure is stored in the neighbourhood N. After generating the synthetic neighbourhood, DICTA queries the black-box function to classify the synthetic documents, resulting in $Y = f(N)$ (line 20).

In our implementation, we use *WordNet* [115] as the ontology T to find semantic replacements for the words. WordNet is a robust lexical database that helps preserve the distribution of document features, ensuring that the synthetic sentences are semantically similar to the original ones according to the distributed semantic principle [41].

Using WordNet offers two significant advantages: *(i) Transparency*: The relations between lexical categories are intuitive and understandable to users regardless of their linguistic knowledge. *(ii) Accessibility*: WordNet is freely available in more than 200 languages and has connectors to many programming languages and systems without requiring fine-tuning. Utilizing WordNet as the primary resource for neighbourhood generation makes our method more appealing for use in limited computing environments, as opposed to sophisticated language models like BERT, which impose a heavy computational overhead on interpretability tasks.

3.2.3 Local Decision Tree and Explanation

After generating the neighbourhood, the neighbourhood N and the corresponding labels Y are used to train an interpretable local surrogate model (line 22 in Algorithm 1, step 6 in Figure 3.4(a)) by training an interpretable decision tree dt (line 22) on N and Y. We adopt a decision tree to explain the black box's local behaviour due to its simplicity and comprehensibility for non-expert users [167]. The decision tree is then used to extract the most important words responsible for the classification, forming the output explanation e (lines 23–24 in Algorithm 1, step 7 in Figure 3.4(a)).

Given the document d, DICTA extracts the importance of words by tracing the conditions triggered by d along the path from the root node to the leaves of dt. The importance of words is obtained as the normalized total reduction of the Gini criterion in the decision tree dt, referred to as *Gini importance* [16].

Quantifying the importance of words enables the construction of a "saliency map" that highlights the important words w_j in d with their corresponding scores. Besides the importance of words, we chose the decision tree as an interpretable surrogate model because its graphical representation allows users to visually and comprehensively trace the decisions of a black box. The words in the tree are structured in a top-down format, with the most important ones appearing closest to the root [37].

An example of an explanation e returned by DICTA is presented in Figure 3.4(b). Here, the relevant words for the positive class are coloured green, while those for the negative class are coloured red. The colour intensity of the highlighted words indicates their importance concerning a specific class.

The advantage of using semantic replacement is that the original structure of a document is preserved, making it easy to observe the importance of other words replaced with the highlighted ones among those in the synthetic neighbourhood. In contrast, the random elimination of words for neighbourhood generation, as performed by LIME or SHAP (Figure 3.4(b)), may result in documents that are too short and do not retain the same structure and meaning as the original document.

Additionally, random elimination may frequently remove a group of words from the document, and their real effect may not be adequately recognized. DICTA overcomes this by understanding the effect of each word by replacing it with words having the same or opposite meanings. For instance, if a word is very influential in labelling a document, replacing it with a word with the opposite meaning will significantly affect the probability of the class label. The comparison shows that DICTA focuses on the most important words and provides a richer explanation by attaching the set of words derived from WordNet and highlighting their impact on text classification.

3.3 Evaluation and Analysis

In this section, we demonstrate FEHAN and DICTA's effectiveness through quantitative and qualitative evaluations[2]. We first describe the experimental settings for both algorithms. Due to the structural differences between these frameworks, we conducted separate sets of experiments using three sentiment analysis datasets. We then present a quantitative evaluation on different metrics compared to local explainers across various datasets. Finally, we provide a qualitative evaluation that practically illustrates the benefits and readability of the explanations returned by these two algorithms.

Datasets

We experimented on four textual datasets commonly used to train classifiers that detect the sentiment of documents:

1. *IMDB*: Movie reviews containing highly polarized opinions [108]. 2. *Yelp*: Business reviews [157]. 3. *Amazon*: Product reviews dataset [61]. 4. *Airline Tweets*: Anonymous tweets related to U.S. airlines [134].

The number of instances, number of categories, and the average/maximum number of words and sentences in each dataset are shown in Table 3.1 (left). We split the data into training (80%), testing (10%), and validation (10%) sets. The table also reports the number of informative sentences DICTA uses, which varies across datasets.

Detailed descriptions of the datasets and the accuracy of various classifiers on each dataset.

Classifiers

Since DICTA is a model-agnostic explainer, we evaluated it by explaining four text classifiers: CNN1D, BiGRU, BiLSTM (all based on deep learning), and a Random Forest (RF) classifier. The

[2]Python code and datasets available at: `https://github.com/MahtabSarvmaili`. Experiments were conducted on Ubuntu 20.04.1 LTS, Intel® Core™ i7 CPU, 16 GB DIMM DDR4 RAM.

Dataset	#docs	#categ	Avg #w	Max #w	Avg #S	Max #S	#infl. sent.	BiLSTM	BiGRU	CNN1d	RF
Yelp	700k	5	9	438	8	150	5	0.60	0.61	0.56	0.52
Amazon	278k	5	8	169	4	122	3	0.66	0.67	0.65	0.60
Airline Tweets	14k	3	7	20	2	9	2	0.78	0.77	0.76	0.70
IMDB	50k	2	13	384	10	117	6	0.86	0.86	0.85	0.79

Table 3.1: Detailed descriptions of the datasets and the accuracy of various text classifiers on each dataset. (left), accuracy (right).

structure of the text classifiers is as follows: 1. *BiLSTM*: Uses one embedding layer initiated with pre-trained vectors, two layers of bidirectional LSTM, followed by a Dense layer and a softmax over the class labels. 2. *BiGRU*: Similar architecture to BiLSTM, but uses Bidirectional GRU instead of LSTM. 3. *CNN1D*: Follows the architecture described by Kim (2014) [79].

The deep learning models were trained with the following parameters: batch size 200, word embedding dimension 200, maximum number of unique tokens 200k, and 10 training epochs. The RF classifier was trained with 150 trees. The accuracy of the various classifiers is reported in Table 3.1 (right).

Metrics

We measured the effectiveness of the explanations returned by DICTA using the following indicators: 1. *Correctness*: Assessed the importance scores of words with respect to the sentiment scores assigned by the sentiment lexicon VADER [65]. 2. *Fidelity*: Measured the fidelity of the local surrogate model with respect to the black-box classifier. 3. *Plausibility and Similarity of the Neighborhood*: Evaluated in terms of outliers present in the generated data and similarity between real data and synthetic neighbourhoods.

Comparisons

We compared DICTA with LIME [136] and SHAP [104]. Due to the structural similarity between DICTA and LIME, we performed comparisons across all the aforementioned measures. Since SHAP does not train a local surrogate or generate a neighbourhood, we limited the evaluation of DICTA and SHAP to the correctness measure. For DICTA, we adopted the following parameter settings based on preliminary experimentation (not reported here due to space constraints): the number of influential sentences selected for each dataset is reported in Table 3.1, and we generated neighbourhoods composed of $n = 100$ synthetic documents.

	IMDB	**Amazon**	**Yelp**	**Airline tweets**
50	$1.94 \times e^{-05}$	$2.81 \times e^{-01}$	$4.6 \times e^{-01}$	$1.51 \times e^{-02}$
100	$3.53 \times e^{-11}$	$1.59 \times e^{-02}$	$5.2 \times e^{-01}$	$6.53 \times e^{-02}$
150	$1.69 \times e^{-15}$	$2.14 \times e^{-01}$	$1.89 \times e^{-01}$	$2.4 \times e^{-02}$
200	$5.23 \times e^{-19}$	$1.7 \times e^{-02}$	$2.57 \times e^{-01}$	$2.38 \times e^{-02}$
250	$3.07 \times e^{-22}$	$8.8 \times e^{-02}$	$1.82 \times e^{-01}$	$5.27 \times e^{-06}$
300	$1.24e - 22 \times e^{-22}$	$4.1 \times e^{-02}$	$1.99 \times e^{-06}$	$1.13 \times e^{-05}$

Table 3.2: The p-values obtained from the Wilcoxon test for FEHAN and LIME across four datasets (IMDB, Amazon, Yelp, and Airline Tweets). These low p-values indicate significant differences between the fidelity of FEHAN and LIME, demonstrating that FEHAN statistically outperforms LIME in all benchmark datasets

3.3.1 Quantitative Evaluation (FEHAN)

To evaluate FEHAN's behaviour for document classification, we generated a set of 300 neighbourhood examples for each given instance and fed these examples back to HAN for classification.

We compared the outcomes of FEHAN against the LIME text explainer, focusing on their fidelity to the black-box model. Fidelity refers to how faithfully an interpretable model imitates the behaviour of the black box in the neighbourhood of a particular data point. This is crucial because the meaningfulness of an explanation depends on it being at least locally faithful. Fidelity can be measured as the accuracy of the local surrogate model c's predictions on the neighbourhood N_d generated for document d, in comparison to the black-box model f's predictions on the same set. Specifically, we compare $y_c = c(N_d)$ with $y_f = f(N_d)$, using the accuracy between y_c and y_f as the evaluation measure.

To measure fidelity, we tested the model with six uniformly sampled sets of test data, consisting of 50, 100, 150, 200, 250, and 300 instances. Figure 3.5 reports the observed fidelity for these sets. The results indicate that FEHAN outperforms LIME in mimicking the black-box behaviour across all datasets. Furthermore, FEHAN exhibits significantly less variance compared to LIME. These findings suggest that FEHAN is more faithful to the black-box behaviour than the local interpretable classifier provided by LIME.

Moreover, we conducted statistical tests on our results to ensure their validity across all datasets. To this end, we employed the Wilcoxon test to analyze the fidelity of these models. The p-values obtained from the Wilcoxon test for all four datasets are reported in Table 3.2. These very low p-values indicate significant differences between FEHAN and LIME, demonstrating that FEHAN statistically outperforms LIME regarding fidelity in all benchmark datasets.

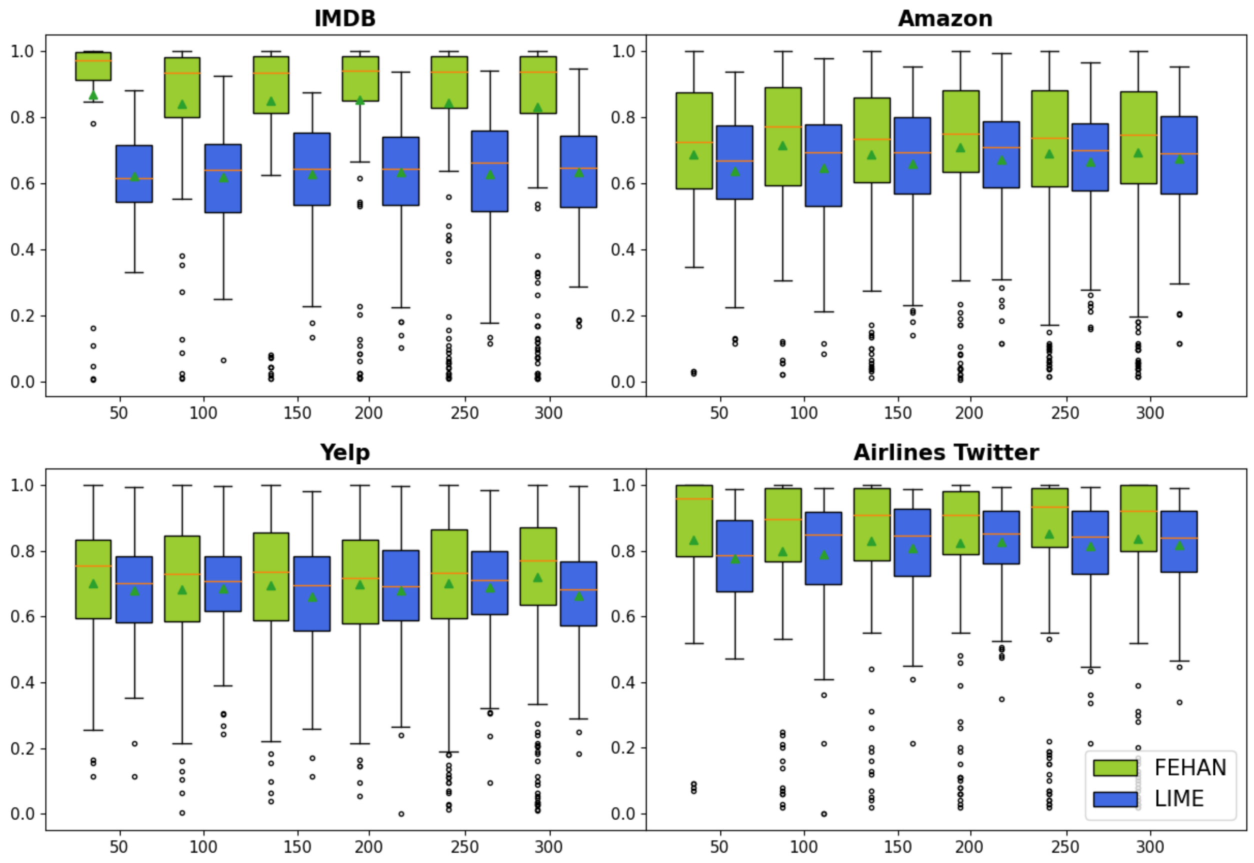


Figure 3.5: Box plots showing the fidelity of FEHAN and LIME across four datasets (IMDB, Amazon, Yelp, and Airline Tweets). The fidelity was measured using six uniformly sampled test sets consisting of 50, 100, 150, 200, 250, and 300 instances. The results demonstrate that FEHAN consistently outperforms LIME in mimicking the black-box model behaviour and exhibits significantly less variance, indicating a higher degree of faithfulness to the black-box behaviour compared to LIME.

Subsequently, we conducted a numerical evaluation of the density and cohesion of neighbourhood data generated by the two explanation approaches [53]. To quantitatively demonstrate that FEHAN produces higher quality neighbourhood text data compared to LIME, we employed two approaches: (i) measuring the cosine distance between the original document and the neighbourhood (i.e., the cohesion of the neighbourhood) and (ii) measuring the Local Outlier Factor (LOF), which assesses the density and compactness of the neighbourhood.

The average cosine distance value between the original document and the neighbourhood data examples indicates the degree of similarity between the neighbourhood data and the original document. The results in Table 3.3 show that the average cosine distance for FEHAN's neighbourhood is

	IMDB		Amazon		Yelp		Airline Twitter	
	FEHAN	**LIME**	**FEHAN**	**LIME**	**FEHAN**	**LIME**	**FEHAN**	**LIME**
50	0.674	0.333	0.719	0.320	0.700	0.319	0.789	0.335
100	0.677	0.324	0.750	0.328	0.685	0.321	0.800	0.330
150	0.713	0.322	0.765	0.330	0.682	0.322	0.796	0.333
200	0.689	0.322	0.739	0.329	0.697	0.322	0.794	0.330
250	0.702	0.323	0.739	0.330	0.705	0.322	0.788	0.334
300	0.706	0.322	0.748	0.330	0.702	0.321	0.792	0.329

Table 3.3: The average cosine distance between the original document and neighbourhood data generated by FEHAN and LIME across four datasets (IMDB, Amazon, Yelp, and Airline Twitter). Higher average cosine distances for FEHAN indicate greater diversity in the vocabulary of the generated documents, suggesting that FEHAN's neighbourhood data exhibits higher quality and diversity compared to LIME.

greater than that for LIME. The higher value of FEHAN's cosine distance across four datasets suggests a greater diversity in the vocabulary of the generated documents. This implies that FEHAN's neighbourhood data exhibits higher quality and diversity.

The Local Outlier Factor (LOF) is a metric for anomaly detection proposed by Breunig et al. [17]. LOF compares a data point's local density against its neighbours' local densities using the k-nearest neighbourhood to identify regions of similar density. Points with significantly lower density compared to their neighbours are considered outliers. In our study, we used LOF to evaluate the quality of the neighbourhood, aiming for denser neighbourhoods with fewer outliers.

Table 3.4 reports the average LOF for FEHAN and LIME across the four datasets. The results indicate that FEHAN generally has a lower LOF compared to LIME, suggesting that FEHAN's neighbourhood generation based on the Markov Chain results in much denser neighbourhoods with fewer outliers.

Comparing the cosine similarity and LOF of the generated data reveals that FEHAN exhibits greater diversity in the number of words. These words are more semantically similar to the original document compared to those generated by LIME. This combination of higher semantic similarity and denser neighbourhoods underscores the superior quality of FEHAN's neighbourhood data.

3.3.2 Qualitative Evaluation (FEHAN)

To evaluate the neighbourhood data qualitatively, we provide examples of neighbours generated by FEHAN and LIME, along with the predicted class labels assigned by their interpretable models.

Figure 3.6 illustrates examples of positive, neutral, and negative neighbourhood documents

	IMDB		Amazon		Yelp		Airline Twitter	
	FEHAN	LIME	FEHAN	LIME	FEHAN	LIME	FEHAN	LIME
50	$1.42 \times e^{-7}$	2.22	1.028	1.78	1.057	$7.55 \times e^{-7}$	1.042	$2.11 \times e^{-7}$
100	$7.80 \times e^{-7}$	2.35	1.027	$2.55 \times e^{-7}$	1.71	$4.12 \times e^{-7}$	1.037	$2.10 \times e^{-7}$
150	$1.15 \times e^{-7}$	2.36	1.028	$5.88 \times e^{-7}$	1.076	$2.52 \times e^{-7}$	$1.96 \times e^{-7}$	$4.88 \times e^{-7}$
200	$1.94 \times e^{-7}$	2.29	1.034	$1.45 \times e^{-7}$	1.067	$3.43 \times e^{-7}$	1.042	$2.11 \times e^{-7}$
250	$1.13 \times e^{-7}$	9.61	1.035	$1.53 \times e^{-7}$	1.063	$3.54 \times e^{-7}$	$1.89 \times e^{-7}$	$5.03 \times e^{-7}$
300	$6.19 \times e^{-7}$	2.30	$1.79 \times e^{-7}$	$4.26 \times e^{-7}$	1.069	$2.50 \times e^{-7}$	1.038	$3.84 \times e^{-7}$

Table 3.4: Average Local Outlier Factor (LOF) for FEHAN and LIME across four datasets (IMDB, Amazon, Yelp, and Airline Twitter). The results show that FEHAN generally has a lower LOF compared to LIME, indicating that FEHAN's neighbourhood generation using the Markov Chain results in denser neighbourhoods with fewer outliers. This suggests that FEHAN's generated neighbourhoods are of higher quality, with greater semantic similarity to the original documents.

generated and classified by FEHAN's decision tree and LIME's text explainer for the Amazon dataset[3]. In this figure, green, blue, and red colours indicate positive, neutral, and negative classes, respectively. The original and pre-processed documents are presented in the first two rows. Since the training process uses the pre-processed data, the generated instances follow the same format.

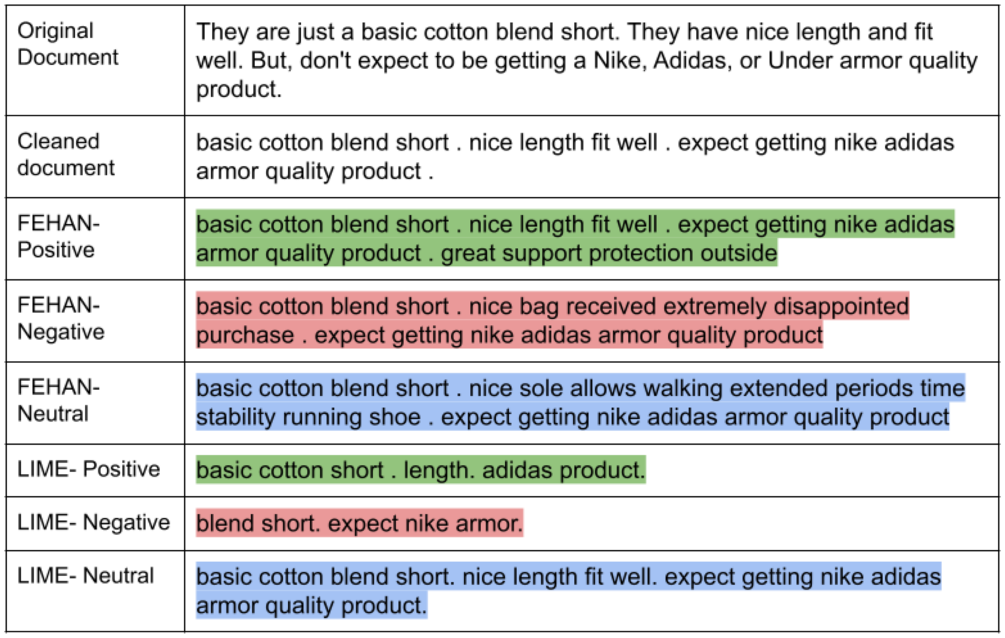
We observe that FEHAN generates documents more similar to the original ones compared to LIME. The integrity of the data is well preserved, with semantically equivalent examples sampled from the original dataset. FEHAN maintains the central concept of the given instance and non-important parts, generating examples that reflect the same context as the original but within the neighbourhood data. In contrast, LIME tends to lose information during neighbourhood generation. This occurs because LIME suppresses words in the document, while FEHAN replaces only the informative sentences with synthetic ones. Additionally, LIME's neighbourhood generation process can result in invalid examples due to the elimination of all input features, a condition that worsens for datasets with generally shorter documents.

3.3.3 Quantitative Evaluation (DICTA)

Salient Scores and Sentimental Polarity Agreement

Inspired by the work of Atanasova et al. (2020) [6], we conducted an experiment to compare the salient scores of words with their sentimental polarity extracted from a sentiment lexicon resource. This experiment aimed to observe the impact of words on the prediction of the black-box model

[3]Although the Amazon dataset has five categories, we have selected the most frequent ones, which are the negative, positive, and neutral instances generated from FEHAN and LIME neighbourhoods.



Original Document	They are just a basic cotton blend short. They have nice length and fit well. But, don't expect to be getting a Nike, Adidas, or Under armor quality product.
Cleaned document	basic cotton blend short . nice length fit well . expect getting nike adidas armor quality product .
FEHAN-Positive	basic cotton blend short . nice length fit well . expect getting nike adidas armor quality product . great support protection outside
FEHAN-Negative	basic cotton blend short . nice bag received extremely disappointed purchase . expect getting nike adidas armor quality product
FEHAN-Neutral	basic cotton blend short . nice sole allows walking extended periods time stability running shoe . expect getting nike adidas armor quality product
LIME- Positive	basic cotton short . length. adidas product.
LIME- Negative	blend short. expect nike armor.
LIME- Neutral	basic cotton blend short. nice length fit well. expect getting nike adidas armor quality product.

Figure 3.6: A neighbourhood example generated by FEHAN and LIME for an instance of the Amazon dataset. The green, blue and red depict this dataset's most positive, neutral and negative classes. FEHAN's decision tree and LIME's text explainer determine the positive and negative labels for the generated neighbourhood data.

regardless of the class label.

We binarized the class labels of the datasets using the middle class as a threshold to assign the two labels. For example, in the Yelp dataset, which has 1-5 star ratings for businesses, ratings from 1-3 were assigned as "negative" reviews, and ratings from 4-5 were assigned as "positive" reviews. A similar procedure was applied to the Amazon dataset, and for the U.S. Airline tweets, ratings of 1-2 were labelled as negative and 3 as positive. We also binarized the importance scores of LIME and SHAP by considering negative scores as a negative contribution and positive scores as a positive contribution towards a label value.

For validation purposes, we extracted the sentiment scores of words from the sentiment lexicon VADER [65] and binarized them into *positive* and *negative* classes. For DICTA, we obtained the polarity of words at each node of the decision tree by examining the normalized ratio of the number of samples falling into each class. If the class label was considered "positive," we assigned a positive annotation to the words; otherwise, they were labeled as "negative." Finally, we measured the percentage of agreement between the classes provided by LIME, SHAP, and DICTA and the sentiment suggested by VADER—the higher the agreement, the better.

The results, shown in Table 3.5, indicate that DICTA outperforms LIME and SHAP on three out

		Yelp	**Amazon**	**Airline Tweets**	**IMDB**
BiGRU	DICTA	0.57	0.65	0.59	0.48
	LIME	0.43	0.48	0.43	0.52
	SHAP	0.47	0.50	0.47	0.84
BiLSTM	DICTA	0.56	0.66	0.61	0.48
	LIME	0.46	0.47	0.41	0.53
	SHAP	0.47	0.46	0.49	0.85
CNN1D	DICTA	0.57	0.65	0.59	0.48
	LIME	0.46	0.43	0.37	0.52
	SHAP	0.47	0.51	0.41	0.84

Table 3.5: Agreement between the sentiment polarity extracted from explainers (DICTA, LIME, SHAP) and the VADER sentiment lexicon across four datasets (Yelp, Amazon, Airline Tweets, and IMDB). The table shows that DICTA generally achieves higher agreement with VADER compared to LIME and SHAP on most datasets, except for the IMDB dataset where SHAP performs best.

of four datasets (Yelp, Amazon, and Airline Tweets). However, SHAP performed best on the IMDB dataset, which requires further investigation to understand the underlying reasons. We found that examining the word importance extracted from the decision tree provided significant insights into word scores. The decision tree structure allowed for a better exploration of the connection between the input document and the lexicon composing it.

In cases where the label assigned by the black-box model to a document is negative, the document typically contains strong negative words that are often found in the top nodes of the decision tree. When tracing down the tree from the root to the leaves to obtain the polarity of words, most neutral or positive words (which could flip the class of a document) are found at intermediate levels or closer to the leaves. Although these words reduce the number of negative instances at their level, many instances may still belong to the negative class, causing the positive words to be assigned a negative sentiment. Therefore, traversing the tree empowers users to understand the relationship between words and their impact on the sentiment label of a document.

Fidelity Evaluation

We compared DICTA against LIME in text classification by measuring the fidelity of the interpretable surrogate model with respect to the black-box model, as described by Doshi-Velez and Kim (2017) and Guidotti et al. (2018) [34, 58].

For DICTA, c is the local decision tree, while for LIME, c is the local regressor. We calculated

	IMDB	Amazon	Yelp	Airline tweets
BiLSTM	5.33×10^{-29}	3.88×10^{-08}	1.15×10^{-28}	1.44×10^{-11}
BiGRU	1.91×10^{-31}	2.48×10^{-09}	3.13×10^{-29}	3.61×10^{-05}
CNN1D	1.88×10^{-41}	3.66×10^{-28}	2.91×10^{-28}	1.79×10^{-15}
RF	1.11×10^{-41}	4.23×10^{-11}	9.82×10^{-43}	6.56×10^{-20}

Table 3.6: The p-values of the Wilcoxon test shows the statistical significance of DICTA's performance improvement over LIME for fidelity.

fidelity by uniformly sampling 300 instances from the test data and measured the accuracy of DICTA and LIME in mimicking the behaviour of black-box decisions for their respective neighbourhoods. The comparison of DICTA and LIME fidelity is presented in the box plots in Figure 3.7. The results demonstrate that DICTA outperforms LIME across all datasets and all text classifiers in imitating the black-box behaviour.

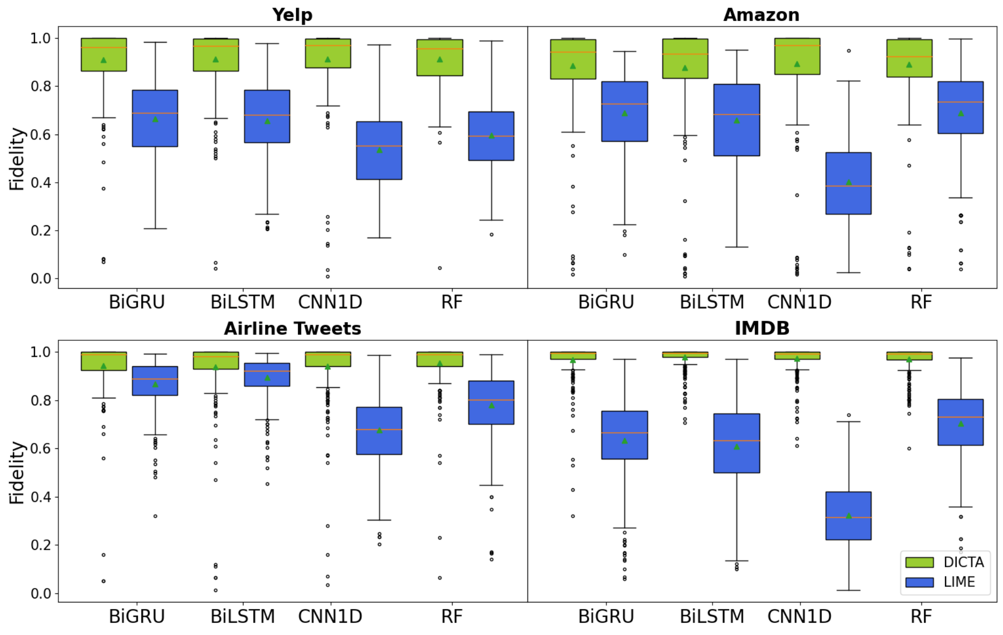


Figure 3.7: Fidelity (accuracy) of DICTA and LIME on datasets

We also evaluated the statistical significance of the improvement in DICTA's performance over LIME. To this end, we employed the Wilcoxon test to analyze the fidelity of these models. The p-values for all datasets and black-box models are reported in Table 3.6. These very low p-values indicate significant differences between DICTA and LIME performances, providing statistical evidence of DICTA's superior fidelity.

		Yelp		**Amazon**		**Airline Twitter**		**IMDB**	
		C	**L**	**C**	**L**	**C**	**L**	**C**	**L**
BiLSTM	DICTA	0.25	1.54	0.28	1.33	0.24	7.25×1e3	0.17	1.35
	LIME	0.27	1.57	0.24	1.31	0.30	1.0	0.21	1.96
BiGRU	DICTA	0.26	1.24	0.28	1.91	0.23	1.41	0.19	1.76
	LIME	0.29	1.25	0.27	1.62	0.29	1.0	0.24	3.41
CNN1D	DICTA	0.28	1.51	0.22	1.5	0.16	1.07	0.14	1.57
	LIME	0.33	1.29	0.25	2.37×1e4	0.28	1.13×1e7	0.20	1.40
RF	DICTA	0.34	1.20	0.45	2.45×1e7	0.38	1.32	0.23	1.68
	LIME	0.33	1.46	0.32	1.69	0.32	3.47	0.32	1.72

Table 3.7: The average cosine distance (**C**), and LOF (**L**) between the original document and neighbourhood data of DICTA and LIME.

Synthetic neighbourhoods Evaluation

In this section, we discuss the evaluation results on the quality of synthetic textual data generated in local neighbourhoods. We quantitatively evaluated the density and cohesion of neighbourhood data, as suggested by Guidotti et al. (2020) [53], to demonstrate that DICTA produces diverse, plausible, and high-quality neighbourhood text data compared to LIME. To achieve this, we employed two approaches:

1. **Cosine Distance**: We measured the average cosine distance between the original document d and the neighborhood N_d. This metric provides evidence about the similarity (cohesion) of the neighbourhood to the original document.

2. **Local Outlier Factor (LOF)**: We measured the Local Outlier Factor (LOF) as proposed by Breunig et al. (2000) [17]. LOF captures the level of neighbourhood density, offering insights into its plausibility and degree of diversity.

The average cosine distance value (columns labelled **C**) between the original document and the synthetic documents in the neighbourhood indicates the degree of similarity between them. The results in Table 3.7 show that the average cosine distance to DICTA's neighbourhood is lower than LIME's neighbourhood, especially for deep text classifiers.

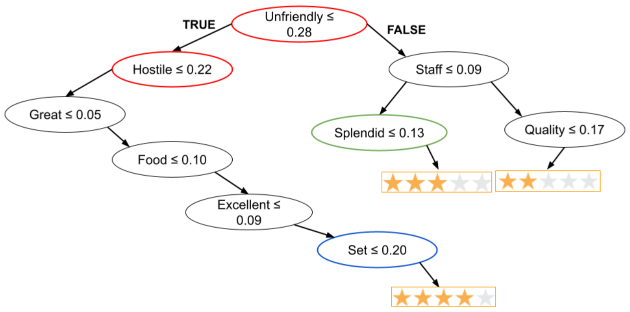
The Local Outlier Factor (LOF) is an anomaly detection approach that compares a data point's local density against its neighbours' local densities to identify regions of similar density. It recognizes these regions using the k-Nearest Neighbors, with points having considerably lower density compared to their neighbours being considered outliers.

We employed LOF to evaluate the compactness and density of the neighbourhood with respect to a reference population given by the original dataset. Table 3.7 reports the average LOF (columns labelled **L**) for DICTA and LIME across the four datasets. The results indicate that DICTA has similar or higher LOF values for most datasets compared to LIME. This suggests that DICTA's neighbourhood generation, based on WordNet, explores at least a similar or wider neighbourhood area around the given document.

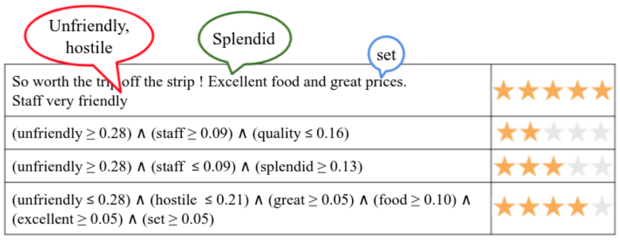
Considering both cosine distance and LOF values (Table 3.7) of the generated data, we can infer that although DICTA's neighbourhood has a greater diversity in terms of the number of different words, the generated text remains semantically similar to the original document.

3.3.4 Qualitative Evaluation (DICTA)

We linearized the decision tree into an understandable rule form that provides a more flexible semantics for representing the classifier [43]. Rules can be extracted from the decision tree by tracing down a decision path from the root node to the leaf.



a


Unfriendly, hostile
Splendid
set

So worth the trip off the strip ! Excellent food and great prices. Staff very friendly	★★★★★
(unfriendly ≥ 0.28) ∧ (staff ≥ 0.09) ∧ (quality ≤ 0.16)	★★☆☆☆
(unfriendly ≥ 0.28) ∧ (staff ≤ 0.09) ∧ (splendid ≥ 0.13)	★★★☆☆
(unfriendly ≤ 0.28) ∧ (hostile ≤ 0.21) ∧ (great ≥ 0.05) ∧ (food ≥ 0.10) ∧ (excellent ≥ 0.05) ∧ (set ≥ 0.05)	★★★★☆

b

Figure 3.8: (a) A local surrogate tree explaining CNN1D for a Yelp review. (b) Decision rules extracted from the tree. The first row of the table shows the document under analysis and its label (five-star review), and the callouts at the top of each word present words from the semantic set (green for synonyms, red for antonyms, and blue for hyponyms and hypernyms) that were replaced with the word. The second to the fourth rows are the decision rules extracted from the decision tree and the final labels assigned to these rules. The antecedents of the rules are in the left column, and their consequents are in the right column.

Figure 3.8 presents an example of a trained decision tree and its traversal for an instance of Yelp reviews. Since we use TF-IDF vectors for training the decision tree, the real value that each feature is compared to translates to the importance of that word in the neighborhood text set. Starting from the root node, we follow paths to reach the leaves. By tracing down the decision tree and following the ensuing rules, we identify words in the order of their importance for the decision at the leaf of the tree. The salient words are presented to the user in the order of their importance for the decision at the leaf of the tree (in the consequent of a decision rule).

As shown in the classical cognitive science literature investigating the explainability of decision rules [45], understanding decision rules in this manner, which suggests causality, provides a better explanation of the rules (and therefore a better explanation of the model) than one consisting of an unordered set of salient words. Referring to Figure 3.8(b), the user may find an explanation starting with "unfriendly" and "hostile staff" a sufficient cause to understand why a restaurant is not recommended, rather than an explanation starting with "set" or "quality."

3.4 Summary

This chapter has introduced two innovative frameworks designed to enhance the interpretability of text classifiers: FEHAN (Framework for Explaining Hierarchical Attention Networks) and DICTA (modularizeD model-agnostic framework for the explanatIon of black box Classifiers for Text dAta). These frameworks address the growing need to understand the decision-making processes of sophisticated text classification models, particularly those based on deep learning, by generating interpretable and meaningful explanations.

FEHAN is tailored to provide local explanations for the Hierarchical Attention Network (HAN), which employs hierarchical structures with attention mechanisms at both the word and sentence levels for document classification. The framework consists of several key components. Firstly, it identifies the most informative sentences that have the highest impact on the classification. These sentences are then replaced with semantically similar ones generated by a Markov Chain model. This approach ensures that the synthetic sentences retain semantic coherence with the original text. An interpretable decision tree is subsequently trained on these synthetic documents to extract important words and construct a saliency map. The saliency map highlights the crucial words contributing to the black-box model's decision, thereby offering a clear and understandable explanation.

Quantitative evaluation of FEHAN involved measuring its fidelity—how well the interpretable model replicates the behavior of the black-box model. This evaluation showed that FEHAN outperforms the LIME text explainer in all datasets and across various metrics. The statistical significance of these results was confirmed using the Wilcoxon test. Additionally, FEHAN's qualitative evaluation demonstrated its ability to generate documents that are semantically similar to the original, maintaining the integrity and context of the data. This ensures that the explanations are not only accurate but also relevant and meaningful.

DICTA expands the scope of interpretability to a broader range of text classifiers through a model-agnostic approach. It leverages lexical databases such as WordNet to generate explanations. The framework begins by identifying influential sentences, similar to FEHAN, and then creates synthetic documents by semantically replacing words within these sentences. These synthetic documents are used to train a local decision tree, which provides the final explanations.

DICTA was subjected to rigorous quantitative evaluation, including fidelity measurements and statistical significance tests, which showed that DICTA consistently outperforms LIME in imitating the black-box behaviour. The evaluation of synthetic neighbourhoods revealed that DICTA generates high-quality, diverse, and semantically coherent text. This was evidenced by lower average cosine distances and comparable or higher Local Outlier Factor (LOF) values, indicating that DICTA explores a similar or broader neighbourhood around the given document while maintaining semantic similarity.

The qualitative evaluation of DICTA involved transforming the decision tree into understandable rule forms, which provide a flexible semantic representation of the classifier. By tracing decision paths from the root to the leaves, the salient words are presented in order of their importance for the decision. This method, rooted in classical cognitive science literature, offers a more intuitive and causal explanation compared to unordered sets of salient words. This approach helps users understand why certain words lead to specific classifications, enhancing the transparency and trustworthiness of the model.

Overall, the chapter underscores FEHAN and DICTA's effectiveness in improving text classifiers' interpretability. Both frameworks exhibit high fidelity, produce high-quality neighbourhood data, and provide clear and understandable explanations. These attributes make them robust tools for explainable AI in natural language processing. By improving our understanding of complex text classifiers, FEHAN and DICTA contribute to the development of more transparent, reliable, and trustworthy AI systems. This advancement is crucial for the broader adoption and acceptance of AI in critical applications where interpretability is essential. Through the rigorous evaluation and compelling results presented in this chapter, FEHAN and DICTA are positioned as significant advancements in the field of explainable AI, offering valuable insights and paving the way for future innovations in this domain.

Chapter 4

Data-centric Graph Explanation

Graph convolutional networks (GCNs) have gained significant traction in various graph mining tasks, including node classification [63, 123, 138], graph classification [8, 87], and link prediction [70, 77]. Despite their outstanding performance, these complex GCN models' non-linear and non-reversible internal functions make them challenging to interpret. GNN explanations enhance model interpretability, transparency, and trust, aiding in debugging, regulatory compliance, and improving user understanding by clarifying complex relationships and decision-making processes within graph data.

Several specifically designed explanation methods for GCNs have been introduced in response to these concerns. Based on the type of provided explanation, we categorize them as follows:

Prototype Explanation (PT): Prototype Explanation (PT) identifies the influential elements of the input that significantly affects the model's prediction. This type of explanation clarifies which aspects of the input lead to the same prediction [52]. Gradient-based methods, such as Grad-CAM and Excitation Backpropagation, are applied to GCNs to generate heatmaps highlighting important regions of the input [131]. Methods like GNNExplainer [175] and PGExplainer [105] perturb the graph structure to mask unrelated nodes and identify the most influential subgraph. SubgraphX [178] uses the Shapley value as a scoring function on selected subgraphs through Monte Carlo Tree Search (MCTS) [150]. GraphSVX [35] constructs a surrogate model by creating a set of perturbed graph samples and computing their contribution to the explained prediction. GStarX [181] proposes a structure-aware scoring function on the nodes based on cooperative game theory. These algorithms primarily identify the most influential subgraph for GCN model predictions.

Counterfactual Explanation (CF): Counterfactual (CF) explanations finds the minimum changes to the input that would alter the model's prediction, thereby revealing the cause-and-effect relationship between the input and the model's prediction [19]. CFGNNExplainer [103] generates CF explanations by identifying edges whose absence changes the model's prediction. CLEAR [107] uses a variational autoencoder to generate counterfactual explanations for graph-level problems. Although CF explanations can provide valuable insights into the cause-and-effect relationships, research on this topic remains limited.

Exemplar Explanation (EXE): identifies modifications that can be made to the input graph without altering the model's prediction [57]. This approach is particularly useful for sparse input graphs. For instance, when dealing with a large but sparse graph, a PT explanation might eliminate too many connections, resulting in an invalid explanation. In such cases, an EXE explanation can help by focusing on all influential connections without losing critical information [44].

In this research, we present the Multi-Perspective GCN Explainer (MPGE), a framework designed to address the aforementioned research gaps in explaining GCN models. MPGE unifies the two existing explanation strategies, Prototype (PT) and Counterfactual (CF) explanations, and introduces a novel EXemplar Explanation (EXE).

Our method employs a perturbation-based technique to learn the expected explanation using targeted loss functions. To provide multi-perspective explanations and obtain the preferred output while considering the degree of changes to the input, we introduce multiple objective functions and train the explanation model accordingly.

We summarize our contributions as follows:

- Introducing a perturbation-based explainer capable of producing multi-perspective explanations, focusing on molecular and drug discovery datasets.
- Generating counterfactual, prototype, and exemplar explanations by perturbing the graph structure.
- Optimizing the perturbation mask to achieve the preferred output for the explanation.
- Controlling the magnitude of perturbation using an elastic net regularizer on the parameters of the perturbation mask.

To demonstrate the effectiveness of our method, we applied it to molecular property prediction tasks. Predicting molecular properties is crucial for drug discovery, and recent GCN developments have been widely adopted for various molecular property prediction and drug discovery concepts. Obtaining expressive explanations that assist in predicting the physical properties of molecular compounds is advantageous for addressing complex problems in these domains. We evaluated our algorithm using criteria such as AUC/ROC, level of sparsity, and graph theory measures on well-known molecular property prediction datasets. The results demonstrated the superiority of our explanation method.

4.1 Graph Neural Networks

Graphs are composed of entities (nodes) and their relations (edges). Let $G = (V, E)$ represent a graph, where $V = \{v_1, v_2, \ldots, v_n\}$ denotes the set of nodes and E denotes the set of edges. The structure of the graph is represented by an adjacency matrix, which is an $n \times n$ square matrix where $a_{i,j} = 1$ if there is an edge connecting node i and j, and $a_{i,j} = 0$ otherwise. The node features are given as an $n \times d$ matrix X, where each row corresponds to a d-dimensional real-valued feature vector of a node.

Graph Neural Networks (GNNs) leverage the graph structure (adjacency matrix) and node features to predict outputs by exploiting a message-passing and message-aggregation mechanism. Message-passing distributes information across the graph, and message aggregation consolidates this information to learn node embeddings. These embeddings are then used to predict downstream tasks such as graph or node classification. In this work, we focus on the node classification problem.

A GNN classifier is modelled by a function $f(A, X; W) \to Y$, where Y is the set of possible predicted classes, A is the adjacency matrix, X represents the node features and W denotes the learned weights of f. GNNs typically follow an information propagation and aggregation scheme to make predictions based on the input. The core idea is that a node's feature is obtained by aggregating information from its neighbors and combining it with its own information. At the l-th propagation level of a GNN, each edge (i, j) passes a message $m_{ij}^l = \text{Message}(h_i^{l-1}, h_j^{l-1})$ between nodes v_i and v_j by using their representations h_i^{l-1} and h_j^{l-1} from the $(l-1)$-th layer. At the 0-th layer, the node representations are initialized as the node features, $h^0 = X$. After message propagation, each node uses an aggregation function

$$m_i^l = \text{Aggregation}\left(m_{ij}^l \mid j \in N_i\right)$$

to aggregate incoming messages from its neighbors N_i. Finally, the aggregated messages are combined with the node's own representation via an update function

$$h_i^l = \text{Update}(m_i^l, h_i^{l-1}).$$

This general mechanism of passing and aggregating information in GNNs can be formulated for a single layer of a Graph Convolutional Network (GCN) as:

$$X_l = \sigma(D^{-1/2}\hat{A}D^{-1/2}X_{l-1}W_{l-1})$$

where X_l denotes the node representations at the l-th layer, calculated by averaging the representations of neighboring nodes and the node itself. The adjacency matrix A indicates which nodes should be considered in the calculation of node representations. The identity matrix $\hat{A} = A + I$ includes the node itself in the new representation calculation. The diagonal degree matrix $\hat{D}$ normalizes $\hat{A}$ to average the node representations, and a non-linear function σ, such as ReLU, is applied to the averaged node representations. This process can be repeated for multiple layers, depending on the depth of the GNN.

The number of layers determines the range of neighbours be included in the network. For instance, after the first iteration, each node aggregates information from its immediate neighbors (1-hop away). For a specific node, calculating its representation involves constructing an individual computation subgraph that includes the relevant nodes and edges needed to compute $f(v)$. This computation subgraph can be represented as a tuple of the subgraph adjacency matrix A_v and feature matrix X_v, denoted as $G_v = (A_v, X_v)$. The node v is then represented as $v = (A_v, x)$, where x is the feature vector for v. The function f maps the node's representation to a probability distribution over a set of classes. For node classification, the computation subgraph identifies the label for the selected node.

4.2 Problem Formulation

In this work, we aim to explain the predictions of a GNN classifier by providing explanations composed of different contrasting components: counterfactuals, prototypes, and exemplars. Our approach is based on perturbing the original graph and defining a loss function that allows the extraction of all desired explanation components.

Let $f(A_v, x)$ be a GNN model that takes a node's features x and the computation subgraph adjacency matrix A_v as input and predicts the class label for the given node. The CF, PT, and EXE explanations can be found by applying a perturbation matrix P to the computation subgraph adjacency matrix A_v of node v. The perturbed subgraph of v is denoted by $\bar{v} = (\bar{A}_v, x)$, where the perturbed adjacency matrix is obtained by the element-wise multiplication with the perturbation matrix, $\bar{A}_v = P \odot A_v$.

The prediction of the model for the CF explanation should be $f(\bar{A}_v, x) \neq f(A_v, x)$, while for the PT and EXE explanations it should be $f(\bar{A}_v, x) = f(A_v, x)$. Although PT and EXE have the same prediction, the main difference lies in the number of connections present in their explanation subgraph: the PT explanation is designed to contain the smallest set of connections crucial for the prediction, whereas the EXE explanation retains most of the original connections, making it very similar to the graph being explained.

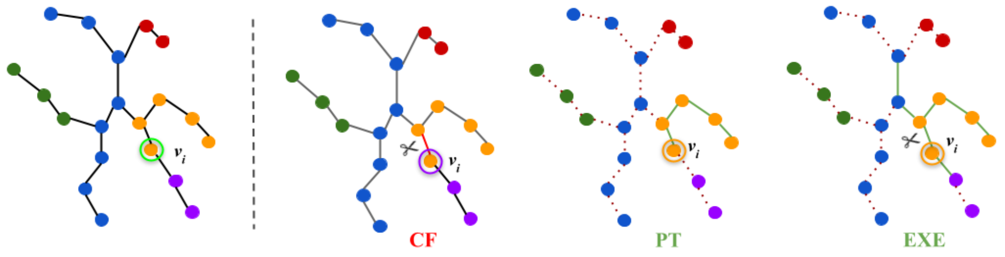


Figure 4.1: A toy example illustrating the CF, PT and EXE explanations for a node v_i. Each node in this graph is shown with the associated colour to its corresponding label. Node v_i is labelled with yellow colour. The counterfactual (CF), prototype (PT), and exemplar (EXE) explanations of v_i's are illustrated in the mentioned order from left to the right. For the CF explanation, the eliminated edge is presented with the red line and it causes the change of node v_i's label (purple colour). The prototype explanation preserves the v_i's label and it shows the most influential subgraph for the prediction of v_i, hence in the extracted PT subgraph only connections among the same label nodes are preserved and the edges that are presented with dashed red lines are eliminated from the graph. The EXE subgraph includes the responsible connections for the prediction of v_i's. The main difference between PT and EXE explanations is the number of present connections in the explanation subgraph.

For the CF explanation, we seek a subgraph of the input where the necessary edges whose absence changes the prediction of the GNN are eliminated. The optimal solution involves making the least number of changes to the graph's structure. In contrast, for the PT explanation, we aim to identify the most influential subgraph of the input. This means that the explanation subgraph should be faithful to the original prediction and characterized by a heavily sparse adjacency matrix, without necessarily being close to the original graph.

For the EXE explanation, we seek the same prediction as the input (similar to PT), but obtained from a subgraph very similar to the original one in terms of connections represented in the adjacency matrix. The EXE explanation should include most of the original connections, requiring less sparsification of the adjacency matrix by minimizing the deletion of edges. This involves controlling the magnitude of the perturbation on the adjacency matrix.

Figure 4.1 provides a toy example illustrating these three explanations. In this example, for the computation subgraph of node v_i, the CF, PT, and EXE explanations are generated. Each node is illustrated with a colour corresponding to its label, with node v_i labelled in yellow.

For the CF explanation, since the label of v_i should change, this explanation is generated by removing v_i's direct connection to another yellow node. The removed edge is shown with a solid red line. Eliminating only this edge results in a different prediction (purple) for v_i.

For the PT explanation, we need to preserve the label of v_i. The PT explanation retains the nodes and connections essential for obtaining the same label. In this explanation, all connections to nodes of different colors are removed, with the removed links indicated by dashed red lines. The PT explanation preserves the most important subgraph for predicting the label of v_i and eliminates the rest of the connections.

For the EXE explanation, we seek a subgraph that is very similar to the original one. Fewer connections are eliminated to obtain this explanation subgraph. The label of v_i is preserved, and the responsible connections for this prediction remain in the subgraph, including some edges that are not the most influential. The eliminated connections are shown with dashed red lines.

In order to address the problem of explaining GNN predictive models for node classification providing the type of explanation described above by a perturbation-based approach, we will present the core elements of our approach: *i)* how to perturb the input graph for our multi-component explanation model; *ii)* how to define the loss functions suitable for deriving the different components of our explanation.

4.3 Multi-component Explanation Method for Graph Classification

As described in the preliminaries section, we propose a perturbation-based method that provides various explanation perspectives. Each perspective offers a different insight into the reasons behind the classification obtained by the GCN model. We now provide a mathematical description of the graph convolution explanation.

Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of links between nodes. The number of nodes and connections are denoted by N and M, respectively. The adjacency matrix corresponding to the graph is described as $\mathbf{A} \in \{0, 1\}^{N \times N}$, with $a_{i,j} = 1$ if there is a connection between node i and node j, and $a_{i,j} = 0$ otherwise (for simplicity, we consider the undirected graph, i.e., $a_{i,j} = a_{j,i}$). The nodes' features matrix $X \in \mathbb{R}^{N \times d}$ is associated with each node $v_i \in \mathcal{V}$, where $x_i = X[i, :]$ is its d-dimensional feature vector.

Let $f_W : (\mathbf{A}, \mathbf{X}) \to \mathbf{Y}$ be a graph prediction model that takes $\mathbf{X}$ and $\mathbf{A}$ as input and predicts

$\mathbf{Y} \in \mathcal{Y}_{1\times C}$, where C is the number of classes. Graph Convolution Networks (GCNs), as noted in [81], employ a message-passing algorithm similar to convolution in CNNs, aggregating local information. A standard layer of f_W (such as the first layer L_1) of GCN, is formulated as follows:

$$L_1 = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}XW) \tag{4.1}$$

where σ denotes the softmax function, $\hat{A} = A + I$ is the normalized adjacency matrix, I is the identity matrix, $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$, $\hat{D}_{ij} = \sum_j \hat{A}_{ij}$ are the entries of this matrix, X is the nodes' features matrix, and W is the weight matrix. This formulation follows the general GCN computation description presented in [81].

In order to explain the GCN's predictions using a perturbation-based algorithm, we need to perturb the input graph and monitor the corresponding changes in the output. Our explanation model employs the idea of perturbing the input graph by masking connections and learning the appropriate perturbations during a training process [163].

We introduce an auxiliary non-parameterized module, denoted as g_P, to support the explanation. This module takes the form $g_P(f_W, A, X; P)$, where f_W is the GCN model, and A and X are the graph's adjacency matrix and feature matrix, respectively. The module returns the requested explanation as $\tilde{G} = (\tilde{A}, X)$.

The explanation module g_P creates a copy of f_W as the explanation model $\tilde{f}_P$, parameterized by a trainable perturbation matrix P. During the explanation procedure, P is iteratively trained by tracking the changes in the graph and their impact on the predictions of $\tilde{f}_P$. At the end of the training, g_P analyzes the generated explanations based on their predicted outputs and a set of evaluative criteria (presented in Section 4.4). Finally, it delivers the selected explanation in the form of $\tilde{G} = (\tilde{A}, X)$.

Our model explains the GCN's predictions from different perspectives: PT, CF, and EXE. The CF, PT, and EXE explanations are obtained by applying a trainable perturbation matrix P to the graph's adjacency matrix, $\tilde{A} = P \odot A$ [163], and monitoring the variations in the output. The predictions of the model for each type of explanation should be:

- **Prototype (PT) Explanation**: For the PT explanation, we seek a subgraph of the input that includes the most influential nodes and edges for the model's prediction. The explanation should contain only the essential and most influential elements. The expected prediction for the PT explanation is $f(X, P \odot A) = f(X, A)$, with $\min |P| > 0$ and $\min |A - P \odot A| > 0$. This influential subgraph significantly impacts the model's prediction, and its removal from the input would cause a substantial change in the predicted probability.

- **Counterfactual (CF) Explanation**: For the CF explanation, we look for a subgraph where the absence of certain edges changes the GNN's prediction. The optimal solution involves making minimal changes to the graph's structure. The expected output is $f(X, P \odot A) \neq f(X, A)$, with $\min |P| > 0$ and $\min |A - P \odot A| < |A|$. Eliminating the CF edges from the graph results in a significant difference in the predicted probability compared to the original prediction.

- **Exemplar (EXE) Explanation**: For the EXE explanation, we aim for the same prediction as the input (similar to PT) but with minimal structural changes. Thus, $f(X, P \odot A) = f(X, A)$, with $\min |P| > 0$, $\min |A - P \odot A| < |A|$, and $\min |A - P \odot A| > 0$. According to the EXE definition, removing the selected edges should have a minimal impact on the predicted probability since all crucial connections for the prediction remain in the graph.

Considering the definition of each explanation and the expected output, we need to control the degree of perturbation and the direction of prediction. This involves guiding the explanation model throughout the training process to obtain the desired prediction. Depending on the type of explanation, the loss function adjusts the model to achieve either a similar or contrasting class label.

4.3.1 Explanation Module

In this section, we elaborate on the details of our algorithm for generating multi-component explanations, including CF, PT, and EXE. To explain the prediction of f_W for a given instance, we utilize a non-parameterized explanation module defined as $g_P(f_W, A, X)$. Within the explanation module, g_P creates a copy of f_W, denoted as the explanation model $\tilde{f}_P$, which is parameterized by P. The explanation model $\tilde{f}_P$ retains the same network structure as the GCN model f_W but keeps the weight matrix W constant and learns the perturbation matrix P throughout the training process.

The objective of $\tilde{f}_P$ is to perturb the original adjacency matrix to produce the desired explanation. Therefore, the matrix P must be multiplied by A and not by $\hat{A}$. This approach prevents the removal of self-loops in the message-passing of $\tilde{f}_P$ (i.e., zeroing out the identity matrix $\hat{A} = A + I$) during the explanation step. To achieve this, we revise the one-layer GCN model in Eq. 4.2 to isolate A:

$$L_1 = \sigma[(D + I)^{-1/2}(A + I)(D + I)^{-1/2}XW] \tag{4.2}$$

Next, we define $\tilde{f}_P$, which consists of the same number of layers as f_W. Computing one layer of $\tilde{f}_P$ is presented in Eq. 4.3.

$$\tilde{L}_1 = \sigma[(\tilde{D})^{-1/2}(P \odot A + I)(\tilde{D})^{-1/2}XW] \tag{4.3}$$

Due to the multiplication of P and A, we need to update the degree matrix D based on $P \odot A$ and add the identity matrix to account for the self-loops. The updated degree matrix is denoted by $\tilde{D}$.

4.3.2 Adjacency Matrix Perturbation

Given the definition of the explanation model $\tilde{f}_P$, we must define the perturbation matrix P to sparsify A. Specifically, we define $\tilde{A} = P \odot A$, where the perturbation matrix P is point-wise multiplied by A to remove certain connections in the graph, resulting in the perturbed adjacency matrix $\tilde{A}$.

Our goal is to modify the graph's structure by learning P to derive explanations where $f_W(A, X) \neq \tilde{f}_P(P \odot A, X)$ (CF) or $f_W(A, X) = \tilde{f}_P(P \odot A, X)$ (PT/EXE). To find the appropriate perturbation matrix, we customize loss functions accordingly.

Following the approach of [153], we instantiate the perturbation matrix as $P_{N\times N}$, where each entry P_{ij} is within the range $[0, 1]$. After training, we binarize $\tilde{P}$ to obtain $\hat{P}$ by applying an element-wise sigmoid function on P and thresholding the entries: values below 0.5 are set to 0, and values equal to or above 0.5 are set to 1.

Once the perturbation matrix is learned, we compute the final explanation by performing point-wise multiplication to obtain $\tilde{A} = \hat{P} \odot A$. Zeros in the perturbation matrix ($\hat{P}_{ij} = 0$) indicate the elimination of the connection between nodes i and j, while ones in $\hat{P}$ indicate the edges that remain intact.

To summarize the process of generating an explanation for a given instance: 1. Compute the binary $\hat{P}$ by applying a pointwise sigmoid function on $\tilde{P}$. 2. Perform point-wise multiplication of the binary perturbation matrix with the adjacency matrix ($P \odot A$) to eliminate edges. 3. Pass $\tilde{A}$ and X through $\tilde{f}_P$ to obtain the final prediction: $\tilde{y} \leftarrow \tilde{f}_P(\hat{P} \odot A, X)$.

4.3.3 Loss Function Optimization

To train the parameters of $\tilde{f}_P$, we define a loss function tailored to obtain the desired prediction. There are two main objectives: one for yielding subgraphs with the same prediction as the input and one for producing subgraphs with contrasting or dissimilar predictions. The overall form of the loss function for all explanation types is as follows:

$$\mathcal{L} = \mathcal{L}_{Pred}(\tilde{f}_P(\tilde{A}, X)) + \beta_1\|P\|_1 + \beta_2\|P\|_2^2 \tag{4.4}$$

The term $\mathcal{L}_{Pred}(\bar{v})$ in Eq. 4.4 is designed to encourage the perturbed input to match the desired prediction. The loss function for deriving CF explanations differs mainly in this term from those for extracting PT and EXE explanations.

For the **CF explanation**, we aim to find a subgraph of the input where the essential edges, whose absence changes the model's prediction, are eliminated. Our goal is to apply minimal changes to the graph's connections to produce a contrasting prediction, i.e., $f_W(A, X) \neq \tilde{f}_P(P \odot A, X)$. Therefore, we train the perturbation matrix to maximize the difference between the prediction of the most probable class label and the original one, i.e., $\text{argmax}_i\ f(A, X)_i \neq \text{argmax}_i\ [\tilde{f}_P(\tilde{A}, X)]_i$ [163]. The CF loss function can be written as:

$$\mathcal{L}_{Pred}^{CF}(\tilde{f}_P(\tilde{A}, X)) = \max\{\tilde{f}_P(\tilde{A}, X)_{f_W(A,X)} - \max_{i \neq f_W(A,X)}[\tilde{f}_P(\tilde{A}, X)]_i, -\kappa\} \tag{4.5}$$

In this formula, $\tilde{f}_P(\tilde{A}, X)_{f_W(A,X)}$ is the probability value of $\tilde{f}_P$ for the perturbed example on the original label. $\max_{i \neq f_W(A,X)}[\tilde{f}_P(\tilde{A}, X)]_i$ is the maximum probability value of other class labels except for the original label. This loss function is minimized when, for any confidence level κ, the predicted probability of the perturbed example for the other classes exceeds the predicted probability for the original label.

For **PT** and **EXE explanations**, we aim to find subgraphs that maintain the same label as the input graph. Thus, the output of $\tilde{f}_P$ for the perturbed subgraph should satisfy $f_W(A, X) = \tilde{f}_P(P \odot A, X)$. Specifically, we seek to obtain a perturbation matrix where the explanation subgraph has the same top-1 prediction as the original instance, i.e., $\text{argmax}_i\ f_W(A, X)_i = \text{argmax}_i[\tilde{f}_P(\tilde{A}, X)]_i$. The perturbed example can be interpreted as a representation of the model's prediction. The PT and EXE (PT-EXE) loss function is defined as:

$$\mathcal{L}_{Pred}^{PT-EXE}(\tilde{A}, X) = \max\{\max_{i \neq f_W(A,X)}[\tilde{f}_P(\tilde{A}, X)]_i - \tilde{f}_P(\tilde{A}, X)_{f_W(A,X)}, -\kappa\} \tag{4.6}$$

This loss function is minimized when, for any given confidence level $\kappa \geq 0$, the predicted probability of the perturbed example for the original label is greater than the predicted probability for any other class by at least κ.

It is important to note that this term alone does not control the magnitude of the perturbation, which may lead to finding subgraphs with only minimal changes to the original A_v. This issue is addressed by the inclusion of the next terms.

L_1 and L_2 Norms Based Terms: The second and third terms in Eq. 4.4 are jointly known as elastic net regularizers [184] (L_1 and L_2 norms). These terms control the magnitude of the

Algorithm 2: Multi-perspective explanation algorithm

```
Input   : Graph $G$=($A$,$X$), trained GNN Model $f$, parameters: $\beta_1$, $\beta_2$, $\kappa$, $K$
Output  : $e$ - explanation
1  $y \leftarrow f_W(A, X)$                                    // GCN Prediction
2  $P \leftarrow J_n$                                          // Initialization
3  $\tilde{f}_P \leftarrow (f_W, P)$                           // initializing explanation model
4  $\tilde{A}^* = []$
5  for $K$ iterations do
6      $\mathcal{L}, \tilde{A}$ = GET_EXPL_EXAMPLE$(A, X)$
7      if $\tilde{A} \neq A$ then
8          $\tilde{A}^* \leftarrow \tilde{A}^* \cup \tilde{A}$
9      if $\mathcal{L}$ began to increase then
10         break
11     $P \leftarrow P + \alpha \nabla_P \mathcal{L}$
12 Function GET_EXPL_EXAMPLE($A, X, y$):
13     $\tilde{A}_{candidate} \leftarrow A$
14     $\hat{P} \leftarrow$ threshold($\sigma(P)$)
15     $\tilde{A} \leftarrow P \odot A$
16     $\tilde{y} \leftarrow f_P(P \odot A, X)$
17     $\mathcal{L} \leftarrow \mathcal{L}(\tilde{y}, y, P)$;                           // Eq 4.4
18     if EXPL= CF $\wedge\ y \neq \hat{y}$ then
19         $\tilde{A}_{candidate} \leftarrow \hat{P} \times A$
20     if EXPL= (PT $\vee$ EXE) $\wedge\ y = \hat{y}$ then
21         $\tilde{A}_{candidate} \leftarrow \hat{P} \times A$
22     return $\mathcal{L}, \tilde{A}_{candidate}$
```

perturbation matrix and serve as efficient feature selection techniques in high-dimensional learning problems. These regularizers penalize the perturbation matrix to remove excessive components and simplify the problem. They are particularly beneficial for PT explanations, where we seek the most influential subgraph of the input to explain the output. By using these regularizers, we can control the size of the explanation, meaning the number of connections present in the explanation subgraph is directly influenced by the values of these regularizers.

In Eq. 4.4, β_1 and β_2 are parameters that control the sparsity of the generated explanations. We conducted a series of experiments to determine the optimal values for these parameters for each type of explanation. The overall procedure of our algorithm is summarized in Algorithm 2.

4.3.4 Explanation procedure

Given a graph (A, X), we initialize the perturbation matrix P as a matrix of ones J_n to initially preserve all edges. We compute $\hat{P}$ by applying the sigmoid function and thresholding the output. Point-wise multiplication of the binarized perturbation matrix with the adjacency matrix provides a candidate explanation. Based on the predicted class label and type of explanation, we decide whether to store or discard the generated explanation.

We train the explanation model g by backpropagating the loss value to the parameters of P. The specific loss function used depends on the type of explanation, as introduced in the previous section. For CF explanations, we train P using the loss function in Eq. 4.5. For both PT and EXE explanations, we use the loss function in Eq. 4.6. The main distinction between PT and EXE lies in the regularization values, which we will discuss later. For each type of explanation, the algorithm produces a set of potential explanations. From this set, only the explanation that aligns with the specified criteria is selected. For instance, in the case of the PT (Path-based) explanation, which requires including the minimal number of present links, the algorithm will choose the explanation that contains the fewest number of connections among those generated.

4.4 Evaluation and Analysis

In this section, we outline our experimental setup to analyze our algorithm. First, we discuss the real-world molecular datasets, baseline methods, and experimental configurations. Next, we present the quantitative evaluation of our method against baseline algorithms for graph and node classification problems. Finally, we demonstrate the explanations obtained for examples from each molecular dataset.

Datasets: To evaluate our algorithm, we considered five molecular classification datasets:

1. **MUTAG** [30] - a dataset of 188 chemical compounds classified based on their mutagenic effect on a bacterium.

2. **Mutagenicity** [76] - consists of 4,337 chemical structures classified as mutagens or nonmutagenic based on their ability to cause mutations in DNA.

3. **AIDS** [137] - includes molecular compounds classified into two states of activity (*active/inactive*) against the HIV virus.

	AIDS	MUTAG	Mutagenicity	BBBP
Graphs	2000	188	4337	2039
AVG nodes	15.69	17.93	30.32	24.1
AVG edges	16.20	19.79	30.77	26.0
Classes	2	2	2	2
Ground Truth Expl	True	True	True	False

Table 4.1: The details of each dataset, including the average number of nodes, the average number of edges, the number of classes, and the presence of ground truth explanations.

4. **BBBP** [111] - the Blood-Brain Barrier Penetration dataset, includes molecular structures classified by their ability to penetrate the blood-brain barrier.

For MUTAG, Mutagenicity, and AIDS, we have ground truth explanations which we used to calculate the explanation accuracy for the PT explanations. The details of these datasets are provided in Table 4.1.

Baselines

We selected two types of baseline explanation algorithms to compare against our multi-component explanation method. For the PT explanation, we chose GNNExplainer [175] and PGExplainer [105], and for the CF explanations, we compared our solution against CF-GNNExplainer.

The GNNExplainer is designed to extract the most relevant subgraph of the input along with the most influential subset of node features for the model's prediction. Although the GNNExplainer paper suggests that removing an influential node or edge from the explanation subgraph can produce a counterfactual explanation, [10] argues that the removal of highly correlated edges or nodes does not necessarily result in a counterfactual explanation due to the high non-convexity of the GNN model. Therefore, we decided to compare the outcomes of GNNExplainer and PGExplainer with our PT explanations. Since we do not extract the most influential subset of node features, we only consider GNNExplainer's masked adjacency matrix as the explanation for evaluation.

For evaluating our CF explanations, we used CF-GNNExplainer as the baseline due to its similarity to our proposed algorithm for generating CF explanations. It is important to note that we followed the training processes suggested in their original papers for all baseline models.

Metrics

To evaluate the quality of the generated explanations, we ran the explanation procedure for a selected graph ten times, each time with a different random seed. We assessed the generated explanations using the following five metrics:

AUC/ROC: Since we have ground truth PT explanations for all datasets except BBBP, we use these explanations to compare the AUC/ROC [15] of our method against the baselines. AUC/ROC metrics can be used to evaluate how closely the GCN's explanations align with this gold standard.

Predicted Probability Shift: Inspired by the deletion metric presented by Petsiuk and et al. [127], we constructed a metric to measure the shift in predicted probability after the removal of the explanation subgraph from the input graph. For PT explanations, the removal of this subgraph should result in a significant change in the predicted probability, as it represents the most influential subgraph. Similarly, for CF explanations, removing the minimal number of edges that cause a change in the predicted probability should lead to a substantial probability shift. Conversely, for EXE explanations, the removal of edges should have minimal impact on the predicted probability. To evaluate this, we first measure the predicted probability of the input graph $y = f_W(A, X)$. After obtaining the explanation, we remove the connections from the input data $\bar{A} = A - \tilde{A}$ and measure its probability $\tilde{y} = f_W(\bar{A}, X)$. We then report the shift in prediction $|\tilde{y} - y|$.

Sparsity: This metric measures the proportion of removed edges from the explanation $\hat{P} \times A$ [177]. A value of 1 indicates that all edges were eliminated. For PT explanations, higher sparsity (close to 1) is preferred, while for CF and EXE explanations, lower sparsity (close to zero) is desired.

Number of Present or Removed Edges: Depending on the type of explanation, we calculate the number of present or removed edges from $\tilde{A}$ relative to the original graph. For CF explanations, we report the number of "Removed Edges" as we aim to make minimal changes to the graph structure. For PT explanations, we measure the number of "Present Edges" as we seek the minimal influential subgraph. For EXE explanations, we aim to find a subgraph similar to the input graph, avoiding the removal of many connections.

To ensure the validity of the generated CF explanations, we verify that the provided solutions do not drastically alter the graph's structure and remain as close as possible to the original graph. Therefore, we consider basic graph measures to evaluate the quality and validity of our solutions.

Betweenness Centrality: This metric measures how often a given node lies on the shortest path between other nodes, indicating which nodes act as bridges in the graph. It is calculated by summing the fraction of all pairs of shortest paths that pass through node v.

Closeness Centrality: This metric evaluates a given node based on its closeness to all other nodes in the network. It scores each node by calculating the shortest path between all nodes and summing the distances of v's shortest paths.

We measure these two criteria for every node in the input graph and the obtained explanation, storing them in two vectors. We then calculate the Euclidean distance between these vectors and report the difference value.

4.4.1 Quantitative Evaluation

Hyperparameter Optimization

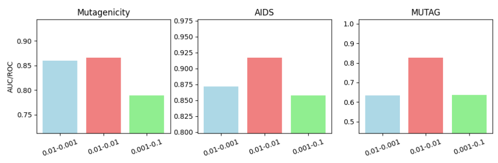
This section outlines the experimental setup for training and generating explanations. Before proceeding with the experimental evaluation, we conducted a series of analyses to determine the optimal elastic net coefficients for each type of explanation. As mentioned earlier, for CF and EXE explanations, we aim to eliminate a small number of connections from the input, while for PT explanations, we seek to include the most significant connections.

We selected each coefficient from the following ranges: $\beta_{L1} = \{0.1, 0.001, 0.0001\}$ and $\beta_{L2} = \{0.1, 0.01, 0.001\}$. We executed the explanation algorithm with the selected coefficients and observed their impact on the AUC/ROC of PT explanations. The β_{L1} and β_{L2} values that resulted in the highest AUC/ROC and the greatest probability shift when connections were removed were selected.

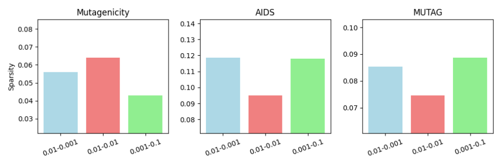
Next, we tested the CF and EXE explanations with different values of β_{L1} and β_{L2} and checked the consistency of the obtained explanations with PT. Based on the type of explanation and the availability of ground truth explanations, we reported the "AUC/ROC," "Sparsity," "Shift in Predicted Probability," and "Number of Removed Connections" for each explanation type in Figure 4.2. To avoid repetition, we presented the evaluated criteria for those values of L_1 and L_2 that were not selected for the explanations. Based on this quantitative analysis, we selected the following values for β_{L1} and β_{L2}:

- PT: $\beta_{L1} = 0.1$ and $\beta_{L2} = 0.001$
- EXE: $\beta_{L1} = 0.0001$ and $\beta_{L2} = 0.01$
- CF: $\beta_{L1} = 0.001$ and $\beta_{L2} = 0.01$

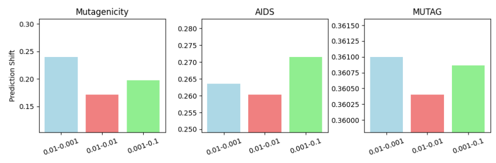
- PT: $\beta_{L1} = 0.1$ and $\beta_{L2} = 0.001$
- EXE: $\beta_{L1} = 0.0001$ and $\beta_{L2} = 0.01$



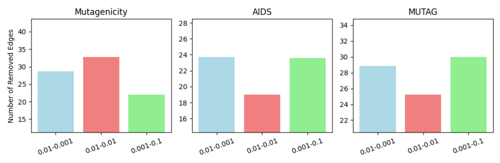
(a) AUC/ROC



(b) Sparsity

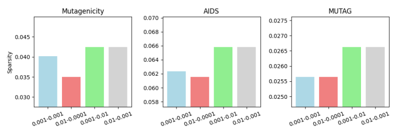


(c) Prediction Shift

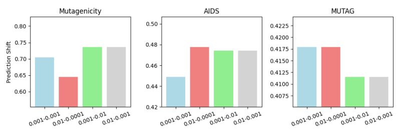


(d) Removed Edges

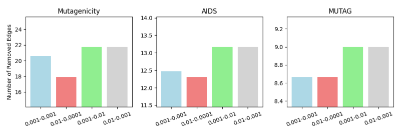
Evaluation of PT



(e) Sparsity

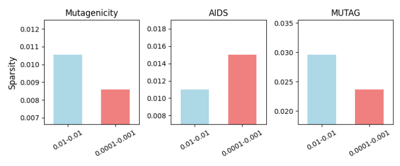


(f) Prediction Shift

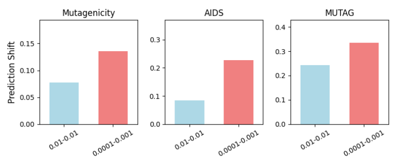


(g) Removed Edges

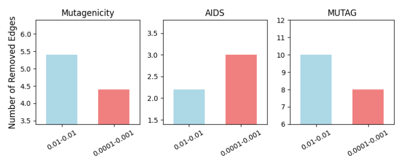
Evaluation of CF



(h) Sparsity



(i) Prediction Shift



(j) Removed Edges

Evaluation of EXE

Figure 4.2: Evaluation of the impact of different elastic net regularization coefficients (β_{L1} and β_{L2}) on the performance of PT, CF, and EXE explanations. Each subfigure presents the results for a specific metric: AUC/ROC, Sparsity, Predicted Probability Shift, and Number of Removed Edges. The selected coefficients are those that maximize AUC/ROC and probability shift for PT explanations while ensuring consistency in the CF and EXE explanations.

- CF: $\beta_{L1} = 0.001$ and $\beta_{L2} = 0.01$

We have considered two sets of experiments to quantitatively and qualitatively evaluate our model's performance.

Baseline Comparison

In the first set of experiments, we measured the AUC/ROC scores of explanations with respect to the ground truth explanation for AIDS, MUTAG, and Mutagenicity datasets and reported them in

Table 4.2. Ground truth explanations allow us to measure the AUC/ROC.

The explanation problem was treated as a binary classification problem where the edges in the ground truth explanation are treated as labels and the perturbation weights P given by the explanation model are viewed as prediction scores. Higher AUC/ROC scores indicate that the explanation algorithm assigned higher scores to edges in the ground truth explanation. This experiment demonstrates that our model outperforms the baselines in identifying the most influential edges for the PT explanation. Our experimental analysis shows that our proposed algorithm is accurate, effective, and consistent in identifying PT explanations.

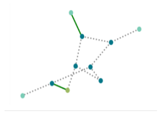
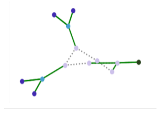
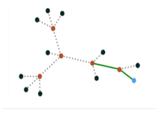
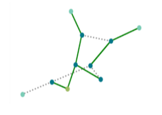
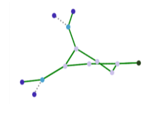
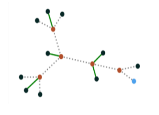
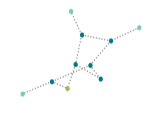
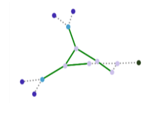
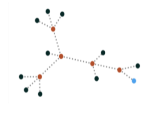
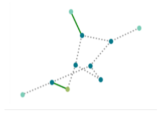
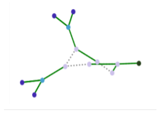
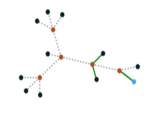
	AIDS	MUTAG	Mutagenicity
Features	Atom type	Atom type	Atom type
	Visualization		
Actual EXPL			
GNNEXplainer			
PGExplainer			
Our Algorithm			
	Explanation AUC		
GNNExplainer	0.37±0.08	0.68±0.0	0.55±0.04
PGExplainer	0.35±0.1	0.14±0.0	0.75±0.02
Our Algorithm	**0.95±0.03**	**0.92±0.0**	**0.89±0.03**

Table 4.2: Illustration of molecular datasets with their AUC/ROC evaluations with respect to our model and alternative baselines.

Analytical Evaluation

Building upon the prior experiment, we conducted a series of investigations to provide a more detailed analysis of the generated explanations, including CF and EXE explanations. We evaluated the explanations generated for each data sample presented in the previous experiment using the criteria outlined in Section 4.4.

		AIDS	MUTAG	Mutagenicity	BBBP
	S	0.18	0.06	0.10	0.11
	PE	4	20	6	4
PT	**AE**	18	6	24	26
	BC	0.66	0.74	1.08	1.08
	CC	1.27	0.38	1.31	0.90
	S	0.07	0.02	0.04	0.02
	PE	12	22	18	36
CF	**AE**	10	4	12	14
	BC	0.53	0	0.76	0.39
	CC	0.26	0	0.51	0.05
	S	0.009	0.02	0.00	0.001
	PE	20	22	26	48
EXE	**AE**	2	4	4	2
	BC	0.0	0.0	0.0	0.0
	CC	0.0	0.0	0.0	0.0
	S	0.01	0.01	0.04	0.001
	PE	18	20	28	48
CFGNN	**AE**	4	4	2	2
	BC	0.0	0	0.76	0.0
	CC	0.0	0	0.51	0.0

Table 4.3: Detailed evaluation of PT, CF, and EXE explanations based on various metrics: Sparsity (S), Number of Preserved Edges (PE), Number of Absent Edges (AE), Betweenness Centrality (BC), and Closeness Centrality (CC) for datasets AIDS, MUTAG, Mutagenicity, and BBBP. For PT explanations, we expected higher sparsity, fewer present connections, and greater betweenness and closeness distances. For CF and EXE explanations, we anticipated lower sparsity, more present edges, and smaller betweenness and closeness distances. The results confirm that PT explanations included the most influential elements, causing significant prediction changes when removed, whereas CF explanations involved minimal edge removal to change predictions, and EXE explanations maintained most influential elements, resulting in minor prediction shifts.

For PT explanations, compared to CF and EXE explanations, we expected to observe higher sparsity, fewer present connections, and greater betweenness and closeness distances. Conversely, for CF and EXE explanations, we anticipated low sparsity, more present edges, and smaller betweenness and closeness distances. The results presented in Table 4.3 align with our expectations. Additionally, we reported the quantitative evaluation of generated explanations by CF-GNNExplainer.

Based on the results presented in Table 4.3, we ensured that our expectations for the explanations were met. Specifically, PT explanations included only the most influential elements of the input (high sparsity), and eliminating the explanation subgraph caused a major change in the predicted probability. Similarly, for CF explanations, we expected to remove a minimal number of edges (low

		Our Algorithm	GNNE	PGE	CF-GNN	Random
PT	AIDS	**0.33±0.05**	0.27±0.05	0.22±0.05	-	0.21±0.08
	MUTAG	**0.58±0.0**	0.36±0.0	0.34±0.0	-	0.35±0.0
	Mutagenicity	**0.40±0.03**	0.15±0.04	0.10±0.04	-	0.10±0.02
	BBBP	**0.49±0.04**	0.31±0.04	0.12±0.03	-	0.13±0.06
CF	AIDS	**0.52±0.03**	-	-	0.48±0.02	0.25±0.01
	MUTAG	**0.56±0.0**	-	-	0.54±0.0	0.42±0.0
	Mutagenicity	**0.71±0.04**	-	-	0.65±0.05	0.22±0.04
	BBBP	**0.39±0.04**			0.13±0.08	0.33±0.04
EXE	AIDS	**0.12±0.05**	-	-	-	0.22±0.06
	MUTAG	**0.23±0.0**	-	-	-	0.35±0.0
	Mutagenicity	**0.07±0.04**	-	-	-	0.09±0.05
	BBBP	0.11±0.04	-	-	-	**0.07±0.04**

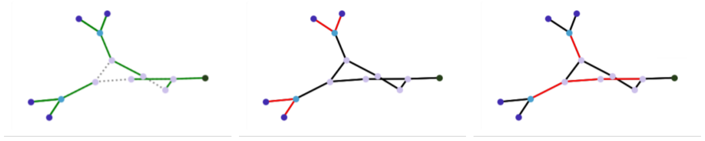
Table 4.4: Difference in predicted probabilities before and after eliminating the explanation subgraph.

sparsity) whose absence changed the model's prediction, thereby drastically altering the graph's prediction. For EXE explanations, we aimed to include all influential elements of the input in the explanation graph (low sparsity), so eliminating the EXE subgraph would not significantly impact the predicted probability.

In this experiment, we removed the explanation subgraph and measured its impact on the predicted probability of the model. First, we measured the predicted probability of GCN $f_w(A, X)$ for the given input. After training the explanation model $\tilde{f}_P$, we obtained the connections to be removed (CF and EXE) or preserved (PT) in the explanation subgraphs and eliminated them from the input graph. Finally, we passed the modified graph to f_W, measured the changes in the final prediction, and calculated the shift from the original prediction. The results are reported in Table 4.4. For CF and PT explanations, we expected to see a high difference value, and for EXE explanations, a low difference was anticipated. We compared the outcome of our algorithm against GNNExplainer, PGExplainer, CF-GNNExplainer, and a random removal of edges as the baseline. One interesting observation is that prediction shifts for most CF explanations were greater than PT, indicating that the removal of the PT subgraph does not necessarily result in a CF explanation.

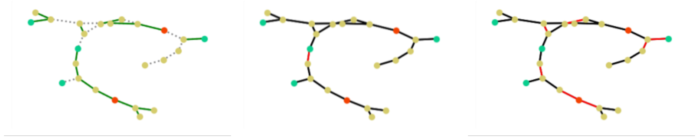
4.4.2 Qualitative Evaluation

After the quantitative evaluation of explanations, we illustrated the generated explanations for each perspective (PT, EXE, and CF) returned from the g module in Figure 4.3. Each of the PT, EXE, and CF explanations for the MUTAG, BBBP, and AIDS datasets is plotted. The node colors indicate node features, which represent atoms, while the edge colors represent the presence or absence of edges in the explanation subgraph. From these figures, we can observe that PT explanations are consistent with EXE and CF explanations. In other words, connections considered to be the most significant for the prediction were not eliminated in these explanations.



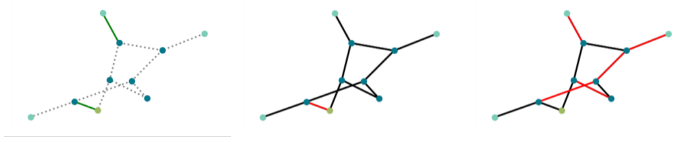
(a) PT (b) EXE (c) CF

Explanations for MUTAG



(d) PT (e) EXE (f) CF

Explanations for BBBP



(g) PT (h) EXE (i) CF

Explanations for AIDS

Figure 4.3: Multi-perspective explanations generated for each dataset. The node colors represent node features, and edge colors demonstrate the presence or absence of an edge. In PT explanations, the removed connections are distinguished with dotted grey lines, and the present connections are shown with green lines. For CF and EXE explanations, the removed edges are shown with red lines.

4.5 Summary

In this chapter, we explored the development and application of a data-centric graph explanation framework named the Multi-Perspective GCN Explainer (MPGE). This framework addresses the challenges of interpreting the predictions made by Graph Convolutional Networks (GCNs), which are widely used for tasks such as node classification, graph classification, and link prediction. Despite their success, GCNs' complex and non-linear nature often makes it difficult to understand their internal decision-making processes.

MPGE locally generates prototype (PT) and counterfactual (CF) explanations and introducing a novel approach called Exemplar Explanation (EXE). Each explanation type provides a different perspective on the model's prediction, making MPGE a comprehensive tool for understanding GCNs.

Prototype explanations focus on identifying the most influential subgraph of the input that is crucial for the model's prediction. These explanations highlight the core elements that drive the decision-making process, helping users understand which parts of the graph are most significant. Counterfactual explanations, on the other hand, identify the minimal set of edges whose removal would change the model's prediction. This approach reveals the cause-and-effect relationships within the graph, providing insights into which connections are essential for maintaining the current prediction.

The newly introduced Exemplar Explanation (EXE) is particularly useful for sparse input graphs or when a detailed view of all influential connections is needed. EXE highlights a subgraph that mirrors the original input graph in terms of label and includes the crucial connections responsible for the classification. This method ensures that the explanation remains close to the original graph structure while focusing on the key elements influencing the prediction.

To implement these explanations, we designed a perturbation-based algorithm that iteratively learns the perturbation matrix during training. This matrix adjusts the adjacency matrix of the input graph, producing the desired explanation by removing or preserving specific connections. The loss functions for PT, CF, and EXE explanations were carefully crafted to ensure the explanations align with the expected outcomes. For example, PT explanations aim to include the most significant connections, while CF explanations seek to identify the minimal changes needed to alter the prediction.

We evaluated MPGE using real-world molecular datasets such as MUTAG, Mutagenicity, AIDS, and BBBP. These datasets provided a robust testing ground due to their complexity and the availability of some ground truth explanations. Our experiments involved both quantitative and qualitative evaluations. Quantitative metrics included AUC/ROC scores, predicted probability shifts, sparsity, and the number of present or removed edges. These metrics helped us assess the effectiveness of our explanations in capturing the most significant elements of the graph and their impact on the model's prediction.

In the qualitative evaluation, we visualized the generated explanations for different datasets, comparing them to the ground truth and other baseline explanation methods like GNNExplainer and PGExplainer. The visualizations demonstrated that our PT explanations were consistent with both EXE and CF explanations, indicating that MPGE successfully identifies the key connections influencing the prediction without eliminating essential elements.

Our results showed that MPGE outperforms baseline methods in terms of explanation accuracy and consistency. The detailed analyses confirmed that PT explanations tend to have higher sparsity, indicating they include only the most critical connections. In contrast, CF explanations exhibited lower sparsity as they aimed to retain as many connections as possible while still changing the prediction. EXE explanations maintained a balance by preserving influential connections and minimizing the removal of edges, ensuring the explanation closely resembles the original graph structure.

Overall, MPGE provides a robust and flexible framework for explaining GCN predictions from multiple perspectives. Integrating PT, CF, and EXE explanations offers a comprehensive understanding of how GCN models derive their predictions, enhancing the transparency and interpretability of graph-based machine learning models. This chapter highlights the importance of developing sophisticated explanation methods to demystify complex AI models, thereby building trust and facilitating their adoption in critical applications such as drug discovery and molecular property prediction.

Chapter 5

Data-centric Prediction Explanation

As one of the decisive factors affecting the performance of a Machine Learning (ML) model, training data points are of great value in promoting the model's transparency and trustworthiness, including explaining prediction results, tracing sources of errors, or summarizing the characteristics of the model [4, 20, 118]. The challenges of example-based prediction explanation mainly come from retrieving relevant data points from a vast pool of training samples or justifying the rationale of such explanations [94, 183].

Method	Explanation of	Need optimization as sub-routine	Whole model explanation		Inference computation complexity bounded by	Memory/cache (**of each training sample**) bounded by
			Theoretical	Practical		
Influence Function	Original Model	Yes (Iterative HVP approximation)	Yes	No	1.$H_\theta^{-1}\nabla_\theta L(\mathbf{x}_t, \theta)$ approximation 2. $< \nabla_\theta L(\mathbf{x}, \theta), H_\theta^{-1}\nabla_\theta L(\mathbf{x}_t, \theta) >$	Size of model parameters
RPS	Fine-tuned Model	Yes (L2 regularized last layer retrain)	No	No	1.last layer representation $\mathbf{f}_t$ 2.$< \alpha_i \mathbf{f}_i, \mathbf{f}_t >$	Size of model parameters of the last layer
TracIn*	Original Model	No	Yes	No	1.$\nabla_\theta L(\mathbf{x}_t, \theta)$ approximation 2. $< \nabla_\theta L(\mathbf{x}, \theta), \nabla_\theta L(\mathbf{x}_t, \theta) >$	Size of model parameters
HD-Explain	Original Model	No	Yes	Yes	1.$\nabla_{\mathbf{x}_t} f(\mathbf{x}_t, \theta)_{y_t}$ 2. Closed-form $k_\theta(\mathbf{x}, \mathbf{x}_t)$ defined by KSD	Size of data dimension

* TracIn typically requires to access the training process. Here, TracIn* refers to a special case that only use the last training checkpoint.

Table 5.1: Summary of existing Post-hoc Example-based Prediction Explanation Methods that work with deep neural networks. Practicality of the whole model explanation is measured by the feasibility of explaining the prediction of ResNet-18 trained on CIFAR-10 with a single A100 GPU machine. CIFAR-10 is a small benchmark data with 50000 training samples.

Modern example-based prediction explanation methods commonly approach the above challenges by constructing an influence chain between training and test data points [90, 118, 160]. The influence chain could be either data points' co-influence on model parameters or their similarity in terms of latent representations. In particular, Influence Function [82], one of the representative, model-aware explanation methods, looks for the shift of the model parameters (due to up-weighting each training sample) as the sample's influence score. Since computing the inverse Hessian matrix is challenging, the approach adapts Conjugate Gradients Stochastic Estimation and the Perlmutter trick to reduce its computation cost. Representer Point Selection (RPS) [174], as another example, reproduces the representer theorem by refining the trained neural network model with $L2$ regularization, such that the influence score of each training sample can be represented as the gradient of the predictive layer. While computationally efficient, RPS is criticized for producing coarse-grained explanations that are more class-level rather than instance-level explanations [154] (In this paper we use the instance level explanation and example-based explanation interchangeably.). Multiple later variants [133, 154] attempted to mitigate the drawbacks above, but their improvements were often limited by the cause of their shared theoretical scalability bounds. The summary of existing Post-hoc Example-based Prediction Explanation Methods that work with deep neural networks is given in Table 5.1.

One limitation of the current influence-based methods is that they attribute the influence of each training data point to the parameters of the trained model as an essential intermediate step. Indeed, as the nature of stochastic gradient descent (the dominating training strategy of neural networks), isolating such contribution is barely possible without 1) relying on approximations or 2) accessing the training process. Unfortunately, either solution would result in performance degradation or heavy computational overhead [141]. Hence, this work delves into the exploration of an alternative influence connection between training and test data points without exploiting the perturbation of model parameters.

5.1 Problem Definition

We consider the task of explaining the prediction of a differentiable classifier $f : \mathbb{R}^d \to \mathbb{R}^l$, given inputs test sample $\mathbf{x}_t \in \mathbb{R}^d$, where d denotes the input dimension and l denotes the number of classes. Specifically, we are interested in explaining a prediction of a model $f(\cdot)$ by returning a subset of its training samples $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ that has strong predictive support to the prediction of test point $\mathbf{x}_t$. While not explicitly stated in the main paper, we treat example-based prediction explanation as a function $\psi(f, D, \mathbf{x}_t) : \mathcal{F} \times \mathcal{D} \times \mathbb{R}^d \to \{\mathbb{R}^d, \mathbb{R}^l\}^k$ such that it takes a trained model f, a training dataset D, and an arbitrary test point $\mathbf{x}_t$ as inputs and output top-k training samples as explanations.

5.1.1 Kernelized Stein Discrepancy

The idea of Kernelized Stein Discrepancy (KSD) [100] can be traced back to a theorem called Stein Identity [75] that states, if a smooth distribution $p(x)$ and a function $\phi(x)$ satisfy $\lim_{||x||\to\infty} p(x)\phi(x) =$

0,

$$\mathbb{E}_{x\sim p}[\phi(x)\nabla_x \log p(x) + \nabla_x \phi(x)] = 0, \quad \forall \phi.$$

The identity can characterize distribution $p(x)$ such that it is often served to assess the goodness-of-fit [85]. The above expression could be further abstracted to use function operator $\mathcal{A}_p$ (a.k.a Stein operator) such that

$$\mathcal{A}_p \phi(x) = \phi(x)\nabla_x \log p(x) + \nabla_x \phi(x),$$

where the operator encodes distribution $p(x)$ in the form of derivative to input (a.k.a score function).

Stein's identity offers a mechanism to measure the gap between two distributions by assuming the variable x is sampled from a different distribution $q \neq p$ such that

$$\sqrt{\mathbb{S}(q,p)} = \max_{\phi \in \mathcal{F}} \mathbb{E}_{x\sim q}[\mathcal{A}_p \phi(x)],$$

where the expression takes the most discriminant ϕ that maximizes the violation of Stein's identity to quantify the distribution discrepancy. This discrepancy is, accordingly, referred as Stein Discrepancy.

5.2 Additional Derivation of Kernelized Stein Discrepancy

While Stein's Identity has been well described in many previous works [29, 100, 101], we briefly recap some key derivations in this paper to seek for self-contained.

As mentioned in the main paper, Stein's Identity states that, if a smooth distribution $p(x)$ and a function $\phi(x)$ satisfy $\lim_{||x||\to\infty} p(x)\phi(x) = 0$, we have

$$\mathbb{E}_{x\sim p}[\phi(x)\nabla_x \log p(x) + \nabla_x \phi(x)] = \mathbb{E}_{x\sim p}[\mathcal{A}_p \phi(x)] = 0, \quad \forall \phi.$$

Intuitively, by using integration by part rules, we can reveal the original assumption from the derived expression such that

$$\int_x \phi(x)\nabla_x \log p(x) + \nabla_x \phi(x) dx = p(x)\phi(x)\Big|_{-\infty}^{+\infty}$$

Stein Discrepancy measures the difference between two distributions q and p by replacing the expectation of distribution p term in Stein's Identity expression with distribution q, which reveals the difference between two distributions by projecting their score functions (gradients) with the function $\phi(x)$

$$\max_{\phi\in\mathcal{F}} \mathbb{E}_{x\sim q}[\mathcal{A}_p\phi(x)] = \max_{\phi\in\mathcal{F}} \mathbb{E}_{x\sim q}[\mathcal{A}_p\phi(x)] - \underbrace{\mathbb{E}_{x\sim q}[\mathcal{A}_q\phi(x)]}_{=0}$$

$$= \max_{\phi\in\mathcal{F}} \mathbb{E}_{x\sim q}[\underbrace{\phi(x)}_{\text{projection coefficients}} \underbrace{(\nabla_x \log p(x) - \nabla_x \log q(x))}_{\text{score function difference}}]$$

Clearly, the choice of projection coefficients (function $\phi(x)$) term is critical to measure the distribution difference.

Kernelized Stein Discrepancy (KSD) addresses the task of searching function ϕ by treating the above challenge as an optimization task where it decomposes the target function ϕ with linear decomposition such that

$$\max_{\phi\in\mathcal{F}} \mathbb{E}_{x\sim q}[\mathcal{A}_p\phi(x)] = \max_{\phi\in\mathcal{F}} \mathbb{E}_{x\sim q}[\mathcal{A}_p \sum_i w_i\phi_i(x)] = \max_{\phi\in\mathcal{F}} \sum_i w_i \mathbb{E}_{x\sim q}[\mathcal{A}_p\phi_i(x)],$$

with linear property of Stein operator $\mathcal{A}_p$. The linear decomposition path is the way to reduce the optimization task into looking for a finite number of the base functions $\phi_i \in \mathcal{F}$ whose coefficient norm is constraint to 1 ($||\mathbf{w}||_{\mathcal{H}} \leq 1$). KSD takes $\mathcal{F}$ to be the unit ball of a reproducing kernel Hilbert space (RKHS) and leverages its reproducing property such that $\phi(x) = \langle\phi(\cdot), k(x,\cdot)\rangle$, which in turn transforms the maximization objective of the Stein Discrepancy into

$$\max_{\phi} \langle\phi(\cdot), \mathbb{E}_{x\sim q}[\mathcal{A}_p k(\cdot, x)]\rangle_{\mathcal{H}}, \quad \textit{s.t.} ||\phi||_{\mathcal{H}} \leq 1.$$

The optimal ϕ is therefore a normalized version of $\mathbb{E}_{x\sim q}[\mathcal{A}_p k(\cdot, x)]$. Hence, KSD is defined as the optimal between the distribution p and q with the optimal solution of ϕ

$$\mathbb{S}(q,p) = \mathbb{E}_{x,x'\sim q}[\kappa_p(x,x')], \quad \text{where} \quad \kappa_p(x,x') = \mathcal{A}_p^x \mathcal{A}_p^{x'} k(x,x').$$

The KSD could be eventually transformed into

$$\mathbb{S}(q,p) = \mathbb{E}_{x,x'\sim q}[\kappa_p(x,x')]$$

where $\kappa_p(x,x') = \mathcal{A}_p^x \mathcal{A}_p^{x'} k(x,x')$ that can work with arbitrary kernel function $k(x,x')$.

In the literature, KSD has been adopted for tackling three types of application tasks – 1)

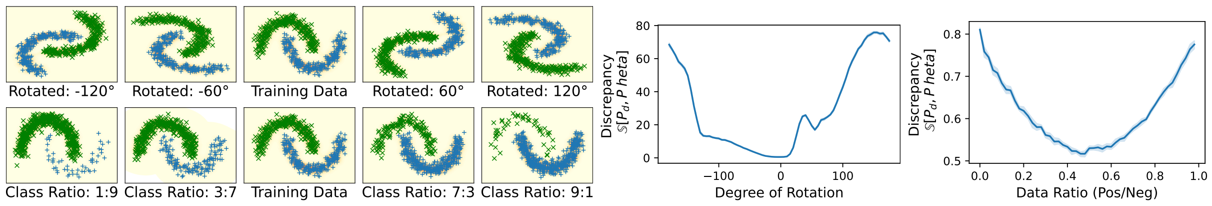


Figure 5.1: Varying of Kernelized Stein Discrepancy given the shift of training data distribution on Two Moon dataset.

parameter inference [11], 2) Goodness-of-fit tests [29, 100, 171], and 3) particle filtering (sampling) [48, 84]. However, to the best of our knowledge, its innate property that uniquely defines model-dependent data correlation has never been exploited, which, we note, is valuable to interpret model behaviour from various aspects, including instance-level prediction explanation and global prototypical explanations.

5.3 Highly-precise and Data-centric Explanation

HD-Explain is an example-based prediction explanation method based on Kernelized Stein Discrepancy. Consider a trained classifier f_θ as the outcome of a training process with Maximum Likelihood Estimation (MLE)

$$\underset{\theta}{\text{argmax}}\, \mathbb{E}_{(\mathbf{x},y)\sim P_D}[\log P_\theta(y|\mathbf{x})].$$

Theoretically, maximizing observation likelihood is equivalent to minimizing a KL divergence between data distribution P_D and the parameterized distribution P_θ such that

$$\begin{aligned}\mathbb{D}_{\text{KL}}(P_D, P_\theta) &= \mathbb{E}_{(\mathbf{x},y)\sim P_D}\left[\log \frac{P_D(\mathbf{x},y)}{P_\theta(\mathbf{x},y)}\right] \\ &= -\underbrace{\mathbb{E}_{(\mathbf{x},y)\sim P_D}[\log P_\theta(y|\mathbf{x})]}_{\text{Likelihood}} + \underbrace{\mathbb{E}_{(\mathbf{x},y)\sim P_D}[\log P_D(y|\mathbf{x})]}_{\text{constant}},\end{aligned}$$

which, in turn, is proven to align with minimizing KSD in the form of gradient descent [101]

$$\nabla_\theta \mathbb{D}_{\text{KL}}(P_D, P_\theta) = -\mathbb{S}(P_D, P_\theta).$$

The chain of reasoning above shows that a well-trained classifier f_θ through gradient-descent should lead to minimum discrepancy between the training dataset distribution and the model encoded distribution $\mathbb{S}(P_D, P_\theta)$ [1]. We empirically verify the connection through simple examples as shown in Figure 5.1, where the changes in training data distribution would result in larger KSD compared to that of the original training data distribution. Intuitively, the connection shows that there is a tie between a model and its training data points, encoded in the form of a Stein kernel function $k_\theta(\cdot,\cdot)$ defined on each pair of data points. As the kernel function is conditioned on model f_θ, we note it is an encoding of data correlation under the context of a trained model, which paves the foundation of the example-based prediction explanation as shown next.

5.3.1 KSD between Model and Training Data

Recall that KSD, $\mathbb{S}(P_D, P_\theta)$, defines the correlation between pairs of training samples through model θ dependent kernel function with closed-form decomposition

$$\begin{aligned}\kappa_\theta((\mathbf{x}_a, y_a), (\mathbf{x}_b, y_b)) &= \mathcal{A}_\theta^a \mathcal{A}_\theta^b k(a, b) \\ &= \nabla_a \nabla_b k(a, b) + k(a, b) \nabla_a \log P_\theta(a) \nabla_b \log P_\theta(b) \\ &\quad + \nabla_a k(a, b) \nabla_b \log P_\theta(b) + \nabla_b k(a, b) \nabla_a \log P_\theta(a),\end{aligned} \tag{5.1}$$

where we denote data point $(\mathbf{x}_a, y_a)$ with a for clean notation. The only model-dependent factor in the above decomposition is a derivative $\nabla_{\mathbf{x},y} \log P_\theta(\mathbf{x}, y)$ (for both data a or b). Hence, to utilize KSD for prediction explanation, we first need to unravel the derivative of the trained model with respect to a whole data point (including its label). Indeed, while the gradient of a classifier $\nabla_\mathbf{x} \log P_\theta(y|\mathbf{x})$ on its input feature $\mathbf{x}$ is straightforward, the gradient to the entire data points is less evident due to the lack of definition of $P_\theta(\mathbf{x})$.

To obtain the joint distribution of $P_\theta(\mathbf{x}, y)$ for facilitating KSD computation, we propose to set $P_\theta(\mathbf{x}) \equiv P_D(\mathbf{x})$ as uniform distribution. Although this setting appears hasty, we note P_D represents the uniform data distribution in dataset D, which, while reflects, but is not the complex unknown distribution from which the data is sampled. In particular, for a data point $(\mathbf{x}, y)$ in dataset D with Independent and identically distributed (I.I.D) assumption, $P_D(\mathbf{x})$ is indeed uniformly distributed, whereas $P_D(y|\mathbf{x})$ is a delta distribution with probability 1.

With the above setting, the score function $\nabla_{\mathbf{x},y} \log P_\theta(\mathbf{x}, y)$ in the Stein operator $\mathcal{A}_\theta$ could be derived as a concatenation of the gradient of model $f_\theta(\mathbf{x})_y$ to its input $\mathbf{x}$ and its probabilistic

[1] Since P_D is discrete distribution while P_θ is continuous, the Discrepancy between the two distributions will not recap Stein Identity ($= 0$) with a limited number of training data points.

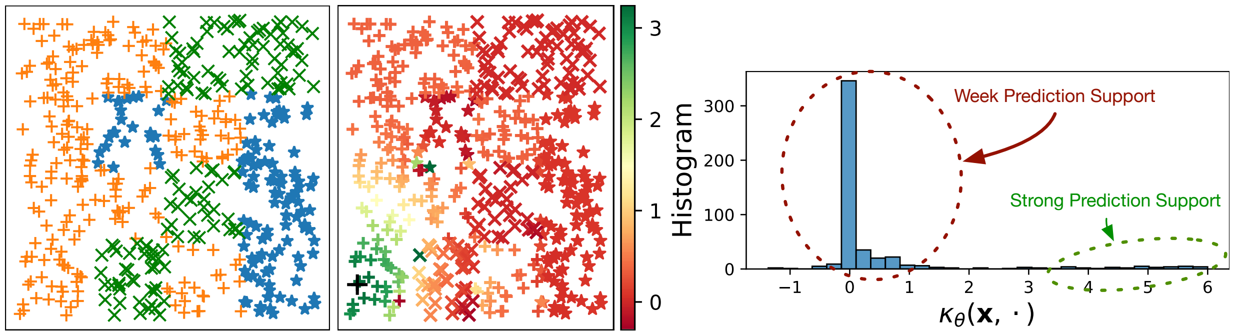


Figure 5.2: Demonstration of HD-Explain on 2D Rectangular synthetic dataset. Left shows the training dataset with three classes. Middle figure shows the explanation support of training data points to a given test point (as black cross), where green shows a higher KSD kernel value. Right shows the distribution of KSD kernel values (over the training set) to the test point, where only a small number of training data points provide strong support to this prediction. The color intensity of this gradient graph, is relevant to Stein Kernel values.

prediction $f_\theta(\mathbf{x})$, since

$$\begin{aligned}\nabla_{\mathbf{x},\mathbf{y}} \log P_\theta(\mathbf{x}, \mathbf{y}) &= \nabla_{\mathbf{x},\mathbf{y}}[\log P_\theta(\mathbf{y}|\mathbf{x}) + \log P_D(\mathbf{x})] \\ &= \nabla_{\mathbf{x},\mathbf{y}} \log P_\theta(\mathbf{y}|\mathbf{x}) + [\nabla_{\mathbf{x}} \log P_D(\mathbf{x})||\nabla_{\mathbf{y}} P_D(\mathbf{x})] \\ &= [\nabla_{\mathbf{x}} \log \mathbf{y}^\top f_\theta(\mathbf{x})||\nabla_{\mathbf{y}} \log \mathbf{y}^\top f_\theta(\mathbf{x})] + [\mathbf{0}||0] \\ &= [\nabla_{\mathbf{x}} \log f_\theta(\mathbf{x})_y||f_\theta(\mathbf{x})],\end{aligned} \tag{5.2}$$

where $[\cdot||\cdot]$ denotes concatenation operation. We use one-hot vector representation $\mathbf{y}$ to represent data label y in the above derivation. Note, since $P_D(\mathbf{x})$ follows a uniform distribution, its gradient to the inputs is a 0 vector.

Combining Equation 5.1 and 5.2, we can estimate the correlation of any pairs of training data points conditioned on the trained machine learning model. Computationally, since a score function $\nabla_{\mathbf{x},y} \log P_\theta(\mathbf{x}, y)$ depends on a single data point, its outputs of the training set could be pre-computed and cached to accelerate the kernel computation. In particular, the output dimension of the score function is simply $m + k$ for data with m dimensional features and k class labels. Compared to the existing solutions, whose training data cache (or influence) are bounded by the dimension of model parameters (such as Influence function, TracIn, RPS, RPS-JLE), the explanation method built on KSD would come with a significant advantage in terms of scalability (see comparison in Table 5.1). This statement is generally true for neural network based classifiers, whenever the size of model parameters is far larger than the data dimension.

5.3.2 Prediction Explanation

The computation of kernel function in Equation 5.1 requires access to features and labels of a data point. While the ground-truth label information is available for the training set, it is inaccessible for a test data point. We consider the predicted class $\hat{y}_t$ of the test data point $\mathbf{x}_t$ as a label to construct a complete data point $(\mathbf{x}_t, \hat{y}_t)$ and apply the KSD kernel function.

For a test data point $\mathbf{x}_t$, we search for top-k training data points that maximize the KSD defined kernel.

Figure 5.2 demonstrates HD-Explain on a 2d synthetic dataset. The distribution of $\kappa_\theta(\mathbf{d}, \cdot)$ in the right most plot shows that only a small number of training data points have a strong influence on a particular prediction.

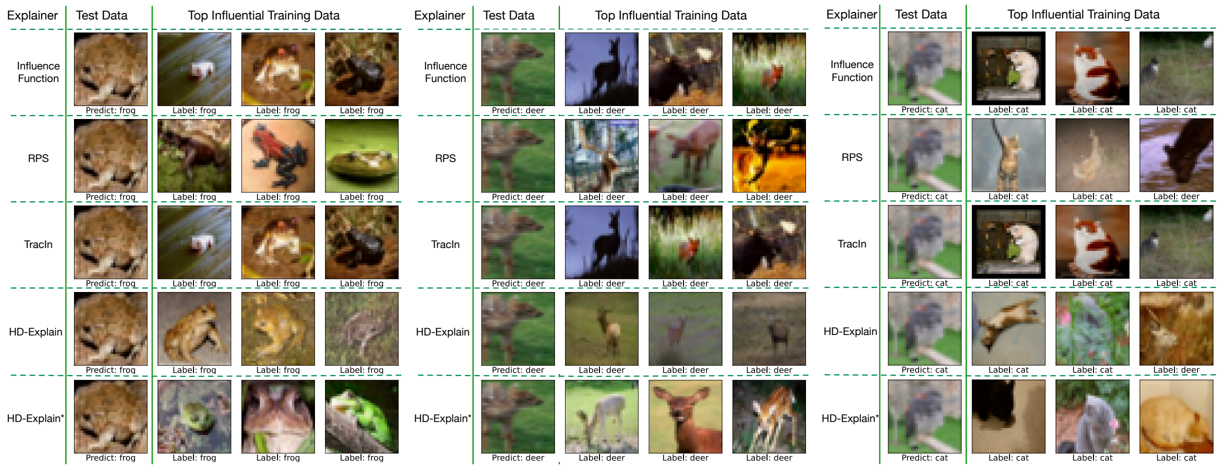


Figure 5.3: Qualitative evaluation of various example-based explanation methods using CIFAR10. We show three scenarios where the target model makes (from left to right) 1) a highly-confident prediction that matches ground truth label, 2) a low-confident prediction that matches ground truth label, 3) low-confident prediction that does not match ground truth label (which is a bird). For each sub plot, we show top-3 influential training data points picked by the explanation methods for the test example.

5.4 Evaluation and Analysis

In this section, we conduct several qualitative and quantitative experiments to demonstrate various properties of HD-Explain and compare it with the existing example-based solutions.

Datasets: We consider multiple disease classification tasks where diagnosis explanation is highly desired. We also introduced synthetic and benchmark classification datasets to deliver the main idea without the need for medical background knowledge. Concretely, we use CIFAR-10 $(32 \times 32 \times 3)$,

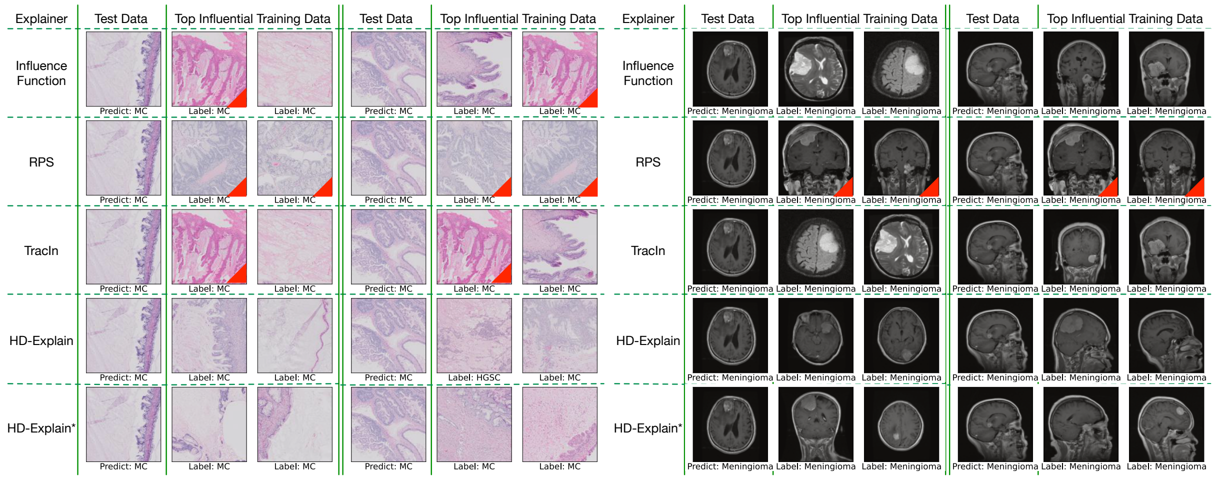


Figure 5.4: Qualitative evaluation of example-based explanation methods on Overian Cancer histopathology (left) and Brain Tumor MRI (right) datasets. We show two test data points that are predicted to belong to the same class in each dataset. Red triangle in the top right corner of an image shows the duplicate explanations across test samples.

Brain Tumor (Magnetic Resonance Imaging, $128 \times 128 \times 3$), and Ovarian Cancer (Histopathology Images, $128 \times 128 \times 3$) datasets.

Dataset	Application	Type	# Size	# Feature Dimension	# Number of Classes	Duplicated Samples	Public Dataset
Two Moons	Synthetic	2D Numeric	500	2	2	No	Shared with code
Rectangulars	Synthetic	2D Numeric	500	2	3	No	Shared with code
CIFAR-10	Classification Benchmark	Image	60,000	$32 \times 32 \times 3$	10	No	Yes
Overian Cancer	Histopathology **(Private)**	Image	20,000	$128 \times 128 \times 3$	5	Yes	No
Brain Tumor	MRI Benchmark	Image	7,023	$128 \times 128 \times 3$	4	Yes	Yes

Table 5.2: Summary of datasets used in the paper.

In this paper, we conducted our experiments on five datasets – two synthetic and three benchmark image classification datasets. As the work concerns the trustworthiness of the machine learning model in high-stakes applications, we also introduced medical diagnosis datasets to provide more insight into the potential benefit the proposed work introduced. To train the target machine learning models, we conducted data augmentations to increase the number of training data samples, including random cropping, rotation, shifting, horizontal flipping, and noise injection. Table 5.2 summarizes more details about the datasets.

Baselines: The baseline explainers used in our experiments include Influence Function, Representer Point Selection, and TracIn. While other variants of these baseline explainers exist [12, 26, 154], we note they don't offer fundamental performance improvements over the classic ones. In addition, as Influence Function and TracIn face scalability issues, we limit the influence of parameters to the last layer of the model so that they can work with models that contain a large number of parameters. Our experiments use ResNet-18 as the backbone model architecture (with around 11 million trainable parameters) for all image datasets. Finally, we also introduce an HD-Explain variant (HD-Explain*) to match the last layer setting of other baseline models, even though HD-Explain can scale up to the whole model without computation pressure.

HD-Explain* variant to match the last layer setting of other baseline models (e.g. RPS). The HD-Explain* is a simple change of HD-Explain in terms of using data representations (the output of the last non-predictive layer of the neural classifier) rather than the raw features. Specifically, we assume a neural network model f_θ could be decomposed into two components $f_{\theta_2} \cdot f_{\theta_1}$, where f_{θ_1} is a representation encoder and f_{θ_2} is a linear model for prediction. With this decomposition, we define the KSD kernel function for HD-Explain* as

$$\begin{aligned}\kappa_\theta((f_{\theta_1}(\mathbf{x}_a), y_a), (f_{\theta_1}(\mathbf{x}_b), y_b)) &= \mathcal{A}^a_{\theta_2}\mathcal{A}^b_{\theta_2} k(a,b) \\ &= \nabla_a \nabla_b k(a,b) + k(a,b)\nabla_a \log P_{\theta_2}(a) \nabla_b \log P_{\theta_2}(b) \\ &+ \nabla_a k(a,b) \nabla_b \log P_{\theta_2}(b) + \nabla_b k(a,b) \nabla_a \log P_{\theta_2}(a),\end{aligned}$$

where we define $a = (f_{\theta_1}(\mathbf{x}_a), y_a)$ and $b = (f_{\theta_1}(\mathbf{x}_b), y_b)$ for short. This setting reduces the prediction explanation to the last layer of the neural network in a similar fashion to RPS.

Metrics: In existing example-based explanation works, the experimental results are often demonstrated qualitatively, as visualized explanation instances, without quantitative evaluation. This results in subjective evaluation. In this paper, we propose several quantitative evaluation metrics to measure the effectiveness of each method.

- **Hit Rate:** Hit rate measures how likely an explanation sample hits the desired example cases where the desired examples are guaranteed to be undisputed. Specifically, we modify a training data point with minor augmentations (adding noise or flipping horizontally) and use it as a test data point, such that the best explanation for the generated test data point should be the original data point in the training set.

- **Coverage:** Given n test data points, the metric measures the number of unique explanation samples an explanation method produces when configuring to return top-k training samples. Formally,

$$\text{Coverage} = \frac{|\cup_{i=1}^{n} e_i|}{n \times k}$$

where e_i is the set of top-k explanations for test data point i. Coverage is motivated to measure the diversity of explanations across a test set where a high value reflects higher granularity (per test point) of the explanation.

- **Run Time:** It measures the run time of an explanation method in wall clock time.

5.4.1 Qualitative Evaluation

Figure 5.3 shows three test cases of the CIFAR10 classification task that cover different classification outcomes, including high-confident correct prediction, low-confident correct prediction, and low-confident incorrect prediction. For both correct prediction cases, we are confident that HD-Explain provides a better explanation than others in terms of visually matching test data points e.g. brown frogs in Figure 5.3 (a) and deer on the grass in Figure 5.3 (b). In contrast, for the misclassified prediction case (Figure 5.3 (c)), we note the HD-Explain produces an example that does not even belong to the same class as the predicted one. This reflects a low confidence in model's prediction for the particular test example and highlight a potential error in the prediction.

RPS also shows such inconsistency in explanation, which aligns with its claim [174]. The other two baseline methods do not offer such properties and still produce explanations that match the predicted label well. It is hard to justify how those training samples support such prediction visually (since no clear shared pattern is obvious to us). In addition, it is interesting to see that Influence Function and TracIn produce near identical explanations, reflecting their similarity in leveraging the perturbation of model parameters.

Figure 5.4 provides additional insights into Ovarian Cancer histopathology and Brain Tumor MRI datasets. HD-Explain again shows a better explanation for producing training samples that appear similar to the test samples (note for the semantic similarity, these explanations should be referred to a medical practitioner). For instance, the explanation of HD-Explain follows the scanning orientation of test points in MRI as shown in Figure 5.4 (b). We note all baseline approaches tend to produce similar explanations to test samples belonging to the same classes. Rather than providing individual prediction explanations, those approaches act closer to per-class interpreters that look for class prototypes. To verify this observation further, we conducted a quantitative evaluation as described in the next section.

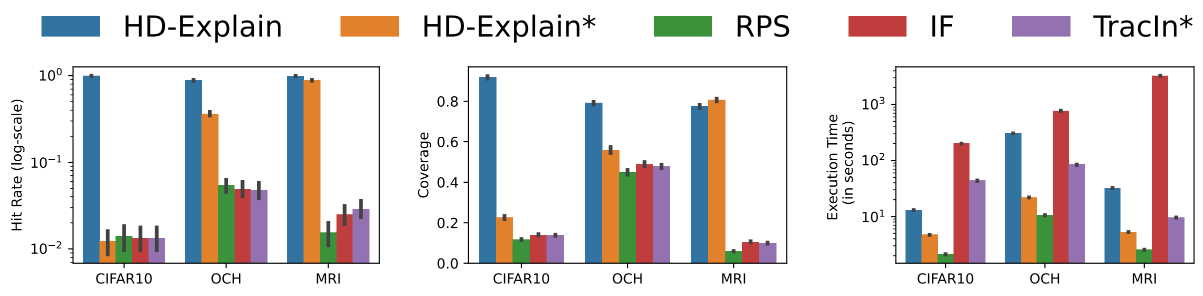


Figure 5.5: Quantitative explanation comparison among candidate example-based explanation methods. From left to right: Hit Rate, Coverage, and Execution Time. Data augmentation strategy used is Noise Injection. Error bar shows 95% confidence interval.

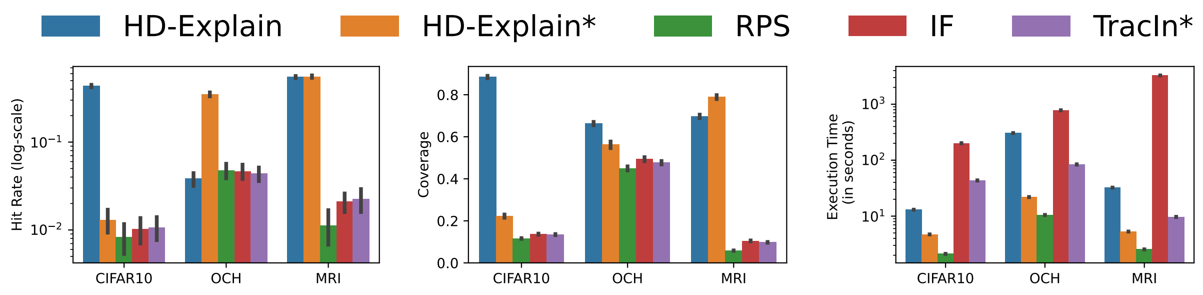


Figure 5.6: Quantitative explanation comparison among candidate example-based explanation methods. From left to right: Hit Rate, Coverage, and Execution Time. Data augmentation strategy used is Horizontal Flip. Error bar shows 95% confidence interval. We reuse the legend of Figure 5.5.

5.4.2 Quantitative Evaluation

In order to perform quantitative evaluation, we limit our experiments to datasets where ground-truth explanation samples are available.

Specifically, given a training data sample $(\mathbf{x}_i, y_i)$, we generate a test point $\mathbf{x}_t$ by adopting two image data augmentation methods:

- **Noise Injection:** $\mathbf{x}_t = \mathbf{x}_i + \epsilon \quad \text{s.t.} \quad \epsilon \sim \mathcal{N}(0, 0.01\sigma)$, where σ is the element-wise standard deviation of features in the entire training dataset.

- **Horizontal Flip:** $\mathbf{x}_t = \text{flip}(\mathbf{x}_i)$, where we flip images horizontally that do not compromise the semantic meaning of images.

We created 30 augmented test points for each training data point ($> 10,000$ data points) in each dataset, resulting in more than $300,000$ independent runs. Since the data augmentation is guaranteed to maintain prediction consistency, the ideally best explanation for the generated test point is the original data point $\mathbf{x}_i$ itself.

Hence, the quantitative evaluation could be a sample retrieval evaluation where **Hit Rate** measures the probability of successful retrieval.

Figure 5.5(a) shows the hit rate comparison among candidate methods on the three image classification datasets under Noise Injection data augmentation.

The existing methods face significant difficulty in retrieving the ideal explanatory sample ($\leq 10\%$), even with such a simple problem setup; only HD-Explain (and its variant) produces a reasonable successful rate ($> 80\%$).

We further investigate the diversity of explanations across a testset using the **Coverage** metric. Here, diversity indirectly reflects the granularity of an explanation when accumulated over the test dataset. Figure 5.5(b) shows the Coverage score, the ratio of explanation samples that are unique over many test points.

It turns out that existing solutions produce only 10% - 50% coverage – many test points receive the same set of explanations, disregarding their unique characteristic. This result is in line with our previous observation in Figure 5.4, where we saw multiple duplicated explanations (as shown with red corner tags). We further observed that the explanations of baselines are often dominated by the class labels; data points predicted as the same class would receive a similar set of explanations. In contrast, HD-Explain shows substantially higher coverage, generating explanations that considers the unique characteristics of each test point.

Regarding computation efficiency, while we have summarized the scalability limitation of the candidate methods in Table 5.1, there was no computational efficiency evaluation conducted in previous works.

We recorded the wall clock execution time of each experiments as shown in Figure 5.5(c). As expected, the Influence Function takes longer to return its explanation than other methods. HD-Explain*, TracIn* and RPS, all use the last layer to generate explanation. RPS showed the lowest compute time since it does not require auto-differentiation for computing the training data influence. HD-Explain* showed the second best compute time and is efficient than TracIn*[2] and IF. HD-Explain considers the whole model for explanation and its compute time is not directly comparable to others. However, it shows better efficiency than IF across all datasets and is better than TracIn* on CIFAR10.

We observe a similar trend in the other data augmentation scenario, Horizontal Flip, where computation time and coverage are roughly the same, as shown in Figure 5.6. However, we do

[2]TracIn* is configured only to compute the gradient of the prediction layer due to its high memory requirements.

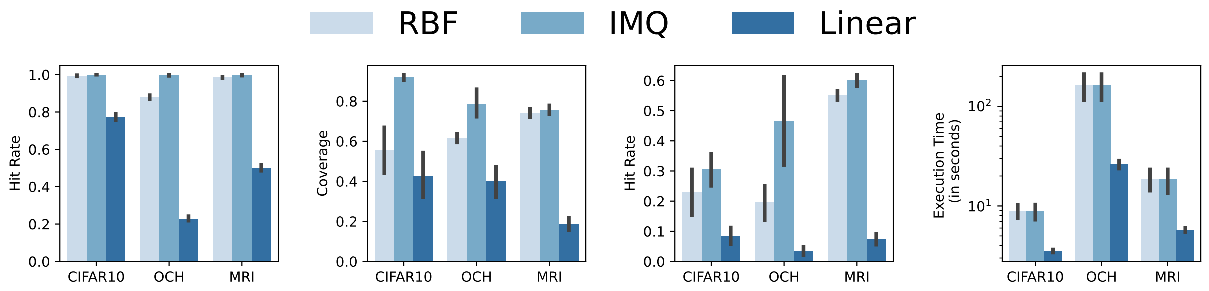


Figure 5.7: Quantitative explanation comparison among HD-Explainers with different kernel functions on all image classification datasets. From left to right, Noise Injection Hit Rate, Coverage for both, Horizontal Flip Hit rate, Execution time for both. Error bar shows 95% confidence interval.

notice that, as the outcome of image flipping, the raw feature (pixel) level similarity between $\mathbf{x}_t$ and $\mathbf{x}_i$ is destroyed. As an outcome, the HD-Explain that works on raw features suffers from performance deduction while other methods, including HD-Explain*, are less impacted. This observation suggests that choosing the layer of explanation might be considered in the practical usage of this approach.

5.4.3 Kernel Options

We use the Radial Basis Function (RBF) as our default choice of kernel. However, another kernel may better fit a particular application domain. In this experiment, we compare three well-known kernels i.e. Linear, RBF, and Inverse Multi-Quadric (IMQ) on the three image classification datasets.

Figure 5.7 presents the results under both data augmentation scenarios. Overall, the IMQ kernel performs better than the RBF kernel regarding explanation quality (Hit Rate). The advantage is significant when the data augmentation scenario is Horizontal Flip (Figure 5.7c, which appears more challenging than Noise Injection. IMQ also showed better performance on Coverage (Figure 5.7b).

The linear kernel performs worse compared to other kernels. However, it is substantially efficient than the others, as shown in Figure 5.7d, highlighting its utility on large datasets. Compared to the baselines presented in Figure 5.5, we note that the Linear kernel is sufficient for HD-Explain to stand out from other methods in both performance and efficiency.

5.4.4 Discussion: Intuition on why HD-Explain works

In HD-Explain, the key metric on measuring the predictive supports of a test point $\mathbf{x}_t$ given a training data $(\mathbf{x}_i, \mathbf{y}_i)$ is the KSD defined kernel $\kappa_\theta([\mathbf{x}_t||\hat{\mathbf{y}}_t], [\mathbf{x}_i||\mathbf{y}_i])$, where $\hat{\mathbf{y}}_t$ denotes the predicted class label by model f_θ in one-hot encoding. By definition, the kernel $\kappa_\theta((\mathbf{x}_a, y_a), (\mathbf{x}_b, y_b)) = k_\theta(a, b)$

between two data points can be decomposed into four terms

$$\underbrace{\text{trace}(\nabla_a \nabla_b k(a,b))}_{①} + \underbrace{k(a,b)\nabla_a \log P_\theta(a)^\top \nabla_b \log P_\theta(b)}_{②} \\ + \underbrace{\nabla_a k(a,b)^\top \nabla_b \log P_\theta(b)}_{③} + \underbrace{\nabla_b k(a,b)^\top \nabla_a \log P_\theta(a)}_{④}.$$

We examine the effect of each term as follows:

- ①: **Raw Feature Similarity**: We note that the first term is often a similarity bias of raw data points given a specified kernel function. In particular, for the RBF kernel $k(a,b) = \exp(-\gamma||a-b||^2)$, the first term is simply $\sum_i^{d+l} 2\gamma k(a,b)$, where $d+l$ refers to the sum of input and output (in one-hot) dimensions of a data point. Intuitively, the term shows how similar the two data points are given the RBF kernel. For linear kernel $k(a,b) = a^\top b$, on another hand, the first term is simply $d+l$ as a constant bias term, which does not deliver any similarity information between the two data points.

- ②: **Score Similarity**: The second term reflects the similarity between two data points in the context of the trained model. In particular, considering the sub-term $\nabla_a \log P_\theta(a)^\top \nabla_b \log P_\theta(b)$, based on our derivation in Equation 5.2 (in the main paper), we note it is equivalent to

$$[\nabla_{\mathbf{x}_a} \log f_\theta(\mathbf{x}_a)_{y_a} || f_\theta(\mathbf{x}_a)]^\top [\nabla_{\mathbf{x}_b} \log f_\theta(\mathbf{x}_b)_{y_b} || f_\theta(\mathbf{x}_b)] \\ = \underbrace{\nabla_{\mathbf{x}_a} \log f_\theta(\mathbf{x}_a)_{y_a}^\top \nabla_{\mathbf{x}_b} \log f_\theta(\mathbf{x}_b)_{y_b}}_{\text{similarity of scores (input gradients)}} + \underbrace{f_\theta(\mathbf{x}_a)^\top f_\theta(\mathbf{x}_b)}_{\text{similarity of predictions}},$$

  where both terms could be viewed as similarity between data points in the context of trained model.

- ③-④: **Mutual Influence of Prediction Shifts** Both of the last two terms examine the alignment between the score of one data point and the kernel derivative of another data point. We conjecture that this alignment reflects how a test prediction would change if there is a training data point present closer to it than before.

HD-Explain is designed to explain a model's prediction for a particular test data point. Its performance depends on the performance of the target model, which we consider a desirable characteristic rather than a weakness. A well-trained model should yield accurate and coherent explanations, whereas the explanations provided by a poorly trained model are likely to diverge from expected outcomes. HD-Explain provides prediction explanations for parameterized models regardless of their training quality. However, our approach can identify poorly trained models, as the quality of the explanations is influenced by the training quality. As demonstrated in our experiment (Figure 9), a poorly trained model can be identified by observing that the explanation samples provided by HD-Explain exhibit a mixture of labels that do not align with the predicted labels.

5.5 Summary

In this chapter, we delve into the significant role of data-centric explainer in enhancing the performance, transparency, and trustworthiness of machine learning (ML) models. Training data points are crucial for explaining prediction results, tracing sources of errors, and summarizing the characteristics of the model. The main challenges in example-based prediction explanation stem from the difficulty in retrieving relevant data points from a vast pool of training samples and justifying the rationale behind such explanations.

We propose Highly-precise and Data-centric Explanation (HD-Explain), a post-hoc, model-aware, example-based explanation solution for neural classifiers. HD-Explain bypasses the perturbation of model parameters and instead leverages the Kernelized Stein Discrepancy (KSD) between the trained predictive model and its training dataset. The Stein operator augmented kernel defines a pairwise data correlation, and its expectation on the training dataset results in the minimum KSD compared to datasets from different distributions. This property allows HD-Explain to reveal a subset of training data points that best support the test point's prediction and identify potential distribution mismatches among training data points.

The key contributions of HD-Explain are:

A novel example-based explanation method that does not rely on model parameter perturbation. Introduction of quantitative evaluation metrics to measure the correctness and effectiveness of generated explanations. Comprehensive evaluation comparing HD-Explain with existing explanation methods across various classification tasks. We conducted extensive qualitative and quantitative experiments to demonstrate the effectiveness and efficiency of HD-Explain. The empirical results show that HD-Explain provides fine-grained, instance-level explanations with remarkable computational efficiency, outperforming well-known example-based prediction explanation methods. Its simplicity, effective performance, and scalability make it suitable for real-world applications where transparency and trustworthiness are crucial.

In conclusion, HD-Explain represents a significant advancement in the field of example-based prediction explanation. By leveraging the properties of Kernel Stein Discrepancy, it offers precise and computationally efficient explanations, enhancing the interpretability of machine learning models. This work contributes to the ongoing efforts to improve the transparency of ML models, making them more understandable and reliable for practical use.

This summary encapsulates the key points and findings discussed in the chapter on data-centric prediction explanation, highlighting the novel contributions of the HD-Explain method.

Chapter 6

Data-centric Assessment of Machine Unlearning Feasibility

Machine unlearning (MU) [21] refers to a process that enables machine learning (ML) models to remove specific training data and revert corresponding data influence on the trained models while preserving the models' generalization. As many countries and territories have promulgated their Right to be Forgotten regulations [1], entitling individuals to revoke their authorization to use their data for machine learning (ML) model training, the demand of MU raised significant interest in the machine learning research community, leading to various types of unlearning approaches, often achieved by either data reorganization [49, 60, 158] or model manipulation [59, 164].

Although existing machine unlearning studies vary based on diverse theoretical foundations, their performance evaluation metrics used are generally common, including 1) Data Erasure Completeness, 2) Unlearning Time Efficiency, 3) Resource Consumption, and 4) Privacy Preservation [146, 170, 172]. While, in a laboratory environment, such high-level evaluation can satisfy the needs of comparing different machine unlearning methods from multiple aspects, they often fall short in assessing the difference of effectiveness of machine unlearning for individual data points. When measuring the success of unlearning, researchers often report the success rate only instead of paying attention to the corner cases, leading to a biased understanding of MU approaches' performance. However, we note that such assumption in practice are not hold; Some data points are more easily to be unlearned while others are not. Such difference could be intrinsic (e.g., augmented data distribution under a trained ML model), regardless which machine unlearning algorithm is used.

While measuring the difficulty of unlearning operations [110, 125] is underexplored, one may come up with a handy metric constructed on top of several obvious factors, such as model complexity, prediction confidence, and the nature of input data features. Indeed, the most straightforward criterion of success of unlearning is to measure the distance of the data sample from the model's decision boundaries [23, 99]; data points closer to the decision boundary are more likely to be forget-able. Unfortunately, we show in this paper that although the factors above have a certain

[1] CCPA in California, GDPR in Europe, PIPEDA in Canada, LGPD in Brazil, and NDBS in Australia.

degree of positive correlation with the difficulty of unlearning, they are often accompanied by noise, leaving them lacking credibility.

In this paper, we investigate the general feasibility/difficulty of machine unlearning operations (regardless of specific unlearning approaches) on individual samples within the training set and identify the criteria contributing to this process. Instead of falling back to predictive confidence-based approaches, we hypothesize that the difficulty of unlearning depends on model-augmented data distribution. Specifically, we show that trained ML models can uniquely define parameterized kernel functions $k_\theta(\cdot,\cdot)$ over training data points, which allow to express the distribution of training samples conditioned on trained model f_θ in the form of a correlation graph. Analyzing the conditional data distribution will lead to more accurate estimation for the difficulty of unlearning. While there are many choices of parameterized kernel functions, we note the kernel defined on Kernelized Stein Discrepancy (KSD) [29, 100] shows a unique advantage in terms of accurately reflecting a trained model's characteristics given its training samples. By analyzing the property of KSD, we show there is a much more reliable metric defined on KSD that can measure the difficulty of unlearning efficiently. With the proposed evaluation metrics, one may reduce unnecessary machine unlearning operations when data points are determined as infeasible to unlearn.

6.1 Preliminaries

6.1.1 Machine Unlearning Definition

Machine Unlearning (MU) is a process of removing specific subsets of training data (along with their influence) from a trained model [14, 21].

Formally, consider a training dataset $\mathcal{D}_t = \{(\mathbf{x}_i, y_i)\}$ comprising n samples, where $\mathbf{x}_i$ and y_i represent the i^{th} data's features and corresponding label. We define two subsets of the dataset for clarity as follows: Let $\mathcal{D}_f \subseteq \mathcal{D}_t$ denote the subset of data designated to be forgotten (a.k.a *forget set*), and $\mathcal{D}_r \subseteq \mathcal{D}_t$ denote the remaining data (a.k.a *remaining set*), such that $\mathcal{D}_f \cup \mathcal{D}_r = \mathcal{D}_t$ and $\mathcal{D}_f \cap \mathcal{D}_r = \emptyset$.

Given an target predictive model f_θ with parameters θ, the common expectation of machine unlearning operation are of adjusting θ to an modified parameter set ϑ such that:

- Increasing of model's error on the forget set $\mathcal{L}_\vartheta(\mathcal{D}_f)$.

- Maintaining original model's error on the remaining set $\mathcal{D}_r$ such that

$$\|\mathcal{L}_\theta(\mathcal{D}_r) - \mathcal{L}_\vartheta(\mathcal{D}_r)\| < \epsilon, \tag{6.1}$$

where ϵ denotes a tolerable performance degradation threshold and $\mathcal{L}$ denotes the loss.

6.1.2 Research Track of Machine Unlearning

The simplest solution for unlearning is to retrain the model from scratch with the *remaining data* after removing the *forget data*. However, even with tricks such as partial retraining [14], that works on decomposible, partial model components, retraining remained resource-intensive. To mitigate the computational overhead of retraining, unlearning operations are often approximated". In particular, Fine-Tuning based approaches [47, 164] suggest to continue training the model on the remaining data such that forget data can be naturally flashed out. Alternatively, Gradient Ascent methods [49, 159] adjust the model's weights in the direction of the gradient to increase the model's error on the data intended for forgetting.

In addition to the straightforward unlearning approaches above, recent studies have frequently utilized the Newton update as a fundamental step for removing data influence [47, 59, 126, 143]. These methods typically leverage the Fisher Information Matrix (FIM) to gauge the sensitivity of the model's output to perturbations in its parameters. For example, Fisher Forgetting [47] employs a scrubbing approach where noise is added to parameters based on their relative importance in distinguishing the forget set from the remaining data set. Mehta et al. [113] employs conditional independence coefficient to identify sufficient sets of parameters for targeted unlearning.

As machine unlearning often involve privacy protection regulations, some methods also incorporated the principles of differential privacy (DP) [1] to ensure the unlearning outcome does not inadvertently reveal information about the data that has been removed. Izzo [67, 179] adhere to the DP framework to ensure a high probabilistic similarity between models before and after unlearning. [59] introduced the concept of certified unlearning, grounded in information theory and specifically tailored to the Fisher Information Matrix. Certified Minimax Unlearning [98] has developed an algorithm specifically for minimax models. This method removes data influences through a total Hessian update and incorporates the Gaussian Mechanism to achieve (ϵ, δ)-minimax unlearning certification, ensuring a balance between data removal and model integrity. [27] propose a data deletion technique that ensures the privacy of deleted records. [161] investigated the machine

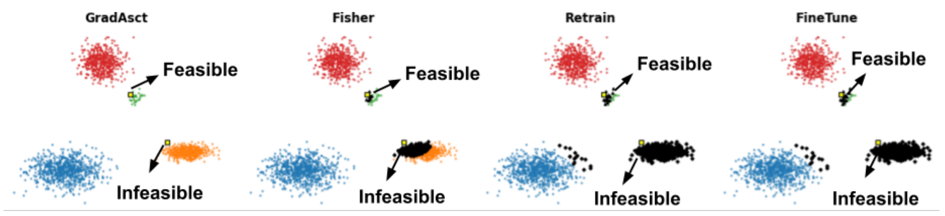


Figure 6.1: Number of similar samples to be forgotten for guaranteed unlearning. The portion of unlearned samples is consistent with the difficulty of unlearning. "Feasible" and "Infeasible" datapoints are shown with yellow markers.

unlearning in the context of SGD and streaming removal requests, but ensured that their method is differentially private. DP algorithms provide the upper bound for the unlearning scheme, but they don't guarantee the full unlearning of requested data [120].

Other research efforts extend into various domains of unlearning, such as knowledge distillation [28], selective forgetting for lifelong learning [147], federated unlearning [22], online unlearning [25, 91], and exploring adversarial attacks using machine unlearning methods [32, 165]. Instead of aiming to follow privacy protection regulation, those approaches focus on reduce the vulnerability of models to adversarial attacks by forgetting training examples [68].

The majority of the literature on machine unlearning primarily concentrates on the development of unlearning algorithms or unlearning approximation techniques for selectively forgetting data from a trained model. As such, the corresponding evaluation metrics are designed to favor the performance difference between algorithms on highly aggregated level (e.g. success rate). In fact, a common assumption underlying much of this research is that unlearning operations are universally feasible for all data points within a dataset, where effectiveness of unlearning will behave consistently across different datasets. This assumption often overlooks the potential variability in unlearning efficacy due to differences in data characteristics or model dependencies, suggesting a need for more nuanced studies that evaluate the specific conditions under which MU can be effectively implemented.

6.2 Understanding Difficulty of Unlearning

In this section, we first investigate the main factors that impact effectiveness of machine unlearning with corresponding data analysis and intuition justifications. We then present our proposed definition of the *unlearning cost* that scores each individual data points with their difficulty of being unlearned from a trained model.

6.2.1 Factors that Affect Difficulty of Unlearning

Intuitively, to quantify success of unlearning operation, one may evaluate the quality of the unlearned model ϑ with respect to the common expectation mentioned in previous section. Particularly, when the forget set contains a single sample $(\mathbf{x}_f, y_f)$, we expect the prediction of the unlearned model on that data point to be same as of a model trained without the data point (a.k.a Certified Unlearning [59]), reflecting that the model's ignorance the to forgotten example. However, it is often impractical to verify the success of unlearning by comparing to a re-trained model; such comparison lose the practical value of unlearning. Hence, evaluation criteria of successful unlearning operation in practice often falls back to measuring 1) the shift of prediction (e.g. altering predicted classification label) along with 2) the model performance preservation criterion (See Eq. 6.1).

Unfortunately, the two metrics mentioned above are often a pair of trade-off, resulting the following factors that jointly pose challenges for defining the difficulty of unlearning.

1. **Size of Unlearning Expansion:** Altering prediction outcome of target sample may negatively impact model performance. When a guaranteed unlearning is desired, one might need to expand unlearning operation to a broader training sample set (the similar data samples) such that unlearning of target sample with respect to decision shift can be successful [23]. While selection of unlearning algorithm may vary the size of the expansion, the need of expanding forget set for guaranteed unlearning remains consistent, as shown in Figure 6.1.

2. **Resistance to Inference Attack:** Also called Membership Attack [24, 46, 152]. In a membership inference attack, adversaries use the model's outputs, such as confidence scores, to ascertain if a specific data point was part of the training set, without needing direct access to the model's internal parameters. With identical unlearning operations (e.g. number of unlearning step, learning rate, and other hyper-parameters), data points often show different level of resistance to the inference attack, which is an alternative factor of measuring the difficulty of unlearning.

3. **Geometric Distance to Decision Boundary:** The distance of a data point to the decision boundary of a model often closely associated with the predictive confidence. A data point with lower predictive confidence (larger uncertainty) is closer to the decision boundary [119]. A data with higher uncertainty might be easier to unlearn compared to those with high predictive confidence [23]. However, predictive confidence does not account for the relationships among similar examples. A data point with low predictive confidence might be deeply embedded

within a complex region of the decision boundary, requiring significant modifications to unlearn. Hence, it remains to be a noisy index.

4. **Tolerance of Performance Deduction:** For an easily unlearn-able data point, guaranteed unlearning is achievable with minimal impact on the model's predictive performance [96]. Here the model performance measurement is on a holdout validation dataset rather than remaining training dataset (as given in Equation 6.1). Conversely, an infeasible data point would require significant changes to the model or the removal of numerous similar samples, leading to a substantial deviation in performance metrics.

5. **Number of Unlearning Steps:** Number of Unlearning Steps evaluates the computational efficiency of the unlearning method, indicating how quickly the model can be updated to forget the specified data. For a given unlearning algorithm, the metric can be approximated through wall clock duration [120].

6. **Distance of Parameter Shift:** valuation often includes layer-wise and activation-wise distances. Layer-wise distance measures the weight differences between the unlearned and original models, while activation-wise distance assesses their activation differences given the same input [47, 158]. These criteria can be used to measure the efficacy of unlearning.

With the identified factors that affect difficulty of unlearning, one my seek for a coherent scoring systems that can rank training data points from the most easy to the most challenging for unlearning operations. In the next section, we will discuss ideas for developing scoring heuristics to differentiate the relationships between training data and their unlearning feasibility.

6.3 Scoring Unlearning Difficulty

By summarizing the unlearning difficulty factors above, we note that they can be clustered into two major factor groups, namely 1) data points with/without strong ties (factor 1, 4-6) and 2) predictive confidence (factor 2-3). Hence, to develop a scoring heuristic, we are looking for a metric that jointly consider the two major factor groups.

6.3.1 Kernel Stein Discrepancy (KSD)

The KSD [100] provides a robust measure to evaluate the goodness-of-fit between a machine learning model and its training dataset. KSD originates from a mathematical theorem known as

Stein's Identity [75], which posits that for a smooth distribution $p(x)$ and a function $\phi(x)$ satisfying the condition $\lim_{||x||\to\infty} p(x)\phi(x) = 0$, the following equation holds for any function ϕ:

$$\begin{aligned}\mathbb{E}_{x\sim p}[\phi(x)\nabla_x \log p(x) + \nabla_x \phi(x)] = \\ \mathbb{E}_{x\sim p}[\mathcal{A}_p \phi(x)] = 0, \quad \forall \phi\end{aligned} \tag{6.2}$$

where $\mathcal{A}_p$ denotes the Stein operator, encapsulating the distribution $p(x)$ in terms of derivatives.

Stein's Discrepancy, derived from Stein's Identity, quantifies the difference between two distributions. Traditionally, finding the optimal function in the space $\phi \in \mathcal{F}$ has been computationally prohibitive, which limited the practical application of native Stein's Discrepancy. Luckily, the introduction of Kernelized Stein Discrepancy mitigated the issue by employing the kernel trick. This approach redefines the search for an optimal function ϕ in $\mathcal{F}$ to finding an appropriate kernel function κ. KSD is defined as:

$$\mathbb{S}(q,p) = \mathbb{E}_{x,x'\sim q}[\kappa_p(x,x')] \tag{6.3}$$

where $\kappa_p(x,x') = \mathcal{A}_p^x \mathcal{A}_p^{x'} k(x,x')$ and can operate with any arbitrary kernel function $k(x,x')$. This formulation allows for a more feasible application in various practical scenarios, making KSD a valuable tool for assessing model-data compatibility. Indeed, in the goodness-of-fitness context, the p distribution often represent predictive distribution of model parameterized with θ, while q distribution denotes the data distribution.

Interestingly, by examining the parameterized kernel defined on pairs of training data points as follows

$$\begin{aligned}
&\kappa_\theta((\mathbf{x}_a, y_a), (\mathbf{x}_b, y_b)) = \mathcal{A}_\theta^a \mathcal{A}_\theta^b k(a,b) \\
&= \nabla_a \nabla_b k(a,b) && \Rightarrow \text{Raw Feature Similarity} \\
&+ k(a,b)\nabla_a \log P_\theta(a) \nabla_b \log P_\theta(b) && \Rightarrow \text{Score Similarity} \\
&\left.\begin{aligned} &+\nabla_a k(a,b)\nabla_b \log P_\theta(b) \\ &+\nabla_b k(a,b)\nabla_a \log P_\theta(a), \end{aligned}\right\} && \Rightarrow \left\{\begin{aligned}&\text{Mutual Influence of} \\ &\text{Prediction Shifts}\end{aligned}\right.
\end{aligned} \tag{6.4}$$

we note it can serve as base of evaluating the difficulty of machine unlearning well given it measures both model augmented data similarity (tie) and hypothetic predictive confidence shifts (as gradient/score).

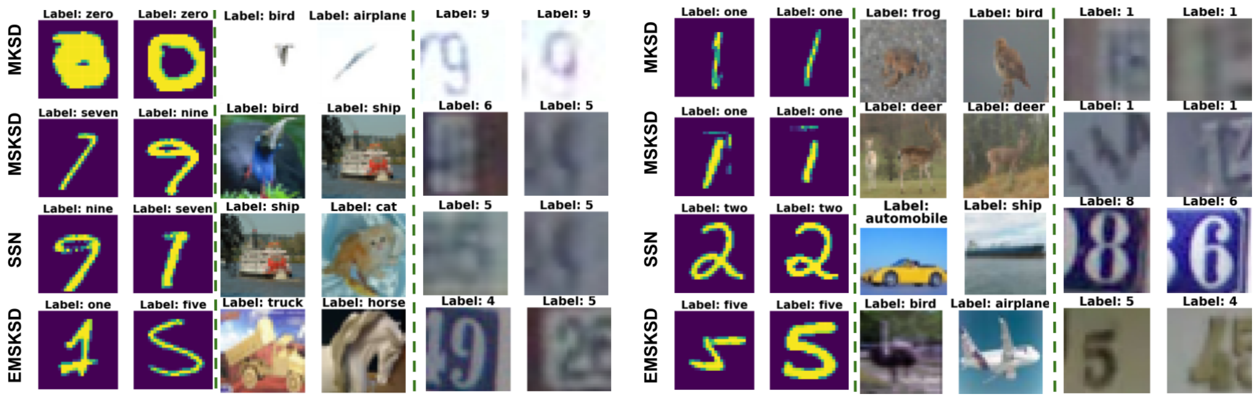


Figure 6.2: Top-2 most easy and hard training samples for machine unlearning flagged by the four proposed unlearning difficulty scoring functions. While the samples flagged vary based on the different scoring function, we note that the easily unlearnable samples are often less representative of the class label, while the hardly unlearnable samples are often typical.

6.3.2 Unlearning Difficulty Scoring with KSD

While the parameterized kernel defined by KSD described has potential to distinguish data points with respect to the difficuty of unlearning, KSD itself does not sufficient to be a scoring function since it is defined on a pair of data points. To corporate Stein Kennel to rank datapoints unlearning difficulty, we need to aggregate pair-wise kernel values for each data point such that

$$\text{Agg}(\{\kappa_\theta((\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j))| \, \forall j \text{ s.t. } (\mathbf{x}_j, y_j) \in D_t\}), \tag{6.5}$$

which requires designing heuristics by carefully examining the distribution of stein kernel values for each data point a.

We analyzed several scoring heuristics to evaluate data points based on their impact on the KSD measure.

- **Marginalized KSD (MKSD):** The MKSD scoring mechanism considers both the immediate proximity of neighboring data points and the extent of strong similarities, as indicated by elevated Stein Kernel values. Consequently, the scoring heuristic sums over the Stein kernel values between a data point and all others in the dataset

$$\text{Agg}_{\text{MKSD}}((\mathbf{x}_i, y_i)) = \Sigma_{j=1}^{n} \kappa_\theta((\mathbf{x}_i, y_i), (\mathbf{x}_j, y_j)) \tag{6.6}$$

A higher Marginalized KSD (MKSD) score suggests greater similarity and a larger “resistance set," implying that more extensive sections of the training data would need to be unlearned alongside the target data point—a scenario that is typically undesirable. To efficiently address this, we rank the training data from the lowest to the highest MKSD, identifying data points that range from easy to difficult to unlearn.

- **Marginalized Standardized KSD (MSKSD):** Intuitively, a data point surrounded by a large number of similar points poses more challenges in unlearning due to its strong connections within the dataset. To effectively highlight such data points, we define the MSKSD which aggregates these standardized exponential values. A higher MSKSD value indicates stronger and more extensive similarities with other data points, suggesting a higher complexity in unlearning these particular instances.

$$\mathrm{Agg}_{\mathrm{MSKSD}}((\mathbf{x}_i, y_i)) = \sum_{j=1}^{n} e^{\hat{\kappa}_\theta((\mathbf{x}_i,y_i),(\mathbf{x}_j,y_j))}, \tag{6.7}$$

  where we denote standardized Stein kernel values as $\hat{\kappa}$. The standardization operation can reduce the influence of magnitude of Stein kernel values in the heuristic; the aggregation will be influenced by the distribution (or skewness) of Stein kernel values.

- **Stein Score Norm (SSN)**: Data points that are proximate to the decision boundary exhibit higher gradient magnitudes, making it easier to adjust the boundary in response to these points compared to those located deeper within each class. We propose that data points with high SSN (see Equation 6.8) are positioned away from the dense centers of their classes, making them prime candidates for unlearning. By identifying and ranking data points with the highest Stein Score Norms, we can effectively pinpoint those that are closer to the decision boundary.

$$\mathrm{Agg}_{\mathrm{SSN}}((\mathbf{x}_i, y_i)) = ||\nabla_\theta \log P_\theta(\mathbf{x}_i, y_i)|| \tag{6.8}$$

- **Entropy Marginalized Standardized KSD (EMSKSD)**: Uncertainty in the predictive confidence of a datapoint can be indication of closeness to decision boundary. Incorporating datapoints predictive uncertainty with the similarity scoring in the following formula:

$$\begin{aligned} \text{Agg}_{\text{EMSKSD}}((\mathbf{x}_i, y_i)) &= \frac{\text{Agg}_{\text{MSKSD}}((\mathbf{x}_i, y_i))}{H(\hat{\mathbf{y}}_i)} \\ H(\hat{\mathbf{y}}_i) &= -\sum_{l=1}^{n} \hat{\mathbf{y}}_{i,l} \log(\hat{\mathbf{y}}_{i,l}) \quad \hat{\mathbf{y}}_{i,l} = f_\theta(\mathbf{x}_i) \end{aligned} \tag{6.9}$$

Here, $H(\hat{\mathbf{y}}_i)$ is the entropy of the predicted probabilities. EMSKSD can obtain points close to the decision boundary with small number of correlated samples.

The scoring heuristics above are not comparable based on our analysis as shown in Figure 6.2; they flag different types of difficulty of unlearning. MKSD tends to flag apparent outliers in the training samples as easily unlearnable training samples, whereas EMSKSD flags less obvious but abnormal training samples, such as truck on paper, fake horse or wrongly labeled numbers.

6.4 Evaluation and Analysis

In this section, we assess the effectiveness of various unlearning approximation methods on data points ranked from easy to hard for unlearning against the scoring heuristics proposed. Specifically, we want to answer the following questions through the experiments:

- Q1: Given easily and hardly unlearnable data points flagged out by the scoring heuristics, do the resulting unleared model show different predictive performance?
- Q2: Which of the proposed scoring heuristics show better predictive alignment with actual unlearning outcome?
- Q3: Which unlearning algorithm show better performance when facing hardly unlearnable data points?

6.4.1 Experimental setups

Datasets and models We conducted our evaluation on three datasets: MNIST, CIFAR-10, and SVHN. Each dataset was classified using a different model: a simple two-layer CNN classifier for the MNIST, and ResNet18 for the CIFAR-10 and SVHN. The following sections provide a detailed analysis of each model's performance on these datasets. Details of datasets and models are presented in Table 6.1.

Dataset	Model	Layers	Batch Size	Number of Classes	Learning Rate	Samples
4 Gaussian Blob	2-layer Neural Network	2	32	4	0.001	2000
MNIST	2-layer CNN	2	150	10	0.001	54000
CIFAR10	ResNet18	18	150	10	0.01	45000
SVHN	ResNet18	18	64	10	0.001	58000

Table 6.1: This table details the datasets and models used in evaluating unlearning algorithms, specifying the models applied to each dataset, including the number of layers, batch sizes, number of classes, learning rates, and sample sizes. The information presented provides insight into the computational frameworks employed to analyze the 4 Gaussian Blob, MNIST, and CIFAR-10 datasets, demonstrating the diversity of approaches used in the study.

Unlearning methods In our experimental setup, we evaluated three primary unlearning baselines in addition to retraining. These baselines include **GradAsc**, **FT**, and **FF**. Additionally, and finally **Retraining** the model without the forgetting set. Although the interest in guaranteed unlearning initially stemmed from the challenges associated with accessing retrained models, achieving guaranteed unlearning for feasible datapoints by retraining presents an intriguing opportunity to explore the characteristics of data points that can be unlearned more readily. If we can achieve guaranteed unlearning for a single data point through retraining, it validates our assumption regarding the varying levels of unlearning feasibility, as it has been shown that generalization ability of ml model [170] prevents unlearning single samples.

Criteria To assess the effectiveness of unlearning, we measured the accuracy and loss of unlearned model. Additionally, we calculated the layer-wise distance between unlearned f_{θ_u} and original f_{θ}. We conducted a Membership Inference Attack (MIA) on the unlearned model (*MIA-Efficacy*) to ascertain how many samples from the forgetting set $\mathcal{D}_f$ were correctly classified as non-training samples. We assessed the MIA-efficacy using a confidence-based attack method [152]. Ideally, post-unlearning, the model θ_u should have effectively "forgotten" the information related to the samples in $\mathcal{D}_f$. Successful unlearning is indicated by the MIA predictor accurately classifying these samples as ones it has not encountered during training.

Training and unlearning setting To ensure consistency in our evaluations, we fixed the hyperparameters for each algorithm. This standardization facilitates the comparison of our scoring heuristic across different conditions, thereby enhancing the reliability of our findings regarding unlearning feasibility. The specific hyperparameters for each algorithm are detailed in Table 6.2.

Dataset	**Algorithm**	**Parameters**
4 Gaussian Blob	Fine Tuning	LR=0.01, Epochs=10
	Gradient Ascent	overfit_threshold=1.0, LR=0.01, Epochs=50
	Fisher Forgetting	alpha=1e-4
	Retraining	LR=0.01, Epochs=50
MNIST	Fine Tuning	LR=0.1, Epochs=10
	Gradient Ascent	overfit_threshold=5.0, LR=1e-4, Epochs=50
	Fisher Forgetting	alpha=1e-5
	Retraining	LR=0.1, Epochs=10
CIFAR10	Fine Tuning	LR=0.01, Epochs=15
	Gradient Ascent	overfit_threshold=5.0, LR=1e-4, Epochs=50
	Fisher Forgetting	alpha=6e-8
	Retraining	LR=0.01, Epochs=100
SVHN	Fine Tuning	LR=0.01, Epochs=15
	Gradient Ascent	overfit_threshold=5.0, LR=1e-4, Epochs=50
	Fisher Forgetting	alpha=6e-8
	Retraining	LR=0.01, Epochs=100

Table 6.2: Reporting the parameters of each unlearning approximation algorithm for each dataset.

6.4.2 Experimental results

Initially, we assess the effectiveness of various unlearning approximation methods on data points ranked from feasible to infeasible for unlearning by the heuristic discussed in 6.3.2. The main objectives of this experimental phase are 1) comparing the predictive performance of the unlearned model for infeasible and feasible data points, 2) examining which of our ranking heuristic 6.3.2 better reflects unlearning feasibility, 3) evaluating the performance of each unlearning approximation algorithm on feasible and infeasible data points, aiming to demonstrate two key points: first, the difference in unlearning outcomes between feasible and infeasible data points, and second, to refute the assumption that unlearning feasibility is uniform across all data points, showing instead that different data points exhibit varying levels of unlearning feasibility. 4) To investigate whether guaranteed unlearning is achievable for feasible data points and if such unlearning can also be accomplished through retraining.

To do this, we first rank the training samples using specific sorting heuristics. Based on their assigned order, we identify the top "Feasible" and "Infeasible" datapoints for unlearning. These

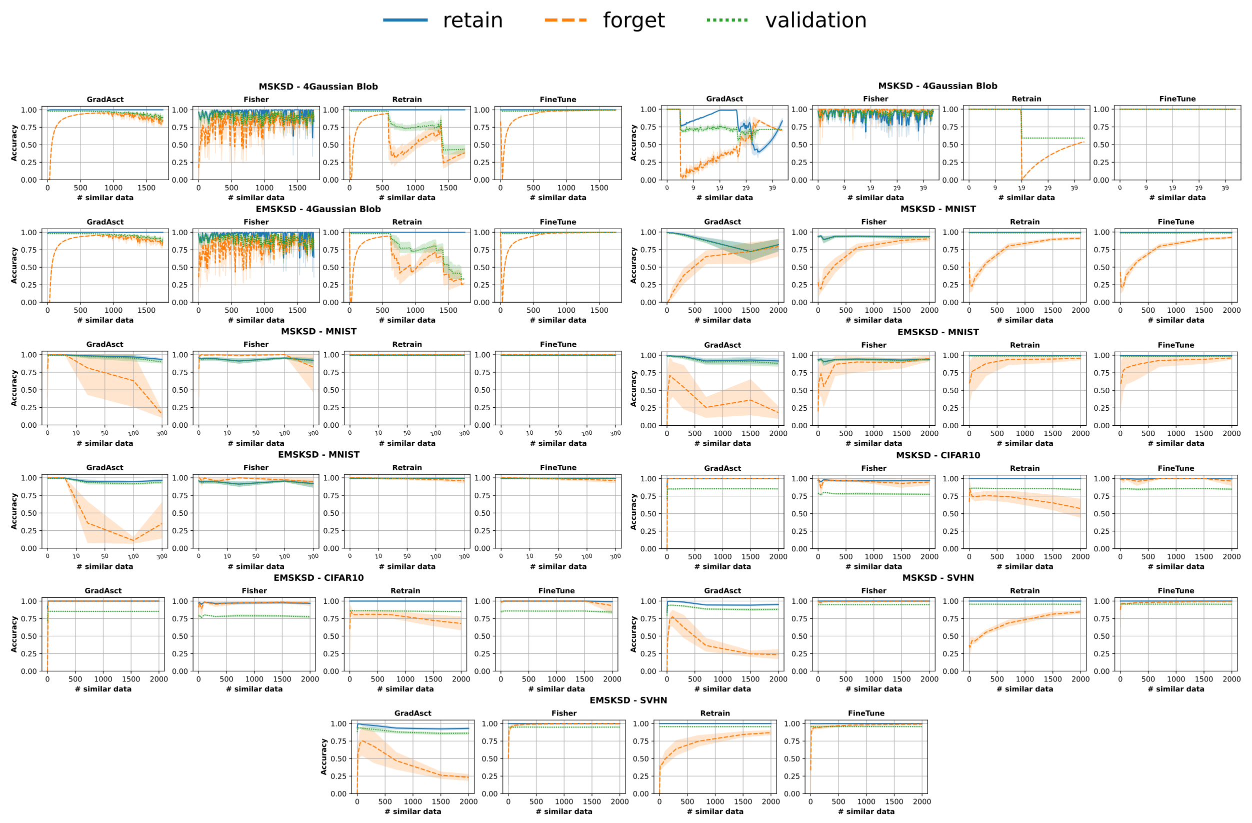


Figure 6.3: Averaged UA, RA, and VA plots for unlearning algorithms on the top 5 feasible and infeasible datapoints recommended by each sorting heuristic described in Section 6.3.2. The plots illustrate various scenarios across different datasets: Blob, MNIST, CIFAR-10, and SVHN, showing how different unlearning algorithms perform under feasible and infeasible conditions.

designations guided the subsequent application of approximation algorithms and retraining strategies to the selected data points. Throughout the experiments, we progressively increased the number of similar samples required to be forgotten alongside the target data points, to investigate the point for obtaining the guaranteed unlearning.

By analyzing Unlearning Accuracy (UA), loss metrics, and Membership Inference Attack (MIA) efficacy, we glean insights into the characteristics that make certain data points more or less amenable to unlearning. These insights help us refine our understanding of why unlearning is straightforward for some data points but challenging for others, thereby enhancing our theoretical and practical approaches to data privacy and machine learning model management.

The presence of similar samples significantly impacts the unlearning performance of FT. These similar samples bolster the generalization capability of the model, complicating the unlearning of specific data points. Sorting heuristics that emphasize the strength of similarity between samples rather than their proximity to the decision boundary are more likely to identify data points that are amenable to guaranteed unlearning 6.3. For example, a comparative analysis between the effectiveness of Marginalized Standardized KSD (MSKSD) and Stein Score Norm (EMSKSD), as illustrated in Figure 6.3, reveals that MSKSD is more adept at pinpointing data points that can be thoroughly unlearned.

The accuracy findings from the 4 Gaussian Blob dataset are consistent with our anticipations regarding the compatibility between the Fine Tuning (FT) approach and the selected sorting algorithms. The top 5 feasible samples ar selected from the cluster with the smallest number of samples. Although the feasible samples are close to the decision boundary (as GA can unlearn them without significant damage to the model's performance), the set of similar samples are incorporating to the predictive performance of FT and Retrain. So, when the segment of remaining data that are strongly similar to the feasible datapoints are set aside, the accuracy of forget set immediately drops. Similar behavior is observed from the MNIST and SVHN.

First, The analysis of the "Feasible" data points recommended by these heuristics, specifically for the MNIST and 4 Gaussian Blob datasets, revealed several key characteristics. The data points often exhibit high predicted error values, suggesting that these points are more challenging to model accurately and, consequently, easier to unlearn. The sorted similar samples centred around the selected datapoint are usually distributed across other classes. For example, in the MNIST dataset, the number of present samples in each class is uniformly distributed and it comprises approximately 0.1 of training samples (n). When attempting to unlearn feasible data points identified by MSKSD and EMSKSD, increasing the number of relevant samples to be unlearned —up to a point that surpasses the general number of samples within that class (i.e., $\|\mathcal{D}_f\| \geq 0.1 \times n$)— results in the remaining dataset $\mathcal{D}_r$ still containing some data points from the original class. Consequently, these residual samples from the original class prevent the model's prediction accuracy for that class from dropping to zero.

To explore the diversity of classes among similar samples associated with a recommended data point, we calculated the Shannon entropy for a subset of these samples. According to the results shown in Table 6.3, data points selected by the EMSKSD method show varied entropy values within their similar sample subsets, indicating a mix of classes. In contrast, data points identified by the MKSD method exhibit the lowest class diversity, suggesting a more homogeneous group of similar samples. For the CIFAR-10 dataset, the recommended data points display a notably higher predicted error, and the similar samples identified by their Stein Kernel values belong to the same

Dataset	EMSKSD	MKSD	MSKSD	SSN
BLOB	0.424 ± 0.006	0.424 ± 0.006	0.420 ± 0.000	0.420 ± 0.000
MNIST	0.427 ± 0.754	0.000 ± 0.000	1.732 ± 0.204	1.092 ± 0.945
CIFAR10	0.008 ± 0.018	0.000 ± 0.000	0.016 ± 0.022	0.016 ± 0.022
SVHN	1.126 ± 0.954	0.000 ± 0.000	1.646 ± 0.386	1.126 ± 0.954

Table 6.3: Class Diversity Measurements for Top Feasible Datapoints: This table shows the class diversity for the top 5 feasible datapoints selected by each heuristic algorithm. For each feasible datapoint, class diversity was counted in the top 50 to 450 most relevant samples sorted based on their Stein Kernel values.

class as the recommended data points. This pattern differs from the MNIST dataset, where the stylistic similarities between different class samples often lead to misclassifications. Unlike MNIST, the classes in CIFAR-10, such as "Car" vs. "Airplane," are distinctly differentiable. Thus, similar samples are typically from the same class. Achieving guaranteed unlearning for CIFAR-10 using Fine Tuning (FT) or retraining would likely require excluding the entire class to effectively remove the targeted data point's influence.

On the other hand, analysis infeasible unlearning revealed that, (**i**) In the 4-Gaussian Blob dataset, the application of Grad Ascent significantly deteriorated the model's performance. (**ii**) Effective unlearning in the MNIST dataset required removing a substantial portion of pertinent data. (**iii**) In CIFAR10, while individual data points could be unlearned, this often severely impaired the model's predictive accuracy. The result demonstrate unlearning infeasible data points is not consistently achievable across datasets. The most drastic measure to eliminate their influence entirely involves unlearning the entire class label associated with these points. Conversely, feasible data points can be effectively removed from the model, achieving guaranteed unlearning. These insights underscore the complexities involved in effectively unlearning specific data points, especially when they are embedded within intricate networks of class relationships and high variability in error prediction. As we further our understanding of these dynamics, it becomes increasingly clear that successful unlearning requires not only sophisticated algorithmic approaches but also a deep understanding of the dataset characteristics and the interaction between data points and model architecture.

6.4.3 Membership Inference Attack Evaluation

We examined the effectiveness of unlearning algorithms on data points deemed infeasible by various sorting heuristics. Each dataset presented unique challenges in addressing these infeasible points,

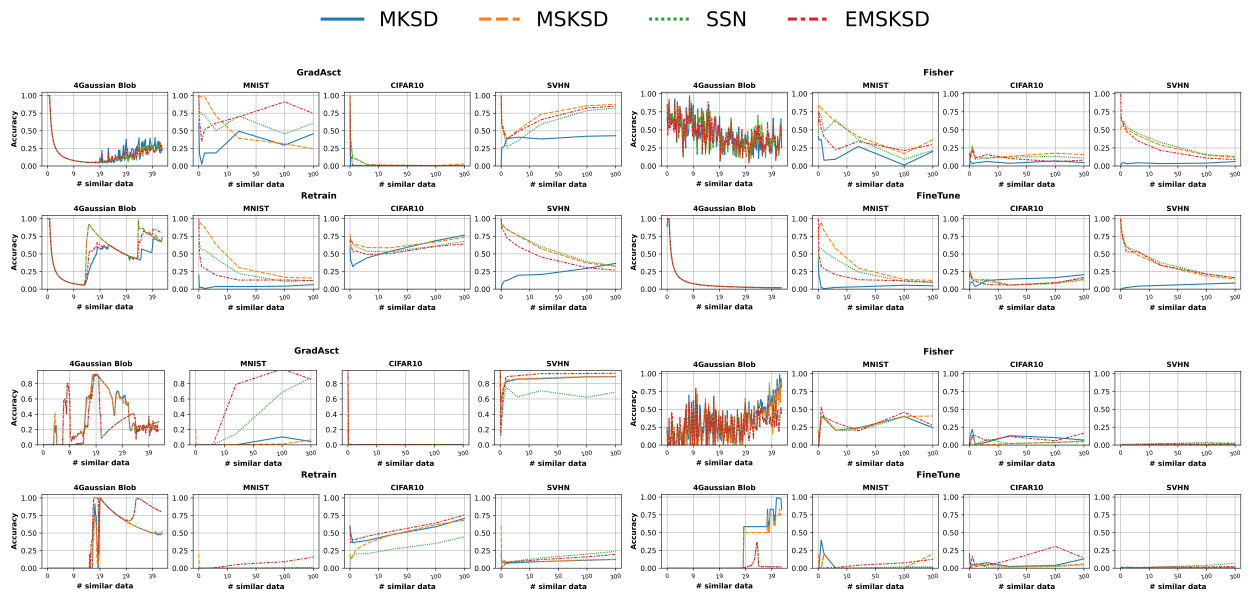


Figure 6.4: Comparison of MIA-Efficacy for feasible and infeasible unlearning across different unlearning approximation methods. The top row presents the efficacy for feasible unlearning methods: Gradient Ascent, Fisher, Retrain, and Fine-Tune. The bottom row presents the efficacy for the same methods under infeasible unlearning conditions. Each plot provides insight into the performance and reliability of the unlearning methods in different scenarios. The legend details the specifics for each method.

underscoring the complexity inherent in the unlearning process. For example, in the 4-Gaussian Blob dataset, unlearning infeasible data points required omitting a substantial portion of related data, which adversely affected both Retraining Accuracy (RA) and Validation Accuracy (VA).

In the case of the MNIST dataset, unlearning infeasible data points either had a negligible impact on Unlearning Accuracy (UA), RA, and VA or required the exclusion of a significant number of related data points. However, the Membership Inference Attack (MIA) efficacy, as depicted in Figure (6.4), indicated that the unlearning efforts were generally unsuccessful, failing to achieve high scores. GradAscent proved to be the most effective algorithm for tackling infeasible data points, yet it still struggled to completely dissociate the influence of these points even when a large number of similar data points were also unlearned. This approach either compromised the model's performance or failed to improve the unlearning metrics (UA, RA, and VA), as illustrated in Figures (6.3), and MIA-efficacy (6.4).

Chapter 7

Conclusion and Future Research

7.1 Conclusion

In this work, we have discussed eXplainable AI (XAI) for non-structured data with a specific focus on text and graph data. Based on the study of previous methods, we began our research with two important questions:

- How to design an explanation algorithm for text classifiers that can: 1) produce the local explanations for varieties of text classifiers, 2) consider the structure of data for designing the explainer, 3) exploit the white box methods or lexical resources to generate the local neighbourhood.
- How to design a GNN explanation algorithm that can: 1) produce different perspectives of explanations, 2) propagate the perturbation to each layer of GNN, 3) control the perturbation magnitude to generate all types of explanations, and 4) keep the generated explanation as close as possible to the original data manifold.

To design a text model explainer with the mentioned characteristics we began our research by designing FEHAN, A modularized Framework for Explaining Hierarchical Attention Network. FEHAN attempted to locally explain the behaviour of the Hierarchical Attention Network (HAN). FEHAN distinguished the Informative Sentences (IS) in a given document using the attention parameters of HAN. Then it generated a set of synthetic documents by replacing the IS with artificial sentences sampled from a probabilistic model that is trained on the original dataset. The new set of synthetic documents was exploited to train an interpretable model - a decision tree - from which the important words can be extracted to construct a saliency map explaining the class label for a given document. The qualitative and quantitative evaluation of FEHAN proved the superiority of obtained explanations by FEHAN with respect to the baseline algorithm. Additionally, evaluating the generated neighbourhood examples indicated that the FEHAN not only preserved the essence of the original document but also enriched it with semantically similar sentences. This feature of our model is significantly strengthened for smaller datasets when the random elimination of words can result in invalid examples.

In the next step, to overcome the limitations of FEHAN for being a model-dependent explainer, we introduced DICTA, a model-agnostic explainer for black box text classifiers. DICTA explained a black box model's behaviour by evaluating the impact of words and their semantic replacement on the class distribution of a document. In this way, each word's value was made explicit, and replacing it with its semantic replacements such as synonyms and antonyms, allowed us to check how it is possible to positively or negatively change the class label of a given document. Hence, the explanations provided by DICTA were more expressive and understandable w.r.t the baseline models and we were allowed to apply it to the varieties of text classifiers. Evaluation of DICTA's synthetic neighbourhoods indicates that DICTA generated semantically similar examples and used them to train the surrogate model.

Due to the fast development of the graph-structured domain and its application in real-life problems, we also focused on the XAI for graph data. We presented a perturbation-based method to provide multiple types explanations for the prediction of GCN model. Our approach perturbs the input graph and attempts to train the perturbation parameters using targeted loss functions. The main goal is to provide users with different explanation perspectives. For this purpose, we designed an algorithm that enables us to generate Counterfactual (CF), Influential Prototypes (PT), and Exemplars (EXE). CF is considered as a subgraph of the input where the minimum set of edges, whose absence changes the prediction of the model, are eliminated; PT is considered as a subgraph of the input with the same label of the graph being explained including only the most important connections responsible for the prediction; and EXE is supposed to be a subgraph of the input with the same label of the graph being explained which is very similar to the input and have common connections responsible for the classification. The most significant aspect of our explanation method is the design of the loss function. We propose a loss function enabling the customization of the type of explanation, the control of the magnitude of modifications to the input (number of removed edges from the graph), and the possibility of keeping the generated explanation close to the data manifold. The extensive experimental evaluation of our method on three common node classification datasets w.r.t. different qualitative criteria such as Fidelity, Sparsity, Closeness, Betweenness and Stability proved our method's outstanding performance in generating different explanations w.r.t. the baselines in terms of evaluative criteria.

Our HD-Explain presents a Kernel Stein Discrepancy-driven example-based prediction explanation method We performed comprehensive qualitative and quantitative evaluation comparing three baseline explanation methods using three datasets. The results demonstrated the efficacy of HD-Explain in generating explanations that are accurate and effective in terms of their granularity level. In addition, compared to other methods, HD-Explain is flexible to apply on any layer of interest and can be used to analyze the evolution of a prediction across layers. HD-Explain serves as an important contribution towards improving the transparency of machine learning models. The development of HD-Explain, a highly precise and data-centric explanation method for neural classifiers, promises to significantly enhance the transparency and trustworthiness of machine learning models across various applications. Furthermore, HD-Explain's scalable and computationally efficient approach makes it feasible for deployment in large-scale, real-world applications. This not only promotes transparency and accountability in AI systems but also paves the way for broader acceptance and integration of AI technologies in society. By bridging the gap between complex model behavior and human understanding, HD-Explain fosters a more informed and trust-based relationship between AI systems and their users. Overall, HD-Explain's contributions to model interpretability and transparency have the potential to drive significant advancements in the responsible and ethical use of AI, ensuring that these technologies are developed and deployed in ways that are understandable, accountable, and aligned with societal values.

7.2 Future research direction

In the past three years, the paradigm of ML research has dramatically shifted from locally trained models to leveraging pre-trained foundation models, whether in computer vision or large language models. As a result, the focus has shifted from local model training to accessible pre-trained models. To explain the behavior of these models, we must adopt a data-centric approach. Data-centric explanation methods are crucial since traditional parameter-based explanation techniques are outdated and inadequate for these large black-box models, particularly state-space models. Data is the primary resource for understanding these complex models, and it must be utilized to its full potential.

Drawing inspiration from a scenario where an observer explains a person's behavior by investigating their behavioral traits and creating a behavior profile, we can apply a similar approach to foundation models. An observer can query the model with variations of given data points and observe the variations in the model's responses. This method can help create a detailed profile of the model's behavior and decision-making process.

Data-Centric Explanation for Large Language Models

Large language models (LLMs) have transformed natural language processing, but their black-box nature presents significant interpretability challenges. Future research should focus on developing data-centric methods to explain LLMs' behavior. This involves analyzing input-output relationships, utilizing attention mechanisms, and examining the impact of training data on model predictions. By studying how these models process data, researchers can gain insights into their decision-making processes.

Data-Centric Explanation for Computer Vision Models

Foundation models in computer vision, such as convolutional neural networks (CNNs) and vision transformers (ViTs), have achieved remarkable success but remain challenging to interpret. Future research should develop data-centric explanation techniques to elucidate how these models process visual information. This could involve studying the influence of specific training images on model predictions, using saliency maps to identify important features, and conducting dataset-level analyses to understand model biases and weaknesses. A data-centric approach can provide a clearer understanding of computer vision models' operations and decision-making processes.

Data-Centric Explanation for State-Space Models

State-space models, commonly used in time series analysis and control systems, present unique interpretability challenges due to their dynamic and complex structure. Traditional parameter-based explanations are insufficient, necessitating a data-centric approach. Future research should explore methods to explain state-space models by analyzing the data they process and the states they transition through. This could include examining the influence of specific data points on state transitions, understanding the role of historical data in shaping model predictions, and developing visualization techniques to illustrate state dynamics. By focusing on the data aspects of state-space models, researchers can better interpret their behavior and improve their transparency.

In summary, the shift towards pre-trained foundation models in ML necessitates a paradigm shift in how we approach model interpretability. Data-centric explanations offer a promising avenue for understanding these complex systems, and future research should prioritize the development and application of such techniques across various domains, including large language models, computer vision models, and state-space models.

Bibliography

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318, 2016.

[2] Mohiuddin Ahmed and AKM Najmul Islam. Deep learning: hope or hype. *Annals of Data Science*, pages 1–6, 2020.

[3] David Alvarez Melis and Tommi Jaakkola. Towards robust interpretability with self-explaining neural networks. *Advances in neural information processing systems*, 31, 2018.

[4] Ariful Islam Anik and Andrea Bunt. Data-centric explanations: explaining training data of machine learning systems to promote transparency. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2021.

[5] Shahin Atakishiyev, Mohammad Salameh, Hengshuai Yao, and Randy Goebel. Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for future research directions. *arXiv preprint arXiv:2112.11561*, 2021.

[6] Pepa Atanasova, Jakob Grue Simonsen, Christina Lioma, and Isabelle Augenstein. A diagnostic study of explainability techniques for text classification. *arXiv preprint arXiv:2009.13295*, 2020.

[7] Steve Azzolin, Antonio Longa, Pietro Barbiero, Pietro Liò, and Andrea Passerini. Global explainability of gnns via logic combination of learned concepts. *arXiv preprint arXiv:2210.07147*, 2022.

[8] Davide Bacciu, Federico Errica, and Alessio Micheli. Contextual graph markov model: A deep and generative approach to graph processing. In *International Conference on Machine Learning*, pages 294–303. PMLR, 2018.

[9] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.

[10] Mohit Bajaj, Lingyang Chu, Zi Yu Xue, Jian Pei, Lanjun Wang, Peter Cho-Ho Lam, and Yong Zhang. Robust counterfactual explanations on graph neural networks. *Advances in Neural Information Processing Systems*, 34:5644–5655, 2021.

[11] Alessandro Barp, Francois-Xavier Briol, Andrew Duncan, Mark Girolami, and Lester Mackey. Minimum stein discrepancy estimators. *Advances in Neural Information Processing Systems*, 32, 2019.

[12] Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. Relatif: Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics*, pages 1899–1909. PMLR, 2020.

[13] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[14] Lucas Bourtoule, Varun Chandrasekaran, Christopher A Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 141–159. IEEE, 2021.

[15] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.

[16] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.

[17] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.

[18] Nadia Burkart and Marco F Huber. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research*, 70:245–317, 2021.

[19] Ruth MJ Byrne. Counterfactuals in explainable artificial intelligence (xai): Evidence from human reasoning. In *IJCAI*, pages 6276–6282, 2019.

[20] Carrie J Cai, Jonas Jongejan, and Jess Holbrook. The effects of example-based explanations in a machine learning interface. In *Proceedings of the 24th international conference on intelligent user interfaces*, pages 258–262, 2019.

[21] Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015 IEEE symposium on security and privacy*, pages 463–480. IEEE, 2015.

[22] Tianshi Che, Yang Zhou, Zijie Zhang, Lingjuan Lyu, Ji Liu, Da Yan, Dejing Dou, and Jun Huan. Fast federated machine unlearning with nonlinear functional theory. In *International conference on machine learning*, pages 4241–4268. PMLR, 2023.

[23] Min Chen, Weizhuo Gao, Gaoyang Liu, Kai Peng, and Chen Wang. Boundary unlearning: Rapid forgetting of deep networks via shifting the decision boundary. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7766–7775, 2023.

[24] Min Chen, Zhikun Zhang, Tianhao Wang, Michael Backes, Mathias Humbert, and Yang Zhang. When machine unlearning jeopardizes privacy. In *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, pages 896–911, 2021.

[25] Yuantao Chen, Jie Xiong, Weihong Xu, and Jingwen Zuo. A novel online incremental and decremental learning algorithm based on variable support vector machine. *Cluster Computing*, 22:7435–7445, 2019.

[26] Yuanyuan Chen, Boyang Li, Han Yu, Pengcheng Wu, and Chunyan Miao. Hydra: Hypergradient data relevance analysis for interpreting deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 7081–7089, 2021.

[27] Rishav Chourasia and Neil Shah. Forget unlearning: Towards true data-deletion in machine learning. In *International Conference on Machine Learning*, pages 6028–6073. PMLR, 2023.

[28] Vikram S Chundawat, Ayush K Tarun, Murari Mandal, and Mohan Kankanhalli. Can bad teaching induce forgetting? unlearning in deep networks using an incompetent teacher. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7210–7217, 2023.

[29] Kacper Chwialkowski, Heiko Strathmann, and Arthur Gretton. A kernel test of goodness of fit. In *International conference on machine learning*, pages 2606–2615. PMLR, 2016.

[30] Asim Kumar Debnath, Rosa L Lopez de Compadre, Gargi Debnath, Alan J Shusterman, and Corwin Hansch. Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity. *Journal of medicinal chemistry*, 34(2):786–797, 1991.

[31] Amit Dhurandhar, Pin-Yu Chen, Ronny Luss, Chun-Chen Tu, Paishun Ting, Karthikeyan Shanmugam, and Payel Das. Explanations based on the missing: Towards contrastive explanations with pertinent negatives. *Advances in neural information processing systems*, 31, 2018.

[32] Jimmy Z Di, Jack Douglas, Jayadev Acharya, Gautam Kamath, and Ayush Sekhari. Hidden poison: Machine unlearning enables camouflaged poisoning attacks. In *NeurIPS ML Safety Workshop*, 2022.

[33] Yinpeng Dong, Hang Su, Jun Zhu, and Bo Zhang. Improving interpretability of deep neural networks with semantic information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4306–4314, 2017.

[34] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[35] Alexandre Duval and Fragkiskos D Malliaros. Graphsvx: Shapley value explanations for graph neural networks. In *Machine Learning and Knowledge Discovery in Databases. Research Track: European Conference, ECML PKDD 2021, Bilbao, Spain, September 13–17, 2021, Proceedings, Part II 21*, pages 302–318. Springer, 2021.

[36] EP EC. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)(text with eea relevance). eli, 2016.

[37] Radwa Elshawi, Mouaz H Al-Mallah, and Sherif Sakr. On the interpretability of machine learning-based model for predicting hypertension. *BMC medical informatics and decision making*, 19(1):1–32, 2019.

[38] Hugo Jair Escalante, Sergio Escalera, Isabelle Guyon, Xavier Baró, Yağmur Güçlütürk, Umut Güçlü, and Marcel Van Gerven. *Explainable and interpretable models in computer vision and machine learning*. Springer, 2018.

[39] Patrick Esser, Robin Rombach, and Bjorn Ommer. A disentangling invertible interpretation network for explaining latent representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9223–9232, 2020.

[40] Evan N Feinberg, Debnil Sur, Zhenqin Wu, Brooke E Husic, Huanghao Mai, Yang Li, Saisai Sun, Jianyi Yang, Bharath Ramsundar, and Vijay S Pande. Potentialnet for molecular property prediction. *ACS central science*, 4(11):1520–1530, 2018.

[41] John Rupert Firth. *Selected papers of JR Firth, 1952-59*. Indiana University Press, 1968.

[42] Alex Fout, Jonathon Byrd, Basir Shariat, and Asa Ben-Hur. Protein interface prediction using graph convolutional networks. *Advances in neural information processing systems*, 30, 2017.

[43] Yoav Freund and Llew Mason. The alternating decision tree learning algorithm. In *icml*, volume 99, pages 124–133. Citeseer, 1999.

[44] Marcello Frixione and Antonio Lieto. Exemplars, prototypes and conceptual spaces. In *Biologically Inspired Cognitive Architectures 2012*, pages 131–136. Springer, 2013.

[45] Mary Gick and Stan Matwin. The importance of causal structure and facts in evaluating explanations. In *Machine Learning Proceedings 1991*, pages 51–54. Elsevier, 1991.

[46] Aditya Golatkar, Alessandro Achille, Avinash Ravichandran, Marzia Polito, and Stefano Soatto. Mixed-privacy forgetting in deep networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 792–801, 2021.

[47] Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Eternal sunshine of the spotless net: Selective forgetting in deep networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9304–9312, 2020.

[48] Jackson Gorham, Anant Raj, and Lester Mackey. Stochastic stein discrepancies. *Advances in Neural Information Processing Systems*, 33:17931–17942, 2020.

[49] Laura Graves, Vineel Nagisetty, and Vijay Ganesh. Amnesiac machine learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11516–11524, 2021.

[50] Christopher Grimsley, Elijah Mayfield, and Julia R.S. Bursten. Why attention is not explanation: Surgical intervention and causal reasoning about neural models. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 1780–1790, Marseille, France, May 2020. European Language Resources Association.

[51] R Guidotti, A Monreale, and L Cariaggi. Investigating neighborhood generation for explanations of image classifiers. PAKDD, 2019.

[52] Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55, 2022.

[53] Riccardo Guidotti and Anna Monreale. Data-agnostic local neighborhood generation. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 1040–1045. IEEE, 2020.

[54] Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. Factual and counterfactual explanations for black box decision making. *IEEE Intelligent Systems*, 2019.

[55] Riccardo Guidotti, Anna Monreale, Stan Matwin, and Dino Pedreschi. Black box explanation by learning image exemplars in the latent feature space. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 189–205. Springer, 2019.

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

[57] Riccardo Guidotti, Anna Monreale, Francesco Spinnato, Dino Pedreschi, and Fosca Giannotti. Explaining any time series classifier. In *2020 IEEE Second International Conference on Cognitive Machine Intelligence (CogMI)*, pages 167–176. IEEE, 2020.

[58] Riccardo Guidotti, Jacopo Soldani, Davide Neri, Antonio Brogi, and Dino Pedreschi. Helping your docker images to spread based on explainable models. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 205–221. Springer, 2018.

[59] Chuan Guo, Tom Goldstein, Awni Hannun, and Laurens Van Der Maaten. Certified data removal from machine learning models. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3832–3842. PMLR, 13–18 Jul 2020.

[60] Varun Gupta, Christopher Jung, Seth Neel, Aaron Roth, Saeed Sharifi-Malvajerdi, and Chris Waites. Adaptive machine unlearning. *Advances in Neural Information Processing Systems*, 34:16319–16330, 2021.

[61] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[62] Robert R Hoffman and Gary Klein. Explaining explanation, part 1: theoretical foundations. *IEEE Intelligent Systems*, 32(3):68–73, 2017.

[63] Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. Hierarchical graph convolutional networks for semi-supervised node classification. *arXiv preprint arXiv:1902.06667*, 2019.

[64] Qiang Huang, Makoto Yamada, Yuan Tian, Dinesh Singh, and Yi Chang. Graphlime: Local interpretable model explanations for graph neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[65] Clayton Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 8, 2014.

[66] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[67] Zachary Izzo, Mary Anne Smart, Kamalika Chaudhuri, and James Zou. Approximate data deletion from machine learning models. In *International Conference on Artificial Intelligence and Statistics*, pages 2008–2016. PMLR, 2021.

[68] Matthew Jagielski, Om Thakkar, Florian Tramer, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Guha Thakurta, Nicolas Papernot, and Chiyuan Zhang. Measuring forgetting of memorized training examples. In *The Eleventh International Conference on Learning Representations*, 2023.

[69] Weiwei Jiang and Jiayun Luo. Graph neural network for traffic forecasting: A survey. *Expert Systems with Applications*, 207:117921, 2022.

[70] Xiaodong Jiang, Ronghang Zhu, Sheng Li, and Pengsheng Ji. Co-embedding of nodes and edges with graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[71] José Jiménez-Luna, Francesca Grisoni, and Gisbert Schneider. Drug discovery with explainable artificial intelligence. *Nature Machine Intelligence*, 2(10):573–584, 2020.

[72] Koganti Krishna Jyothi, Subba Reddy Borra, Koganti Srilakshmi, Praveen Kumar Balachandran, Ganesh Prasad Reddy, Ilhami Colak, C Dhanamjayulu, Ravikumar Chinthaginjala, and Baseem Khan. A novel optimized neural network model for cyber attack detection using enhanced whale optimization algorithm. *Scientific Reports*, 14(1):5590, 2024.

[73] Antonis Kakas and Loizos Michael. Abduction and argumentation for explainable machine learning: A position survey. *arXiv preprint arXiv:2010.12896*, 2020.

[74] Jaykumar Kakkad, Jaspal Jannu, Kartik Sharma, Charu Aggarwal, and Sourav Medya. A survey on explainability of graph neural networks. *arXiv preprint arXiv:2306.01958*, 2023.

[75] Sudheesh Kumar Kattumannil. On stein's identity and its applications. *Statistics & probability letters*, 79(12):1444–1449, 2009.

[76] Jeroen Kazius, Ross McGuire, and Roberta Bursi. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of medicinal chemistry*, 48(1):312–320, 2005.

[77] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11–20, 2019.

[78] Siwon Kim, Jihun Yi, Eunji Kim, and Sungroh Yoon. Interpretation of nlp models through input marginalization. *arXiv preprint arXiv:2010.13984*, 2020.

[79] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[80] Gary King and Langche Zeng. Logistic regression in rare events data. *Political analysis*, 9(2):137–163, 2001.

[81] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[82] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pages 1885–1894. PMLR, 2017.

[83] Ron Kohavi, George H John, et al. Wrappers for feature subset selection. *Artificial intelligence*, 97(1-2):273–324, 1997.

[84] Anna Korba, Pierre-Cyril Aubin-Frankowski, Szymon Majewski, and Pierre Ablin. Kernel stein discrepancy descent. In *International Conference on Machine Learning*, pages 5719–5730. PMLR, 2021.

[85] Tatsuya Kubokawa. Stein's identities and the related topics: an instructive explanation on shrinkage, characterization, normal approximation and goodness-of-fit. *Japanese Journal of Statistics and Data Science*, pages 1–45, 2024.

[86] Orestis Panagiotis Lampridis, Riccardo Guidotti, Salvatore Ruggieri, and Grigorios Tsoumakas. Explaining sentiment prediction by generating exemplars in the latent space. *Undergraduate thesis, Aristotle University of Thessaloniki, School of Informatics*, 2019.

[87] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. Graph classification using structural attention. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1666–1674, 2018.

[88] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.

[89] Qiaomei Li, Rachel Cummings, and Yonatan Mintz. Optimal local explainer aggregation for interpretable prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 12000–12007, 2022.

[90] Xiao-Hui Li, Caleb Chen Cao, Yuhan Shi, Wei Bai, Han Gao, Luyu Qiu, Cong Wang, Yuanyuan Gao, Shenjia Zhang, Xun Xue, et al. A survey of data-driven and knowledge-aware explainable ai. *IEEE Transactions on Knowledge and Data Engineering*, 34(1):29–49, 2020.

[91] Yuantong Li, Chi-Hua Wang, and Guang Cheng. Online forgetting process for linear regression models. In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 217–225. PMLR, 13–15 Apr 2021.

[92] Yu Liang, Siguang Li, Chungang Yan, Maozhen Li, and Changjun Jiang. Explaining the black-box model: A survey of local interpretation methods for deep neural networks. *Neurocomputing*, 419:168–182, 2021.

[93] Q Vera Liao, Daniel Gruen, and Sarah Miller. Questioning the ai: informing design practices for explainable ai user experiences. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–15, 2020.

[94] Brian Y Lim, Qian Yang, Ashraf M Abdul, and Danding Wang. Why these explanations? selecting intelligibility types for explanation goals. In *IUI Workshops*, 2019.

[95] Stan Lipovetsky and Michael Conklin. Analysis of regression in game theory approach. *Applied Stochastic Models in Business and Industry*, 17(4):319–330, 2001.

[96] Hengzhu Liu, Ping Xiong, Tianqing Zhu, and Philip S Yu. A survey on machine unlearning: Techniques and new emerged privacy risks. *arXiv preprint arXiv:2406.06186*, 2024.

[97] Hui Liu, Qingyu Yin, and William Yang Wang. Towards explainable NLP: A generative explanation framework for text classification. In Anna Korhonen, David R. Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 5570–5581. Association for Computational Linguistics, 2019.

[98] Jiaqi Liu, Jian Lou, Zhan Qin, and Kui Ren. Certified minimax unlearning with generalization rates and deletion capacity. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[99] Mengda Liu, Guibo Luo, and Yuesheng Zhu. Machine unlearning with affine hyperplane shifting and maintaining for image classification. In *International Conference on Neural Information Processing*, pages 215–227. Springer, 2023.

[100] Qiang Liu, Jason Lee, and Michael Jordan. A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284. PMLR, 2016.

[101] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

[102] Ana Lucic, Harrie Oosterhuis, Hinda Haned, and Maarten de Rijke. Focus: Flexible optimizable counterfactual explanations for tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5313–5322, 2022.

[103] Ana Lucic, Maartje A Ter Hoeve, Gabriele Tolomei, Maarten De Rijke, and Fabrizio Silvestri. Cf-gnnexplainer: Counterfactual explanations for graph neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4499–4511. PMLR, 2022.

[104] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[105] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized explainer for graph neural network. *Advances in neural information processing systems*, 33:19620–19631, 2020.

[106] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[107] Jing Ma, Ruocheng Guo, Saumitra Mishra, Aidong Zhang, and Jundong Li. Clear: Generative counterfactual explanations on graphs. *arXiv preprint arXiv:2210.08443*, 2022.

[108] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[109] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[110] Neil G Marchant, Benjamin IP Rubinstein, and Scott Alfeld. Hard to forget: Poisoning attacks on certified machine unlearning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7691–7700, 2022.

[111] Ines Filipa Martins, Ana L Teixeira, Luis Pinheiro, and Andre O Falcao. A bayesian approach to in silico blood-brain barrier penetration modeling. *Journal of chemical information and modeling*, 52(6):1686–1697, 2012.

[112] Kyrylo Medianovskyi and Ahti-Veikko Pietarinen. On explainable ai and abductive inference. *Philosophies*, 7(2):35, 2022.

[113] Ronak Mehta, Sourav Pal, Vikas Singh, and Sathya N Ravi. Deep unlearning via randomized conditionally independent hessians. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10422–10431, 2022.

[114] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[115] George A Miller. *WordNet: An electronic lexical database*. MIT press, 1998.

[116] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1–38, 2019.

[117] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[118] Chang S Nam, Jae-Yoon Jung, and Sangwon Lee. *Human-Centered Artificial Intelligence: Research and Applications*. Academic Press, 2022.

[119] Hieu T Nguyen and Arnold Smeulders. Active learning using pre-clustering. In *Proceedings of the twenty-first international conference on Machine learning*, page 79, 2004.

[120] Thanh Tam Nguyen, Thanh Trung Huynh, Phi Le Nguyen, Alan Wee-Chung Liew, Hongzhi Yin, and Quoc Viet Hung Nguyen. A survey of machine unlearning. *arXiv preprint arXiv:2209.02299*, 2022.

[121] Danilo Numeroso and Davide Bacciu. Explaining deep graph networks with molecular counterfactuals. *arXiv preprint arXiv:2011.05134*, 2020.

[122] Sixun Ouyang, Aonghus Lawlor, Felipe Costa, and Peter Dolog. Improving explainable recommendations with synthetic reviews. *arXiv preprint arXiv:1807.06978*, 2018.

[123] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5363–5370, 2020.

[124] Namyong Park, Andrey Kan, Xin Luna Dong, Tong Zhao, and Christos Faloutsos. Estimating node importance in knowledge graphs using graph neural networks. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 596–606, 2019.

[125] Martin Pawelczyk, Jimmy Z Di, Yiwei Lu, Gautam Kamath, Ayush Sekhari, and Seth Neel. Machine unlearning fails to remove data poisoning attacks. *arXiv preprint arXiv:2406.17216*, 2024.

[126] Alexandra Peste, Dan Alistarh, and Christoph H Lampert. Ssse: Efficiently erasing samples from trained machine learning models. *arXiv preprint arXiv:2107.03860*, 2021.

[127] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[128] Vitali Petsiuk, Rajiv Jain, Varun Manjunatha, Vlad I Morariu, Ashutosh Mehra, Vicente Ordonez, and Kate Saenko. Black-box explanation of object detectors via saliency maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11443–11452, 2021.

[129] Pedro O Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1713–1721, 2015.

[130] Douglas EV Pires, Tom L Blundell, and David B Ascher. pkcsm: predicting small-molecule pharmacokinetic and toxicity properties using graph-based signatures. *Journal of medicinal chemistry*, 58(9):4066–4072, 2015.

[131] Phillip E Pope, Soheil Kolouri, Mohammad Rostami, Charles E Martin, and Heiko Hoffmann. Explainability methods for graph convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10772–10781, 2019.

[132] Karl Popper. *Conjectures and refutations: The growth of scientific knowledge*. routledge, 2014.

[133] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33:19920–19930, 2020.

[134] A. Rane and A. Kumar. Sentiment classification system of twitter data for us airline service analysis. In *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, volume 01, pages 769–773, 2018.

[135] Patrick Reiser, Marlen Neubert, André Eberhard, Luca Torresi, Chen Zhou, Chen Shao, Houssam Metni, Clint van Hoesel, Henrik Schopmans, Timo Sommer, et al. Graph neural networks for materials science and chemistry. *Communications Materials*, 3(1):93, 2022.

[136] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[137] Kaspar Riesen and Horst Bunke. Iam graph database repository for graph based pattern recognition and machine learning. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, SSPR & SPR 2008, Orlando, USA, December 4-6, 2008. Proceedings*, pages 287–297. Springer, 2008.

[138] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. *arXiv preprint arXiv:1907.10903*, 2019.

[139] Mahtab Sarvmaili, Riccardo Guidotti, Anna Monreale, Amilcar Soares, Zahra Sadeghi, Fosca Giannotti, Dino Pedreschi, and Stan Matwin. A Modularized Framework for Explaining Black Box Classifiers for Text Data. *Proceedings of the Canadian Conference on Artificial Intelligence*, may 27 2022. https://caiac.pubpub.org/pub/71c292m6.

[140] Mahtab Sarvmaili, Amilcar Soares, Riccardo Guidotti, Anna Monreale, Fosca Giannotti, Dino Pedreschi, and Stan Matwin. A modularized framework for explaining hierarchical attention networks on text classifiers. *Proceedings of the Canadian Conference on Artificial Intelligence*, jun 8 2021. https://caiac.pubpub.org/pub/zzjy8kzu.

[141] Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8179–8186, 2022.

[142] Michael Sejr Schlichtkrull, Nicola De Cao, and Ivan Titov. Interpreting graph neural networks for nlp with differentiable edge masking. *arXiv preprint arXiv:2010.00577*, 2020.

[143] Ayush Sekhari, Jayadev Acharya, Gautam Kamath, and Ananda Theertha Suresh. Remember what you want to forget: Algorithms for machine unlearning. *Advances in Neural Information Processing Systems*, 34:18075–18086, 2021.

[144] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[145] Neel Shah, Ahmed Arshad, Monty B Mazer, Christopher L Carroll, Steven L Shein, and Kenneth E Remy. The use of machine learning and artificial intelligence within pediatric critical care. *Pediatric research*, 93(2):405–412, 2023.

[146] Thanveer Shaik, Xiaohui Tao, Haoran Xie, Lin Li, Xiaofeng Zhu, and Qing Li. Exploring the landscape of machine unlearning: A survey and taxonomy. *arXiv preprint arXiv:2305.06360*, 2023.

[147] Takashi Shibata, Go Irie, Daiki Ikami, and Yu Mitsuzumi. Learning with selective forgetting. In *IJCAI*, volume 3, page 4, 2021.

[148] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *arXiv preprint arXiv:1704.02685*, 2017.

[149] Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.

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

[151] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[152] Liwei Song, Reza Shokri, and Prateek Mittal. Privacy risks of securing machine learning models against adversarial examples. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 241–257, 2019.

[153] Suraj Srinivas, Akshayvarun Subramanya, and R Venkatesh Babu. Training sparse neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 138–145, 2017.

[154] Yi Sui, Ga Wu, and Scott Sanner. Representer point selection via local jacobian expansion for post-hoc classifier explanation of deep neural networks and ensemble models. *Advances in neural information processing systems*, 34:23347–23358, 2021.

[155] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Gradients of counterfactuals. *arXiv preprint arXiv:1611.02639*, 2016.

[156] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[157] Duyu Tang, Bing Qin, and Ting Liu. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1422–1432, 2015.

[158] Ayush K Tarun, Vikram S Chundawat, Murari Mandal, and Mohan Kankanhalli. Fast yet effective machine unlearning. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[159] Anvith Thudi, Gabriel Deza, Varun Chandrasekaran, and Nicolas Papernot. Unrolling sgd: Understanding factors influencing machine unlearning. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*, pages 303–319. IEEE, 2022.

[160] Che-Ping Tsai, Chih-Kuan Yeh, and Pradeep Ravikumar. Sample based explanations via generalized representers. *Advances in Neural Information Processing Systems.*, 36, 2023.

[161] Enayat Ullah, Tung Mai, Anup Rao, Ryan A Rossi, and Raman Arora. Machine unlearning via algorithmic stability. In *Conference on Learning Theory*, pages 4126–4142. PMLR, 2021.

[162] Luca Veyrin-Forrer, Ataollah Kamal, Stefan Duffner, Marc Plantevit, and Céline Robardet. What does my gnn really capture? on exploring internal gnn representations. In *International Joint Conference on Artificial Intelligence 2022*, 2022.

[163] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[164] Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. *arXiv preprint arXiv:2108.11577*, 2021.

[165] Shaokui Wei, Mingda Zhang, Hongyuan Zha, and Baoyuan Wu. Shared adversarial unlearning: Backdoor mitigation by unlearning shared adversarial examples. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[166] Oliver Wieder, Stefan Kohlbacher, Mélaine Kuenemann, Arthur Garon, Pierre Ducrot, Thomas Seidel, and Thierry Langer. A compact review of molecular property prediction with graph neural networks. *Drug Discovery Today: Technologies*, 37:1–12, 2020.

[167] Mike Wu, Sonali Parbhoo, Michael C Hughes, Volker Roth, and Finale Doshi-Velez. Optimizing for interpretability in deep neural networks with tree regularization. *arXiv preprint arXiv:1908.05254*, 2019.

[168] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. *arXiv preprint arXiv:2101.00288*, 2021.

[169] Zhenxing Wu, Jike Wang, Hongyan Du, Dejun Jiang, Yu Kang, Dan Li, Peichen Pan, Yafeng Deng, Dongsheng Cao, Chang-Yu Hsieh, et al. Chemistry-intuitive explanation of graph neural networks for molecular property prediction with substructure masking. *Nature Communications*, 14(1):2585, 2023.

[170] Jie Xu, Zihan Wu, Cong Wang, and Xiaohua Jia. Machine unlearning: Solutions and challenges. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2024.

[171] Jiasen Yang, Qiang Liu, Vinayak Rao, and Jennifer Neville. Goodness-of-fit testing for discrete distributions via stein discrepancy. In *International Conference on Machine Learning*, pages 5561–5570. PMLR, 2018.

[172] Jiaxi Yang and Yang Zhao. A survey of federated unlearning: A taxonomy, challenges and future directions. *arXiv preprint arXiv:2310.19218*, 2023.

[173] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

[174] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018.

[175] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[176] Hao Yuan, Jiliang Tang, Xia Hu, and Shuiwang Ji. Xgnn: Towards model-level explanations of graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 430–438, 2020.

[177] Hao Yuan, Haiyang Yu, Shurui Gui, and Shuiwang Ji. Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[178] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. On explainability of graph neural networks via subgraph explorations. In *International Conference on Machine Learning*, pages 12241–12252. PMLR, 2021.

[179] Binchi Zhang, Yushun Dong, Tianhao Wang, and Jundong Li. Towards certified unlearning for deep neural networks. In *Forty-first International Conference on Machine Learning*, 2024.

[180] Quan-shi Zhang and Song-Chun Zhu. Visual interpretability for deep learning: a survey. *Frontiers of Information Technology & Electronic Engineering*, 19(1):27–39, 2018.

[181] Shichang Zhang, Yozen Liu, Neil Shah, and Yizhou Sun. Gstarx: Explaining graph neural networks with structure-aware cooperative games. *Advances in Neural Information Processing Systems*, 35:19810–19823, 2022.

[182] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[183] Jianlong Zhou, Amir H Gandomi, Fang Chen, and Andreas Holzinger. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5):593, 2021.

[184] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 2005.