A Journey in Representation Learning: Textual Emotion Detection and Structural Graph Encodings

Nurudin Alvarez-Gonzalez

DOCTORAL THESIS UPF / 2023

THESIS SUPERVISORS
Dr. Andreas Kaltenbrunner
Dr. Vicenç Gómez

Dept. of Information and Communication Technologies

upf. Universitat Pompeu Fabra *Barcelona*

A mi madre.

Acknowledgements

This thesis is dedicated to my parents. You taught me to focus on the few things that matter and to insist. I insisted. To my mother, who could not study as much as she wanted nor see me finish the studies she proudly saw me start. You will never see me graduate, and no dedication is enough.

I am thankful to my advisors Andreas and Vicenç. They persevered with me during all difficulties of the Industrial Doctorate program when the sponsoring company went bankrupt, and I was left without a job and moving to a foreign country that people often cannot place on a map. Their guidance has made me a better scientist, a better writer, and taught me to research more effectively. This thesis has been a marathon, and without them, I would not have been able to reach the finish line—starting with them as 'doktorvateren' that by the end became 'doktorgroßvateren'.

I am also thankful for all my colleagues, both past and present. Whether discussing specific technical details of papers or brainstorming solutions to hard problems all the way to the little nonsense to fill the hours, they helped me push forward during the Industrial Doctorate program and beyond. There are many names, too many to mention—but I keep all the fond memories with Nacho, Diego, Joan, Dmitry, Matteo, Andreas, Sara, Soner, Gural, Atilla while we sprinted to build a Search Engine together and my research aimed to make it ever-so-slightly better. I also dedicate the thesis to Christian, Pierre-Antoine and Amin, who had to deal with me 'serving two masters' while working with them, and the many and merry discussions with Paul, Andreas (this is a different one!), Ezgi, Zeno, Lucas, Leo, Ahmed, Thomas, and of course both Nacho and Amin again. Intense work days followed by doctorate research nights sting less when in good company, and I couldn't have it another way.

Finally, my work is dedicated to my wife, Luna. All research and writing on the thesis ran parallel to our little adventure: meeting again, getting married, and building a little life together. A pandemic in the middle, an explosion in the port of Beirut, a legal journey for her to come meet me in tiny, unexpected Luxembourg. It was not easy: she put up with my sleepless nights, the deadlines Anywhere-on-Earth, the tiresome venting all the same. She cheerfully went along whenever I asked her to look at my figures, or see if I could explain whatever I was working on for her to understand. Above all, she was there, tugging along patiently—with love, and Mateo, our first son, being born along the way.

Abstract

Automatically learning useful representations that can be used in downstream tasks is a key problem in modern machine learning. In recent years, researchers have discovered approaches to represent modalities as diverse as sounds, images, text, or graphs—all encompassed by the field of Representation Learning. Recently, the domains of textual and graph data, representation learning techniques have experienced significant successes with the advent of Large Language Models (LLMs) and Graph Neural Networks (GNNs). The common thread across these advances is the design of learning architectures that can efficiently capture the information contained in large datasets in order to make future predictions. Such learning architectures imply a set of assumptions (a choice of inductive bias) which is determined by the specific characteristics of the data domain and the type of learning task.

In this work, we explore two different limits on representation learning for textual and graph data: emotion detection and graph structures. In particular, we study the spectrum from traditional non-learnable (pre-defined) representations, to representations that are automatically extracted from data during learning. On textual data, we concern ourselves with a specific task—emotion detection—and investigate the impact of methods and labelling regimes in learned representations. We explore the impact of learnable and non-learnable representations, and seek to understand how the sources of data can affect the representations learned by different models. On the domain of graphs, we study to which degree the expressivity of recent GNN representations can be achieved through explicit structural attributes of graphs. We identify and design signals that are theoretically connected to several approaches to measure expressivity in graphs. Experimentally, we find that our approach improves the performance of GNN models when added as input features or used during the learning process. Representing textual information is key to provide natural language interfaces that understand humans on their own terms. Likewise, identifying emotions from text is crucial for varied tasks like mining signals for financial models or identifying toxic social networks interactions. In this thesis, we consider the two largest now-available corpora for emotion classification: GoEmotions, with 58k messages labelled by readers, and Vent, with 33M writer-labelled messages. First, we design a benchmark to evaluate several learnable and non-learnable representations as input to several learning methods, including two simple yet novel models on top of BERT that outperform previous strong baselines on GoEmotions. Second, we analyze differences between how writers express emotions and how readers perceive them through an experiment with human participants where we assess the quality of labelling work on the Vent dataset. We open-source our models and share a public web interface for researchers to explore them.

Similarly, identifying similar network structures is key to capture graph isomorphisms and learn representations that exploit structural information encoded in graph data. This work focuses on ego-network sub-graphs, and identify two practical approaches to encode sub-graph structures with varying degrees of expressivity. Our methods are more expressive than the classic Weisfeiler-Lehman (1-WL) test for isomorphism testing, and we show that they can be used as pre-processing steps to produce features that augment node representations. We characterize three sources of structural information—degrees, distances and edges—and design a model to enrich Message Passing (MP) Graph Neural Networks (GNNs) beyond 1-WL and its more powerful 3-WL variant by capturing edge-level information. We formally study the expressivity, algorithmic complexity, and relationship of our approaches to existing literature. Experimentally, we evaluate our methods across more than ten GNN architectures and 17 graph datasets, finding that they match or outperform existing models while providing a trade-off between memory and predictive power with up to 18.1$\times$ lower memory costs.

Resumen

Aprender representaciones útiles que puedan ser utilizadas en tareas arbitrarias es un problema fundamental en el aprendizaje automático moderno. En los últimos años, se han presentado enfoques para representar modalidades de datos tan diversas como sonidos, imágenes, texto o grafos, todo ello englobado en el campo del Aprendizaje de Representaciones. Recientemente, en dominios de datos textuales y de grafos, las técnicas de aprendizaje de representaciones han experimentado éxitos significativos con la llegada de los Large Language Models (LLMs, Modelos de Lenguaje a Gran Escala) y las *Graph Neural Networks* (GNNs, redes neuronales en grafos). El hilo conductor común en estos avances es el uso de representaciones aprendibles en conjunto con modelos de *Deep Learning* y cantidades ingentes de datos, y los compromisos entre el conocimiento de expertos y las arquitecturas especializadas frente a las colecciones de datos más grandes y los costes computacionales.

En este trabajo, exploramos dos límites diferentes en el aprendizaje de representaciones para datos textuales y de grafos: la detección de emociones y de estructuras de grafos, respectivamente. En particular, estudiamos el espectro que va desde representaciones tradicionales no aprendibles a las representaciones que se extraen automáticamente de los datos durante el aprendizaje. En el dominio textual, nos centramos en una tarea específica—la detección de emociones—e investigamos el impacto de los métodos y regímenes de anotación en las representaciones aprendidas. Estudiamos el impacto de representaciones aprendibles y no aprendibles, y buscamos comprender cómo las distintas fuentes de anotación pueden afectar a las representaciones aprendidas por diferentes modelos. En el ámbito de los grafos, estudiamos hasta qué punto se puede lograr la expresividad de las representaciones de las GNNs a través de atributos estructurales explícitos de los grafos. Identificamos y diseñamos un conjunto de nuevas señales que están teóricamente relacionadas con varios enfoques para medir la expresividad en grafos. Experimentalmente, encontramos que nuestro enfoque mejora el rendimiento de las GNNs cuando se agregan como atributos de entrada o se utilizan durante el proceso de aprendizaje.

Identificar emociones en el texto es crucial para diversas tareas, como la extracción de señales para modelos financieros o la identificación de interacciones tóxicas en las redes sociales. En esta tesis, consideramos los dos mayores corpus disponibles actualmente para la clasificación de emociones: GoEmotions, con 58.000 mensajes etiquetados por lectores, y Vent, con 33 millones de mensajes etiquetados por sus autores. Primero, diseñamos un marco de evaluación para varias representaciones aprendibles y no aprendibles usadas como entrada para varios métodos de aprendizaje, incluyendo dos modelos simples pero novedosos basados en BERT que superan el anterior estado del arte en GoEmotions. En segundo lugar, analizamos las diferencias entre cómo los escritores expresan emociones y cómo los lectores las perciben mediante un experimento con participantes humanos en el que evaluamos la calidad del trabajo de etiquetado en el conjunto de datos Vent. Hacemos públicos nuestros modelos de manera *open-source* y compartimos una interfaz web pública para que los investigadores puedan explorarlos.

De manera similar, identificar estructuras de red similares es fundamental para capturar isomorfismos de grafos y aprender representaciones que aprovechen la información estructural codificada en los datos de los grafos. Este trabajo se centra en subgrafos 'ego' centradas en un nodo, a partir de los que identificamos dos enfoques prácticos para codificar estructuras de subgrafos con diferentes grados de expresividad. Demostramos que nuestros enfoques son más expresivos que la clásica prueba de isomorfismos de grafo de Weisfeiler-Lehman (1-WL), y mostramos que se pueden utilizar como un preprocesamiento para producir atributos y aumentar las representaciones de nodos y aristas. Caracterizamos tres fuentes de información estructural: grados, distancias y aristas, y diseñamos un modelo para enriquecer las GNNs basadas en Paso de Mensajes (MP) más allá de 1-WL y de 3-WL, una variante más expresiva, al capturar información a nivel de aristas. Para los enfoques propuestos, estudiamos formalmente la expresividad, la complejidad algorítmica y la relación de nuestros enfoques con la literatura existente. Experimentalmente, evaluamos nuestros métodos en más de diez arquitecturas de GNN y 17 conjuntos de datos de grafos, encontrando que igualan o superan a los modelos existentes, pudiendo intercambiar entre la memoria y la capacidad predictiva con costes de memoria hasta 18.1$\times$ menores.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Artificial Intelligence is the field of research concerned with making machines do things that people consider to require intelligence.

— Marvin Minsky, in Society of the Mind (1968).

A prerequisite for performing intelligent tasks is finding suitable representations for the information needed to solve them—varying across domains in terms of complexity, difficulty and generality. For example, to understand a sentence and identify some property about it, should we look at the characters making up the words, the words themselves, or how they are laid out, one after the other? If we are looking for typos, we will focus on something different than if we are detecting any emotions expressed in the sentence, or summarizing it as part of a larger paragraph.

Different representations might be better suited to some domains and tasks, with trade-offs between computational costs, model performance, and out-of-distribution behavior. As a result, it is natural to seek *generic* ways to construct representations *systematically*—whether they capture text, images, sound, relational data, or a combination of them and other modalities.

1.1 Motivation and Context

The origin of this work was an industrial doctorate collaboration to learn useful and efficient representations of textual and graph data in a Web Search domain. In this setting, documents are both composed of their textual (and potentially multi-media) contents, and the hyper-links between them—which define a web-graph across websites linking to one another.

Our initial goal was to learn representations that could be used by a Search Engine to power document matching and ranking. The end-to-end task in such a system requires to identify and index high-quality documents, ensure that they are retrievable for a user-provided query, and rank them in such a way that the most relevant documents appear on top. In this setting, automatically classifying from textual and graph data into content topics, sentiment in news or reviews, or ideological biases and spam domains could help provide more meaningful results to user requests.

We aimed to improve the search components that represented both documents and their interactions as a graph by identifying properties that could be pre-computed for previously unseen documents. Specifically, we aimed to address cold-start challenges associated to previously unseen queries, documents, and customer intents by encoding textual content and relevant graph-level signals in a shared latent representation that could be efficiently computed and improved through user feedback. However, our collaboration stopped due to unforeseen financial constraints on the industrial partner.

In this context, we modified our research plan to continue our work in two directions. First, we continued our natural language processing work by focusing on an applied emotion detection task, studying the impact of different textual representation, model architectures, and annotation schemes from the point-of-view of readers and writers. Second, we shifted our work on graphs from large web-scale graphs towards identifying representations that can be used to represent unseen nodes, edges and sub-graphs in machine learning tasks. Furthermore, we explored the theoretical properties of graph representations in relation to existing algorithms used in tasks such as isomorphism detection. The underlying thread to our research is *representation learning*: how to *learn* representations that are suited for *downstream tasks*, studying the limitations, properties, and empirical aspects of textual and graph representations.

1.2 Representation Learning

Traditional machine learning techniques focused on leveraging human expertise to design features that encoded the necessary information to perform intelligent tasks—with handcrafted statistical signals being fed into a variety of learning algorithms. The *feature engineering* approach allows practitioners to efficiently encode their expertise and deliver solutions for the task at hand. However, the dependency on human expertise and narrow, domain-specific designs led researchers towards asking—"*is it possible to automatically generate features suitable for downstream learning tasks?*".

In the last decade, techniques to automatically extract useful representations from data exploded in popularity within the field of Representation Learning (Bengio, Courville, and Vincent 2013) and often in combination with Deep Learning (DL) (LeCun, Bengio, and Hinton 2015). Learnable representations have led to a continued successes in disparate domains—including image classification, segmentation and synthesis as examples of Computer Vision (CV) tasks (Goodfellow et al. 2014; He et al. 2016; Kirillov et al. 2023; Kolesnikov et al. 2021; Krizhevsky, Sutskever, and Hinton 2012; Rombach et al. 2021); text classification, generation, translation, summarization and other Natural Language Processing (NLP) problems (Bojanowski et al. 2017a; Brown et al. 2020; Lewis et al. 2020; Mikolov et al. 2013; Radford et al. 2018, 2019; Raffel et al. 2020; Sutskever, Vinyals, and Le 2014; Vaswani et al. 2017); and graph machine learning applied to isomorphism detection, classification and regression of properties in social networks, molecular graphs and beyond (Abboud, Dimitrov, and Ceylan 2022; Balcilar et al. 2021; Gori, Monfardini, and Scarselli 2005; Grover and Leskovec 2016; Gutteridge et al. 2023; Kipf and Welling 2017; Mitton and Murray-Smith 2023; Scarselli et al. 2005; Veličković et al. 2018; Ying et al. 2021; Zhao et al. 2022).

In Section 1.3, we highlight how a large component of the success of representation learning has been the availability of larger data sets, computational resources, and improved input signals or model architectures. Our work in this thesis continues along these directions, studying the impact of novel data sets, annotation regimes, model inputs, and algorithmic improvements on the two domains introduced in Section 1.1.

1.3 Inductive Biases, Pre-training, & Fine-tuning

The success of Representation Learning owes to the recent availability of large data sets on which models can be pre-trained on *unsupervised* or *self-supervised* tasks[1], which enable models to automatically learn properties on large-scale data without human annotation.
The choice of the modelling approach depends on available computational power, expert knowledge, and data availability, where practitioners can explore the spectrum between (a) feature engineering, (b) input pre-training and architecture design, (c) large self-supervised pre-training and fine-tuning, and (d) generative modelling and few-or-zero-shot prompting. In this section we provide an overview of previous work in each of these directions in the two domains we cover in this disertation—text and graph data.

1.3.1 Learning Representations from Text

We first provide an overview of the history of text representations from early statistical methods to the current state-of-the-art neural large language modelling generative methods that can solve zero-shot tasks. In Figure 1.1, we provide a visual overview of the section, where we introduce influential representation techniques in chronological order.

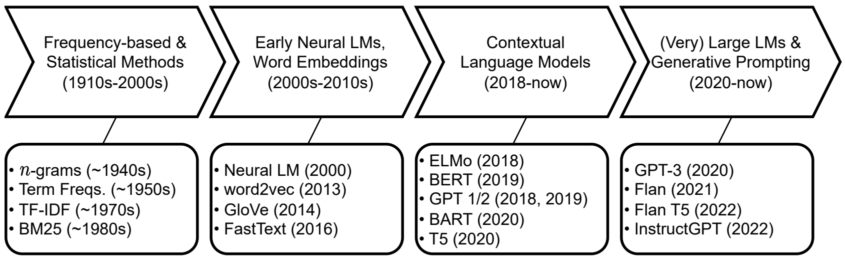


Figure 1.1: Historical overview of text representations from statistical methods to neural language models. We highlight influential works that are relevant to this dissertation and their publication dates.

[1]'Self-supervised' often refers to learning objectives capturing properties of the input itself, e.g. across words in sentences or image patches in computer vision settings.

Before representation learning techniques became ubiquitous to represent textual data, practitioners often relied on statistical methods and prior knowledge drawn from linguistics to represent words, syntactic structures, or documents (Manning and Schütze 1999). When processing documents within corpora, downstream models were trained on n-gram representations captured through frequency-based sparse document vectors such as bags-of-words (i.e. uni-gram term frequencies), term-frequency over inverse document frequency (TF-IDF), and parametrized variants of TF-IDF like BM25 (Robertson and Zaragoza 2009).

Early learnable text representations could be understood as dense vector-space neural language-modelling alternatives to the former statistical representations (Bengio, Ducharme, and Vincent 2000). Eventually, efficient methods for constructing such representations were presented—word2vec captured word co-occurrence patterns by minimizing the skip-gram and continuous bag-of-words (CBOW) objectives (Mikolov et al. 2013), GloVe produced global embedding vectors as the product of a log-bilinear regression model (Pennington, Socher, and Manning 2014), and FastText extended the word modelling of word2vec to unseen words representing words as linear combinations of n-grams embeddings (Bojanowski et al. 2017a). The common thread across word embedding techniques was to capture relevant information between words within a corpora in a compact latent space to be used as input in downstream tasks—often further tuning the embedded vectors if this was possible within the model architecture (e.g. a neural network trained through back-propagation).

Building on top of pre-trained representations of the input data, research initially focused on introducing modifications on top of downstream model architectures, encoding *inductive biases* that help capture the target hypotheses. The broad term 'inductive bias' is used widely accepted by the machine learning research community, and captures loosely defined notions of the knowledge or structures implicitly included in model design that boost its ability to learn and inductively generalize from training data. One such example of this research approach was the trend on neural methods for NLP, where authors produced variants of recurrent neural network (RNN) architectures such as the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) to encode tree information (Tai, Socher, and Manning 2015) or attention pointers (See, Liu, and Manning 2017).

Over time, given the abundance of enormous web corpora and cheaper computation costs, NLP researchers have focused on training larger contextualized embedding models on a variety on objectives. In rough chronological order, this materialized into bidirectional language modelling with ELMo (Peters et al. 2018), and later through predicting masked tokens or next sentences as is the case with BERT (Devlin et al. 2019), predicting the original terms out of corrupted tokens as in T5 (Raffel et al. 2020), the reconstruction losses proposed for BART (Lewis et al. 2020), or next token prediction in GPT models (Radford et al. 2018, 2019).

In recent years, models have grown in size and number of parameters, and researchers have turned to produce and curate larger datasets derived from sources such as Wikipedia, or Common Crawl[2], with datasets such as the Colossal Clean Common Crawl (C4)[3] and the Pile (Gao et al. 2020) aiming to provide diverse and meaningful text for pre-training.

In this context, larger models leveraging pre-training techniques yield *foundation models* that can represent the underlying data domain—whether it is text, images, audio, or multiple combined modalities. However, they may not show acceptable performance in specific tasks that were not seen during training or captured by the optimization objective. A variety of methods have been proposed to direct models pre-trained on generic domains to perform well on specific objectives. Recently, owing to architectures like Transformers that can learn contextual representations by leveraging the underlying information from larger datasets, work has focused on *fine-tuning* pre-trained models by retraining them on task-specific data and supervised labels as for models like BERT, BART and T5 (Demszky et al. 2020; Lewis et al. 2020; Raffel et al. 2020).

Another avenue of work has focused on framing prediction tasks as language generation tasks. In this setting, models are *prompted* with an initial snippet, with the aim that the tokens predicted afterwards will contain the expected output. Prompting often involves providing examples ('few-shot') of the task through text (Brown et al. 2020), alternatively providing only a description of the task without providing examples ('zero-shot') (Wei et al. 2022). In this direction, recent work has further fine-tuned LLMs to follow instructions and improve prompting performance (Ouyang et al. 2022).

[2]https://commoncrawl.org/

[3]https://www.tensorflow.org/datasets/catalog/c4

As textual foundation models encode the accumulated knowledge captured within their input corpora, they require a massive number of parameters and several orders-of-magnitude increased computation costs compared to shallower neural methods or statistical approaches. Thus, beyond recent state-of-the-art metrics that LLMs achieve, there exist trade-offs between model complexity and computation costs, data and annotation budgets—as evidenced by ongoing research to train compute-optimal model for their parameters, and efforts to achieve comparable performance to Transformers (Hoffmann et al. 2022) using compute-efficient RNNs like the Receptance Weighted Key Value (RWKV) (Peng et al. 2023) model. Practitioners may also concern themselves with the cost of computation and deployment, and whether human annotations are available for a given task, language or domain, and at which volume—such as low resource scenarios (Hedderich et al. 2021).
These recent advances on models capable of producing human-like responses (Bubeck et al. 2023) have spurred novel research in emotion detection, both in terms of opportunities and risks. For instance, recent works have shown that large language models can be used to learn from limited code-switching emotion data (Kumar et al. 2023) but have also highlighted potential privacy concerns with user identification (Staab et al. 2023).
Our work in this thesis predates recent advances in generative modelling. At the time, a broad line of study in NLP focused on identifying effective strategies to fine-tune pre-trained language models to improve their performance on out-of-distribution tasks. As we will present in Section 1.5, our work followed a similar approach. However, our study focused on understanding the limits of learning from large emotion data sets and label spaces, as well as between the perspectives of readers and writers, which we believe remains valid after the popularization of large generative language models.

1.3.2 Graph Representation Learning

Another emerging domain of representation learning is graphs. Graphs are powerful data structures capable of expressing a wide variety of concepts—including textual information as a sequence, or image data as a pixel grid. This makes graphs an attractive target to learn from as ubiquitous and generic data structures.

However, graph data contrasts with other domains like text or images, where the structure of inputs to a model can vary widely. For instance, graphs can have variable sizes and structures (e.g. as grids, random graphs, and denser regions in a network), and methods need to be robust to permutations of node and edge-labels. Furthermore, graph tasks are highly diverse, as they might target predictions at the level of nodes, edges, or 'full' graphs.

Comparing graph representations can be challenging owing to the diversity of graph data. However, graphs are abstract structures which have been thoroughly studied as a field of their own. Previous work on graph theory can be leveraged in the formal study of learnable graph representations. Different analytical frameworks can help surface research directions in terms of the computations that certain representations cannot capture. For instance, graph neural networks are often evaluated by whether the representations they produce can distinguish non-isomorphic graphs.

Two graphs are said to be isomorphic if there exists a permutation between the node labels in one graph that yields the set of connected edges in the other one and vice-versa. The connection with isomorphism detection is interesting as the underlying Graph Isomorphism problem is NP-Intermediate, but it is known that polynomial approximations can still distinguish non-isomorphic graphs with high probability (Babai and Kucera 1979).

This formalization connects the *expressivity* of graph representations with the known problem of Graph Isomorphism detection, and the approximations to that problem—the most common of which is the Weisfeiler-Lehman (1-WL) color refinement test (Weisfeiler and Leman 1968). When the 1-WL test produces different representations for two graphs, they are not isomorphic; if the representations match, they are likely to be isomorphic. The existence of expressivity hierarchies allows researchers to reason about the demonstrable abilities of their models and inductive biases.

Early work to learn on graphs resembled the hand-engineered features that had been used when applying statistical methods to NLP. A common representation were Graph Kernels (GKs), used in downstream statistical learning models—such as diffusion (Kondor and Lafferty 2002), random walk (Vishwanathan et al. 2010), and Weisfeiler-Lehman kernels (Shervashidze et al. 2010) inspired by the namesake 1-WL test.

Learnable graph representations mirrored research on NLP, extrapolating insights from word embedding methods like word2vec to learn node representations by observing the frequencies of nodes co-occurring in random walks sampled from graph data (Grover and Leskovec 2016; Perozzi, Al-Rfou, and Skiena 2014). These approaches were limited: they only represented nodes and required all nodes to be known at training and inference time—which known as a *transductive* model. From these limitations, researchers focused on finding *inductive* approaches that could learn to represent relevant aspects from unseen graphs, nodes or edges with matching structural properties or attributes—most commonly exemplified by early Graph Neural Networks (GNNs) (Gao, Wang, and Ji 2018; Hamilton, Ying, and Leskovec 2017; Kipf and Welling 2017).

Due to the heterogeneity of graph datasets and tasks—which can be specified over graphs as a whole, edges or nodes—research has focused on designing model architectures that incorporate expressive *inductive biases* to increase model performance. For instance, authors introduced Chebyshev filters on the spectral domain (Defferrard, Bresson, and Vandergheynst 2016), attention mechanisms (Veličković et al. 2018), or efficient distributed implementations (Samanta et al. 2020) in network architectures, identifying that computations can be modelled as messages passed across neighbouring nodes (Gilmer et al. 2017) (dubbed MP-GNNs).

However, despite MP-GNNs being successful in several domains, their representational power has been found to be at most as capable as the 1-WL test (Morris et al. 2019; Xu et al. 2019). As a result, recent work has focused on augmenting MP-GNN models with inductive biases in the form of additional features or architectural modifications to improve their expressivity in the k-WL family of tests. This has been motivated by the aforementioned fact that the expressivity of these models can be formally reasoned about. MP-GNN models have been extended by means of structural features (Bouritsas et al. 2021) or spectral information (Balcilar et al. 2021), introducing dropout mechanisms that enhance isomorphism detection (Papp et al. 2021), implementing more powerful network representations at increased computational costs (Maron et al. 2019b), or enriching network architectures with structural attributes during learning (Abboud, Dimitrov, and Ceylan 2022; Bevilacqua et al. 2022; Nikolentzos, Dasoulas, and Vazirgiannis 2020; Zhang and Li 2021; Zhao et al. 2022).

1.4 Objectives and Research Questions

This thesis aims to study learnable representations on two domains: text emotion detection and structural graph encodings. Driven by the initial business motivation outlined in Section 1.1, the overarching objective was to connect both domains. Although the main goal was to improve the performance of a web search engine where emotional text and graph data are used, we first identified independent research questions that could be addressed on their own. For textual emotion detection, this thesis aims to answer two research questions:

A1. What are the performance limits in textual Emotion Detection?

A2. Are there differences in how *writers* express and *readers* perceive emotions in social media?

Underlying questions **A1** and **A2** is the objective to gain a deeper understanding to build emotion detection systems through useful inductive biases, pre-training techniques, and choices of data sources and emotion labels. Likewise, on the study of structural graph properties, our objective was to identify a representation or a family of representations to improve existing models.
On structural graph encodings, our work has similar objectives of improving existing models as in the textual emotion domain. However, rather than learning representations that deal with the subjective nature of emotions, we concern ourselves with the theoretical expressivity and empirical performance of graph neural networks. We formulate two research questions with a common preface:

Is it possible to enhance graph neural networks through the incorporation of a set of predefined structural features,

B1. in terms of expressive power?

B2. in terms of empirical performance?

Research questions **B1** and **B2** aim at identifying effective methods to improve graph representations at scale.
Having introduced our objectives, the following section provides an overview of our results, contributions, and the structure of the document.

1.5 Thesis Contributions and Structure

In this section, we describe the research questions for this thesis, highlight our main contributions, and outline the structure of the document to present our research.

1.5.1 Main Results and Contributions

We now highlight main results from our work, directing the reader to each chapter as outlined in Subsection 1.5.2 for additional details.

Textual Emotion Detection

On textual emotion detection, we implement a benchmark with four text representation algorithms and five model types on the two largest available emotion text corpora. We study whether there are differences in how writers express and readers perceive emotions through an experiment with human annotations. We open-source code for (a) a framework to compare statistical and neural methods, (b) an exploration tool for multi-label emotion detection models, and (c) an emotion-labelling interface for Amazon MTurk. Finally, we release our data set of reader annotations on the Vent dataset, obtaining a multi-perspective emotion dataset with labels provided by *readers* and *writers* containing a total of 2640 text snippets.

Structural Graph Encodings

On structural graph encodings, we introduce two methods that represent sub-graphs as sparse feature vectors to complement node or edge features. We formally study the expressivity of our method and show that it is more expressive than the 1-WL isomorphism test. We implement an experimental benchmark and find that introducing our features boosts the performance of 11 different graph neural models in 6 graph learning tasks. Furthermore, we show that introducing edge-level information in our structural encodings improves expressivity. We formally show that our variants of ego-network encodings have different levels of expressivity and connect them to recent graph neural network models like Shortest Path Neural Networks. Experimentally, we find that our ego-network encodings can produce comparable results to state-of-the-art Subgraph GNNs at reduced memory costs.

1.5.2 Thesis Structure

We now describe the structure of the thesis in relation to the objectives and research questions (RQ) outlined in Section 1.4. The thesis is structured in three main parts: two parts covering facets of representation learning in textual emotion detection and structural graph encodings, and one part that weaves them together to provide final conclusions, opportunities, and future work.

Part I. Textual Emotion Detection

The first part focuses on textual emotion detection. We seek to understand the impact of different text representation techniques, model architectures, data volumes, and labelling approaches.
In Chapter 2, we provide an overview for state-of-the-art emotion taxonomies, corpora, and modelling approaches. After this introduction, Chapter 3 introduces our results evaluating a variety of statistical methods and fine-tuning on the two largest available Emotion Detection data sets to answer RQ **A1**. In this chapter we also assess the impact of annotation sources—whether emotion labels are provided by the *authors* expressing the emotion or the *readers* perceiving them, answering RQ **A2**.

Publication venues: The work from Chapter 3 has been published in *Findings of the Association for Computational Linguistics: EMNLP* 2021 (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2021b) and a short paper on our open-source contributions was published in the journal *Software Impacts, Volume 10* (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2021a).

Part II. Structural Graph Encodings

In the second part of the thesis, we study the impact of introducing certain structural encodings in learnable graph representations. In Chapter 4, we provide an overview of graph representation learning techniques and

the different formal frameworks that have been proposed to study their expressivity. We then propose two different representations that improve the performance of Graph Neural Network models, which answer to RQs **B1** and **B2** in the following two chapters.
In Chapter 5, we introduce an encoding algorithm for ego-network subgraphs considering degree and distance information.
We formally analyze the expressivity of our method in the context of the classic Weisfeiler-Lehman color refinement algorithm (1-WL). We then identify a family of features that can be computed efficiently from our encoding and evaluate their empirical performance across a large benchmark of tasks and graph neural network architectures.

Publication venues: The work from Chapter 5 has been first published in non-archival form in *The First Learning on Graphs Conference* 2022 (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2022) and as a full journal paper with extended results on *Machine Learning and Knowledge Extraction, Volume 5* 2023 (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2023a)

As a continuation of this work, Chapter 6 introduces an extended encoding of ego-network information that captures edge-level signals. We formally study the differences in expressivity of the extended encoding. Empirically, we evaluate the features both as raw input signals to downstream and as learnable embeddings that provide additional inductive biases on the structure of graphs to MP-GNN and sub-graph GNN models.

Publication venues: The work from Chapter 6 is under review in the journal *Transactions on Machine Learning Research* 2023 (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2023b).

Part III. Conclusions & Future Directions

We conclude the thesis with Chapter 7. In this chapter, we summarize our work and findings, connecting our results with the motivation outlined in Section 1.1. We also discuss the ethical considerations and broader impact of this thesis, and outline opportunities for future research based on the results of our work.

We supplement the conclusions with Appendices that extend the results presented on the two previous parts. For the work presented in Chapter 3, Chapter 5 and Chapter 6, we provide additional details about experimental setups, data set details, pseudo-code implementations, and hyper-parameter values.

Part I

Representation Learning for Emotion Detection

Chapter 2

Background

2.1 Introduction

Identifying emotional signals is key to a series of downstream tasks. For instance, emotion detection is necessary for empathetic chat-bots that can respond to the emotional needs of their users (Fung et al. 2018). Distinguishing emotional content is required to study viral (Guerini and Staiano 2015), educational (Ortigosa, Martín, and Carro 2014), political (Mohammad et al. 2015), or incendiary (Brassard-Gourdeau and Khoury 2019) interactions on social media. Capturing the evolution of user-provided emotional content can help prevent harassment (Agrawal and Awekar 2018) or develop early indicators for depression (Husseini Orabi et al. 2018; Ramírez-Cifuentes et al. 2020). Facial expressions (Li and Deng 2020), speech (El Ayadi, Kamel, and Karray 2011), body movements (Noroozi et al. 2018) and text (Poria et al. 2019) are sources from which emotions may be automatically extracted.

Emotion analysis contrasts sentiment analysis, which characterizes text in terms of polarity (positive, negative or neutral), by involving a larger set of classes, often influenced by aspects such as ambiguity, misunderstandings, irony or sarcasm (Chauhan et al. 2020; Mohammad 2021). Recent progress in the field has been enabled by the success of pre-trained language models, such as Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al. 2019), and the release of high-quality large-scale annotated datasets.

In this thesis, we focus on studying textual emotion detection on the largest available data sets when it was written. We seek to understand how data volume, text representation, model architecture, and label source all impact the performance of emotion detection systems. In the following sections, we provide a frame of reference for the problem of detecting emotions from text to preface our work in Chapter 3.

2.2 Related Work

This section summarizes three domains of related work. In Subsection 2.2.1, we discuss the existing taxonomies to represent emotions. Subsection 2.2.2 provides an overview of existing corpora used to build and evaluate emotion detection systems. Finally, Subsection 2.2.3 describes the NLP approaches that may be used to implement text-based emotion detection systems.

2.2.1 Emotion Taxonomies

The landscape of human emotion has been represented by several different taxonomies and approaches. Ekman (Ekman and Friesen 1971) proposed 6 basic emotions expressed through facial expressions across cultures: *joy, sadness, anger, surprise, disgust and fear.* Independently, Plutchik (Plutchik 1980) introduced a similar taxonomy that added *anticipation* and *trust*, characterizing emotions through his Wheel of Emotion (Plutchik 1980). Finer-grained emotion taxonomies have been recently proposed, capturing high-dimensional relationships of up to 600 different emotions, clustering emotion concepts using machine learning techniques (Cowen et al. 2019). These taxonomies show the multifaceted nature of emotions across cultures in vocalization (Cowen et al. 2018), music (Cowen et al. 2020), or facial expressions (Cowen and Keltner 2019).

Beyond these discrete categorizations, other models such as the affective-circumplex model of emotions (Russell 2003), have captured emotions as proportions on three dimensions rather than discrete categories: Valence (positive or negative), Activation (active or passive), and Dominance (dominant or submissive). Finally, a small number of works extend these taxonomies to include the perspective of senders and receivers of emotional communication (Buechel and Hahn 2017a,b; Mohammad and Turney 2013;

Ptaszynski et al. 2009). In this chapter, we focus on categorical approaches with recent emotional taxonomies covering a rich spectrum of emotions from the perspectives of senders and receivers.

2.2.2 Emotion Detection Text Corpora

To build Emotion Detection systems, practitioners require text data containing emotional signals. Early works like SentiStrength (Thelwall et al. 2010) and ANEW (Nielsen 2011) used lexical associations for sentiment analysis, capturing whether text was positive, negative or neutral and to which degree. Lexical approaches can be used in rule-based systems, where words contribute to a sentiment or emotion signal. Sophisticated rule-based models like VADER (Hutto and Gilbert 2014) rely on human-annotated word sentiments, alongside with slang, modifiers, emphasis or punctuation. Other approaches went beyond polarity: LIWC (Tausczik and Pennebaker 2010) presented labelled dictionaries mapping words to their emotional and polarity probabilities. Likewise, EmoLex (Mohammad, Kiritchenko, and Zhu 2013) crowd-sourced a word-emotion association lexicon labelling words in terms of sentiment and emotion following Plutchik's taxonomy.
Another approach is to treat emotion detection as a supervised learning problem, with corpora including emotional information varying in size, scope and labelling approach (Bostan and Klinger 2018). Early datasets such as Affective Text (Strapparava and Mihalcea 2007) were small, with 1,250 headlines labelled for valence and Ekman's emotions. GoEmotions (Demszky et al. 2020) is a dataset of human-labelled Reddit comments with 27 emotions and 'neutral'. To reduce data acquisition costs, some works explore corpora mined from emotion-rich environments such as social networks. For instance, Crowdflower's emotion dataset (Crowdflower 2016) labelled 40K tweets expanding Ekman's emotions with *enthusiasm, fun, hate, neutral, love, boredom, relief* and *empty*. Similarly, EmoNet (Abdul-Mageed and Ungar 2017) applied distant supervision by labelling tweets using hashtags among the 'circles' within Plutchik's Wheel. Finally, (Malko et al. 2021) used Vent data on a limited number of emotion categories overlapping with Ekman's emotions, and concluded that the self-annotated labels of Vent are indicative of emotional contents expressed in the text, supporting more detailed analyses of emotion expression.

If emotion detection is treated as a learning problem, datasets present a trade-off between size and quality. Human-annotated datasets span thousands of samples, often targeting a small number of emotions, such as SocialNLP 2019 EmotionX challenge or Crowdflower (Crowdflower 2016; Shmueli and Ku 2019; Strapparava and Mihalcea 2007). Approaches like EmoNet (Abdul-Mageed and Ungar 2017) can comprise millions of samples collected from social media using distantly supervised labels, allowing for larger datasets at the time of publication. However, datasets collected from social media may be private, with direct dataset sharing often being forbidden, and content routinely getting deleted, limiting reproducibility. Additionally, labels produced by distant supervision using hashtags might not align with how humans generally perceive or express emotions across domains.

2.2.3 NLP Emotion Detection Models

Approaches to Emotion Detection are often constrained by the difficulties of extracting emotional signals from small collections of data (Acheampong, Wenyu, and Nunoo-Mensah 2020; Alswaidan and Menai 2020), using both feature-engineered and neural models. Some feature-based models introduce emotional priors by using word-emotion associations as features for supervised classifiers (Mohammad, Kiritchenko, and Zhu 2013). Other approaches employ statistical methods such Bag-of-Words to represent documents, placing the effort of learning emotional associations on the model (Silva, Hruschka, and Hruschka 2014).
Recently, larger datasets have allowed to train neural models that outperform their traditional counterparts. For instance, EmoNet involved collecting a 1.6M tweet dataset, and training RNN-based models on the same data (Abdul-Mageed and Ungar 2017). After the release of BERT (Devlin et al. 2019), an explosion of novel work has focused on fine-tuning transformer models to learn from scarce emotion data. For example, the top performing models on the SocialNLP 2019 EmotionX Challenge (Shmueli and Ku 2019) outperform the best previous existing model by a 19% increase in micro-F1.
In this thesis, we explore simple Transformer-based baselines on this common ground, in which we use BERT as a representation layer. Our modelling introduces lightweight models on top of the contextualized embed-

dings, outperforming previous BERT baselines on micro-F1 by 11.8%. During our research, we evaluated other pre-trained models but found comparable performance at the time.
However, we also study non-neural and non-Transformer methods that remain popular in industry. Our contribution covers different approaches on large scale emotion datasets with rich label spaces and 58K / 9.75M sample texts. Rather than narrowly benchmarking variations of specific architectures, e.g., fine-tuning transformer language models, we work with a variety of established methods to help practitioners choose a modelling approach in terms of predictive performance and model complexity.

2.3 Emotion Detection in Text

A text snippet can have several associated emotions in cases of ambiguity, or when expressing multiple feelings, as seen in Table 2.1. As such, we represent the problem as multiple valid labels being possible for a snippet.

Text	**Emotion**
Wow. I just read the synopsis, and that's really what happens.	Surprise
What do you think? If you look at my question above? Last thing I should do is to say sorry?	Confusion
And then everyone clapped and cheered.	Joy, Admiration

Table 2.1: Labelled text-emotion pairs from the GoEmotions dataset. The third example shows an instance with multiple emotional associations.

In this thesis, we study emotion detection as a multi-label classification task. Given an input text $\mathbf{s} \in \mathcal{U}^*$ and a set of N emotions, our task is to produce (learn) a function $\phi : \mathcal{U}^* \mapsto \mathcal{P}^N$ that maps $\mathbf{s}$ into independent probabilities for each emotions $y_1, y_2, ..., y_N$.
Treating emotion detection as a multi-label task allows us to apply the same architecture on multi-class data sets. It also allows us to account for ambiguity, even in data sets where only a single output class is expected, such as Vent. We use this modelling approach in the entirety of Chapter 3.

Chapter 3

Uncovering the Limits of Text-based Emotion Detection

In this chapter, we analyze the limits of text-based emotion detection and address the limitations of limited data sets with few emotion labels covered in Subsection 2.2.2. We study emotion detection on the two largest now-available corpora: GoEmotions (Demszky et al. 2020) and Vent (Lykousas et al. 2019). GoEmotions contains 58k Reddit comments tagged with possibly multiple labels out 28 emotions, annotated by third-person *readers*. The raw Vent corpus (Lykousas et al. 2019) includes 33M messages tagged with one out of 705 emotions by their original first-person *writers*. The unprecedented volume of these datasets makes them suitable to study textual emotion detection at scale from different perspectives.

The chapter is structured in three sections. In Section 3.1 we describe the GoEmotions and Vent corpora in detail, and introduce a benchmark to measure performance of several feature spaces and learning algorithms on the Emotion Detection as defined in Section 2.3. Our benchmark evaluates statistical and neural models, including a strong BERT baseline on GoEmotions, and let us identify settings in which statistical methods such as TF-IDF outperform more complex word-level embeddings such as FastText. To aid in exploring and sharing our results, we release a web interface showcasing our models. We analyze analyze the hierarchical structure of the label space for models trained on Vent in Section 3.2. Our models capture the cluster structure defined by emotion categories to a large extent, despite not explicitly observing emotion categories during training. Finally, we design an experiment with human participants in Section 3.3, through which we evaluate our model and the differences between emotions provided by writers and those perceived by participants (readers). As we will show, our models outperform readers at predicting emotions intended by writers, and they also predict the emotions annotated by readers even more accurately, with important implications for emotion analysis.

3.1 Experimental Setting

Here we briefly describe the GoEmotions and Vent datasets, our multi-label classification benchmark design, and the representation and modelling approaches that serve as the building blocks for our emotion detectors. We include extensive details on the experimental design, hyper-parameters, additional results, and data analyzes in Appendix A.1 and Appendix A.2.

3.1.1 GoEmotions

The GoEmotions dataset (Demszky et al. 2020) contains 58,009 text snippets collected from English Reddit comments. A minimum of 3 raters labelled each snippet into multiple emotions from 27 emotion categories plus a neutral category, keeping only snippets where 2+ raters agree on one or more labels. Figure 3.1 shows the frequencies and types of each emotion in the dataset. Raters were asked to label comments as "Neutral" when they were not able to clearly assign an emotion to a comment. The dataset is accompanied with a strong baseline fine-tuned BERT classifier built on top of BERT-base (Devlin et al. 2019) that predicts the 28 emotions from the contextualized embedding of the last token, achieving 0.46 macro-F1 and 0.51 micro-F1 scores.

3.1.2 Vent

The Vent dataset (Lykousas et al. 2019) contains more than 33M comments from a social network and its accompanied mobile app, predominantly in English. Each comment or "vent" is self-annotated by the author, which we will refer to as "*venter*", according to their emotional state. Emotions are structured into 63 emotion categories covering 704 emotions.

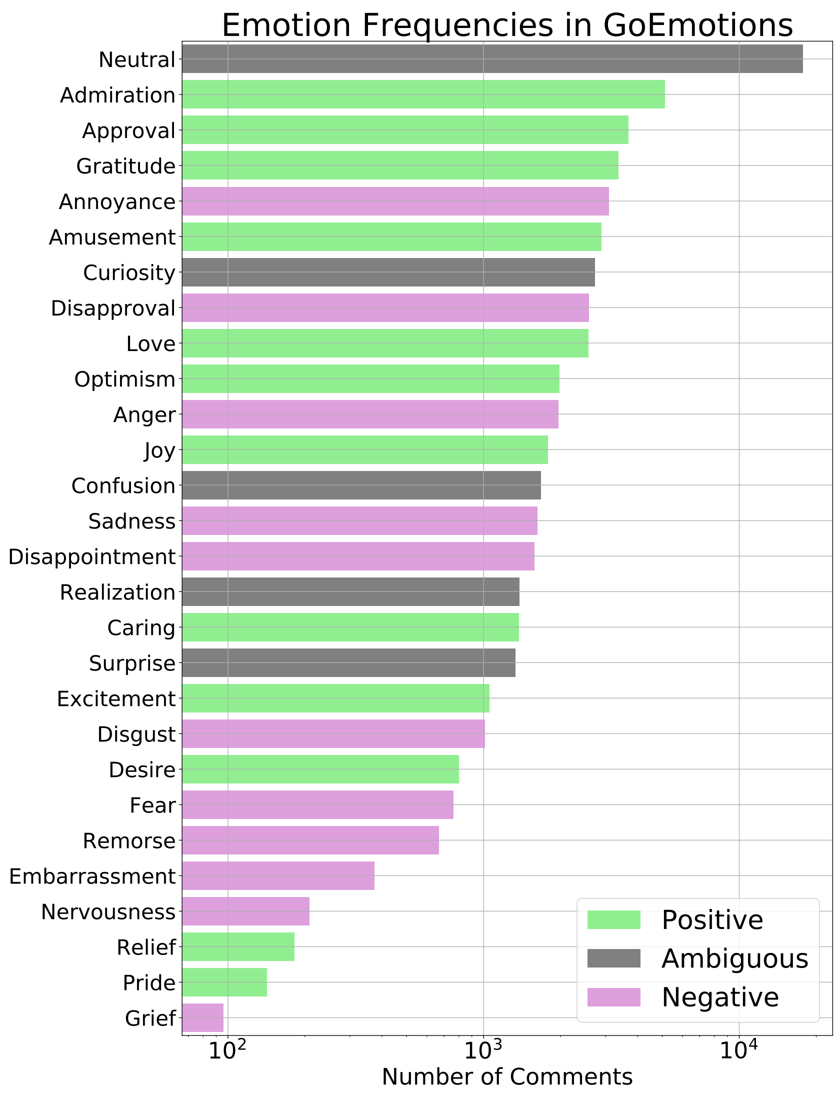


Figure 3.1: GoEmotions label frequencies and the three categories in (Demszky et al. 2020): Positive, Ambiguous and Negative.

In contrast to GoEmotions where comments are labelled with "*reader*" emotions that annotators infer from text, Vent is a "*writer*"-labelled emotion dataset. Every Vent comment is labelled with one subjective emotional label provided by its writer. The dataset is provided as-is, with minimal anonymisation of user and URL references. To normalize the corpus, we *(a)* remove stylistic highlighting such as italics, *(b)* eliminate extraneous white space, and *(c)* map user and URL references to special fixed tokens. Since the length of Vents is heterogeneous, we limit comment length to the range between 3 and 32 tokens (75-th percentile). We restrict ourselves to categories that contain at least one unambiguous emotion, and ignore emotions marked as disabled by Vent, not used once a month, or whose meaning is unclear, e.g., those containing emoji like "Mushy", which includes a mushroom emoji. The filtered data contains 9.75M comments in 88 emotions that are grouped into 9 categories—as shown in Figure A.5.

3.1.3 Multi-Label Benchmark Design

We design a common architecture that lets us train, evaluate, and optionally transfer our models, and only use methods that may be applied in a streaming mini-batch fashion. Our approach is composed of three components, shown as sequential steps in Figure 3.2.
We implement two statistical and two embedding representations, combined with five different learning algorithms. For statistical representations we use **Bag-of-Words** and **TF-IDF**, training **Naive Bayes**, **Logistic Regression**, and **Incremental Random Forests**. For neural-LM representations, we evaluate pre-trained English models, exploring word-level representations with **FastText** and contextualized representations using **BERT** with fine-tuning. For neural-LM representations, we use two simple neural architectures (described below) which process embedded input sequences and pool over them to produce their outputs—shown in Figure 3.3. We implement a multi-label classification objective, minimising the average binary cross-entropy loss over N target emotions.

1. **(Pooled) Deep Neural Network.** We apply a DNN over every embedded token independently in parallel.

2. **(Bi)-LSTM.** We apply stacked (Bi)-LSTMs consuming the embedded sequence in a sequence to sequence manner.

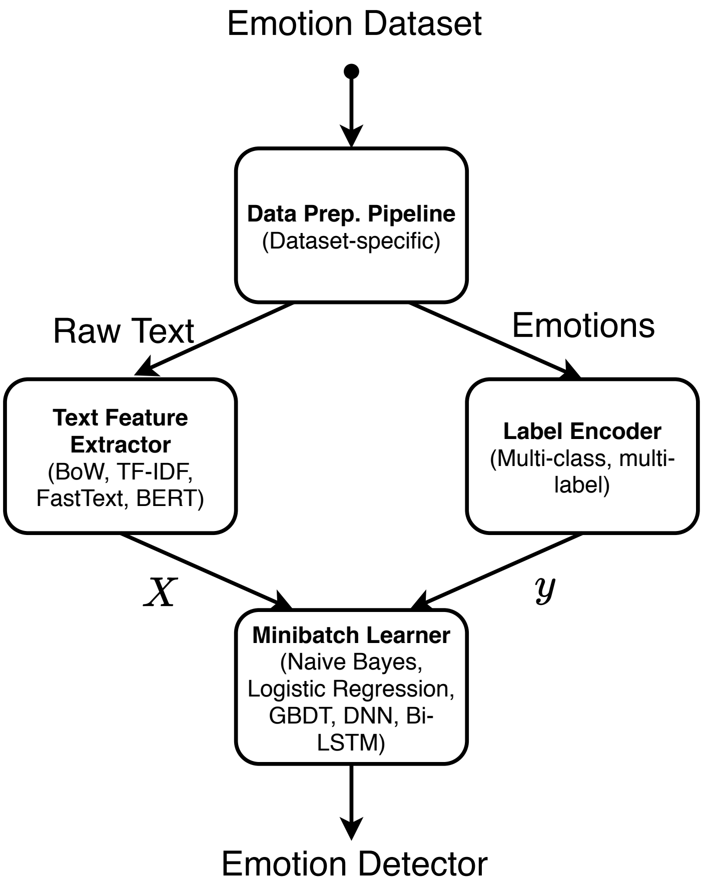
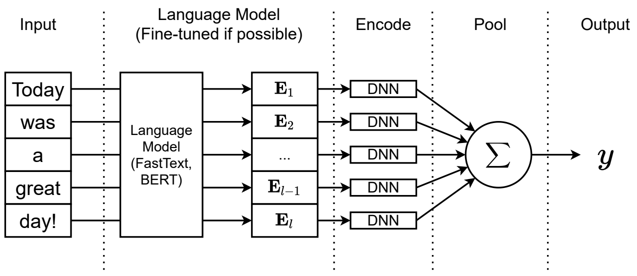


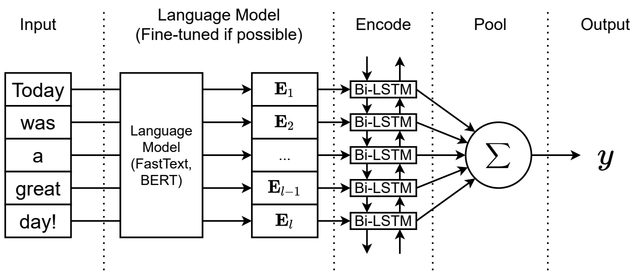
Figure 3.2: High level architecture. We implement several modules to transparently execute the different combinations of text representation and learning algorithms. The options between parenthesis detail the alternative implementations available on every step of the benchmark.

3.2 Experimental Results

In this section, we study the results of our benchmark, evaluate the behaviour of the trained models, and analyze the differences between *writer* and *reader* emotions on Vent.



(a) Pooled DNN Architecture.



(b) Bi-LSTM Architecture.

Figure 3.3: Showcase of the neural architectures in the benchmark. BERT is fine-tuned while FastText is not due to implementation constraints.

3.2.1 Emotion Benchmark Evaluation

First, we evaluate the emotion detection methods that we introduced in Subsection 3.1.3. We focus on four multi-label classification metrics: **M**acro F1, **m**icro F1, and micro-averaged **Pre**cision and **Rec**all. We use the original splits from the GoEmotion datasets, while we split Vent in 80 / 10 / 10 splits ordered by publication time for training, validation and testing. Our analysis is described on a per-dataset basis, with Table 3.1 and Table 3.2 containing the results for GoEmotions and Vent respectively, showing the best performing models in **bold**.

BERT representations outperform every other configuration in both datasets. Furthermore, our results show that Logistic Regressions outperform Naive Bayes and Incremental Random Forests on configurations using Bag-of-Words or TF-IDF. On GoEmotions, our highest micro-F1 score is 0.57, while on Vent the highest micro-F1 score is 0.21.

Repr.	Model	M-F1	m-F1	Pre	Rec
BoW	N. Bayes	0.34	0.46	0.43	0.52
	Log. Reg.	0.45	0.53	0.48	0.61
	R. Forest	0.45	0.52	0.50	0.59
TF-IDF	N. Bayes	0.33	0.44	0.43	0.49
	Log. Reg.	0.47	0.53	0.49	0.60
	R. Forest	0.45	0.52	0.48	0.60
FT	DNN Pool	0.42	0.49	0.45	0.61
	Bi-LSTM	0.44	0.54	0.51	0.58
BERT	Baseline*	0.46	0.51	—	—
	DNN Pool	**0.48**	0.55	0.52	0.61
	Bi-LSTM	0.47	**0.57**	**0.53**	**0.62**

Table 3.1: GoEmotions results. All stddevs ≤ 0.01. * denotes results as reported by (Demszky et al. 2020).

Our results on GoEmotions outperform the previous strong baseline using BERT, increasing macro-F1 from 0.46 to 0.47 and micro-F1 from 0.51 to 0.57. The positive results indicate that our design approach for the benchmark was appropriate to achieve a robust comparison between neural and non-neural methods.

A possible explanation for the performance gains in our models over the baseline (Demszky et al. 2020) is that our approach pools over each token in the input while the baseline uses the contextualized embedding of the last token in the sentence for its prediction. We observe a modest improvement from statistical methods over the baseline, which might be due to emotional information being largely encoded at the word level.

On Vent, Table 3.2 shows that the BERT model with pooled Bi-LSTM layers significantly outperforms other methods. In comparison with GoEmotions, non-BERT models perform significantly worse than BERT-based models. A possible explanation for this is the noisy nature of venter-annotated text from social media posts, whose chosen emotions might at times be arbitrary, and the increased ambiguity from the 88 emotion labels in Vent compared to the 28 targets in GoEmotions.

Repr.	**Model**	**M-F1**	**m-F1**	**Pre**	**Rec**
—	Random	0.02	0.04	0.02	0.69
BoW	N. Bayes	0.13	0.15	0.12	0.23
	Log. Reg.	0.13	0.15	0.12	0.20
	R. Forest	0.11	0.13	0.12	0.18
TF-IDF	N. Bayes	0.14	0.16	0.13	0.21
	Log. Reg.	0.14	0.16	0.14	0.21
	R. Forest	0.11	0.13	0.12	0.19
FT	DNN Pool	0.08	0.10	0.08	0.30
	Bi-LSTM	0.08	0.11	0.08	0.21
BERT	DNN Pool	0.17	0.19	0.16	0.24
	Bi-LSTM	**0.19**	**0.21**	**0.19**	**0.26**

Table 3.2: Vent results with Temporal Splits. All stddevs ≤ 0.01, except FastText models. Random is a baseline classifier that produces a random score between 0 and 1 for every snippet and label.

When we vary the fraction of the whole training set used for training, we find that statistical models perform closer to BERT when there is a reduced amount of data (see results in Section 3.2.1). A possible explanation for this behaviour might be due to our design approach, namely, by the introduction of layers on top of BERT that slow down convergence, as shown in Figure 3.3.

To our surprise, we find that FastText consistently underperforms in comparison to statistical models. In particular, the TF-IDF logistic regression model significantly outperforms FastText on Vent. Furthermore, TF-IDF models using logistic regressions or random forests show similar performance to the FastText model using pooled Bi-LSTMs on GoEmotions despite the relative simplicity of the models. A possible explanation for the the observed difference in performance between Vent and GoEmotions might be that pretrained FastText models are unable to represent slang, typos and platform-specific vocabulary from Vent despite their underlying n-gram model.

Model Performance vs Dataset Size

We perform additional analyses to evaluate the performance of our models as a function of their input data size. We do so by training on randomly sampled sub-sets of the training set from Vent. Figure 3.4 shows the micro-F1 score curves for a given percentage of data available.
We find that statistical models show stable performance, which we explain by the fact that the vocabulary sizes are chosen by searching on a limited set of values. We also observe that FastText slowly improves but requires a much higher data intensity than other methods, which we believe is caused by the inability of re-training the FastText model through backpropagation due to limitations in the official implementation.

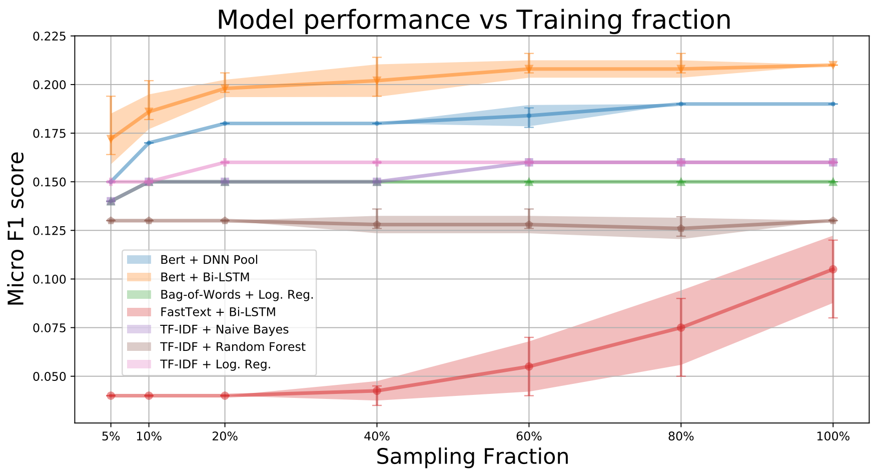


Figure 3.4: Model performance (micro-F1 score) against the whole test set given a percentage of training data.

Temporal Impact Emotional Data

We also train with Vent using uniformly random splits to study whether the data is non-stationary. We take the Robust subset of Vent and divide randomly it in 3 splits whose sizes match the previous 80 / 10 / 10 setting.

This contrasts with the approach in Subsection 3.2.1 where the same splits are taken in chronological order for the train / validation / test sets.

In this setting, we find consistently worse results, as observed in Table 3.3. For instance, the performance of the BERT / Bi-LSTM model in terms of micro-F1 is 0.19, against the 0.21 observed for the model trained with temporal splits in Table 3.2. We believe that this behavior might be caused by the homophilic nature of social networks: as the network grows, new members join existing communities and over time their vocabularies homogenize, becoming more predictable over time.

Such a behavior would be consistent with previous studies on the socio-linguistic evolution of communities (Danescu-Niculescu-Mizil et al. 2013). These studies predict the duration of the life span of a user in a community given their posting behaviour. Users that do not align with the linguistic expectations of the community deciding to leave, and those that fall in line reinforcing their cultural norms.

Repr.	**Model**	**M-F1**	**m-F1**	**Pre**	**Rec**
—	Random	0.02	0.04	0.02	0.60
BoW	N. Bayes	0.13	0.15	0.12	0.22
	Log. Reg.	0.13	0.15	0.13	0.20
	R. Forest	0.11	0.13	0.12	0.17
TF-IDF	N. Bayes	0.14	0.16	0.13	0.20
	Log. Reg.	0.14	0.16	0.14	0.20
	R. Forest	0.11	0.13	0.12	0.17
FT	DNN Pool	0.10	0.13	0.10	0.21
	Bi-LSTM	0.04	0.05	0.03	**0.32**
BERT	DNN Pool	0.16	0.18	0.16	0.23
	Bi-LSTM	**0.18**	**0.19**	**0.17**	0.24

Table 3.3: Averaged results on the Vent dataset with Random Splits. Best performing models are in **bold**. The metrics are **M**acro F1, **m**icro F1, and micro-averaged **Pre**cision and **Rec**all. All standard deviations ≤ 0.01.

3.2.2 Hierarchy of Emotions in Vent

Vent data features a single emotion provided by the writer, in contrast to datasets annotated by readers like GoEmotions. Futhermore, our multi-label approach produces probabilities for each emotion given a text message. This lets us analyze the complex structure of emotions our model learns from Vent. To do so, we build a (normalized) confusion matrix M, where $M_{i,j}$ is the proportion of observing predicted emotion j when a vent message is labelled with the i-th emotion.

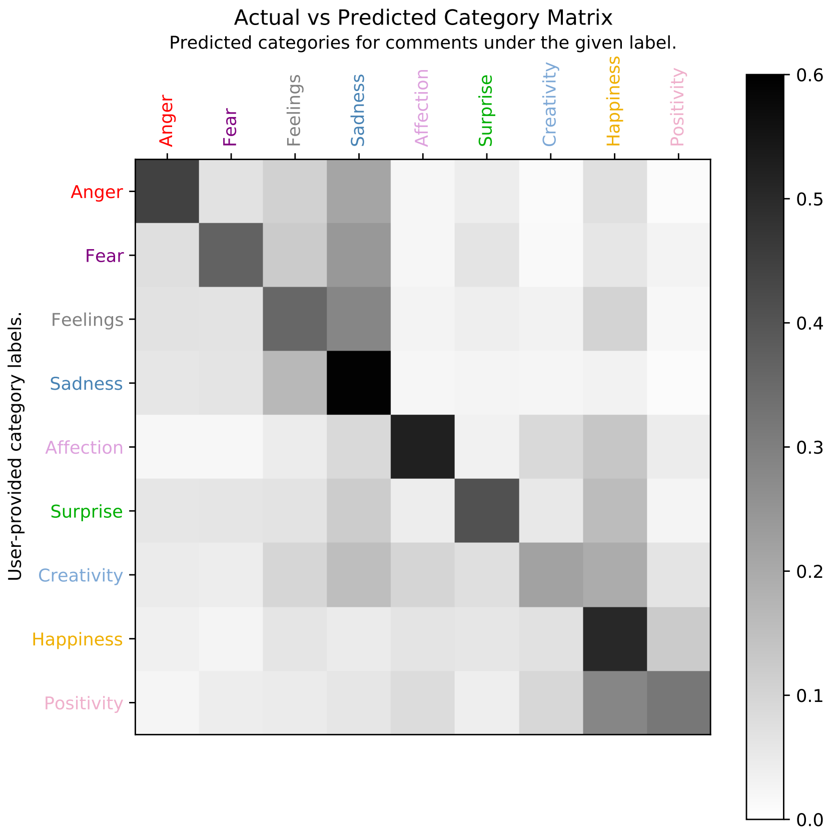


Figure 3.5: Normalized confusion matrix between categories, performing predictions by max-pooling over emotions as indicators of the category.

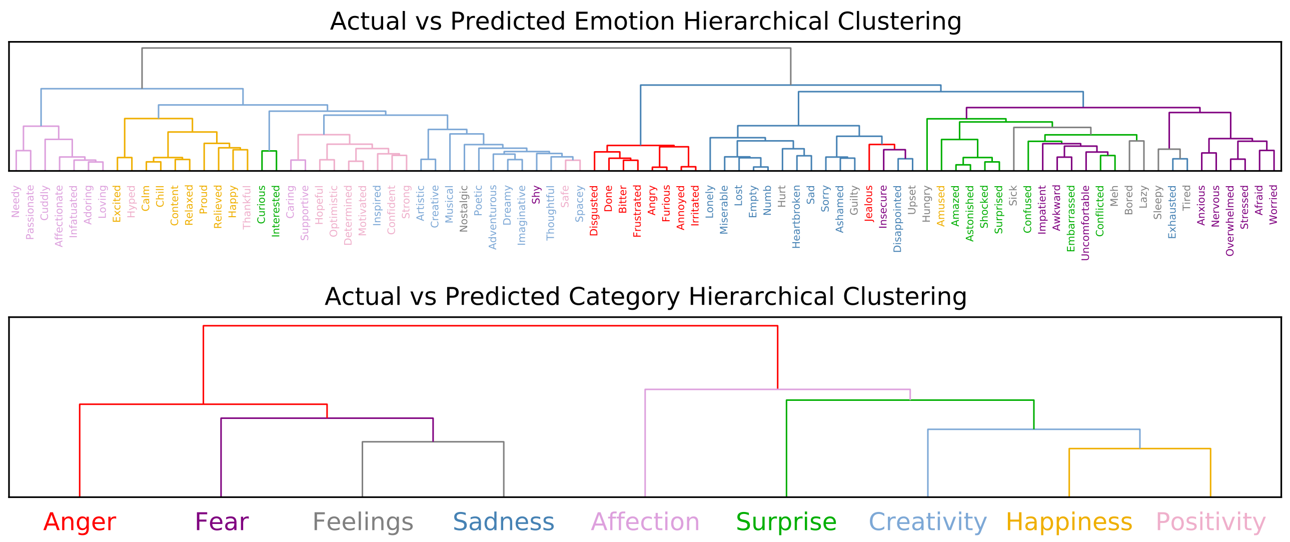


Figure 3.6: Emotion (top) and Category (bottom) dendrograms obtained from the normalized activations of the model captured by the emotion-level equivalent of Figure 3.5. Colors indicate emotion categories.

The resulting confusion matrix at the category level is shown in Figure 3.5 (Appendix A.3 shows results at the emotion level). We observe that negative categories (Anger, Fear, Feelings, and Sadness) and positive categories (Affection, Surprise, Creativity, Happiness, and Positivity) are more often 'confused' within each group than they are across both groups.

Each row in matrix M represents an emotion activation pattern in the target space. We perform agglomerative clustering to discover the hierarchical structure of such space. Figure 3.6 shows the obtained dendrogram using the Euclidean distance between each pair of rows in M as linkage metric. We show dendrograms at the emotion and category levels, where category-level activations are computed by max-pooling the activation patterns across emotions within a category.

Interestingly, clusters of emotions generally align with emotion categories, despite category labels *not being used during training*. Top levels of the hierarchy clearly split positive and negative categories, while subsequent levels provide a sensible hierarchy of emotion clusters at different scales. At the emotion level, we find cases where clusters contain emotions belonging to different categories, but which appear to be semantically meaningful, e.g., the clusters formed by Jealous, Insecure, Disappointed, and Upset, or the cluster formed by Sleepy, Exhausted, and Tired in Figure 3.6.

3.3 Human Evaluation

In this section, we describe our design of an annotation workflow leveraging Human Intelligence Tasks (HIT) on Amazon's Mechanical Turk (MTurk) to evaluate the model and the differences between reader and writer emotions.

HIT Design

We sample 30 Vent comments at random from the test split for each of the 88 emotions, building a dataset of 2,640 comments. We normalize snippets in the same manner as described in Section 3.1. Additionally, we exclude comments that contain either `vent` or `nsfw` (Not Safe For Work) terms to reduce the amount of self-referential or inappropriate content readers are exposed to.

Comments are grouped in HITs of 10 comments, with 5 different readers assigned to each HIT, subject to an approval process to ensure quality annotations. We submit 264 HITs and receive work from 84 different readers with an average emotion accuracy of 11.43%—and we provide extended details on the benchmark in Appendix A.3.1.

Based on reader annotations, we analyse inter-annotator agreement and find that on average 1.81 (± 0.88) readers out of the assigned 5 agree with the most frequently assigned emotion.

At the category level, 2.95 (± 1.03) readers agree on average with the most frequent label. The majority of readers can agree upon an emotion category, while generally not agreeing upon specific emotions within a category. We provide additional details on the inter-annotator agreement of readers in Appendix A.3.1.

Human Annotators vs Model Performance

Interestingly, our best-performing model outperforms human annotators (readers) on the subset of snippets submitted for labelling at the category level–0.395 vs 0.383 micro-F1, respectively. Readers show higher recall (0.725 vs 0.472) but lower precision (0.261 vs 0.356). This is expected, as we do not perform additional post-processing, i.e., filtering to select the majority label, to the readers' annotations. We note that Vent data is collected from a social network with unknown user demographics, emotion labels provided by writers might be noisy and/or biased.

We therefore leverage reader annotations to study the appropriateness of Vent for learning emotions at scale. To do so, we invert the task: we predict the *annotations provided by readers*, using the model trained on Vent. We seek to measure the overlap between the model, readers, and writers, evaluating whether the model aligns with what readers expect more than the original Vent labels (i.e. writers).

Table 3.4 shows emotion and category level macro- and micro-F1 scores for the model, readers, and writers. The model outperforms readers on the *writer* prediction task—that is, when the labels are given by *writers* and predictors are either our model or readers—by 19.9% and 3.1% at the emotion and category levels in terms of relative micro-F1 (0.181 vs. 0.151 and 0.395 vs. 0.383 respectively).

Task	Label	Predictor	M-F1	m-F1
Emotion	Writer	Reader	0.151	0.151
		Model	0.181	0.181
	Reader	Model	0.241	0.246
		Writer	0.151	0.136
Category	Writer	Reader	0.383	0.383
		Model	0.395	0.395
	Reader	Model	0.467	0.471
		Writer	0.383	0.382

Table 3.4: Comparison of results for predicting reader or writer labels ($N = 2640$) using our proposed model or the other available perspective.

Surprisingly, the model is also capable of predicting the ambiguous emotional labels from *readers* for the same texts, despite *not being explicitly trained on the task*. The model achieves 35.9% higher micro-F1 when evaluated against readers rather than writers at the emotion level (0.246 vs. 0.181), and 19.2% at the category level (0.471 vs. 0.395). As a control experiment, we also compare with writer-provided emotions as a predictor of worker-provided emotions. We observe a micro-F1 gap which may be caused by non-uniform biases from readers towards certain emotions, which impact per-emotion support[1].

Our findings show that models trained on large amounts of writer-provided emotional labels from Vent are capable of capturing emotions perceived by readers. Our model achieves better performance when measured against readers rather than writers in terms of micro-F1, which aligns with existing literature on perspective-based emotion detection. For instance, previous work (Buechel and Hahn 2017a) found readers' perspectives to be superior than writers' in terms of inter-annotator agreement on the VAD emotion model. These findings may explain the predictable nature of worker annotations, and the lower performances we observe when evaluating on writer-provided emotions. Our results are the first to consider both author and reader perspectives on categorical taxonomies in large English corpora

[1]Differences between readers and writers depend on the support (# examples per target label) when micro-averaging, as values for precision/recall swap at the label level.

collected from social media, which to our knowledge had not been studied in the emotion detection literature.

3.4 Data Biases and Ethical Considerations

We present techniques to extract emotion information using text data from two social networks: Reddit and Vent. We believe that the user-generated nature of the data limits the generality of our results. For instance, it is understood that young adult males are over represented in the user base of Reddit (Center 2016). For Vent, no demographic data are available, which should further warrant caution on the part of researchers and practitioners building upon our work. Although there is evidence for language use differences across genders, age-groups and personality traits (Schwartz et al. 2013), data mined from social settings might amplify spurious relationships and lead to incomplete and biased accounts of such differences.
As emotional aspects of language are neither objective nor universal, we expect annotation biases in the target labels. In the case of GoEmotions, each Reddit comment was annotated by either 3 or 5 human judges given a list of 27 emotion definitions. However, the judges all share the same location, despite known cultural differences in the expression and understanding of emotions (Jackson et al. 2019; Scollon et al. 2004). As such, the labels might not generalise across different cultures even when the language–English–is the same. In the case of Vent, emotions are provided by each person according to their own emotional state within a cultural group (the Vent community). In this sense, the usage patterns of may have shaped variation in how writers in Vent conceptualize emotions.
There are privacy and discrimination considerations on both data and models. The GoEmotions dataset is anonymised to ensure that the identity of the commenting users is kept safe. On Vent, user identifiers and hyperlinks in comments are masked to prevent user linking. However, due to the size of the dataset, it is possible that other personal details such as real names are contained in the complete text dumps. As such, the authors of (Lykousas et al. 2019) provide the dataset on a private at-request basis. Large language models retrained on either dataset, and particularly those trained on Vent, might end up encoding personal information in a way that can later be extracted (Carlini et al. 2020). However, we believe

that in the context our work, the risk is reduced as we do not fine-tune the underlying language model in a generative task that would promote BERT to memorise parts of Vent.
A broader risk of our work is the potential emotion detection models to amplify and produce abuse. In recent years, generative methods to produce comments for news articles have been published at large academic venues (Yang et al. 2019), raising concerns of their potential to shape public perception or augment the reach of fake news stories. We believe that emotion detection systems may be used to further enhance comment generation models, allowing their designers to adversarially craft content aiming shape the emotional perceptions of readers. Models trained on a large-scale dataset such as Vent might be used to condition generated comments so that they foster a desired kind of emotional discourse, positive or negative. Because of this, we will provide our models on request to interested researchers rather than making them fully available upon publication to limit their usage by unknown third parties.

3.5 Conclusions

This chapter presents a principled analysis of representation learning techniques applied to emotion detection on the two largest available datasets to date: GoEmotions and Vent. Our benchmark shows how different models behave, with BERT-based architectures consistently achieving the best performance across datasets. As a by-product of our work, we release EmotionUI, a web interface for researchers to explore our models[2][3], and share our code, tools, annotations and experiment data[4] as EmotionCore (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2021a)—an open-source framework for emotion analysis we that describe in Appendix A.4. Our results also provide practical directions for fine-tuning language models like BERT on emotion data—and improve performance previous work. On GoEmotions, our best performing model shows an improvement over the previous baseline by 11.8% relative micro-F1. On Vent, we train a model that outperforms readers at predicting emotions provided by writ-

[2]http://emotionui.nur.systems/
[3]https://github.com/nur-ag/emotion-ui/
[4]https://github.com/nur-ag/emotion-classification/

ers by 19.9% relative micro-F1. Surprisingly, the same model shows better performance when evaluated against emotional labels provided by readers by 35.9% relative micro-F1 despite not being trained on this task.
Our findings show that models trained on Vent outperform readers in the task of predicting writer-provided emotions, even though the model performs better on reader-provided emotions rather than the original ground truth labels provided by Vent users (writers). These findings open new research directions on the automatic detection of emotions between readers and writers in discrete emotion taxonomies, and its implications for emotion detection systems. Our work suggests that the task of predicting the emotions that writers aim to express is harder than detecting those perceived by readers. However, the majority of current annotated emotion detection studies focus on reader emotions.
Finally, we released our labelled dataset containing 2640 reader-annotated samples to study emotion detection leveraging the perspectives of readers and writers in Vent—i.e. 88 emotions and 9 categories. Our dataset supplements previous multi-perspective corpora on the Stanford Sentiment Treebank (Sst) and Manually-Anotated Sub-Corpus (Masc), each containing 40 annotated snippets (Buechel and Hahn 2017a,b).

Part II

Representation Learning for Graph Structures

Chapter 4

Background

This chapter provides an overview of graph representation learning, reviewing notation, related work, and relevant concepts before presenting our work. We begin the chapter by providing a brief introduction to modern graph representation learning in Section 4.1. Afterwards, Section 4.2 introduces the necessary notation and definitions. In Section 4.3, we provide an extended literature review and context to discuss existing methods and provide a reference frame for our work. We finish the chapter with Section 4.4, which outlines research gaps in existing work and motivates our research.
To address some of these limitations, we will introduce two algorithms that connect expressivity with graph attributes in the two subsequent chapters. In particular, we will introduce a representation leveraging node degrees and distances between nodes in Chapter 5, and additionally edge positions in Chapter 6. In each chapter, we will introduce our methods, provide details on their algorithmic and theoretical properties, and showcase their performance in several experimental benchmarks.

4.1 Modern Representation Learning on Graphs

Representation learning approaches on graph data have appeared under the paradigms of Graph Neural Networks (Scarselli et al. 2009) and Geometric Deep Learning (Bronstein et al. 2021). These methods learn end-to-end representations of nodes, edges, or graphs, in a data-driven way.

Graph neural networks can efficiently learn representations that consider both graph structures and the features of nodes and edges in large-scale graph data sets. In recent tears, the most popular choice of neural graph architecture is the Message Passing Graph Neural Network (MP-GNN). MP-GNNs represent nodes by repeatedly aggregating feature 'messages' from their neighbours, which has been been successfully applied in a wide variety of domains (Battaglia et al. 2016; Duvenaud et al. 2015; Gilmer et al. 2017; Samanta et al. 2020).

MP-GNNs operate by processing the input graph (data) with the computational graph (model) to learn useful representations via message passing between direct neighbours in the input graph. This identification facilitates the theoretical analysis of MP-GNNs (e.g., in terms of their expressive power (Xu et al. 2019), or the characterization of problematic phenomena such as *over-squashing* (Alon and Yahav 2021) or *over-smoothing* (Oono and Suzuki 2022)). As a result and despite practical successes, theoretical work has shown that there is a limit on the representational power of MP-GNNs bounded by the computationally efficient Weisfeiler-Lehman (1-WL) (Weisfeiler and Leman 1968) graph isomorphism test (Morris et al. 2019; Xu et al. 2019).

Establishing a connection between GNN architectures and their formal expressivity has lead to a better theoretical understanding of the performance of MP-GNNs and many possible generalizations at the price of additional computational cost (Barceló et al. 2020; Brijder et al. 2019; Geerts 2021; Grohe 2017; Morris et al. 2021).

To improve the expressivity of MP-GNNs, recent methods have extended the vanilla message-passing mechanism in various ways. For example, some recent works use higher order k-vertex tuples (Morris et al. 2019) leading to k-WL generalizations, introduce relative positioning information for network vertices (You, Ying, and Leskovec 2019), or propagate messages beyond direct neighborhoods (Nikolentzos, Dasoulas, and Vazirgiannis 2020). Other methods leverage concepts from algebraic topology (Bodnar et al. 2021), or combine sub-graph information in different ways (Bevilacqua et al. 2022; Bouritsas et al. 2021; Frasca et al. 2022; Michel et al. 2023; Vignac, Loukas, and Frossard 2020; Zhang and Li 2021; Zhao et al. 2022). Similarly, Provably Powerful Graph Networks (PPGN) (Maron et al. 2019b) have been proposed as an architecture with

3-WL expressivity guarantees, at the cost of quadratic memory and cubic time complexities with respect to the number of nodes. In (Balcilar et al. 2021), authors proposed GNNML3, a modified MP-GNN with linear time and memory complexity which is more expressive than the 1-WL test, and experimentally as powerful as 3-WL. However, GNNML3 achieves its performance by leveraging a spectral decomposition pre-processing step with cubic worst-case time complexity. All aforementioned approaches improve expressivity by extending MP-GNNs architectures with novel information, often evaluating on standarized benchmarks (Hu et al. 2020a; Morris et al. 2020; You, Ying, and Leskovec 2020). However, identifying the optimal approach on novel domains remains unclear and requires costly architecture search, especially as those expressive architectures introduce additional computation costs.

In recent years, researchers have explored the idea of decoupling the input graph from the computational graph. This is the basis of leading-edge learning approaches, such as Subgraph MP-GNNs (Abboud, Dimitrov, and Ceylan 2022; Frasca et al. 2022; Zhao et al. 2022), perturbation methods (Dwivedi et al. 2022; Papp et al. 2021), or Graph Transformers (Kim et al. 2022; Rampášek et al. 2022; Ying et al. 2021; Yun et al. 2019). These methods extend the message passing mechanism to more general structures induced by the graph, beyond direct neighbors, for example between the nearest neighbours of a node at a given distance (ego-networks), or by enabling nodes to observe the whole graph (graph transformers).

The increase in flexibility extends the expressive power of basic MP-GNNs, at the cost of additional computational footprint and a departure of the inductive bias contained in the input graph, which is required to be learned again. Furthermore, research in this domain has focused on identifying novel architectures that can perform well on some definition of expressivity, whether it is the 1-WL test and its k-WL variants (Morris et al. 2021; Xu et al. 2019), matrix query-languages like MATLANG (Brijder et al. 2019), 2-variable counting logics (Barceló et al. 2020; Grohe 2021), or graph biconnectivity (Zhang et al. 2023).

4.2 Notation and Definitions

In this work, $G = (V, E)$ denotes a graph with $n = |V|$ nodes and $m = |E|$ edges. $l_G(u, v)$ is the shortest path length between two nodes $u, v \in V$. $d_G(v)$ is the degree of v in G and $d_{\mathtt{max}}$ is the maximum degree. Double brackets $\{\!\!\{\cdot\}\!\!\}$ denote multi-sets while $\bigcup$ and $\bigcap$ respectively indicate set and multi-set union and intersection. We use short-hand x^r notation to signify x is contained r times, where $y = \{\!\!\{x^r\}\!\!\}$ reads as "x appears r times in y".

We use $\mathcal{S}_v^k = (\mathcal{V}_v^k, \mathcal{E}_v^k) \subseteq G$ to denote the k-depth induced ego-network sub-graph of G centered on v (abbreviated $\mathcal{S}$ in equations). $\mathcal{V}_v^k$ denotes the neighbours at depth k of v in G. Likewise, we use $\mathcal{S}_{\langle u,v\rangle}^k = (\mathcal{V}_u^k \bigcap \mathcal{V}_v^k, \mathcal{E}_u^k \bigcap \mathcal{E}_v^k)$ to denote the intersection of ego-networks across edge $\langle u, v\rangle$. Feature vectors are shown in **bold**, as $\mathbf{x}_v$ for node v, $\mathbf{x}_{\langle u,v\rangle}$ for edge $\langle u, v\rangle$, we denote vector concatenation by $\|$, and the Hadamard product by $\odot$. Finally, we represent a learnable embedding of a discrete input, e.g., degree or distance signals as $\mathtt{Emb}(\cdot)$, and a learnable weight matrix as $\mathbf{W}$.

4.3 Literature Review and Context

In this section, we extend the overview from Section 4.1. We first provide a formal introduction to Graph Neural Networks and Message-Passing in Subsection 4.3.1. Afterwards, Subsection 4.3.2 provides a review of existing frameworks to analyze the expressivity of GNNs. In each sub-section, we will formally introduce three frameworks: in Section 4.3.2 we introduce the Weisfeiler-Lehman test and its variants, which are the most common approach to study expressivity. Section 4.3.2 introduces the expressivity hierarchy of languages defining matrix operations while Section 4.3.2 introduces the framework of Boolean counting logics as a measure of expressivity.

Finally, Subsection 4.3.3 concludes the section by describing works that have been proposed to improve the expressivity of graph neural networks beyond the Weisfeiler-Lehman test.

4.3.1 Graph Neural Networks

Graph neural networks are deep learning architectures for learning representations on graphs. The most popular choice of architecture is the Message Passing Graph Neural Network (MP-GNN). In MP-GNNs, there is a direct correspondence between the connectivity of network layers and the structure of the input graph. Because of this, the representation (embedding) of each node depends directly on its neighbors and only indirectly on more distant nodes, as shown in Figure 4.1.

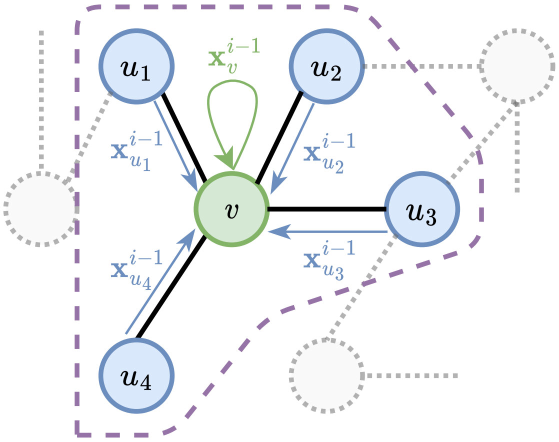


Figure 4.1: A diagram of an MP-GNN updating the representation of node v based on its 4 neighbours. The representation of v at step i, $\mathbf{x}_v^i$ will depend on its previous representation $\mathbf{x}_v^{i-1}$ and the representation of its neighbours: $\{\mathbf{x}_u^{i-1} | \forall u \in \mathcal{V}_v^1\}$.

Each layer of an MP-GNN computes an embedding for a node by iteratively aggregating its attributes and the attributes of their neighboring nodes. Aggregation is expressed via two parametrized functions: Msg, which represents the computation of joint information for a vertex and a given neighbor, and Update, a pooling operation over messages that produces a vertex representation. Let $\mathbf{x}_v^0 \in \mathbb{R}^w$ denote an initial w-dimensional feature vector associated with $v \in V$. Each i-th GNN layer computes the i-th message passing step, such that μ_v^i is the multi-set of messages received by v:

$$\mu_v^i = \left\{\!\!\left\{ \textsc{Msg}_G^i(\mathbf{x}_v^{i-1}, \mathbf{x}_u^{i-1}) \mid \underset{u \neq v}{\forall}\ u \in \mathcal{V}_v^1 \right\}\!\!\right\}, \tag{4.1}$$

and $\mathbf{x}_v^i$ is the output of a permutation-invariant Update function over the message multi-set and the previous vertex state:

$$\mathbf{x}_v^i = \textsc{Update}_G^i\Big(\mu_v^i, \mathbf{x}_v^{i-1}\Big). \tag{4.2}$$

For machine learning tasks that require graph-level embeddings, additional parametrized functions named Readout produce graph-level representations r_G^i, pooling all vertex representations at step i:

$$r_G^i = \textsc{Readout}\Bigg(\left\{\!\!\left\{ \mathbf{x}_v^i \mid \forall\ v \in V \right\}\!\!\right\}\Bigg). \tag{4.3}$$

The functions Msg, Update, and Readout are differentiable with respect to their parameters, which are optimized during learning via gradient descent. The choice of functions gives rise to a broad variety of GNN architectures (Defferrard, Bresson, and Vandergheynst 2016; Gao, Wang, and Ji 2018; Hamilton, Ying, and Leskovec 2017; Kipf and Welling 2017; Veličković et al. 2018). However, it has been shown that all MP-GNNs defined by Msg, Update, and Readout are at most as expressive as the 1-WL test when distinguishing non-isomorphic graphs (Xu et al. 2019).
This increased interest in expressivity within the community in the formal study of MP-GNNs (Papp and Wattenhofer 2022), and to boost message-passing formats. For instance, there have been works that introduce with spectral, positional, subgraph, and structural signals, or their combination (Abboud, Dimitrov, and Ceylan 2022; Balcilar et al. 2021; Bevilacqua et al. 2022; Bodnar et al. 2021; Dwivedi et al. 2022; Frasca et al. 2022; Li et al. 2020; Morris et al. 2019; Nikolentzos, Dasoulas, and Vazirgiannis 2020; Rampášek et al. 2022; Ying et al. 2021; You, Ying, and Leskovec 2019; Zhang and Li 2021; Zhao et al. 2022). A more detailed description of these works will be presented in Subsection 4.3.3 and describe how they connect to our work in the following chapters.
In the next sub-sections, we also summarize related work on the theoretical ability of models to express certain computations—*expressivity* in the abstract—and describe methods that improve the theoretical expressivity and empirical performance of MP-GNNs.

4.3.2 Expressivity In Graph Representation Learning

GNN expressivity has commonly been studied throgh the 1-WL test (Weisfeiler and Leman 1968) and its k-WL variants (Morris et al. 2019)—capturing whether GNNs can distinguish certain non-isomorphic graphs. Other perspectives have explored the sequences of matrix operations that GNNs can compute (Balcilar et al. 2021), or in terms of graph biconnectivity, capturing the ability to identify cut nodes and edges of the graph as proposed with GD-WL (Zhang et al. 2023). Finally, expressivity has also been explored through the lens of 2-variable counting logics (Barceló et al. 2020; Grohe 2021)—studying what Boolean statements that MP-GNNs can express. We summarize these frameworks in the rest of the section.

Weisfeiler-Lehman and its variants

The classic Weisfeiler-Lehman algorithm (1-WL), also known as color refinement, is shown in Algorithm 4.1. The algorithm starts with an initial color assignment to each vertex c_v^0 according to their degree and proceeds updating the assignment at each iteration.

Algorithm 4.1 1-WL (Color refinement).

```
Input: $G = (V, E)$
1: $c_v^0 := \texttt{hash}(\{\!\{d_G(v)\}\!\}) \ \forall\ v \in V$
2: do
3:     $c_v^{i+1} := \texttt{hash}(\{\!\{c_v^i, c_u^i : \ \forall\ u \in \mathcal{V}_v^1\}\!\})$
4: while $c_v^i \neq c_v^{i-1}$
Output: $c_v^i : V \mapsto \mathbb{N}$
```

The update aggregates for each v, its color c_v^i and the colors of its neigbours c_u^i, hashing the color multi-set and mapping it into a new color to be used in the next iteration c_v^{i+1}. The algorithm converges to a stable color assignment which can be used to test graphs for isomorphism. Note that neighbor aggregation in 1-WL can be understood as a message passing step, where `hash` is analogous to Update steps in MP-GNNs.

Two graphs, G_1, G_2, are not isomorphic if they are distinguishable (that is, their stable color assignments do not match). However, if they are not distinguishable (that is, their stable color assignments match), they are

likely to be isomorphic (Babai and Kucera 1979). To reduce the likelihood of false positives when color assignments match, one can consider k-tuples of vertices instead of single vertices, leading to higher order variants of the WL test (denoted k-WL) which assign colors to k-vertex tuples.
In this case, G_1 and G_2 are said to be k-WL equivalent, denoted $G_1 \equiv_{k-\text{WL}} G_2$, if their stable assignments are not distinguishable. k-WL tests are more expressive than their $(k-1)$-WL counterparts for $k > 2$, with the exception of 2-WL, which is known to be as expressive as the 1-WL color refinement test (Grohe 2017). For more details on the algorithm, we refer to (Huang and Villar 2021; Morris et al. 2021). Expressivity ultimately refers to the families of non-isomorphic but k-WL indistinguishable graphs that a given method *also* fails to distinguish—including GNNs, or any other graph representation (Morris et al. 2021).

Matlang: Expressivity through matrix query-languages

Among the various characterizations of k-WL expressivity, the relationship with sequences of operations expressed in matrix query-languages defined by Matlang (Brijder et al. 2019) has also found applications for graph representation learning (Balcilar et al. 2021).
Matlang is a language of operations on matrices, where sentences are formed by sequences of operations. There exists a subset of Matlang—$\mathbf{ML}_1$—that is as expressive as the 1-WL test. Another subset—$\mathbf{ML}_2$—is strictly more expressive than the 1-WL test but less expressive than the 3-WL test. Finally, there exists another subset $\mathbf{ML}_3$ that is as expressive as the 3-WL test (Geerts 2021).
Specifically, Matlang operations include matrix multiplication $(\cdot)$, addition $(+)$, transpose $(\ ^\intercal)$, element-wise multiplication $(\odot)$, column vector of ones $(\mathbf{1})$, vector diagonalization $(\texttt{diag})$, matrix trace $(\texttt{tr})$, scalar-matrix/vector multiplication $(\times)$, and element-wise function application (f). Given Matlang's operations:

$$\mathbf{ML} = \{\cdot, +, \ ^\intercal, \odot, \mathbf{1}, \texttt{diag}, \texttt{tr}, \times, f\},$$

(Geerts 2021) shows that a language containing $\mathbf{ML}_1 = \{\cdot, \ ^\intercal, \mathbf{1}, \texttt{diag}\}$ is as powerful as the 1-WL test, $\mathbf{ML}_2 = \{\cdot, \ ^\intercal, \mathbf{1}, \texttt{diag}, \texttt{tr}\}$ is strictly more expressive than the 1-WL test, but less expressive than the 3-WL test, and $\mathbf{ML}_3 = \{\cdot, {}^\intercal, \mathbf{1}, \mathtt{diag}, \mathtt{tr}, \odot\}$ is as powerful as the 3-WL test. For $\mathbf{ML}_1, \mathbf{ML}_2, \mathbf{ML}_3$, enriching the language with $\{+, \times, f\}$ has no impact on expressivity. In other words, MATLANG connects the k-WL hierarchy with a *computational* model of the operations that a given family of GNNs can learn to express.

GNNs and Logical Statements

Finally, another framework to explore expressivity has been to identify the kinds of logical statements that a GNN can express about an input graph. In particular, (Barceló et al. 2020; Grohe 2021) studied the kind of Boolean statements MP-GNNs can express through counting logics. This framing models expressivity as follows: "*what first-order logic statements about the nodes in a graph can a GNN encode?*"
Logical expressivity has been framed around (a) identifying the properties of nodes and edges, e.g. "*is the node red?*", and then (b) building statements that may consider the structure of the graph: "do all blue nodes have at most two red neighbours at h-hops¿'. This approach has been used to evaluate the ability of GNN models to capture proximity properties, for instance whene valuating the recently published Shortest Path Neural Network model (Abboud, Dimitrov, and Ceylan 2022).

4.3.3 GNNs beyond the Weisfeiler-Lehman test

Besides studying flavours of expressivity, researchers have also focused on improving performance for GNNs. We summarize relevant families of novel network architectures in connection with our work. For a high-level overview on augmented message-passing methods, see (Veličković 2022).

Structural MP-GNNs

Graph Substructure Networks (**GSNs**) (Bouritsas et al. 2021) incorporate topological features by counting local substructures (such as the presence of cliques or cycles). GSNs require expert knowledge on what features are relevant for a given task with modifications to MP-GNN architectures.
GNNML3 (Balcilar et al. 2021) performs message passing in spectral-domain with a custom frequency profile. While this approach achieves

good performance on graph classification, it requires an expensive preprocessing step for computing the eigendecomposition of the graph Laplacian and $\mathcal{O}(k)$-order tensors to achieve k-WL expressiveness, with cubic time complexity.

Sub-graph MP-GNNs

More recently, a series of methods represent vertices or graphs as aggregations over sub-graphs. The sub-graph information is pooled or introduced during message-passing at an additional cost that varies depending on each architecture. Consequently, they require generating the subgraphs (or effectively replicating the nodes of every subgraph of interest) and pay an additional overhead due to the aggregation.
These approaches include **k-hop MP-GNNs (k-hop)** (Nikolentzos, Dasoulas, and Vazirgiannis 2020), which propagate messages beyond immediate vertex neighbors, effectively using ego-network information in the vertex representation; Distance Encoding GNNs (**DE-GNN**) (Li et al. 2020) propose to improve MP-GNN by using extra node features that encode distances to a subset of p nodes; Identity-aware GNNs (**ID-GNNs**) (You et al. 2021), which embed each node incorporating identity information in the GNN and apply rounds of heterogeneous message passing; Nested GNNs (**NGNNs**) (Zhang and Li 2021), which perform a two-level GNN using rooted sub-graphs and consider a graph as a bag of sub-graphs; GNN-as-Kernel (**GNN-AK**) (Zhao et al. 2022), which follow a similar idea but introduce additional positional and contextual embeddings during aggregation; Equivariant Subgraph Aggregation Networks (**ESAN**) (Bevilacqua et al. 2022), encode graphs as bags of sub-graphs showing that this leads to increased expressive power; Shortest Path Neural Networks (**SPNNs**) (Abboud, Dimitrov, and Ceylan 2022) introduced a model aggregating across shortest-path distances, but not edges or messages across neighbours, whose expressivity differs from 1-WL and addresses the over-squashing problem (Alon and Yahav 2021); Subgraph Union Networks (**SUN**) (Frasca et al. 2022), which unify and generalize previous sub-graph GNN architectures and connects them to invariant graph networks (Maron et al. 2019a); and Subgraph Permutation Equivariant Networks (**SPEN**) (Mitton and Murray-Smith 2023), which leverage equivariant sub-graph information locally in contrast with the global permutation equivariance of PPGNs.

Remarkably, recent work (Zhang et al. 2023) has shown that the implicit encoding of the pairwise distance between nodes, plus the degree information which can be extracted via aggregation are fundamental to provide a theoretical justification of **ESAN**. Furthermore, the work on **SUN** (Frasca et al. 2022) showed that *node-based sub-graph* GNNs are provably as powerful as the 3-WL test. This result matches recent analyses on the hierarchies of model expressivity (Papp and Wattenhofer 2022). Finally, another perspective on sub-graph methods has been to enhance network connectivity via edges that are found at different distances from an ego-network root, as in Dynamic Graph Rewiring (**DRew**) (Gutteridge et al. 2023).

Perturbation methods

Beyond sub-graph methods, random perturbations of the graph structure like DropGNN (Papp et al. 2021) or Random Node Initializations (Abboud et al. 2021) have also shown surprising performance in expressivity tasks. Furthermore, random-walks based methods have been shown to be effective at capturing structural information, including positional information (**RWPE**) (Dwivedi et al. 2022).

Graph Transformers

Similarly, the extension of Transformer models to Graph tasks has led to increased research interest, notably with the introduction of Graphormer (Ying et al. 2021)—which included positional and degree encoding similar to Elene, but using only *absolute* in/out degrees. More recently, Pure Graph Transformers (Kim et al. 2022) removed graph-specific architecture choices, directly encoding nodes and edges as tokens processed through self-attention and a global read-out.

Finally, a series of works have yielded high-performance recipes for graph transformers such as GPS (Rampášek et al. 2022)—combining strong inductive biases from MP-GNNs, as well as global and local encoding to build high-performance Graph Transformers. Graph transformers can be understood as *fully-connected* graph processors, which (Abboud, Dimitrov, and Ceylan 2022) showed can be emulated through an SPNN.

4.4 Research Gaps

Graph representation methods at the time this thesis was written have already become extremely versatile, with a wide array of architectures being suitable in many domains. However, these models are nevertheless limited in certain cases, either because of their computational requirements, additional model complexity, or poorly understood theoretical properties. In this section, we highlight some of the limitations and opportunities that we researched as part of this thesis.

First, the approaches to boost the expressivity of MP-GNNs discussed in Subsection 4.3.3 all introduce additional complexity and computation costs either in the model or the computational graph. Some, like perturbation methods, alter the structure of the input graph or its features so that structures that otherwise would be indistinguishable can be identified. Other methods, like spectral, structural, or sub-graph GNNs, introduce new signals in the message-passing mechanism that require additional computation costs. The underlying difference is that the computation graph is no longer defined solely by the input data, as in the standard MP-GNNs presented in Figure 4.1.

Second, approaches to boost expressivity introduce new signals as part of the learning process in the model architecture. This may allow models to represent richer computations. However, it is not clear whether model performance and the ability to distinguish graphs in the k-WL hierarchy requires learning, which involves processing the same sub-graphs during forward and backward passes.

In the next two chapters, we draw inspiration from those opportunities to address the research questions in Section 1.4. We will introduce two ego-network encoding approaches that can be used to boost the expressivity of any graph neural network, both as standalone features or introduced during learning.

Chapter 5

Beyond 1-WL with Local Ego-Network Encodings

5.1 Introduction

Identifying similar network structures is key to capture graph isomorphisms and learn representations that exploit structural information from graph data. Recent methods improve the expressivity of MP-GNNs by extending the vanilla message passing mechanism in the ways described in Subsection 4.3.3. Those approaches improve expressivity by extending specific MP-GNN architectures and evaluating on standarized benchmarks (Hu et al. 2020a; Morris et al. 2020; You, Ying, and Leskovec 2020). However, identifying the optimal approach on novel domains requires costly architecture search. This chapter shows that ego-networks can produce a structural encoding scheme with greater expressivity than the Weisfeiler-Lehman (1-WL) test in any MP-GNN architecture.
We present **IGEL**, an **I**nductive **G**raph **E**ncoding of **L**ocal information. IGEL is a preprocessing step producing features that augment node attributes, encoding ego-networks into sparse vectors that enrich MP-GNNs beyond 1-WL expressivity. We formally describe the relation between IGEL and 1-WL, and characterize its expressive power and limitations. Experiments show that IGEL matches the empirical expressivity of state-of-the-art methods on isomorphism detection while improving performance on seven GNN architectures and five graph machine learning tasks.

The main contributions in this chapter are:

C1 We present a novel structural encoding scheme for graphs, describing its relationship with existing graph representations and MP-GNNs.

C2 We formally show that the proposed encoding has more expressive power than the 1-WL test, and identify expressivity upper-bounds for graphs that are indistinguishable using state of the art methods.

C3 We experimentally asses the performance of eight model architectures enriched with our proposed method on five tasks and ten graph data sets, and find that it consistently improves downstream model performance.

In this chapter, we directly consider distances and degrees in the ego-network, explicitly providing the structural information encoded by more expressive GNN architectures. In contrast to previous work, IGEL aims to be a *minimal* yet *expressive* representation of network structures *without learning* that is amenable to formal analysis as shown in Section 5.3. This connects ego-network properties to sub-graph GNNs, corroborating the 3-WL upper-bound of **SUN**, and the expressivity analysis of **ESAN**—to which IGEL is strongly related in its ego-networks policy (EGO+). Furthermore, these relationships may explain the comparable empirical performance of IGEL to the state-of-the-art, as shown in Section 5.4.
We describe IGEL in Section 5.2 and analyze IGEL's expressivity in Section 5.3. In Section 5.4, we evaluate its performance experimentally while we finally discuss our findings and summarize our results in Section 5.5.

5.2 Local Ego-Network Encodings

The idea behind the IGEL encoding is to represent each vertex v by compactly encoding its corresponding ego network $\mathcal{S}_v^k$ at depth k. The choice of encoding consists of a histogram of vertex degrees at distance $d \leq k$, for each vertex in $\mathcal{S}_v^k$. Essentially, IGEL runs a breadth-first traversal up to depth k, counting the number of times the same degree appears at distance $d \leq k$. We postulate that such a simple encoding is sufficiently expressive and at the same time computationally tractable to be considered as vertex features.

Figure 5.1 shows an illustrative example for $k = 2$. In this example, the green node is encoded using a (sparse) vector of size $5 \times 3 = 15$, since the maximum degree at depth 2 is 5 and there are three distances considered: $0, 1, 2$. The ego network contains six nodes, and all (distance, degree) pairs occur once, except for degree 3 at distance 2, which occurs twice. Algorithm 5.1 describes the steps to produce the IGEL encoding iteratively.

Algorithm 5.1 IGEL Encoding.

```
Input: $G = (V, E), k : \mathbb{N}$
1: $e_v^0 := \{\{(0, d_G(v))\}\} \ \forall\, v \in V$
2: for $i := 1$; $i \mathrel{+}= 1$ until $i = k$ do
3:     $e_v^i := \bigcup(e_v^{i-1}, \{\{(i, d_{\mathcal{S}_v^k}(u)) \ \forall u \in \mathcal{V}_v^k \mid l_G(u, v) = i\}\})$
4: end for
Output: $e_v^k : V \mapsto \{\{(\mathbb{N}, \mathbb{N})\}\}$
```

Similarly to 1-WL, information of the neighbors of each vertex is aggre-

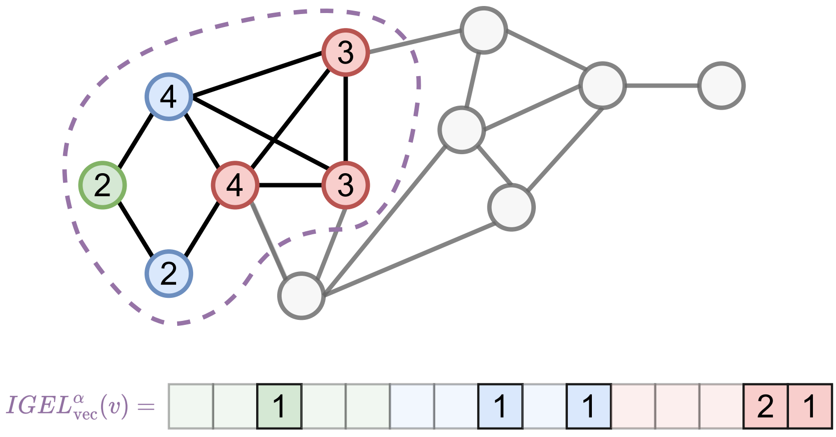


Figure 5.1: IGEL encoding of the green vertex. Dashed region denotes $\mathcal{S}_v^k (k = 2)$. The green vertex is at distance 0, blue vertices at 1 and red vertices at 2. Labels show degrees in $\mathcal{S}_v^k$. The frequency of distance degree (l, d) tuples forming $\text{IGEL}_{\texttt{vec}}^k(v)$ is: $\{(0, 2) : 1, (1, 2) : 1, (1, 4) : 1, (2, 3) : 2, (2, 4) : 1\}$.

gated at each iteration. However, 1-WL does not preserve distance information in the encoding due to the hashing step. Instead of hashing the degrees into equivalence classes, IGEL keeps track of the distance at which a degree is found, generating a more expressive encoding.
The cost of such additional expressiveness is a computational complexity that grows exponentially in the number of iterations. More precisely, the time complexity follows $\mathcal{O}(n \cdot \min(m, (d_{\texttt{max}})^k))$, with $\mathcal{O}(n \cdot m)$ when $k \geq \texttt{diam}(G)$. In Appendix B.1.3, we provide a possible breadth-first search (BFS) implementation that is embarrassingly parallel and thus can be computed over p processors following $\mathcal{O}(n \cdot \min(m, (d_{\texttt{max}})^k)/p)$ time complexity.
The encoding produced by Algorithm 4.1 can be described as a multi-set of path-length and degree pairs (l, d) in the k-depth ego-graph of v, $\mathcal{S}_v^k$:

$$e_v^k = \left\{\!\!\left\{ \left(l_{\mathcal{S}_v^k}(u, v), d_{\mathcal{S}_v^k}(u) \right) \middle| \; \forall\, u \in \mathcal{V}_v^k \right\}\!\!\right\}, \tag{5.1}$$

which also results in exponential space complexity. However, e_v^k can be represented as a (sparse) vector $\text{IGEL}_{\texttt{vec}}^k(v)$, where the i-th index contains the frequency of path-length and degree pairs (l, d)[1], as in the example shown in Figure 5.1:

$$\text{IGEL}_{\texttt{vec}}^k(v)_i = \left| \{\!\{ (l, d) \in e_v^k \text{ s.t. } f(l, d) = i \}\!\} \right|, \tag{5.2}$$

which has linear space complexity $\mathcal{O}(k \cdot n \cdot d_{\texttt{max}})$, conservatively assuming every node requires $d_{\texttt{max}}$ parameters[2] at every k depth from the center of the ego-network. The complexity can be further reduced by making use of sparse vector implementations.
We finish this section with two remarks. First, the produced encodings can differ only for values of $k < \texttt{diam}(G)$, since otherwise the ego-network is the same for all nodes, $\mathcal{S}_v^k = \mathcal{S}_v^{k+1} = G, \forall v$, and thus the resulting encodings, i.e., $e_v^k = e_v^{k+1}$, for $k \geq \texttt{diam}(G)$.

[1] In practice, we may normalize raw counts by e.g. applying log1p-normalization.

[2] $d_{\texttt{max}} = \mathcal{O}(n)$ in the worst-case when a vertex is fully connected. For real-world graphs, $d_{\texttt{max}} \ll n$, and the probability of larger degrees often decays by $\zeta \in \mathbb{R}$, an exponential factor typically within 1 and 3 such that $\mathbb{P}[d_G(v)] \sim d_G(v)^{-\zeta}$ (Albert and Barabási 2002).

Second, the sparse formulation in Equation 5.2 can be understood as an inductive analogue of Weisfeiler-Lehman Graph Kernels (Shervashidze et al. 2010), which we explore in Section 5.3.3.

5.3 Which Graphs are IGEL-Distinguishable?

We now present results about the expressive power of IGEL. We discuss the increased expressivity of IGEL with respect to 1-WL, and identify upper-bounds on the expressivity on graphs that are also indistinguishable under MATLANG and the 3-WL test. We assess expressivity by studying whether two graphs can be distinguished by comparing the encodings obtained by the k-WL test and the IGEL encodings for a given value of k. Similarly to the definition of k-WL equivalence, we say that $G_1 = (V_1, E_1)$ and $G_2 = (V_1, E_1)$ are IGEL-equivalent if the sorted multi-set of node representations is the same for G_1 and G_2:

$$G_1 \equiv^k_{\text{IGEL}} G_2 \iff \{\!\{e^k_{v_1} : \forall v_1 \in V_1\}\!\} = \{\!\{e^k_{v_2} : \forall v_2 \in V_2\}\!\}.$$

5.3.1 Distinguishability on 1-WL Equivalent Graphs

We first show IGEL is more powerful than 1-WL, following Lemma 1 and Lemma 2:

Lemma 1. IGEL *is at least as expressive as 1-WL. For two graphs, G_1, G_2, which are distinguished by 1-WL in k iterations ($G_1 \not\equiv_{\text{1-WL}} G_2$) it also holds that $G_1 \not\equiv^k_{\text{IGEL}} G_2$ for $k = k + 1$. If* IGEL *does not distinguish two graphs G'_1 and G'_2, 1-WL also does not distinguish them: $G'_1 \equiv^k_{\text{IGEL}} G'_2 \Rightarrow G'_1 \equiv_{\text{1-WL}} G'_2$.*

Lemma 2. *There exist at least two non-isomorphic graphs, G_1, G_2, that* IGEL *can distinguish but that 1-WL cannot distinguish; i.e., $G_1 \not\equiv^k_{\text{IGEL}} G_2$ while $G_1 \equiv_{\text{1-WL}} G_2$.*

First, we formally prove Lemma 1, i.e., that IGEL is at least as expressive as 1-WL. For this, we consider a variant of 1-WL which removes the hashing step. This modification can only increase the expressive power of 1-WL and makes it possible to directly compare such (possibly more expressive) 1-WL encodings with the encodings generated by IGEL. Intuitively, after

k color refinement iterations, 1-WL considers nodes at k hops from each node, which is equivalent to running IGEL with $k = k + 1$, i.e., using ego networks that include information of *all* nodes that 1-WL would visit.

Proof of Lemma 1: For convenience, let $c_v^{i+1} = \{\!\{c_v^i; c_u^i \ \forall\ u \in \mathcal{V}_v^1 \mid u \neq v\}\!\}$ be a recursive definition of Algorithm 4.1 where hashing is removed and $c_v^0 = \{\!\{d_G(v)\}\!\}$. Since the hash is no longer computed, the nested multi-sets contain strictly the same or more information as in the traditional 1-WL algorithm.
For IGEL to be less expressive than 1-WL, it must hold that there exist two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ such that $G_1 \not\equiv_{\text{1-WL}} G_2$ while $G_1 \equiv^k_{\text{IGEL}} G_2$.
Let k be the minimum number of color refinement iterations such that $\exists\ v_1 \in V_1$ and $\forall\ v_2 \in V_2, c^k_{v_1} \neq c^k_{v_2}$. We define an equally or more expressive variant of the 1-WL test 1-WL* where hashing is removed, such that $c^k_{v_1} = \{\!\{\{\!\{...\{\!\{d_G(v_1)\}\!\}, \{\!\{d_G(u) \forall u \in \mathcal{V}^1_{v_1}\}\!\}...\}\!\}\}\!\}$, nested up to depth k. To avoid nesting, the multi-set of nested degree multi-sets can be rewritten as the union of degree multi-sets by introducing an indicator variable for the iteration number where a degree is found:

$$\begin{aligned} c^k_{v_1} = & \left\{\!\!\left\{ (0, d_G(v_1)) \right\}\!\!\right\} \bigcup \\ & \left\{\!\!\left\{ (1, d_G(v_1)); (1, d_G(u)) \ \forall\ u \in \mathcal{V}^1_{v_1} \right\}\!\!\right\} \bigcup \\ & \left\{\!\!\left\{ (2, d_G(v_1)); (2, d_G(u)) \ \forall\ u \in \mathcal{V}^1_{v_1}; (2, d_G(w)) \ \forall\ w \in \mathcal{V}^1_u \right\}\!\!\right\} ... \end{aligned}$$

At each step i, we introduce information about nodes up to distance i of v_1. Furthermore, by construction, nodes will be visited on every subsequent iteration—i.e., for $c^2_{v_1}$, we will observe $(2, d_G(v_1))$ exactly $d_G(v_1) + 1$ times, as all its $d_G(v_1)$ neighbors $u \in \mathcal{V}^1_v$ encode the degree of v_1 in c^1_u. The flattened representation provided by 1-WL* is still equally or more expressive than 1-WL, as it removes hashing and keeps track of the iteration at which a degree is found.
Let IGEL-W be a less expressive version of IGEL that does not include edges between nodes at $k+1$ hops of the ego network root. Now, consider the case in which $c^k_{v_1} \neq c^k_{v_2}$ from 1-WL*, and let $k = k+1$ so that IGEL-W considers

degrees by counting edges found at k to $k+1$ hops of v_1 and v_2. Assume that $G_1 \equiv^k_{\text{IGEL-W}} G_2$. By construction, this means that $\{\{e^k_{v_1} : \forall\, v_1 \in V_1\}\} = \{\{e^k_{v_2} : \forall\, v_2 \in V_2\}\}$. This implies that all degrees and iteration counts match as per the distance indicator variable at which the degrees are found, so $c^k_{v_1} = c^k_{v_2}$ which contradicts the assumption $c^k_{v_1} \neq c^k_{v_2}$ and therefore implies that also $G_1 \equiv_{\text{1-WL}^*} G_2$. Thus, $G_1 \equiv^k_{\text{IGEL-W}} G_2 \Rightarrow G_1 \equiv_{\text{1-WL}^*} G_2$ for $k = k+1$ and also $G_1 \not\equiv_{\text{1-WL}^*} G_2 \Rightarrow G_1 \not\equiv^k_{\text{IGEL-W}} G_2$. Therefore, by extension IGEL is at least as expressive as 1-WL. $\square$

To prove Lemma 2, we show graphs that IGEL can distinguish despite being indistinguishable by 1-WL and the MATLANG sub-languages $\mathbf{ML}_1$ and $\mathbf{ML}_2$. In Section 5.3.1, we provide an example where IGEL distinguishes 1-WL/$\mathbf{ML}_1$ equivalent graphs, while Section 5.3.1 shows that IGEL also distinguishes graphs that are known to be distinguishable in the strictly more expressive $\mathbf{ML}_2$ language.

$\mathbf{ML}_1$ / 1-WL Expressivity: *Decalin and Bicyclopentyl*

Decalin and Bycyclopentyl (in Figure 5.2) are two molecules whose graph representations are not distinguishable by 1-WL despite their simplicity. The graphs are non-isomorphic, but 1-WL identifies 3 equivalence classes in both graphs: central nodes with degree 3 (purple), their neighbors (blue), and peripheral nodes farthest from the center (green).
Figure 5.2 shows the resulting IGEL encoding for the central node (in purple) using $k = 1$ (top) and $k = 2$ (bottom). For $k = 1$, the encoding field of IGEL is too narrow to identify substructures that distinguish the two graphs (Figure 5.3, top). However, for $k = 2$ the representations of central nodes differ between the two graphs (Figure 5.3, bottom). In this example, any value of $k \geq 2$ can distinguish between the graphs.

$\mathbf{ML}_2$ Expressivity: *Cospectral 4-Regular Graphs*

IGEL can also distinguish $\mathbf{ML}_2$-equivalent graphs. Recall that $\mathbf{ML}_2$ is strictly more expressive than 1-WL, as described in Subsection 4.3.2. It is known that d-regular graphs of the same cardinality are indistinguishable by the 1-WL test in Algorithm 4.1 and that co-spectral graphs cannot be distinguished in $\mathbf{ML}_2$:

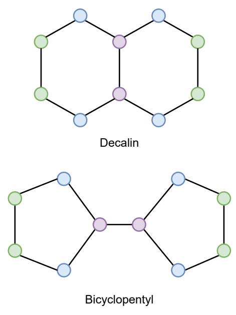


Figure 5.2: Decalin (top) and Bicyclopentyl (bottom). 1-WL (Algorithm 4.1) and produces equivalent colorings in both graphs, hence they are 1-WL equivalent. The colorings match between central nodes (purple), their immediate neighbours (blue), and peripheral nodes farthest from the center (green).

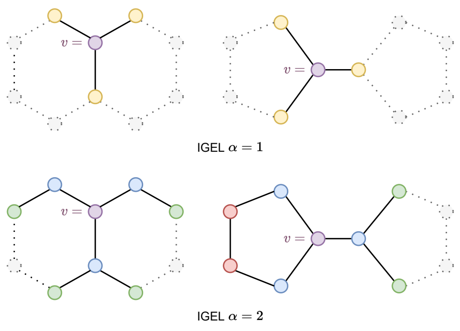


Figure 5.3: IGEL encodings ($k \in \{1, 2\}$) for Decalin and Bicyclopentyl computed for purple vertices (v). Dotted sections are not encoded. Colors denote different (l, d) tuples. IGEL$(k = 2)$ distinguishes the graphs since Decalin nodes at distance 2 from v have degree 1 (green) while their Bicyclopentyl counterparts have degrees 1 (green) and 2 (red).

Definition 1. *$G = (V, E)$ is d-regular with $d \in \mathbb{N}$ if $\forall\ v \in V, d_G(v) = d$.*

Remark 1. *For any pair of n-vertex d-regular graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, $G_1 \equiv_{1-WL} G_2$ (see (Grohe 2017), Example 3.5.2, p. 81).*

Definition 2. *Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are co-spectral if their adjacency matrices have the same multi-set of eigenvalues.*

Remark 2. *For any pair of n-vertex co-spectral graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, $G_1 \equiv_{\boldsymbol{ML}_2} G_2$ (see (Geerts 2021), Proposition 5.1).*

In contrast to 1-WL, we can find examples of non-isomorphic d-regular graphs that IGEL can distinguish, as the generated encodings will differ for any pair of graphs whose sets of sorted degree sequences do not match at any path length less than k. Furthermore, we can find examples of co-spectral graphs that can be distinguished by IGEL encodings.

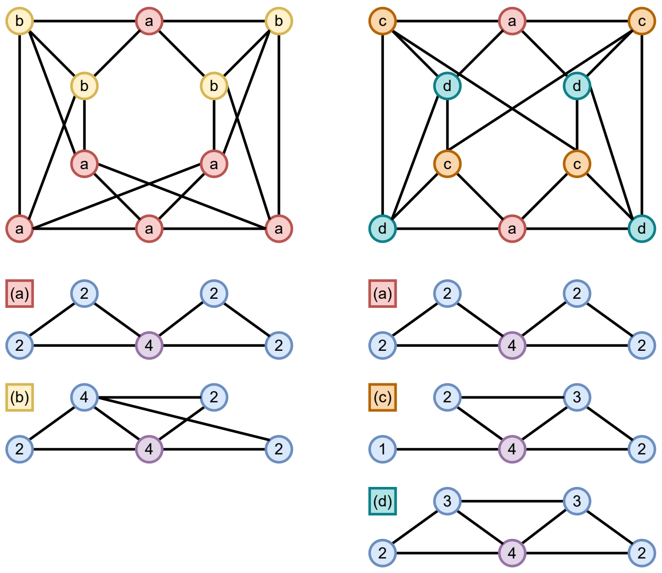


Figure 5.4: IGEL ($k = 1$) encodings for two co-spectral 4-regular graphs from (Van Dam and Haemers 2003). IGEL distinguishes four kinds of distance / degree structures within the graphs (associated with labels for every purple root node as a, b, c, and d). The two graphs can be distinguished since the encoded structures and their frequencies do not match.

In both cases, the intuition is that the ego-network encoding generated by IGEL discards the edges that connect nodes beyond the sub-graph. Consequently, the generated encoding will depend on the actual connectivity at the boundary of the ego-network, and provide IGEL with increased expressivity compared to other methods.

Figure 5.4 shows two co-spectral 4-regular graphs taken from (Van Dam and Haemers 2003), and the structures obtained using IGEL encodings with $k = 1$ on each graph. The 1-WL test assigns a single color to all nodes and stabilizes after one iteration. Likewise, any $\mathbf{ML}_2$ sentences executed as operations on the adjacency matrices of both graphs produce equal results. However, IGEL identifies four different structures (denoted a, b, c, d). Since the IGEL encodings between both graphs do not match, they are distinguishable. This is the case for any value of $k \geq 1$.

5.3.2 Indistinguishability on *Strongly Regular Graphs*

We identify an upper bound on the expressivity of IGEL: non-isomorphic **S**trongly **R**egular **G**raphs with equal parameters. In this case, two non-isomorphic graphs are not indistinguishable by IGEL.

Definition 3. *A n-vertex d-regular graph is strongly regular—denoted* $\texttt{SRG}(n, d, \lambda, \mu)$*—if all adjacent vertices have* λ *vertices in common and all non-adjacent vertices have* γ *vertices in common.*

Remark 3. *For any* $G = \texttt{SRG}(n, d, \lambda, \mu)$, $\texttt{diam}(G) \leq 2$ *(Brouwer and Van Maldeghem 2022).*

Theorem 1. IGEL *cannot distinguish two SRGs when n, d, and* λ *are the same, and between any value of* μ *(equal or not).*
Formally, given $G_1 = \texttt{SRG}(n_1, d_1, \lambda_1, \mu_1)$, $G_2 = \texttt{SRG}(n_2, d_2, \lambda_2, \mu_2)$*:*
— $G_1 \equiv^1_{\text{IGEL}} G_2 \iff n_1 = n_2 \wedge d_1 = d_2 \wedge \lambda_1 = \lambda_2$*, and*
— $G_1 \equiv^2_{\text{IGEL}} G_2 \iff n_1 = n_2 \wedge d_1 = d_2$*, and*
— *no values of* μ *can be distinguished by* $\equiv^1_{\text{IGEL}}$ *or* $\equiv^2_{\text{IGEL}}$.

Proof. Recall that for any graph G, IGEL encodings are equal for all $k \geq \texttt{diam}(G)$ and per Remark 3, SRGs have diameters of two or less. Let $G = \texttt{SRG}(n, d, \lambda, \mu)$, only $k \in \{1, 2\}$ produce different encodings for nodes of G. By construction, $\forall v$ in G, v has encoding e^k_v—so the encoding of G is $e^k_G = \{\!\{e^k_v\}\!\}^n$. Furthermore, $\{\!\{e^1_v\}\!\}^n$ only encodes n, d and λ, and $\{\!\{e^2_v\}\!\}^n$ only encodes n and d by expanding e^k_v in Algorithm 5.1:
— Let $k = 1$: $\forall\ v \in V, \mathcal{S}^1_v = (V', E')$ *s.t.* $V' = \mathcal{S}^1_v$. Since G is d-regular, v is the center of $\mathcal{S}^1_v$, and has d-neighbours. By def., d neighbours of v have λ shared neighbours with v each, plus an edge with v, and $\mathcal{S}^1_v$ does not include μ edges beyond its neighbours. Thus, for SRGs G_1, G_2 where $n_1 = n_2$, $d_1 = d_2$, and $\lambda_1 = \lambda_2$, $e^1_{G_1} = e^1_{G_2} = \{\!\{e^1_v\}\!\}^n$ where:

$$e^1_v = \left\{\!\!\left\{ (0, d) \right\}\!\!\right\} \bigcup \left\{\!\!\left\{ (1, \lambda + 1) \right\}\!\!\right\}^d .$$

— Let $k = 2$: $\forall\ v \in V, \mathcal{S}^2_v = G$ as $\forall\ u \in V, u \in \mathcal{V}^2_v$ when $\texttt{diam}(G) \leq 2$. G is d-regular, so $\forall\ v \in V, d = d_{\mathcal{S}^2_v}(v) = d_G(v)$. Thus, for any SRGs G_1, G_2 s.t. $n_1 = n_2$ and $d_1 = d_2$, $e^2_{G_1} = e^2_{G_1} = \{\!\{e^2_v\}\!\}^n$ where:

$$e^2_v = \left\{\!\!\left\{ (0, d) \right\}\!\!\right\} \bigcup \left\{\!\!\left\{ (1, d) \right\}\!\!\right\}^d \bigcup \left\{\!\!\left\{ (2, d) \right\}\!\!\right\}^{n-d-1}$$

thus, IGEL with $k \in \{1, 2\}$ can only distinguish between *different* values of n, d and λ. $\square$

5.3.3 Properties and Relationships with other methods

In this section, we identify properties of IGEL and connect the algorithm with several graph representation methods—traditional and neural. Namely, we first show that IGEL is permutation-invariant, and then connect it with Weisfeiler-Lehman graph kernels, SPNNs, and MATLANG.

IGEL is Permutation Equivariant and Invariant

Lemma 3. *Given any $v \in V$ for $G = (V, E)$ and given a permuted graph $G' = (V', E')$ of G produced by a permutation of node labels $\pi : V \to V'$ such that $\forall v \in V \Leftrightarrow \pi(v) \in V'$, $\forall (u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E'$.*
The IGEL *representation is permutation equivariant at the graph level*

$$\pi(\{\!\{e^k_{v_1}, \dots, e^k_{v_n}\}\!\}) = \{\!\{e^k_{\pi(v_1)}, \dots, e^k_{\pi(v_n)}\}\!\}.$$

The IGEL *representation is permutation invariant at the node level*

$$e^k_v = e^k_{\pi(v)}, \forall v \in G.$$

Proof. Note that e^k_v in Algorithm 5.1 can be expressed recursively as:

$$e^k_v = \left\{\!\!\left\{ \left(l_{\mathcal{S}^k_v}(u, v), d_{\mathcal{S}^k_v}(u) \right) \middle| \; \forall\, u \in \mathcal{V}^k_v \right\}\!\!\right\}.$$

Since IGEL only relies on node distances $l_G(\cdot, \cdot)$ and degree nodes $d_G(\cdot)$, and both $l_G(\cdot, \cdot)$ and $d_G(\cdot)$ are permutation invariant (at the node level) and equivariant (at the graph level) functions, the IGEL representation is permutation equivariant at the graph level, and permutation invariant at the node level. $\square$

Relationship to Weisfeiler-Lehman Graph Kernels

The Weisfeiler-Lehman algorithm has inspired the design of graph kernels for graph classification, as proposed by (Shervashidze and Borgwardt 2009; Shervashidze et al. 2010).

In particular, Weisfeiler-Lehman graph kernels (Shervashidze et al. 2010) produce representations of graphs based on the labels resulting of evaluating several iterations of the 1-WL test described in Algorithm 4.1. The resulting vector representation resembles IGEL encodings, particularly in vector form—$\text{IGEL}^{k}_{\texttt{vec}}(v)$. In the case of WL kernels, the `hash` function maps sorted label multi-sets in terms of their position in lexicographic order within the iteration. Figure 5.5 illustrates an iteration of the algorithm:

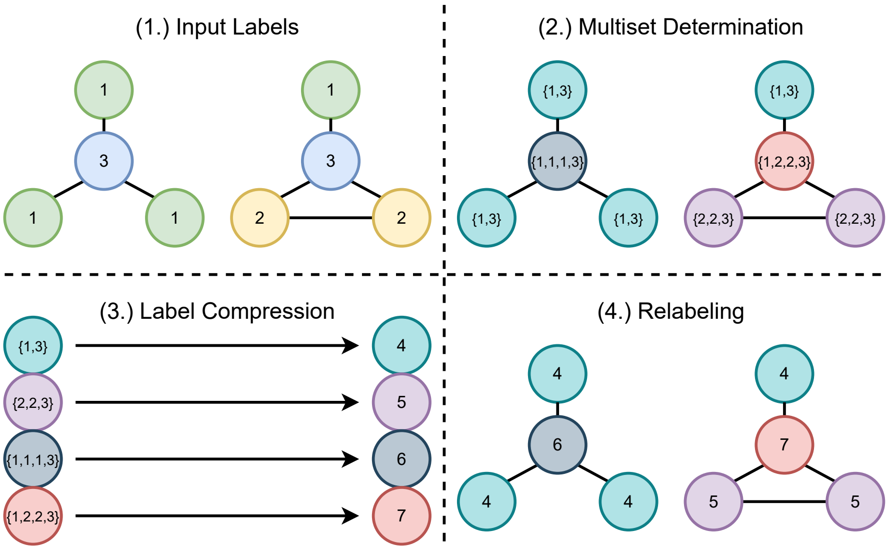


Figure 5.5: One iteration of the Weisfeiler-Lehman graph kernel where input labels are node degrees. Given an input labeling, one iteration of the kernel computes a new set of labels based on the sorted multi-sets produced by aggregating the labels of each node's neighbors.

After k iterations, graphs are represented by counting the frequency of distinct labels c^i_v that were observed at each iteration $1 \leq i \leq k$. The resulting vector representation can be used to compare whether two graphs are similar and as an input to graph classification models. However, WL graph kernels can suffer from generalization problems, as the label compression step assigns different labels to multi-sets that differ on a single element within the multi-set.

If a given graph contains a previously unseen label, each iteration will produce previously unseen multi-sets, propagating at each iteration step and potentially harming model performance. Recent works generalize certain iteration steps of WL graph kernels to address these limitations, introducing topological information (Rieck, Bock, and Borgwardt 2019) or Wasserstein distances between node attribute to derive labels (Togninalli et al. 2019). IGEL can be understood as another work in that direction, removing the hashing step altogether and simply relying on target structural features—path length and distance tuples—that can be inductively computed in unseen or mutating graphs.

Connecting IGEL and SPNNs

It is possible to connect IGEL with the Shortest Path Neural Network (**SPNNs**) model (Abboud, Dimitrov, and Ceylan 2022) and show that IGEL is strictly more expressive than SPNNs on unattributed graphs.

Proposition 1. IGEL *is strictly more expressive than SPNNs on unattributed graphs.*

Proof. First, we show that IGEL encodings contain at least the same the information as SPNNs in unattributed graphs. Let $\mathcal{L}_G^k(v) = \{u | u \in V \wedge l_G(u, v) = k\}$ be the nodes in G exactly at distance k of v. In (Abboud, Dimitrov, and Ceylan 2022), the embedding of v at layer $t+1$ ($\mathbf{h}_v^{t+1}$) in a k-depth SPNN is defined based on the following aggregation over the $1, ..., k$-distance neighbourhoods:

$$(1+\epsilon) \cdot \mathbf{h}_v^t + \sum_{i=1}^{k} \psi_i \sum_{u \in \mathcal{L}_G^i(v)} \mathbf{h}_u^t, \tag{5.3}$$

where ϵ and ψ_i are learnable parameters that modulate aggregation of node embeddings for node v, and node embeddings for nodes at a distance i, respectively.
For unattributed graphs, Equation 5.3 captures information equivalent to counting the frequency of all node distances to node v in $\mathcal{S}_v^k$. This can be written as a reduced IGEL representation following Equation 5.1:

$$\mathtt{SPNN}_v^k = \left\{\!\!\left\{ l_{\mathcal{S}_v^k}(u, v) \middle| \; \forall \; u \in \mathcal{V}_v^k \right\}\!\!\right\}. \tag{5.4}$$

This representation captures an encoding that only considers *distances*. While $\texttt{SPNN}_v^{k=1}$ can be used to compute the degree of $v \in V$ in the entire graph G, it does not capture the degree of any node u only within the ego-network $\mathcal{S}_v^k$. Thus, by definition, the IGEL encoding of Equation 5.1, contains *at least* all the necessary information to construct Equation 5.4—showing that IGEL is at least as expensive as SPNNs in unattributed graphs.

Second, we show that there exist unattributed graphs that IGEL can distinguish but that cannot be distinguished by SPNNs. Figure 5.6 shows an example. Thus, IGEL is strictly more expressive than SPNNs. $\square$

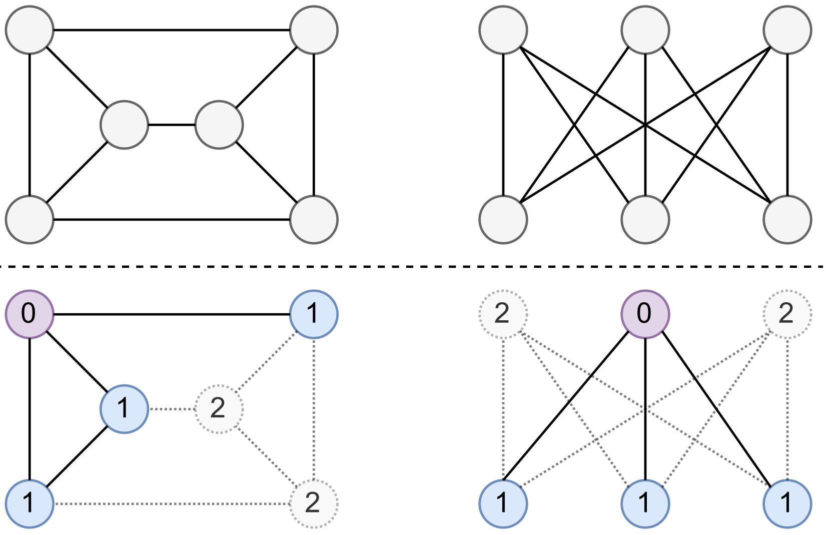


Figure 5.6: IGEL encodings for two SPNN-indistinguishable graphs (Abboud, Dimitrov, and Ceylan 2022) from (Kriege et al. 2018) (top). IGEL with $k = 1$ distinguishes both graphs (bottom) as all ego-networks form tailed triangles on the right graph, and stars on the left. However, SPNNs (as well as 1-WL) fail to distinguish them as all ego-network roots (purple, distance 0) have three adjacent nodes (blue, distance 1) and two non-adjacent nodes at 2-hops (dotted, distance 2).

IGEL and MATLANG

A natural question is to explore how IGEL relates to MATLANG—whether IGEL can be expressed in $\mathbf{ML}_1$, $\mathbf{ML}_2$ or $\mathbf{ML}_3$, and whether some MATLANG operations (and hence languages containing them) can be reduced to IGEL. We focus on the former to evaluate whether IGEL can be computed for a given node as an expression in MATLANG.

Can IGEL be represented in MATLANG?

Let $A \in \{0,1\}^{n\times n}$ be the adjacency matrix of $G = (V, E)$ where $n = |V|$. Our objective is to find a sequence of operations in:

$$\mathbf{ML} = \{\cdot, +, {}^{\intercal}, \odot, \mathbf{1}, \mathtt{diag}, \mathtt{tr}, \times, f\}$$

such that when applied to A, we can express the same information as $\text{IGEL}^{k}_{\mathtt{vec}}(v)$ for all $v \in V$. To compute the IGEL encoding of v for a given k, we must write **ML** sentences to compute:

(a) $\mathcal{S}^k_v = (V', E')$, the ego-network of v in G at depth k.

(b) The degrees of every node in $\mathcal{S}^k_v$.

(c) The shortest path lengths from every node in $\mathcal{S}^k_v$ to v.

Let $A^k_v \in \{0,1\}^{n\times n}$ denote the adjacency matrix of $\mathcal{S}^k_v$. Provided we can compute A^k_v from A, computing the degree is a trivial operation in $\mathbf{ML}_1$, as we only need to compute the matrix-vector product $(\cdot)$ of A^k_v with the vector of ones $(\mathbf{1})$. Recall that $d_{\mathcal{S}^k_v}(u)$ is the degree of vertex $u \in \mathcal{S}^k_v$, and let $(\cdot)_i$ denote the i-th index in the resulting vector:

$$d_{\mathcal{S}^k_v}(u) = (A^k_v \cdot \mathbf{1})_u$$

Furthermore, if we can find A^k_v where $k = 1$, it is also possible to find $A^k_v\ \forall\ k \in \{2, ..., k\}$. Thus, we can compute the distance of every node to v by progressively expanding k, mapping the degrees of nodes to the value of k being processed by means of unary function application (f), and then computing the minimum value. Let ξ denote a distance vector containing entries for all vertices found at distance k of v:

$$\xi_{\mathcal{S}_v^k} = A_v^k \cdot \mathbf{1} f_{\mathtt{k}}$$

where:

$$f_{\mathtt{k}}(x) = \begin{cases} k+1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

The distance $l_{\mathcal{S}_v^k}(u)$ between a given node u to v in $\mathcal{S}_v^k$ can then be computed in terms of $\mathbf{ML}_1$ operations that may be applied recursively:

$$l_{\mathcal{S}_v^k}(u) = \left(\xi_{\mathcal{S}_v^k} - l_{\mathcal{S}_v^{k-1}}(u)\right) f_{\mathtt{min}}^k$$

Where $f_{\mathtt{min}}^k$ is a unary function that retrieves the minimum length from having computed $k - l_{\mathcal{S}_v^{k-1}}(u)$:

$$f_{\mathtt{min}}^k(x) = \begin{cases} k & \text{if } x = k \\ k - x & \text{if } x < k \wedge x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Thus, computing the distance is also possible in $\mathbf{ML}_1$ given A_v^k. However, in order to compute A_v^k, we must be able to extract a subset of V contained in $\mathcal{S}_v^k$. Following the approach of (Geerts 2021) (see Proposition 7.2), we may leverage a node indicator vector $\mathbf{1}_{V'} \in \{0,1\}^n$ for $V' \subseteq V$ where $\mathbf{1}_{V'_v} = 1$ when $v \in V'$ and 0 otherwise. We can then compute A_v^k via a diagonal mask matrix $M \in \{0,1\}^{n \times n}$ such that $M_{i,i} = 1$ when $v_i \in V'$, and $M_{i,i} = 0$ when $v_i \notin V'$, computed as $M = \mathbf{1}_{V'} \cdot \mathbf{1}_{V'}^{\intercal}$. Finding A_v^k involves:

$$A_v^k = M \cdot A \cdot M \tag{5.5}$$

It follows from Equation 5.5 that if we are provided M, it is possible to compute A_v^k in $\mathbf{ML}_1$. However, since indicator vectors are not part of $\mathbf{ML}$, it is not possible to extract the ego-subgraph for a given node v. As such IGEL cannot be expressed within MATLANG unless indicator vectors are introduced.

A natural question is whether it is possible to express IGEL with an operator that requires no knowledge of V' unlike indicator vectors, which require computing $\mathcal{S}_v^k = (V', E')$ beforehand. One possible approach is to only allow indicator vectors for single vertices, encoding any $V' \subseteq V$ only if $|V'| = 1$. We denote single-vertex indicator vectors as $\texttt{one-hot}_v$—an operation that represents the one-hot encoding of v. Note that for any $V' \subseteq V$, its indicator vector $\mathbf{1}_{V'}$ can be computed as the sum of one-hot encoding vectors: $\mathbf{1}_{V'} = \sum_{v \in V'} \texttt{one-hot}_v$. Thus, introducing `one-hot` is as expressive as introducing indicator vectors.
We now express IGEL in terms of the `one-hot` operation. Consider the following $\mathbf{ML}_1$ expression where $\mathbf{Z}_v^0 = \texttt{one-hot}_v$:

$$\mathbf{Z}_v^{i+1} = \Big(A \cdot \mathbf{Z}_v^i\Big) f_{\texttt{bin}}$$

For $\mathbf{Z}_v^1$, we obtain an indicator vector containing the neighbours of v—matching $\mathcal{V}_v^1$—which is binarized (mapping non-zero values to 1—e.g. applying $f_{\texttt{bin}}$[3]). Furthermore, when computed recursively for k steps, $\mathbf{Z}_v^k$ matches the indicator vector of $\mathcal{V}_v^k$—that can trivially be used to compute $\xi_{\mathcal{S}_v^k}$. We can then find M as used in Equation 5.5:

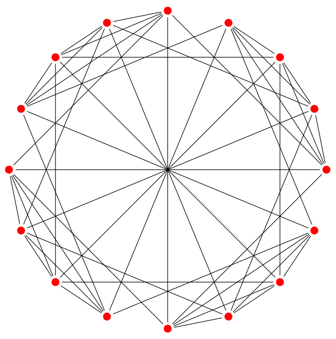
$$M = \bigg(\texttt{diag}\Big(\mathbf{Z}_v^k\Big)\bigg) f_{\texttt{bin}}$$

Thus introducing $\texttt{one-hot}_v$ is sufficient to express IGEL within MATLANG as $\mathbf{ML}_1$. Given our results in Section 5.3 and Section 5.4, introducing $\texttt{one-hot}_v$ in MATLANG produces matrix languages that are at least as expressive as IGEL. We leave the study of the expressivity of MATLANG after introducing a transductive operation such as $\texttt{one-hot}_v$ as future work.

5.3.4 Expressivity Implications

As expected from Theorem 1, IGEL cannot distinguish the Shrikhande and 4×4 Rook graphs (shown in Figure 6.1), which are known to be $\mathbf{ML}_3$-equivalent graphs (Arvind et al. 2020; Balcilar et al. 2021) with $\texttt{SRG}(16, 6, 2, 2)$ parameters despite not being isomorphic.

[3] $f_{\texttt{bin}}(x)$ returns 0 when x is 0, and 1 otherwise.


(a) 4x4 Rook Graph.

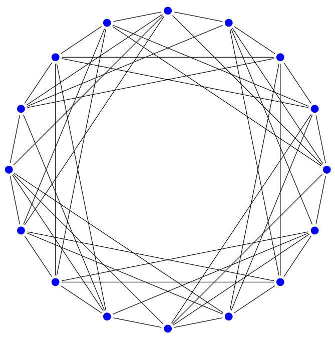

(b) Shrikhande Graph.

Figure 5.7: 3-WL equivalent 4x4 Rook (a, red) and Shrikhande (b, blue) graphs are indistinguishable by IGEL as they are non-isomorphic SRGs with parameters $\texttt{SRG}(16, 6, 2, 2)$.

Our findings show that IGEL is a powerful permutation-equivariant representation (see Section 5.3.3), capable of distinguishing 1-WL equivalent graphs such as Figure 5.4—which as cospectral graphs, are known to be expressable in strictly more powerful MATLANG sub-languages than 1-WL (Geerts 2021). Furthermore, in Section 5.3.3, we connect IGEL to SPNNs (Abboud, Dimitrov, and Ceylan 2022) and show that IGEL is strictly more expressive than SPNNs on unattributed graphs. Finally, we note that the upper bound on Strongly Regular Graphs is a hard ceiling on expressivity since SRGs are commonly known to be indistinguishable by 3-WL (Arvind et al. 2020; Frasca et al. 2022; Morris et al. 2021; Papp and Wattenhofer 2022; Zhao et al. 2022).

IGEL formally reaches an expressivity upper bound on SRGs, distinguishing SRGs with different values of n, d and λ. These results are similar with sub-graph methods implemented within MP-GNN architectures, such as Nested GNNs (Zhang and Li 2021) and GNN-AK (Zhao et al. 2022), which are known to be at last as powerful as 3-WL, and the ESAN framework when leveraging ego-networks with root-node flags as a subgraph sampling policy (EGO+), which is as powerful as 3-WL on SRGs (Bevilacqua et al. 2022). However, in contrast to these methods, IGEL cannot distinguish different values of μ.

In Section 5.4, we study IGEL experimentally and find that the expressivity limitations that IGEL exhibits on SRGs do not have significant implications in downstream tasks. Summarizing, IGEL distinguishes non-isomorphic graphs that are undistinguishable by the 1-WL test. Furthermore, Lemma 2 shows that IGEL can distinguish any graphs that 1-WL can distinguish. We derive a precise expressivity upper-bound for IGEL on SRGs, showing that IGEL cannot distinguish SRGs with equal parameters, or between values of μ. Overall, the expressive power of IGEL on SRGs is similar to other state-of-the-art methods including k-hop GNNs (Nikolentzos, Dasoulas, and Vazirgiannis 2020), GSNs (Bouritsas et al. 2021), NGNNs (Zhang and Li 2021), GNN-AK (Zhao et al. 2022) and ESAN (Bevilacqua et al. 2022).

5.4 Empirical Validation

We evaluate IGEL as a sparse local ego-network encoding following Equation 5.2, extending vertex/edge attributes on six experimental tasks: graph classification, graph isomorphism detection, graphlet counting, graph regression, link prediction, and vertex classification. With our experiments, we seek to evaluate the following empirical questions:

Q1. Does IGEL improve MP-GNN performance on graph-level tasks?

Q2. Can we empirically validate our results on the expressive power of IGEL compared to 1-WL?

Q3. Are IGEL encodings appropriate features for unattributed graphs?

Q4. How do GNN models compare with more traditional neural network models when they are enriched with IGEL features?

5.4.1 Overview of the Experiments

For graph classification, isomorphism detection, and graphlet counting, we reproduce the benchmark proposed by (Balcilar et al. 2021) on eight graph data sets. For each task and data set, we introduce IGEL as vertex/edge attributes, and compare the performance of including or excluding IGEL

on several GNN architectures—including linear and MLP baselines (without message-passing), GCNs (Kipf and Welling 2017), GATs (Veličković et al. 2018), GINs (Xu et al. 2019), Chebnets (Defferrard, Bresson, and Vandergheynst 2016), and GNNML3 (Balcilar et al. 2021). We measure whether IGEL improves inductive MP-GNN performance while validating our theoretical expressivity analysis (**Q1.** & **Q2.**). We also evaluate on the ZINC-12K and PATTERN datasets from Benchmarking GNNs (Dwivedi et al. 2020) to test IGEL on larger, real-world datasets.
For link prediction, we experiment on two unattributed social network graphs. We train self-supervised embeddings on IGEL encodings and compare them against standard transductive vertex embeddings[4]. We detail the self-supervised embedding approach in Appendix B.3. We compare our results against strong vertex embedding models, namely DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and Node2Vec (Grover and Leskovec 2016), seeking to validate IGEL as a theoretically grounded structural feature extractor in unattributed graphs (**Q3.**).
For vertex classification, we train using IGEL encodings and vertex attributes as inputs on DNN models without message-passing on an inductive Protein-to-Protein Interactions (PPI) multi-label classification problem. We evaluate the impact of introducing IGEL on top of vertex attributes, and compare the performance of IGEL-inclusive models with MP-GNNs (**Q4.**).

5.4.2 Experimental Methodology

On graph-level tasks, we introduce IGEL encodings concatenated to existing vertex features[5] into the best performing model configurations found by (Balcilar et al. 2021) without any hyper-parameter tuning (e.g number of layers, hidden units, choice pooling and activation functions). We evaluate performance differences with and without IGEL on each task, data set and model on 10 independent runs, measuring statistical significance of

[4] *Transductive* methods require that all nodes in the graph are known at training and inference time, while *inductive* methods can be applied to unseen nodes, edges, and graphs. Inductive methods may be applied in transductive settings, but not vice-versa. Since IGEL is label and permutation invariant, its output is an inductive representation.

[5] We also introduce IGEL as edge-level features representing an edge as the element-wise product of node-level IGEL encodings at each end of the edge.

the differences through paired t-tests. On Benchmark datasets, we reuse the best reported GIN-AK^{+} baseline from (Zhao et al. 2022) and simply introduce IGEL as additional node features with $k \in \{1, 2\}$, with no hyper-parameter changes.
On vertex and edge-level tasks, we report best performing configurations after hyper-parameter search. Each configuration is evaluated on 5 independent runs. Our results are compared against strong standard baselines from the literature, and we provide a breakdown of the best performing hyper-parameters found in Appendix B.1.

Results and Notation.

In the following sub-sections, text formatting denotes significant (as per paired t-tests) **positive (in bold)**, *negative (in italic)*, and insignificant differences (no formatting) after introducing IGEL. On each task and dataset, the best results are underlined.

5.4.3 Graph Classification: TU Graphs

We evaluate IGEL on four graph classification tasks on the TU molecule data sets (Morris et al. 2020). Nodes represent atoms and edges represent their atomic bonds. There are no edge features while node features are a one-hot encoded vector of the atom represented by that node. Table 5.1 shows our results, where we evaluate differences in mean accuracies with and without IGEL through paired t-tests, denoting significance intervals of $p < 0.01$ as * and $p < 0.0001$ as $^{\diamond}$.
Our results show that IGEL in the Mutag and Proteins data sets improves the performance of all MP-GNN models, including GNNML3, contributing to answer **Q1.** By introducing IGEL in those data sets, MP-GNN models reach similar performance to GNNML3. Introducing IGEL achieves this at $\mathcal{O}(n \cdot \min(m, (d_{\mathtt{max}})^k))$ preprocessing costs compared to $\mathcal{O}(n^3)$ worst-case eigen-decomposition costs associated with GNNML3's spectral supports.
As IGEL is an inductive method, the worst-case $\mathcal{O}(n \cdot (d_{\mathtt{max}})^k)$ when $k < \mathtt{diam}(G)$ cost is only required when the graph is first processed. Afterwards, encodings can be reused, recomputing them for nodes neighbouring new nodes or updated edges within k. This contrasts with GNNML3's spectral supports, which must be recalculated when nodes or edges change.

Table 5.1: Per-model classification accuracy metrics on TU data sets. Each cell shows the average accuracy of the model and data set in that row and column, with IGEL (left) and without IGEL (right).

Model	**Enzymes**	**Mutag**	**Proteins**	**PTC**
MLP	**41.10>26.18**$^{\diamond}$	**87.61>84.61**$^{\diamond}$	75.43~75.01	**64.59>62.79**$^{\diamond}$
GCN	**54.48>48.60**$^{\diamond}$	**89.61>85.42**$^{\diamond}$	**75.67>74.50***	65.76~65.21
GAT	54.88~54.95	**90.00>86.14**$^{\diamond}$	**73.44>70.51**$^{\diamond}$	66.29~66.29
GIN	**54.77>53.44***	89.56~88.33	**73.32>72.05**$^{\diamond}$	61.44~60.21
Chebnet	61.88~62.23	**91.44>88.33**$^{\diamond}$	**74.30>66.94**$^{\diamond}$	64.79~63.87
GNNML3	*61.42<62.79*$^{\diamond}$	**92.50>91.47***	**75.54>62.32**$^{\diamond}$	*64.26<66.10*$^{\diamond}$

On the Enzymes and PTC data sets, results are mixed: for all models other than GNNML3, IGEL either significantly improves accuracy (on MLPNet, GCN, and GIN on Enzymes), or does not have a negative impact on performance. GAT outperforms GNNML3 in PTC, while GNNML3 is the best performing model on Enzymes. Additionally, GNNML3 performance degrades IGEL is introduced on the Enzymes and PTC data sets. We believe this degradation may be caused by overfitting due to a lack of additional parameter tuning, as GNNML3 models are deeper in Enzymes and PTC (four GNNML3 layers) when compared Mutag and Proteins (three and two GNNML3 layers respectively).
It may be possible to improve GNNML3 performance with IGEL by re-tuning model parameters, but due to computational constraints we do not test this hypothesis. Nevertheless, all models improve in at least two data sets after introducing IGEL without hyper-parameter tuning, which we believe indicates our results are a conservative performance lower-bound.
We also compare the best IGEL results from Table 5.1 with state-of-the-art expressive GNNs. Table 5.2 summarizes the reported results for k-hop GNNs (Nikolentzos, Dasoulas, and Vazirgiannis 2020), GSNs (Bouritsas et al. 2021), NestedGNNs (Zhang and Li 2021), ID-GNNs (You et al. 2021), GNN-AK (Zhao et al. 2022), and ESAN (Bevilacqua et al. 2022). When we compare IGEL and the best performing baseline for every data set, none of the differences are statistically significant ($p > 0.01$) except for ID-GNN in Proteins (where $p = 0.009$).

Table 5.2: Mean ± stddev of best IGEL configuration and state-of-the-art results reported on (Bevilacqua et al. 2022; Bouritsas et al. 2021; Nikolentzos, Dasoulas, and Vazirgiannis 2020; You et al. 2021; Zhang and Li 2021; Zhao et al. 2022) with best performing baselines underlined.

Model	**Mutag**	**Proteins**	**PTC**
IGEL (ours)	92.5 ± 1.2	75.7 ± 0.3	66.3 ± 1.3
k-**hop**^{†,1}	*87.9±1.2*$^{\diamond}$	75.3 ± 0.4	—
GSN^{†,2}	92.2 ± 7.5	76.6 ± 5.0	68.2 ± 7.2
NGNN^{†,3}	87.9 ± 8.2	74.2 ± 3.7	—
ID-GNN^{†,4}	$\underline{93.0 \pm 5.6}$	$\underline{77.9 \pm 2.4}^{*}$	62.5 ± 5.3
GNN-AK^{†,5}	91.7 ± 7.0	77.1 ± 5.7	67.7 ± 8.8
ESAN^{†,6}	91.1 ± 7.0	76.7 ± 4.1	$\underline{69.2 \pm 6.5}$

†: Results as reported by [1]: (Nikolentzos, Dasoulas, and Vazirgiannis 2020), [2]: (Bouritsas et al. 2021), [3]: (You et al. 2021), [4]: (Zhang and Li 2021), [5]: (Zhao et al. 2022), and [6]: (Bevilacqua et al. 2022).

Overall, introducing IGEL yields comparable performance to state-of-the-art methods (**Q2**) without architectural modifications—including when compared to five strong baseline models focused on WL expressivity.

5.4.4 Graph Isomorphism Detection

In this subsection we evaluate IGEL on isomorphism detection tasks on two data sets: Graph8c[6], and EXP (Abboud et al. 2021). On the Graph8c data set, we identify isomorphisms by counting the number of graph pairs for which randomly initialized MP-GNN models produce equivalent outputs. Equivalence is measured by the Manhattan distance between graph on 100 independent initialization runs[7]. The EXP data set contains 1-WL equivalent pairs of graphs where the objective is a balanced binary classification task to detect whether they are isomorphic or not. Table 5.3 reports our results on both tasks.

[6]Simple 8 vertices graphs from: `http://users.cecs.anu.edu.au/~bdm/data/graphs.html`

[7]Models are only initialized and not trained, as in (Balcilar et al. 2021).

On Graph8c, introducing IGEL significantly reduces the amount of graph pairs erroneously identified as isomorphic for all MP-GNN models. Furthermore, IGEL allows a linear baseline employing a sum readout function over input feature vectors, then projecting onto a 10-component space, to identify all but 1571 non-isomorphic pairs compared to the erroneous pairs GCNs (4196 errors) or GATs (1827 errors) can identify without IGEL.

We also find that all Graph8c graphs can be distinguished if the IGEL encodings for $k = 1$ and $k = 2$ are concatenated. We do not study the expressivity of concatenating combinations of k in this chapter, but based on our results we hypothesize it produces strictly more expressive representations. We evaluate this hypothesis in Chapter 6, as concatenating multiple encodings captures information about edges between sub-graphs.

Table 5.3: Graph isomorphism detection results. The IGEL column denotes whether IGEL is used or not in the configuration. For Graph8c, we describe graph pairs erroneously detected as isomorphic. For EXP classify, we show the accuracy of distinguishing non-isomorphic graphs in a binary classification task.

Model	**+ IGEL**	**Graph8c (#Errors)**	**EXP Class. (Accuracy)**
Linear	No	6.242M	50%
	Yes	**1571**	**97.25%**
MLP	No	293K	50%
	Yes	**1487**	**100%**
GCN	No	4196	50%
	Yes	**5**	**100%**
GAT	No	1827	50%
	Yes	**5**	**100%**
GIN	No	571	50%
	Yes	**5**	**100%**
Chebnet	No	44	50%
	Yes	**1**	**100%**
GNNML3	No	0	100%
	Yes	0	100%

On EXP, introducing Igel is sufficient to identify all non-isomorphic graphs for all standard MP-GNN models, as well as the MLP baseline. Furthermore, the linear baseline can reach 97.25% classification accuracy with Igel, despite only computing a global sum readout before a single-output fully connected layer. Results on Graph8c and EXP validate our theoretical claims that Igel is more expressive than 1-WL and can distinguish graphs that would be indistinguishable under 1-WL—answering **Q2**.
We also evaluate Igel on the SR25 data set[8], which contains 15 Strongly Regular 25 vertex non-isomorphic graphs known to be indistinguishable by 3-WL where we can empirically validate Theorem 1. Balcilar et al. (2021) showed that all models in our benchmark are unable to distinguish any of the 105 non-isomorphic graph pairs in SR25. We find that introducing Igel does not improve distinguishability—as expected from Theorem 1.

5.4.5 Graphlet Counting

We evaluate Igel on a graphlet counting regression task aiming to minimize Mean Squared Error (MSE) on the normalized graphlet counts[9]. Table 5.4 shows results of introducing Igel in five graphlet counting tasks on the RandomGraph data set (Chen et al. 2020). We identify 3-stars, triangles, tailed triangles and 4-cycle graphlets, as shown in Figure 5.8, plus a custom structure with 1-WL expressiveness proposed in (Balcilar et al. 2021) to evaluate GNNML3. We highlight statistically significant differences when introducing Igel ($p < 0.0001$).
Introducing Igel improves the ability of 1-WL GNNs to recognize triangles, tailed triangles and custom 1-WL graphlets from (Balcilar et al. 2021). Stars can be identified by all baselines, and introducing Igel only produces statistically significant differences on the linear baseline. Interestingly, Igel on the linear model produces results outperforming MP-GNNs without Igel for star, triangle, tailed triangle and custom 1-WL graphlets. By introducing Igel on the MLP baseline, it obtains the best performance (lower MSE) on the triangle, tailed-triangle and custom 1-WL graphlets, even when compared to GNNML3 *and* sub-graph GNNs—including when Igel encodings are input to GNNML3.

[8]SRGs$(25, 12, 5, 6)$ from: `http://users.cecs.anu.edu.au/~bdm/data/graphs.html`
[9]Counts are normalized by the standard deviation counts across the data set for MSE values to be consistent across graphlet types, as in (Balcilar et al. 2021).

Table 5.4: Graphlet counting results. On every cell, we show mean test set MSE error (lower is better), with stat. sig, ($p < 0.0001$) results **highlighted** and best results per task $\underline{\text{underlined}}$. For comparison, we also report strong literature results from two sub-graph GNNs: GNN-AK$^+$ using a GIN base (Zhao et al. 2022) and SUN (Frasca et al. 2022).

Model	+ IGEL	Star	Triangle	Tailed Tri.	4-Cycle	Custom
Linear	No	1.60×10^{-1}	3.41×10^{-1}	2.82×10^{-1}	2.03×10^{-1}	5.11×10^{-1}
	Yes	$\mathbf{4.23 \times 10^{-3}}$	$\mathbf{4.38 \times 10^{-3}}$	$\mathbf{1.85 \times 10^{-2}}$	1.36×10^{-1}	$\mathbf{5.25 \times 10^{-2}}$
MLP	No	$\underline{2.66 \times 10^{-6}}$	2.56×10^{-1}	1.60×10^{-1}	1.18×10^{-1}	4.54×10^{-1}
	Yes	8.31×10^{-5}	$\underline{\mathbf{5.69 \times 10^{-5}}}$	$\underline{\mathbf{5.57 \times 10^{-5}}}$	$\mathbf{7.64 \times 10^{-2}}$	$\underline{\mathbf{2.34 \times 10^{-4}}}$
GCN	No	4.72×10^{-4}	2.42×10^{-1}	1.35×10^{-1}	1.11×10^{-1}	1.54×10^{-3}
	Yes	8.26×10^{-4}	$\mathbf{1.25 \times 10^{-3}}$	$\mathbf{4.15 \times 10^{-3}}$	$\mathbf{7.32 \times 10^{-2}}$	$\mathbf{1.17 \times 10^{-3}}$
GAT	No	4.15×10^{-4}	2.35×10^{-1}	1.28×10^{-1}	1.11×10^{-1}	2.85×10^{-3}
	Yes	4.52×10^{-4}	$\mathbf{6.22 \times 10^{-4}}$	$\mathbf{7.77 \times 10^{-4}}$	$\mathbf{7.33 \times 10^{-2}}$	$\mathbf{6.66 \times 10^{-4}}$
GIN	No	3.17×10^{-4}	2.26×10^{-1}	1.22×10^{-1}	1.11×10^{-1}	2.69×10^{-3}
	Yes	6.09×10^{-4}	$\mathbf{1.03 \times 10^{-3}}$	$\mathbf{2.72 \times 10^{-3}}$	$\mathbf{6.98 \times 10^{-2}}$	$\mathbf{2.18 \times 10^{-3}}$
Chebnet	No	5.79×10^{-4}	1.71×10^{-1}	1.12×10^{-1}	8.95×10^{-2}	2.06×10^{-3}
	Yes	3.81×10^{-3}	$\mathbf{7.88 \times 10^{-4}}$	$\mathbf{2.10 \times 10^{-3}}$	7.90×10^{-2}	$\mathbf{2.05 \times 10^{-3}}$
GNNML3	No	8.90×10^{-5}	2.36×10^{-4}	2.91×10^{-4}	$\underline{6.82 \times 10^{-4}}$	9.86×10^{-4}
	Yes	9.29×10^{-4}	2.19×10^{-4}	4.23×10^{-4}	6.98×10^{-4}	4.17×10^{-4}
GIN-AK$^+$ (Zhao et al. 2022)	No	1.6×10^{-2}	1.1×10^{-2}	1.0×10^{-2}	1.1×10^{-2}	—
SUN (Frasca et al. 2022)	No	6.0×10^{-3}	8.0×10^{-3}	8.0×10^{-3}	1.1×10^{-2}	—

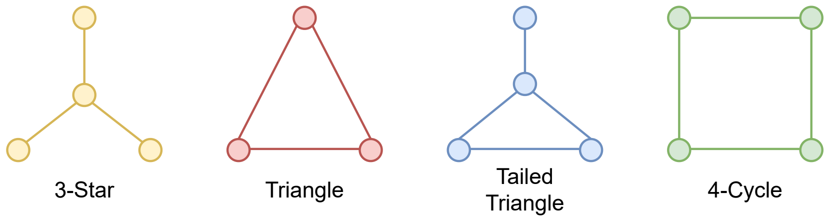


Figure 5.8: Graphlet types in the counting task.

Results on linear and MLP baselines are interesting as neither uses message passing, indicating that raw IGEL encodings may be sufficient to identify certain graph structures in simple linear models. For all graphlets except 4-cycles, introducing IGEL outperforms or matches GNNML3 performance at lower pre-processing and model training/inference costs—without the need for costly eigen-decomposition or message passing, answering **Q1.** and **Q2.** IGEL moderately improves performance counting 4-cycle graphlets, but the results are not competitive when compared to GNNML3.

5.4.6 Benchmark results with Subgraph GNNs

Given the favorable performance of IGEL compared to related sub-graph aware methods such as NGNN and ESAN as shown in Table 5.2, we also explore introducing IGEL on a sub-graph GNN, namely GNN-AK proposed by (Zhao et al. 2022). We follow a similar experimental approach as in previous experiments, reproducing the results of GNN-AK using GINs (Xu et al. 2019) with edge-feature support (Hu et al. 2020b)[10] as the base GNN on two real-world benchmark data sets: ZINC-12K and PATTERN from Benchmarking GNNs (Dwivedi et al. 2020).
Without performing any additional hyper-parameter tuning or architecture search, we evaluate the impact of introducing IGEL with $k \in \{1, 2\}$ on the best performing model configuration and code published by the authors of GNN-AK (Zhao et al. 2022). Furthermore, we also evaluate the setting in which sub-graph information is not used to assess whether IGEL

[10]GINs are the best performing base model in three out of the four data sets, as reported in (Zhao et al. 2022).

can provide comparable performance to GNN-AK without changes to the network architecture. Table 5.5 summarizes our results.

IGEL maintains or improves performance in both cases when introduced on a GIN, but only in the case of PATTERN we find a statistically significant difference ($p < 0.05$). When introducing IGEL on a GIN-AK model, we find statistically significant improvements on both ZINC-12K. Introducing IGEL on GIN-AK$^+$ in PATTERN produces unstable losses on the validation set, with model performance showing larger variations across epochs. We believe that this instability might explain the loss in performance, and that further hyper-parameter tuning and regularization (e.g. tuning dropout to avoid overfitting on specific IGEL features) could result in improved model performance.

Finally, we note that despite our constrained setup, introducing IGEL is also interesting from a runtime and memory standpoint. In particular, introducing IGEL on a GIN for PATTERN yields performance only -0.2% worse than its GNN-AK (86.711 vs 86.877), while executing 3.87 times faster (62.21s vs 240.91s per iteration) and requiring 20.51 times less memory (26.2GB vs 1.3GB). This is in line with our theoretical analysis in Section 5.2, as IGEL can be computed once as a pre-processing step and then introduces a negligible cost on the input size of the first layer, which is amortized in multi-layer GNNs. Together with our results on graph classification, isomorphism detection, and graphlet counting, our experiments show that IGEL is also an efficient way of introducing sub-graph information without architecture modifications in large real-world datasets.

Table 5.5: Results on benchmark data sets in combination with GNN-AK (Zhao et al. 2022), highlighting **positive** and *negative* stat. sig, ($p < 0.05$) results when IGEL is added with best per-dataset results underlined.

Model	**+ IGEL**	**ZINC-12K** (MSE)	**PATTERN** (Acc)
GIN	No	0.155 ± 0.003	85.692 ± 0.042
	Yes	0.155 ± 0.005	$\underline{\mathbf{86.711 \pm 0.009}}$
GIN-AK$^+$	No	0.086 ± 0.002	86.877 ± 0.006
	Yes	$\underline{\mathbf{0.078 \pm 0.003}}$	*86.737 ± 0.062*

5.4.7 Link Prediction

We also test IGEL on a link prediction task, following the experimental approach of (Grover and Leskovec 2016) to compare with well-known transductive node embedding methods on the Facebook and ArXiv AstroPhysics data sets (Leskovec and Krevl 2014). We model the task as follows: for each graph, we generate negative examples (non-existing edges) by sampling random unconnected node pairs. Positive examples (existing edges) are obtained by removing half of the edges, keeping the pruned graph connected after edge removals[11]. Both sets of vertex pairs are chosen to have the same cardinality.

We learn *self-supervised* IGEL embeddings with a DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) approach (described in Appendix B.3), and model the link prediction task as a logistic regression problem whose input is the representation of an edge—as the element-wise product of IGEL embeddings of vertices at each end of an edge without fine-tuning[12].

Table 5.6: Area Under the ROC Curve (AUROC) link prediction results on Facebook and AP-arXiv. Embeddings learned on IGEL encodings outperform transductive methods. IGEL stddevs < 0.005.

Method	**Facebook**	**arXiv**
DeepWalk	0.968	0.934
node2vec	0.968	0.937
IGEL ($k = 2$)	**0.976**	**0.984**

In Table 5.6, we report AUC results reported on 5 independent executions and compared against previous reported baselines. In this case, we perform hyper-parameter search, with results described on Appendix B.1, providing additional details on unsupervised parameters in our repository.

IGEL with $k = 2$ significantly outperforms standard transductive methods on both data sets. This is despite the fact that we compare against methods

[11]Keeping the graph connected is not required for IGEL. It is required by transductive methods, which fail to learn meaningful representations on disconnected graph components.

[12]This is the best edge representation reported by (Grover and Leskovec 2016).

that are aware of node identities, and that several vertices can share the same IGEL encodings. Furthermore, IGEL is an inductive method that may be used on unseen nodes and edges[13]—unlike transductive methods DeepWalk or node2vec.
Additionally, IGEL significantly underperforms compared to other baselines when $k = 1$. We believe this might be caused by the fact that when $k < 2$, it is not possible for the model to assess whether two vertices are neighbours based on their IGEL representation. Overall, our link prediction results show that IGEL encodings can be used as a potentially inductive feature generation approach in unattributed networks, without degrading performance when compared to standard vertex embedding methods—answering **Q3**.

5.4.8 Vertex Classification

We evaluate IGEL on a vertex classification task where we measure to which extent the task can be solved by leveraging IGEL structural features without message passing (**Q4.**), in light of our graph-level results on graphlet counting. We introduce IGEL in a DNN model and evaluate against several MP-GNN baselines. Our comparison includes supervised baselines proposed by GraphSAGE (Hamilton, Ying, and Leskovec 2017), LCGL (Gao, Wang, and Ji 2018), and GAT (Veličković et al. 2018) on a multi-label classification task in a Protein-to-Protein Interaction (PPI) (Hamilton, Ying, and Leskovec 2017) dataset. The aim is to predict 121 binary labels, given graph data where every vertex has 50 attributes. We tune the parameters of a multi-layer perceptron (MLP) whose input features are either IGEL, vertex attributes, or both through randomized grid search[14].
Table 5.7 reports Micro-F1 averaged over 5 independent runs. Using IGEL and vertex attributes in an MLP outperforms MP-GNNs like GraphSAGE or LGCL despite not using message-passing—only accessing vertex attributes without additional context from its neighbours during learning, answering **Q4**. Furthermore, even though IGEL underperforms compared to GAT, results from (Veličković et al. 2018) use a 3-layer GAT passing messages through 3-hops, while IGEL with $k = 1$ is our best configuration.

[13]We do not explore the inductive setting as it would unfairly favor IGEL, and cannot be directly applied to DeepWalk or Node2vec.

[14]We provide a detailed description of the grid-search parameters in Appendix B.1.

Table 5.7: Multilabel class. Micro-F1 scores on PPI. IGEL plus node features as input for an MLP outperforms LGCL and GraphSAGE. IGEL stddevs < 0.01.

Method		**PPI**
Only Features (MLP, ours)		0.558
GraphSAGE-GCN[†]		0.500
GraphSAGE-mean[†]		0.598
GraphSAGE-LSTM[†]		0.612
GraphSAGE-pool[†]		0.600
GraphSAGE (no sampling)[‡]		0.768
LGCL[*]		0.772
IGEL ($k = 1$)	Graph Only	0.736
	Graph + Feats	*0.850*
IGEL ($k = 2$)	Graph Only	0.506
	Graph + Feats	0.741
Const-GAT[‡]		0.934
GAT[‡]		**0.973**

Results as reported by †: (Hamilton, Ying, and Leskovec 2017) ‡: (Veličković et al. 2018) $*$: (Gao, Wang, and Ji 2018).

We believe the information captured by IGEL at 1-hop might be sufficient to improve performance on tasks where local information is critical, potentially reducing the hops required by downstream models (e.g. GATs).

5.5 Discussion and Conclusions

IGEL is a novel and simple vertex representation algorithm that increases the expressive power of MP-GNNs beyond the 1-WL test. Empirically, we found IGEL can be used as a vertex/edge feature extractor on graph-, edge-, and node-level settings. On four different graph-level tasks, IGEL significantly improves the performance of nine graph representation models without requiring architectural modifications—including Linear and MLP baselines, GCNs (Niepert, Ahmed, and Kutzkov 2016), GATs (Veličković et al. 2018), GINs (Xu et al. 2019), ChebNets (Defferrard, Bresson, and Vandergheynst 2016), GNNML3 (Balcilar et al. 2021), and GNN-AK (Zhao et al. 2022). We introduce IGEL without performing hyper-parameter search on an existing baseline, which suggests that IGEL encodings are informative and can be introduced in a model without costly architecture search.

Although structure-aware message passing (Frasca et al. 2022; Nikolentzos, Dasoulas, and Vazirgiannis 2020; Zhang and Li 2021; Zhao et al. 2022), substructure counts (Bouritsas et al. 2021), identity (You et al. 2021), and subgraph pooling (Bevilacqua et al. 2022) may also be combined with existing MP-GNN architectures, IGEL reaches comparable performance as related models while simply augmenting the set of vertex-level feature without tuning model hyper-parameters. IGEL consistently improves performance on five data sets for graph classification, one data set for graph regression, two data sets for isomorphism detection, and five different graphlet structures in a graphlet counting task. Furthermore, even though MP-GNNs with learnable subgraph representations are expected to be more expressive since they can freely learn structural characteristics according to optimization objectives, our results show that introducing IGEL produces comparable results on three different domains and improves the performance of a strong GNN-AK$^+$ baseline on ZINC-12K. Additionally, introducing IGEL on a GIN model in the PATTERN data set achieves 99.8% of the performance of a strong GIN-AK$^+$ baseline at 3.87 times lower memory costs. The computational efficiency is a key benefit of IGEL as it only requires a single preprocessing step that extends vertex/edge attributes and can be cached during training and inference.

On link prediction tasks evaluated in two different graph data sets, IGEL-based DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) embeddings outperformed transductive methods based on embedding node identities such as DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) and Node2vec (Grover and Leskovec 2016). Finally, IGEL with $k = 1$ outperformed certain MP-GNN architectures like GraphSAGE (Hamilton, Ying, and Leskovec 2017) on a protein–protein interaction node-classification task despite being used as an additional input to a DNN without message passing. More powerful MP-GNNs—namely a three-layer GAT (Veličković et al. 2018)—outperformed the IGEL-enriched DNN model, albeit at potentially higher computational costs due to the increased depth of the model.

A fundamental aspect in our analysis of the expressivity of IGEL is that the connectivity of nodes is different depending on whether they are analyzed as part of the subgraph (ego network) or within the entire input graph. In particular, edges at the boundary of the ego network are a subset of the edges in the input graph. IGEL exploits this idea in combination with a simple encoding based on frequencies of degrees and distances. This is a novel idea, which allows us to connect the expressive power of IGEL with other analyses like the 1-WL test, as well as Weisfeiler-Lehman kernels (see Section 5.3.3), Shortest Path Neural Networks (see Section 5.3.3), and MATLANG (see Section 5.3.3). Furthermore, the ego-network formulation allows us to identify an upper-bound on expressivity in Strongly Regular Graphs—matching recent findings on the expressivity of sub-graph GNNs. Although we have presented IGEL on unattributed graphs, the principle underlying its encoding can be also applied to labelled or attributed graphs. Appendix B.2 outlines possible extensions in this direction. In Subsection B.2.3, we also connect IGEL with k-hop GNNs and GNN-AK—drawing a direct link between subgraph GNNs and our proposed encoding.

Overall, our results show that IGEL can be efficiently used to enrich network representations on a variety of tasks and data sets, which we believe is an attractive baseline in expressivity-related tasks. This opens up interesting future research directions by showing that explicit network encodings like IGEL can perform competitively compared to more complex learnable representations while being more computationally efficient. To encourage reproducibility and future work, we make all our code and research scripts available in our main code repository[15].

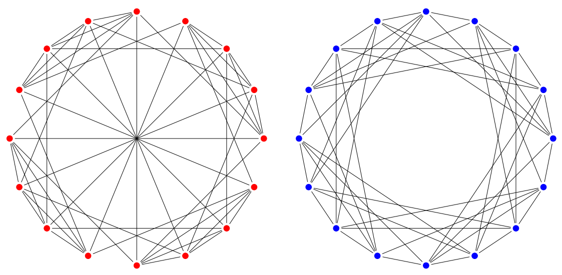
[15]`https://github.com/nur-ag/IGEL`

Chapter 6

Improving Subgraph-GNNs via Edge-Level Ego-Network Encodings

In the previous chapter, we introduced IGEL, an ego-network encoding method that can be used as node-level features to improve the expressivity of MP-GNNs without modifying their architecture. One key finding is that IGEL is *unable* to distinguish Strongly Regular Graphs. Furthermore, as highlighted in Section 4.3, the majority of work to improve expressivity has identifying novel (learnable) GNN architectures that improve on some expressivity framework.

In this chapter, we present an alternative approach to previous *purely learning* approaches. Rather than learning on subgraphs, we introduce a systematic procedure to generate structural features (i.e., no learning) which can be integrated in MP-GNNs. Similar approaches have been proposed recently (Bouritsas et al. 2021), and IGEL can be thought as a step in the same direction. Crucially, in this chapter we propose features that capture information at the edge-level, including signals contained in the ego-networks of adjacent nodes in the input graph. We call this encoding ELENE, for **E**dge-**L**evel **E**go-**N**etwork **E**ncodings. The benefits of such a representation are diverse: the encodings are interpretable and amenable for theoretical analysis, efficiently computable as a pre-processing step, and they reach comparable performance with state-of-the-art learning methods.



(a) 4×4 Rook Graph. (b) Shrikhande Graph.

Figure 6.1: Expressive power is typically analyzed in terms of the families of non-isomorphic graphs *that models fail to distinguish*: 4×4 Rook (a) and Shrikhande (b) graphs are indistinguishable by node-only sub-graph GNNs (Frasca et al. 2022).

As an illustrative example, consider **S**trongly **R**egular **G**raphs (SRGs) (Balcilar et al. 2021). They are known to be *indistinguishable* by node-based subgraph GNNs (Frasca et al. 2022; Morris et al. 2021; Papp and Wattenhofer 2022; Zhao et al. 2022), as exemplified by the non-isomorphic 4×4 Rook and Shrikhande graphs in Figure 6.1. We theoretically show that Elene is as expressive as node-only sub-graph GNNs and more expressive than Igel. Furthermore, we show it is expressive enough to differentiate certain classes of SRGs like those in Figure 6.1.

Another example of a challenging benchmark is the h-Proximity task (see Figure 6.2). It requires the ability to capture graph properties that depend both on the graph structure (shortest path distances) and node attributes (colors) (Abboud, Dimitrov, and Ceylan 2022). In this case, an enriched (learnable) MP-GNN with Elene features—called Elene-L—outperforms current baselines. In real-world benchmarks, Elene-L matches the performance of state-of-the-art learning methods at significantly lower memory costs, as we show experimentally in Section 6.4.

The chapter is organized as follows. Section 6.1 defines and motivates Elene. Section 6.2 introduces Elene-L and Section 6.3 analyzes its expressivity. Finally, Section 6.4 evaluates our methods in three benchmarks. Section 6.5 summarizes our results.

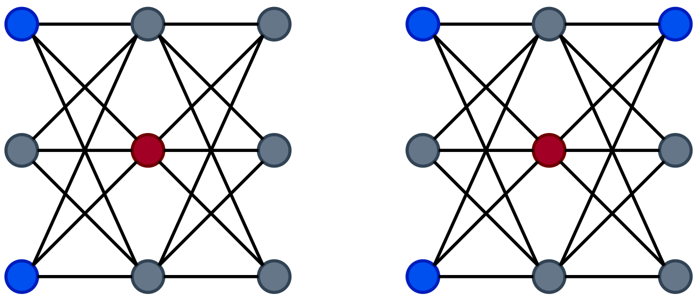

(a) Positive graph. (b) Negative graph.

Figure 6.2: h-Proximity binary classification task—A pair of positive (a) and negative (b) 1-proximity graph examples. An h-Proximity graph is positive if all red nodes have at most 2 blue neighbours up to distance h, and negative otherwise.

6.1 Defining Elene

In this section we first present the proposed edge-level encodings and then illustrate their expressive power.

6.1.1 Constructing an Edge-Level Ego-Network Encoding

The main idea behind Elene encodings is to capture higher-order interactions that go beyond the node-centric perspective used by MP-GNNs. We look at the structure resulting not only from the ego-network of every node, but also from the combination of two ego-networks of adjacent nodes in the input graph, and design a pool of features based on that structure. Consider the k-depth ($k > 1$) ego-network $\mathcal{S}_v^k = (\mathcal{V}_v^k, \mathcal{E}_v^k)$ surrounding node v. We may ask: *how many edges of a neighbour u of v reach nodes that are 1-hop closer to v, at the same distance as u, or 1-hop farther from v?* The proposed Elene encodings elaborate on this idea to capture interactions between nodes and edges in ego-network sub-graphs.
More formally, consider a node u contained in $\mathcal{S}_v^k$ and let $d_{\mathcal{S}}(u|v)$ count the edges from u to nodes at a distance $l_{\mathcal{S}}(u,v) + p$ of v, $p \in \{-1, 0, +1\}$:

$$d_{\mathcal{S}}^{(p)}(u|v) = \Big|(u, w) \in \mathcal{E}_v^k,\ \forall w \in \mathcal{V}_v^k : l_{\mathcal{S}}(v,w) = l_{\mathcal{S}}(u,v) + p\Big|.$$

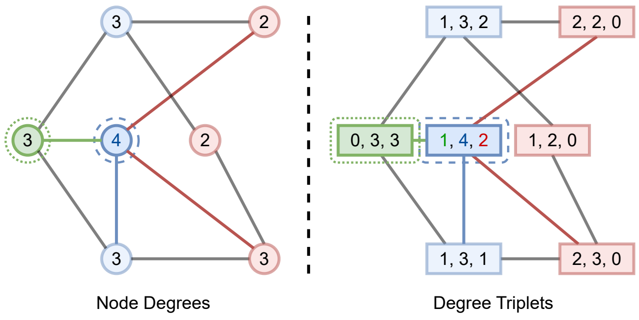


Figure 6.3: Example graph (right) and its (left) degree triplets for nodes in the 2-hop ego-network rooted on the green node. The dashed blue node has one edge to the 0-hop root ($d_{\mathcal{S}}^{(\text{-}1)} = 1$), a degree of 4, and two edges 2-hops from the root ($d_{\mathcal{S}}^{(+1)} = 2$, red), so its degree triplet is (1, 4, 2).

The degree of node u decomposes as the sum of these *relative* degrees corresponding to these three different subsets of neighbours of u:

$$d_{\mathcal{S}}(u) = d_{\mathcal{S}}^{(\text{-}1)}(u|v) + d_{\mathcal{S}}^{(0)}(u|v) + d_{\mathcal{S}}^{(+1)}(u|v).$$

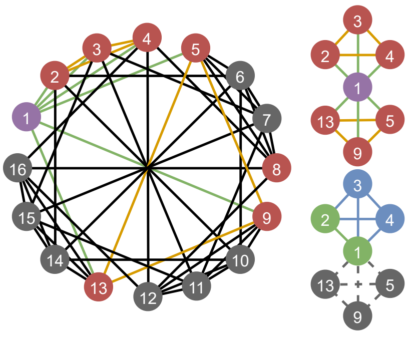
Figure 6.3 (left) shows an example graph, with all nodes labeled with their degree and colored according to the distance to the root node of the ego network, in this case the node in green. The plot on the right shows a degree triplet for each node, which counts the *relative* degrees, or edges closer and further to the root (1st and 3rd components), together with the degree (2nd component). Leveraging relative degrees yields ELENE, an ego-network encoding as a multi-set of quadruplets counting all instances of distance and degree triplets in sub-graph $\mathcal{S}$:

$$e_v^k = \left\{\!\!\left\{ \left(l_{\mathcal{S}}(u,v), d_{\mathcal{S}}^{(\text{-}1)}(u|v), d_{\mathcal{S}}(u), d_{\mathcal{S}}^{(+1)}(u|v) \right) \middle| \forall u \in \mathcal{V}_v^k \right\}\!\!\right\}. \tag{6.1}$$

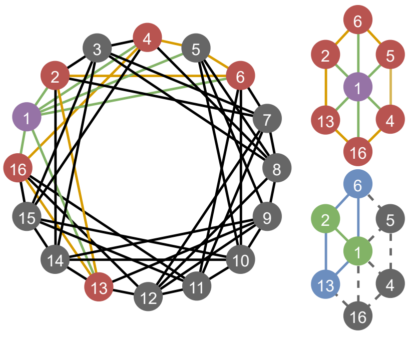
We can construct an Edge (**ED**) Centric analogue of the Node (**ND**) Centric encoding of Equation 6.1 by also encoding edge-wise sub-graph intersections for edge $\langle u, v\rangle$ as $e^k_{\langle u,v\rangle}$ and counting quadruplets across $\mathcal{S}^k_{\langle u,v\rangle}$ with distances to both u and v. Using **ED** or **ND** encodings leads to different expressive power, as we show formally in Section 6.3. In both cases, Subsection 6.3.4 shows that ELENE encodings are permutation equivariant.

6.1.2 Illustrating Elene

To illustrate Elene, we focus on Strongly Regular graphs. In line with the expressivity analysis for Igel on SRGs Subsection 5.3.2, we denote Strongly Regular Graphs as $\texttt{SRG}(n, d, \lambda, \mu)$. Recall that Strongly Regular graphs with equal parameters are *indistinguishable* by the 1-WL (Weisfeiler and Leman 1968) test—a classic graph algorithm known to distinguish graphs that are not isomorphic with high probability (Babai and Kucera 1979)—and its more powerful $k = 3$-WL variant (Arvind et al. 2020; Balcilar et al. 2021)—whose ability to distinguish graphs has been shown to be the expressivity upper-bound for node-only Sub-graph GNNs (Bevilacqua et al. 2022; Frasca et al. 2022; Zhao et al. 2022). A natural question follows: *what structural information is sufficient to distinguish SRGs?*



(a) 4 × 4 Rook Graph.



(b) Shrikhande Graph.

Figure 6.4: 3-WL equivalent 4 × 4 Rook (a) and Shrikhande (b) graphs are indistinguishable by 3-WL, as SRGs with equal parameters $\texttt{SRG}(16, 6, 2, 2)$ (Arvind et al. 2020; Frasca et al. 2022). Elene (**ND**, top $k = 1$ ego-network sub-graphs rooted on purple nodes) is also unable to distinguish the graphs, while Elene (**ED**, bottom intersecting ego-network sub-graphs around green edges) counts different numbers of edges. Edge colors indicate distance for the root (node or edge).

Figure 6.4 shows 1-depth ego-networks $\mathcal{S}_{v_1}^{k=1}$ for purple vertices labelled with '1' with 1-hop neighbours colored in red (top smaller sub-graphs)[1].

[1]Note that for these two graphs, any node will have matching ego-networks regardless of their label—as per the proof for Theorem 3.

We represent both graphs in terms of $n = 16$ equal sub-graphs (one per node), analyzing whether the sub-graph pairs can be distinguished. Both sub-graphs have the same number of nodes (7), edges (12), and matching degree multisets $\{\!\{3^6, 6^1\}\!\}$. Furthermore, by coloring the edges as connected to the ego-network root (in green) or connecting adjacent neighbours of the root (in orange), the count of edge colors also matches. The Node-Centric (**ND**) Elene encoding, as shown in Equation 6.1, corresponds to such a coloring and is thus unable to distinguish the pair of graphs. In Section 6.3, we formally prove this upper bound for Elene (**ND**), which coincides with the expressive power of node-based Sub-graph GNNs.
In contrast, if we consider the 1-hop common neighbours (in blue) of adjacent nodes labelled '1' and '2', the intersecting sub-graphs are distinguishable (bottom smaller sub-graphs). Indeed, the number of edges differs between the 4×4 Rook graph (6 edges) and the Shrikhande's graph (5 edges). This corresponds to Edge-Centric Elene (**ED**), and illustrates it has more expressive power than the Node-Centric (**ND**) one.

6.1.3 Computational Complexity of ELENE

The Elene encoding for a single node v requires traversing all the edges in the ego-network $\mathcal{E}_v^k$. This can be computed via Breadth-First Search (BFS) bounded by depth k, which has worst-case complexity $\mathcal{O}(d_{\texttt{max}}^k)$. If k is greater than the diameter in the graph, Elene must traverse all m edges for the n ego-networks with each node as the root. Encoding the entire graph thus has time complexity $\mathcal{O}(n \cdot \min\{m, d_{\texttt{max}}^k\})$. Note that the more expressive edge-centric implementation requires executing the BFS from both nodes alongside an edge, with asymptotically no additional cost.
Elene is best suited for sparse graphs, where $d_{\texttt{max}} \ll n$. For fully connected graphs, $m = |\mathcal{E}_v^k| = n \cdot (n-1)/2$ which results in time complexity $\mathcal{O}(n^3)$, matching the computational worst-case complexity of GNN-AK, NGNNs, SUN, ESAN, or SPEN.
In terms of memory, Elene encodings require a sparse $3 \cdot (k+1) \cdot (d_{\texttt{max}} + 1)$-component vector for each node $v \in V$ to represent the multi-set of quadruplets in Equation 6.1. Accordingly, each entry holds the count of observed relative degrees at each distance from v. In Appendix C.1, we describe a BFS implementation of Elene that can be parallelized over p processors yielding $\mathcal{O}(n \cdot \min\{m, d_{\texttt{max}}^k\}/p)$ time complexity.

6.2 Learning with Elene: Elene-L

We now introduce two approaches for leveraging Elene encodings in practical learning settings—a simple concatenation of Elene over network attributes, and a fully learnable variant called Elene-L that updates node and edge representations during the learning process. The first approach represents Elene multi-as sparse vectors containing frequencies of each quadruple q—which can be attributes concatenated to $\mathbf{x}_v$ or $\mathbf{x}_{\langle u,v\rangle}$ if processing e_v^k or $e_{\langle u,v\rangle}^k$:

$$\text{Elene}_{\texttt{vec}}^k(v)_i = \left|\left\{\!\!\left\{ q \in e_v^k \middle| f(q) = i \right\}\!\!\right\}\right|. \tag{6.2}$$

where $f(q)$ is an indexing function mapping each unique quadruplet to an index in the sparse vector.

The concatenation approach is the most memory efficient, using only as much memory as the encodings themselves, and can be computed once and reused during training or inference. Furthermore, this approach is directly applicable to any downstream learning model e.g., an MP-GNN, without changing its architecture.

Certain tasks, however, require structural information *within the sub-graph* to be combined with node or edge *attributes* during learning. One example are h-Proximity tasks, which require joint representations that integrate the Elene encodings with attributes and structure.

Elene-L addresses the limitations of concatenating Elene with node and edge attributes by learning over both *structures* and *attributes* at once. Elene-L learns non-linear functions (Φ, e.g. a Dense Neural Network—DNN) to represent nodes, edges, and Node or Edge-Centric sub-graphs by representing u in $\mathcal{S}_v^k$ via a learnable function $\Phi_{\texttt{nd}}$[2]:

$$\Phi_{\texttt{nd}}^t(u|v) = \Phi_{\texttt{nd}}\Big(\mathbf{x}_v^t \Big\| \mathbf{x}_u^t \Big\| \texttt{Emb}(u|v)\Big) \tag{6.3}$$

where $\mathbf{x}_v^t$ and $\mathbf{x}_u^t$ are features of u and v at time-step t (i.e. after t layers, such that $t = 0$ are 'input' features), and $\texttt{Emb}(u|v)$ is a learnable embedding of Elene encodings which we describe in Subsection 6.2.1.

[2]We compress notation by using Φ^t for the output of Φ at step t.

As with ELENE, we produce a learnable representation of edge $\langle u, w\rangle$ in $\mathcal{S}_v^k$ via a learnable $\Phi_{\texttt{ed}}$:

$$\Phi^t_{\texttt{ed}}(u, w|v) = \Phi_{\texttt{ed}}\Big(\mathbf{x}^t_v \Big\| \mathbf{x}^t_{\langle u,w\rangle} \Big\| \mathbf{x}^t_u \odot \mathbf{x}^t_w \Big\| \texttt{Emb}(u, w|v)\Big).$$

The representation of the Node-Centric ego-network root at time t is a learnable $\Phi_{\texttt{ND}}$ applied over the aggregation of every node and edge in the sub-graph given a pooling function ($\sum$):

$$\Phi^t_{\texttt{ND}}(v) = \Phi_{\texttt{ND}}\left(\mathbf{x}^t_v \middle\| \sum_u^{\mathcal{V}_v^k} \Phi^t_{\texttt{nd}}(u|v) \middle\| \sum_{\langle u,w\rangle}^{\mathcal{E}_v^k} \Phi^t_{\texttt{ed}}(u, w|v)\right). \tag{6.4}$$

Similarly, the Edge-Centric representation for edge $\langle u, v\rangle$ at time t is a learnable $\Phi_{\texttt{ED}}$ consuming the aggregation over ego-networks containing the edge, as shown in Figure 6.5:

$$\Phi^t_{\texttt{ED}}(u, v) = \Phi_{\texttt{ED}}\left(\sum_w^{\mathcal{V}^k_{\langle u,v\rangle}} \Phi^t_{\texttt{ed}}(u, v|w)\right). \tag{6.5}$$

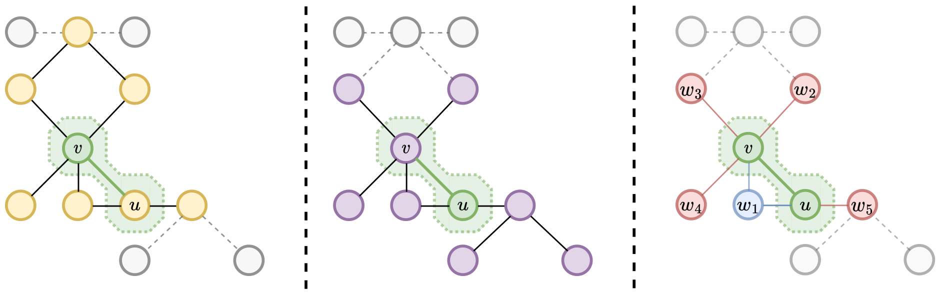


Figure 6.5: k-depth ego-network intersection following Equation 6.5 for the green edge. The ego-networks of u and v (yellow (left) and purple (center) respectively), intersect on five nodes around $\langle u, v\rangle$ (dotted, right). We show $\mathcal{V}^{k=2}_{\langle u,v\rangle} = \{u, v, w_1, w_2, w_3, w_4, w_5\}$ (right), indicating nodes reachable in 0 or 1-hops, exactly 1-hop or 1 or 2-hops from u and v.

Node and edge representations at $t+1$ update via a learnable parameter γ gating the flow of ELENE updates:

$$\mathbf{x}_v^{t+1} = \mathbf{x}_v^t + \gamma_{\texttt{ND}} \cdot \Phi_{\texttt{ND}}^t(v). \tag{6.6}$$

We follow the same update-rule at the edge-level:

$$\mathbf{x}_{\langle u,w\rangle}^{t+1} = \mathbf{x}_{\langle u,w\rangle}^t + \gamma_{\texttt{ED}} \cdot \Phi_{\texttt{ED}}^t(u, w) \tag{6.7}$$

We may use $\mathbf{x}_v^{t+1}$ and $\mathbf{x}_{\langle u,w\rangle}^{t+1}$ directly in the downstream task, or as inputs into an MP-GNN layer during learning—boosting its expressivity. We follow the latter approach in this work.

6.2.1 Defining ELENE-L Embeddings

The representations in Equation 6.6 and Equation 6.7 leverage attributes and ELENE encodings through $\texttt{Emb}(u|v)$ and $\texttt{Emb}(u, w|v)$. To define ELENE-L embeddings, three hyper-parameters determine the shapes of embedding matrices: ω, the length of the embedding vectors; ρ, the max. degree to be encoded (by default, $\rho = d_{\texttt{max}}$); and third k, the maximum distance to be encoded (i.e., the ego-network depth). For the quadruplet of u, we abbreviate:

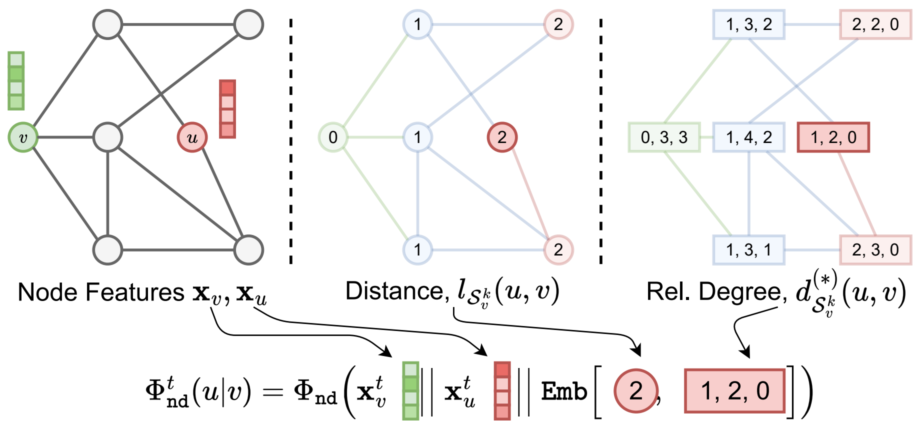
$$q_u = (l_u, d_u^1, d_u^2, d_u^3) = \left(l_{\mathcal{S}}(u, v), d_{\mathcal{S}}^{(\text{-}1)}(u|v), d_{\mathcal{S}}(u), d_{\mathcal{S}}^{(+1)}(u|v)\right)$$

as defined in Equation 6.1 and *jointly* embed distance and relative degrees. The embedding $\texttt{Emb}(u|v)$ of u in $\mathcal{S}_v^k$ is given by:

$$\texttt{Emb}\left[l_u, d_u^1, d_u^2, d_u^3\right] = \left(\mathbf{W}^{1,\text{nd}}_{(l_u, d_u^1)} \middle\| \mathbf{W}^{2,\text{nd}}_{(l_u, d_u^2)} \middle\| \mathbf{W}^{3,\text{nd}}_{(l_u, d_u^3)}\right), \tag{6.8}$$

where $\mathbf{W}^{1,\text{nd}}$, $\mathbf{W}^{2,\text{nd}}$, and $\mathbf{W}^{3,\text{nd}} \in \mathbb{R}^{S \times \omega}$ are three node embedding matrices with $S = (\rho + 1) \cdot (k + 1)$ entries—one for each distance and relative degree pair. A visual representation of the attributes and ELENE encodings of u is shown in Figure 6.6.
To embed edge $\langle u, w\rangle$ in $\mathcal{S}_v^k$, we use the quadruplets of u and w, q_u and q_w, and increase the granularity of distances to capture the *relative* direction of the edge, following the different edge colors in Figure 6.4:

$$\delta_{uw} = l_u - l_w + 1 \in \{0, 1, 2\}.$$



v
u
1
2
0
1
2
1
2
1, 3, 2
2, 2, 0
0, 3, 3
1, 4, 2
1, 2, 0
1, 3, 1
2, 3, 0
Node Features $\mathbf{x}_v, \mathbf{x}_u$
Distance, $l_{\mathcal{S}_v^k}(u, v)$
Rel. Degree, $d^{(*)}_{\mathcal{S}_v^k}(u, v)$
$\Phi^t_{\mathtt{nd}}(u|v) = \Phi_{\mathtt{nd}}\Big(\mathbf{x}^t_v \,\|\, \mathbf{x}^t_u \,\|\, \mathtt{Emb}\big[\,2,\ 1, 2, 0\,\big]\Big)$

Figure 6.6: ELENE-L encoding of u in the $k = 2$ ego-network of v. The representation contains the feature vectors of both nodes ($\mathbf{x}_v$ and $\mathbf{x}_u$, left), the distance information of u to v (2, center) and the relative degree information ($[1, 2, 0]$, right).

We embed $\langle u, w\rangle$ in a permutation-invariant by summing embeddings bidirectionally so that $\mathtt{Emb}(u, w|v) = \mathtt{Emb}(w, v|v)$:

$$\begin{aligned}
&\mathtt{Emb}(u, w|v) = \qquad (6.9)\\
&\Big(\mathbf{W}^{1,\mathsf{ed}}_{(l_u, \delta_{uw}, d^1_u)}\Big\|\mathbf{W}^{2,\mathsf{ed}}_{(l_u, \delta_{uw}, d^2_u)}\Big\|\mathbf{W}^{3,\mathsf{ed}}_{(l_u, \delta_{uw}, d^3_u)}\Big) + \\
&\Big(\mathbf{W}^{1,\mathsf{ed}}_{(l_w, \delta_{wu}, d^1_w)}\Big\|\mathbf{W}^{2,\mathsf{ed}}_{(l_w, \delta_{wu}, d^2_w)}\Big\|\mathbf{W}^{3,\mathsf{ed}}_{(l_w, \delta_{wu}, d^3_w)}\Big).
\end{aligned}$$

$\mathbf{W}^{1,\mathsf{ed}}$, $\mathbf{W}^{2,\mathsf{ed}}$, and $\mathbf{W}^{3,\mathsf{ed}} \in \mathbb{R}^{3\times S\times\omega}$ are edge-level embedding matrices with $3\times$ more entries to represent the three possible values of δ_{uw}.

6.3 Expressive Power

We now analyze the expressive power of ELENE—formally answering our question on *which information is sufficient to distinguish* `SRG`*s*.
We extend our previous results on IGEL to ELENE in Section 5.3 and show that Edge-Centric and Node-Centric ELENE are strictly more expressive than previous methods relying on degrees and distances by comparing their

expressivity on SRGs. We then show that ELENE-L is at least as expressive as ELENE, and prove that edge-aware ELENE-L is more expressive than node-centric ELENE-L or ELENE. Finally, we connect our framework with SPNNs (Abboud, Dimitrov, and Ceylan 2022), showing that the latter can be expressed by an instance of node-centric ELENE-L without edge-degree information—motivating our analysis on *attributed* tasks in Section 6.4.

6.3.1 Expressive Power of ELENE

Previous work has shown that encoding-based and sub-graph MP-GNN methods are unable to distinguish 3-WL equivalent SRGs (Arvind et al. 2020; Balcilar et al. 2021; Frasca et al. 2022). Furthermore, in Chapter 5 we showed that IGEL is a simple, sparse node feature vector containing counts of distance and degree tuples in an ego-network—and strictly more expressive than the 1-WL test. Following Equation 6.2, ELENE multi-sets may also be represented as sparse vectors—which can then be used as feature vectors, but also to distinguish ego-network sub-graphs.
We build on top of the results from Chapter 5 and show ELENE is at least as expressive as IGEL. We then find an upper-bound of expressivity for IGEL, which is at most able to distinguish between n, d or λ parameters of SRGs, but not μ, and show Node-Centric and Edge-Centric ELENE is strictly more expressive on SRGs as it can explicitly encode all SRG parameters by counting edges:

Theorem 2. *Node-Centric* ELENE *is at least as expressive as* IGEL, *and transitively more expressive than 1-WL.*

Proof. The IGEL encoding is a simpler version of Equation 6.2 that only considers distance (l_u) and *absolute* degree (d_u^2):

$$\text{IGEL}^k_{\texttt{vec}}(v)_i = \left| \left\{\!\!\left\{ (l_u, d_u^1, d_u^2, d_u^3) \in e_v^k \,\middle|\, f'(l_u, d_u^2) = i \right\}\!\!\right\} \right|.$$

where $f'(l_u, d_u^2)$ is a bijective function that does not consider *relative* degrees, in contrast with ELENE's f. Thus, for any ego-network, ELENE includes all information required to construct IGEL vectors, so it is at least as expressive as IGEL. □

Theorem 3. Elene *(**ND**) encodes and distinguishes* SRG*s with different parameters of n, d, λ and μ.*

Proof. Consider $\texttt{SRG}(n, d, \lambda, \mu) = (V, E)$. The maximum diameter of an SRG is 2 (Brouwer and Van Maldeghem 2022), so we focus on the case where $k = 2$. The Elene (**ND**) encoding of $v \in V$ according to Equation 6.1 is:

$$e_v^2 = \left\{\!\!\left\{ \Big(0, 0, d, d\Big)^1, \Big(1, 1, d, d\text{-}\lambda\text{-}1\Big)^d, \Big(2, d - \mu, d, 0\Big)^{n\text{-}d\text{-}1} \right\}\!\!\right\}$$

By definition, any Node-Centric ego-network in an SRG has a single root with d neighbours, d neighbours with one edge with the root and $d - \lambda - 1$ edges to the next layer, and the remaining $n - d - 1$ non-adjacent nodes to the root each have $d - \mu$ edges with the d neighbours of the root. Consider $\texttt{SRG}'(n', d', \lambda', \mu') = (V', E')$. If any of the parameters between SRG and SRG$'$ differ, so will e_v^2 from $e_{v'}^2$. This is not the case for Igel, which at most capture n, d, and λ:

$$\text{IGEL}_v^1 = \left\{\!\!\left\{ \Big(0, d\Big)^1, \Big(1, 1 + \lambda\Big)^d \right\}\!\!\right\}$$
$$\text{IGEL}_v^2 = \left\{\!\!\left\{ \Big(0, d\Big)^1, \Big(1, d\Big)^d, \Big(2, d\Big)^{n\text{-}d\text{-}1} \right\}\!\!\right\}$$

Thus, Elene (**ND**) can encode and distinguish all parameters of SRGs—outperforming Igel. However, Elene (**ND**) *cannot distinguish* non isomorphic SRGs when $n = n'$, $d = d'$, $\lambda = \lambda'$, and $\mu = \mu'$. □

Corollary 1. Elene *(**ND**) is more expressive than* Igel *and 1-WL, per Theorem 2 & Theorem 3.*

Corollary 2. Elene *(**ND**) signals at the node-level are not capable of distinguishing between non-isomorphic* SRG*s with equal parameters—e.g. the graphs in Figure 6.4.*

Proposition 2. Elene *(**ED**, leveraging both e_v^k $\forall v \in V$ and $e_{\langle u,v\rangle}^k$ $\forall (u, v) \in E$) is strictly more expressive than* Elene *(**ND**), as it can distinguish the pair of graphs in Figure 6.4.*

6.3.2 Expressive Power of ELENE-L.

Theorem 4. ELENE-L *with the sum as the pooling operator is at least as expressive as* ELENE.

Proof. We first show ELENE-L (**ND**) is at least as expressive as ELENE (**ND**). We then show that the **ED** variants are at least as expressive as **ND** variants (Proposition 3), and show through Figure 6.4 that ELENE-L (**ED**) is more powerful than Node-Centric ELENE-L (**ND**).

ELENE-L (ND). $\forall v \in V$, the ELENE-L($\mathbf{x}_v^t$) representation of v is given by $\Phi_{\texttt{ND}}^t(v)$ as per Equation 6.4. $\Phi_{\texttt{ND}}^t(v)$ is the result of applying $\Phi_{\texttt{ND}}$ to the concatenation of $\mathbf{x}_v^t$ and the combined representations of every $u \in \mathcal{V}_v^k$ and $\langle u, w\rangle \in \mathcal{E}_v^k$. Let $\Phi_{\texttt{out}}$ and $\Phi_{\texttt{nd}}$ be the identity function, we exclude edge-level information by discarding the output of $\Phi_{\texttt{ed}}$. We now expand $\hat{\Phi}_{\texttt{ND}}^t(v)$, which is $\Phi_{\texttt{ND}}^t(v)$ with the changes to the learnable Φ:

$$\hat{\Phi}_{\texttt{ND}}^t(v) = \left(\mathbf{x}_v^t \middle\| \sum_u^{\mathcal{V}_v^k} \left(\mathbf{x}_v^t \middle\| \mathbf{x}_u^t \middle\| \texttt{Emb}(u|v)\right)\right).$$

We discard repeated $\mathbf{x}_v^t$ terms, and rewrite the representation of v, distributing the sum over the concatenated vector:

$$\hat{\Phi}_{\texttt{ND}}^t(v) = \left(\mathbf{x}_v^t \middle\| \sum_u^{\mathcal{V}_v^k} \mathbf{x}_u^t \middle\| \sum_u^{\mathcal{V}_v^k} \texttt{Emb}(u|v)\right).$$

Let $\mathbf{W}^{1,\mathbf{nd}}, \mathbf{W}^{2,\mathbf{nd}}, \mathbf{W}^{3,\mathbf{nd}} \in \mathbb{R}^{S\times S}$ used by `Emb` be identity matrices so every relative degree and distance pair out of $S = d_{\texttt{max}}\cdot(k+1)$ has a single position in $\mathbf{W}$. By using the sum as the pooling function, we obtain the frequency of each relative degree and distance pair, matching ELENE in Equation 6.2. Thus, $\hat{\Phi}_{\texttt{out}}^t(v)$ contains the information contained in the ELENE multi-set, reaching at least the same expressivity. □

Proposition 3. ELENE-L ***(ED)*** *variants with the sum as the pooling operator are at least as expressive as* ELENE.

ELENE-L (ED). We had discarded $\Phi_{\texttt{ed}}$, showing edge-aware variants are at least as expressive as node-centric variants, since the concatenation of edge-level information can only match or boost expressivity. Thus, all variants of ELENE-L are as expressive as ELENE. □

Theorem 5. ELENE-L *(**ED**) is more expressive than* ELENE-L *(**ND**) and* ELENE *(**ND**).*

Proof. There is at least a pair of non-isomorphic SRGs that ELENE-L (**ED**) can distinguish. In Subsection 6.1.2, we show that the Shrikhande and 4×4 Rook graphs (Arvind et al. 2020; Balcilar et al. 2021) (parametrized as $\texttt{SRG}(16, 6, 2, 2)$) can be distinguished by Edge-Centric counts that ELENE-L (**ED**) captures despite being undistinguishable by 3-WL (by implementing Equation 6.1 at the *edge* level). Following from Theorem 2 and Theorem 3, both graphs are indistinguishable by ELENE (**ND**) or ELENE-L (**ND**), as well as sub-graph GNNs like GNN-AK or SUN.
Intuitively, SRGs are indistinguishable with node-centric k ego-network subgraph encodings when $k \in \{1, 2\}$ since all nodes produce identical representations, as shown in Theorem 3. However, the graphs can be distinguished by edge-level information as per Equation 6.7, as the intersection of k-depth ego-networks for $\langle v_1, v_2\rangle$ differ in edge counts between both SRGs—as observed in Section 6.1. We can see the 4×4 Rook Graph has 6 edges (i.e. $|\mathcal{E}^k_{\langle v_1,v_2\rangle})| = 6$) while the Shrikhande graph has $|\mathcal{E}^k_{\langle v_1,v_2\rangle})| = 5$, hence the graphs are distinguishable by ELENE-L (**ED**), but not ELENE-L (**ND**) or ELENE (**ND**). □

6.3.3 Linking ELENE and Shortest Path Neural Networks.

Remark 4. *A Graphormer with max. shortest path length M and global readout is an instance Shortest Path Neural Networks (SPNNs) with $k = M - 1$ depth (Abboud, Dimitrov, and Ceylan 2022).*

Theorem 6. ELENE-L *(**ND**) is as expressive as Shortest Path Neural Networks (SPNNs), and transitively, Graphormers.*

Proof. Let $\mathcal{L}^k_G(v) = \{u | u \in V \wedge l_G(u, v) = k\}$ be the nodes in G exactly at distance k of v. In (Abboud, Dimitrov, and Ceylan 2022), a k-depth SPNN updates the hidden state of node v an aggregation over the $1, ..., k$ exact-distance neighbourhoods:

$$\mathbf{x}^{t+1}_v = \Phi_{\texttt{sp}}\Big((1 + \epsilon) \cdot \mathbf{x}^t_v + \sum_{i=1}^{k} \alpha_i \sum_{u\in\mathcal{L}^i_G(v)} \mathbf{x}^t_u\Big). \tag{6.10}$$

We show that ELENE-L (**ND**) can implement SPNNs. First, let $\gamma_{\texttt{nd}} = 1$ in Equation 6.6, such that:

$$\mathbf{x}_v^{t+1} = \Phi_{\texttt{ND}}^t(v) = \\ \Phi_{\texttt{ND}}\left(\mathbf{x}_v^t \middle\| \sum_{u}^{\mathcal{V}_v^k} \Phi_{\texttt{nd}}^t(u|v) \middle\| \sum_{\langle u,w\rangle}^{\mathcal{E}_v^k} \Phi_{\texttt{ed}}^t(u,w|v)\right).$$

We drop $\Phi_{\texttt{ed}}^t(u,w|v)$ as SPNNs ignore edge-level signals[3]. Let $\Phi_{\texttt{ND}}(\cdot)$ be composed of two functions $\Phi_{\texttt{sp}}(g(\cdot))$[4] where:

$$g\left(\mathbf{x}_v^t \middle\| \sum_{u}^{\mathcal{V}_v^k} \Phi_{\texttt{nd}}^t(u|v)\right) = \left((1+\epsilon)\cdot \mathbf{x}_v^t + \sum_{u}^{\mathcal{V}_v^k} \Phi_{\texttt{nd}}^t(u|v)\right).$$

We replace $\Phi_{\texttt{ND}}$ by $\Phi_{\texttt{sp}}$ and $g(\cdot)$, and expand $\Phi_{\texttt{nd}}^t(u|v)$:

$$\mathbf{x}_v^{t+1} = \Phi_{\texttt{sp}}\left((1+\epsilon)\cdot \mathbf{x}_v^t + \sum_{u}^{\mathcal{V}_v^k} \Phi_{\texttt{nd}}\left(\mathbf{x}_v^t \middle\| \mathbf{x}_u^t \middle\| \texttt{Emb}(u|v)\right)\right).$$

We then instantiate $\Phi_{\texttt{nd}}(\cdot)$ as:

$$\Phi_{\texttt{nd}}(\cdot) = \sum_{i}^{k} \alpha_i \cdot \texttt{if}[i = l_{\mathcal{S}}(u,v)] \cdot \mathbf{x}_u^t$$

$\texttt{if}[\cdot]$ is a boolean function returning 1 if a condition holds, 0 otherwise. It can be implemented through the distance and degree signals in `Emb`, such we can check if the node distance matches a specific value[5]. Substituting in $\mathbf{x}_v^{t+1}$ above yields:

$$\mathbf{x}_v^{t+1} = \Phi_{\texttt{sp}}\left((1+\epsilon)\cdot \mathbf{x}_v^t + \sum_{u}^{\mathcal{V}_v^k} \sum_{i}^{k} \alpha_i \cdot \texttt{if}[i = l_{\mathcal{S}}(u,v)] \cdot \mathbf{x}_u^t\right),$$

[3]Including edge-level signals may bring ELENE-L (**ED**) to parity with Pure Graph Transformers (Kim et al. 2022). We do not explore this connection.

[4]$g(\cdot)$ is a linear combination over concatenated input vectors, learnable by a first layer of $\Phi_{\texttt{out}}$ without activations.

[5]This is not necessary during learning: a one-hot 'decoder' can be implemented using a two-layer perceptron with ReLU activations.

This expression is equivalent to Equation 6.10, and shows ELENE-L (**ND**) can learn like SPNNs—and, transitively through Remark 4, that Graphormers can be emulated by ELENE-L (**ND**). □

6.3.4 ELENE Is Permutation Equivariant and Invariant

We show that ELENE is permutation equivariant at the graph level, and permutation invariant at the node level, following a similar approach to when we showed that IGEL also has these properties, as shown in Section 5.3.3.
Intuitively, as all operations that ELENE requires are permutation equivariant at the graph level, and permutation invariant at the node level, the same holds for ELENE representations.

Lemma 4. *Given any $v \in V$ for $G = (V, E)$ and given a permuted graph $G' = (V', E')$ of G produced by a permutation of node labels $\pi : V \to V'$ such that $\forall v \in V \Leftrightarrow \pi(v) \in V'$, $\forall (u, v) \in E \Leftrightarrow (\pi(u), \pi(v)) \in E'$.*
All ELENE *representations are permutation equivariant at the graph level:*

$$\pi(\{\{e^k_{v_1}, \dots, e^k_{v_n}\}\}) = \{\{e^k_{\pi(v_1)}, \dots, e^k_{\pi(v_n)}\}\}.$$

Furthermore, ELENE *representations are permutation invariant at the node level:*

$$e^k_v = e^k_{\pi(v)}, \forall v \in V, \pi(v) \in V'.$$

Proof. Note that e^k_v in Equation 6.1 can be expressed in terms of $d^{(p)}_G(u|v)$ and $l_G(u, v)$. Both $l_G(\cdot, \cdot)$ and $d^{(p)}_G(\cdot|\cdot)$ are permutation invariant functions at the node level and equivariant at the graph level, as they rely on the distance between nodes, which will not change when permutation $\pi(\cdot)$ is applied.
Thus, ELENE representations are permutation equivariant at the graph level, and permutation invariant at the node level. □

6.4 Experimental Results

We now study the effect of introducing ELENE and ELENE-L in a variety of graph-level settings, evaluating where *purely structural* ELENE underper-

forms ELENE-L, and the practical impact of ELENE variants. We provide reproducible code, hyper-parameters, and analysis scripts on Github[6]. Our experimental study is split in sections which describe our results in four experimental benchmarks:
A) Expressivity. Evaluates whether models distinguish non-isomorphic graphs (on 1-WL EXP (Abboud et al. 2021) and 3-WL SR25 (Balcilar et al. 2021) equiv. datasets), count sub-graphs (in RandomGraph (Chen et al. 2020)), and evaluate graph-level properties (Corso et al. 2020).
B) Proximity. Measures whether models learn long-distance *attributed* node relationships in h-Proximity datasets (Abboud, Dimitrov, and Ceylan 2022).
C) Real World Graphs. Evaluates performance on five large-scale graph classification/regression datasets from Benchmarking GNNs (ZINC, CIFAR10, PATTERN) (Dwivedi et al. 2020), and the Open Graph Benchmark (MolHIV, MolPCBA) (Hu et al. 2020a).
D) Memory Scalability. Evaluates the memory consumption of ELENE-L on d-regular graphs, varying n and $d_{\texttt{max}}$ to validate the algorithmic complexity analysis in Subsection 6.1.3 and compare memory usage against GIN-AK, GIN-AK$^+$ and SPEN.

6.4.1 Experimental Protocol

Reporting. When reported in the original studies, we show standard deviations for experiments with more than two runs following (Zhao et al. 2022), and highlight best models per task in **underlined bold**. ELENE denotes Equation 6.2 as additional features, while ELENE-L denotes representations of Equation 6.6 and Equation 6.7. **(ED)** denotes ELENE-L with Edge-Centric signals, while **(ND)** denotes a Node-Centric variant that ignores edge-information for ablation studies. '†' indicates results from literature.
Environment. Experiments ran on a shared server with a 48GB Quadro RTX 8000 GPU, 40 CPU cores & 502GB RAM. To measure memory and time costs without sharing resources, we also reproduced our experiments on real-world graphs on a SLURM cluster with nodes equipped with 22GB Quadro GPUs.

[6]`https://github.com/nur-ag/ELENE`

Experimental Setup. We explore sub-sets of Elene hyper-parameters via grid search with $k \in \{0, 1, 2, 3, 5\}$ parameter ranges for Elene and Elene-L, and test the ED/ND variants for Elene-L with embedding params. $\omega \in \{16, 32, 64\}$, $\rho = d_{\texttt{max}}$, using masked-mean pooling for stability. For h-Proximity (Abboud, Dimitrov, and Ceylan 2022), we compare against SPNNs (Abboud et al. 2021) and Graphormer(Ying et al. 2021) as originally reported. For Expressivity and Real World Graphs, we reuse hyper-parameters and splits from GIN-AK$^+$ in (Zhao et al. 2022) without architecture search, comparing against strong MP-GNN baselines from literature where GNN-AK$^+$ underperforms: CIN (Bodnar et al. 2021) for ZINC and SUN (Frasca et al. 2022) for sub-graph counting. We choose GINE (Hu et al. 2020b), an edge-aware variant of GIN (Xu et al. 2019), as our base MP-GNN given that GIN-AK$^+$ outperforms its uplifted counterparts for GCN-AK$^+$ and PNA-AK$^+$ (Corso et al. 2020; Kipf and Welling 2017; Zhao et al. 2022), without running into out-of-memory issues like PPGN (Maron et al. 2019b) in the PPGN-AK instantiation. Finally, for scalability we compare with GNN-AK on benchmark datasets (Zhao et al. 2022) and SPEN (Mitton and Murray-Smith 2023).
Experimental Objectives. We connect *expressivity* and its relation to graph *attributes*, comparing against methods that *do not* perturb graph structure, e.g. DropGNN (Papp et al. 2021); leverage random walks, e.g. RWPE (Dwivedi et al. 2022); or require costly pre-processing e.g. $\mathcal{O}(n^3)$ spectral eigendecompositions as GNNML3, LWPE, or GraphGPS (Balcilar et al. 2021; Dwivedi et al. 2022; Rampášek et al. 2022). Per Subsection 6.3.3, Elene relates to Graphormers via SPNNs, so we focus on sub-graph GNNs and SPNNs.

6.4.2 Expressivity

We test all Elene variants on four MP-GNN expressivity datasets, with results captured in Table 6.1. Elene signals improve GINs as shown by our single-run results on EXP and SR25, which are consistent with Section 6.3. Namely, Elene and Elene-L (**ND**) and (**ED**) all reach 100% accuracy on the 1-WL equivalent EXP task, as expected from Theorem 2. Furthermore, Elene-L (**ED**) can distinguish all 3-WL equivalent `SRG`s in the challenging SR25 dataset—providing empirical evidence for our formal analysis in Theorem 5.

Table 6.1: Expressivity benchmark results. In EXP and SR25, models where ELENE-L is introduced reach the best performance per task, shown in **underlined bold**. We highlight the *best performing configurations from* ELENE *variants on GIN in italics*, which we consistently observe in the Node-Centric (**ND**) configuration except for isomorphism tasks.

Model	**EXP (Acc.)**	**SR25 (Acc.)**	**Count. Substr. (MAE)**				**Graph Prop. ($\log_{10}$(MAE))**		
			Tri.	**Tail Tri.**	**Star**	**4-Cycle**	**IsCon.**	**Diam.**	**Radius**
GIN	50%	6.67%	0.357	0.253	0.023	0.231	-1.914	-3.356	-4.823
SUN$^{\dagger,\text{(Frasca et al. 2022)}}$	—	—	**0.008**	**0.008**	**0.006**	**0.011**	-2.065	-3.674	**-5.636**
GIN-AK$^{\dagger,\text{(Zhao et al. 2022)}}$	**100%**	6.67%	0.093	0.0751	0.017	0.073	-1.993	-3.757	-5.010
GIN-AK^{+}	**100%**	6.67%	0.011	0.010	0.016	**0.011**	-2.512	-3.917	-5.260
GIN+ELENE	**100%**	6.67%	0.024	0.023	0.020	0.041	-2.218	-3.656	-5.024
GIN+ELENE-L (ND)	**100%**	6.67%	*0.012*	*0.015*	*0.014*	*0.016*	*-2.620*	*-3.815*	*-5.117*
GIN+ELENE-L (ED)	**100%**	**100%**	0.023	0.023	0.017	0.023	-2.497	-3.541	-4.755
Best (GIN / GIN-AK) + (ELENE / ELENE-L)	**100%**	**100%**	0.010	0.010	0.014	**0.011**	**-2.715**	**-4.072**	-5.267

On Graph Properties and Counting Substructures (2 runs averaged, as in (Zhao et al. 2022)), a GIN + ELENE-L (**ND**) model consistently outperforms GIN-AK *without context encoding.* In Counting, both ELENE variants and GIN-AK$^+$ are outperformed by SUN, but GIN+ELENE matches or outperforms GIN on every task, showing that ELENE features are informative and can boost performance by themselves.

On both tasks, we find that GIN+ELENE-L (**ED**) performs poorly—outperforming GIN+ELENE but not our baselines. This might be caused by model over-parametrization, as six node and edge-level embedding matrices are learned for 3 and 6 layers on Counting Substructures and Graph Properties respectively[7]. On the Graph Properties tasks of `IsConnected` and `Diameter`, a GIN-AK$^+$ with ELENE-L outperforms state-of-the-art results—and interestingly a GIN with ELENE-L (**ND**) outperforms all existing baselines on the `IsConnected` task. This can be further improved by using a GIN-AK$^+$ with ELENE-L (ND).

6.4.3 *h*-Proximity

We evaluate ELENE-L on h-Proximity (Abboud, Dimitrov, and Ceylan 2022) tasks (10-fold averaged)—where nodes are assigned colors including red and blue, and models must classify where all red nodes have at most two blue nodes within h hops (positive) or otherwise (negative), as shown in Figure 6.2. Models must learn which colors are relevant for the target and capture long-ranging dependencies during learning. Edge information is irrelevant, and pre-computed encodings like ELENE cannot capture interactions of distances and node attributes.

In (Abboud, Dimitrov, and Ceylan 2022), it is reported that MP-GNNs perform well on $h = 1$-Proximity, so we focus on the $h \in \{3, 5, 8, 10\}$ variants. Table 6.2 shows our results, where ELENE-L (**ND**) outperforms strong baselines from SPNNs and Graphormer (Abboud et al. 2021). As expected, a GIN + ELENE did not meaningfully improve over GIN. Our numerical results provide empirical validation for Theorem 6.

[7]Weight sharing may help over-parametrization by learning a single structural representation, trading off expressivity.

Table 6.2: h-Proximity binary classification results (mean accuracy $\pm$ standard deviation). ELENE-L **(ND)** without degree information outperforms baselines strong SPNNs and Graphormer baselines from[†] (Abboud et al. 2021).

	3-Prox.	**5-Prox.**	**8-Prox.**	**10-Prox.**
GCN[†]	50.0 ± 0.0	50.0 ± 0.0	50.1 ± 0.0	49.9 ± 0.0
GAT[†]	50.4 ± 1.0	49.9 ± 0.0	50.0 ± 0.0	50.0 ± 0.0
SPNN ($k = 1$)[†]	50.5 ± 0.7	50.2 ± 1.0	50.0 ± 0.9	49.8 ± 0.8
SPNN ($k = 5$)[†]	95.5 ± 1.6	96.8 ± 0.7	96.8 ± 0.6	96.8 ± 0.6
Graphormer[†]	94.7 ± 2.7	95.1 ± 1.8	97.3 ± 1.4	96.8 ± 2.1
GIN+ELENE	52.0 ± 2.0	51.8 ± 1.2	52.4 ± 2.6	51.4 ± 1.1
GIN+ELENE-L (ND)	$\underline{\mathbf{98.3 \pm 0.5}}$	$\underline{\mathbf{98.6 \pm 0.5}}$	$\underline{\mathbf{99.0 \pm 0.5}}$	$\underline{\mathbf{99.2 \pm 0.3}}$

6.4.4 Real World Graphs

We also evaluate ELENE and ELENE-L on five large real-world graph classification and regression tasks. We test ELENE and ELENE-L on ZINC, MolHIV, PATTERN, CIFAR10 and MolPCBA and report our results in Table 6.3. Given increased memory and computation costs and poorer results of ELENE-L (**ED**) in Subsection 6.4.2, we only evaluate ELENE-L (**ND**). On ZINC, GIN + ELENE-L (3 averaged runs) achieves comparable results to existing baselines, including SUN (Frasca et al. 2022). Furthermore, by introducing ELENE-L on GIN-AK$^+$, the model matches the previous strong baseline achieved by CIN (Bodnar et al. 2021). On PATTERN (3 averaged runs), GIN + ELENE-L achieves comparable results to GIN-AK$^+$, but does not meet the best reported performance of GCN-AK$^+$ by a 0.07% delta. We did not achieve to independently reproduce GIN-AK$^+$ results (Zhao et al. 2022) on MolHIV (5 averaged runs)—finding that GIN with ELENE or ELENE-L is not significantly different ($p < 0.01$) to GIN, while the performance of GIN-AK$^+$ is statistically inferior.
Table 6.4 shows the memory and time performance of ELENE compared to state-of-the-art methods. A GIN + ELENE-L model outperforms GIN-AK in CIFAR and a GIN-AK$^+$ baseline in MolPCBA with reduced time and memory costs. Furthermore, our setup of GIN layers combined with ELENE always outperforms the memory consumption of GNN-AK$^+$.

Table 6.3: Results on real world benchmark datasets (mean ± standard deviation). We compare with published results and reproduce the experiments of (Zhao et al. 2022). Adding ELENE variants to GIN and GIN-AK$^+$ yield state-of-the-art results on ZINC and MolPCBA, and match existing methods on PATTERN and MolHIV.

	ZINC (MAE)	**PATTERN** (Acc.)	**MolHIV** (ROC)	**CIFAR10** (Acc.)	**MolPCBA** (AP)
GSN†	0.115 ± 0.012	—	77.99 ± 1.00	—	—
NGNN†	—	—	78.34 ± 1.86	—	28.32 ± 0.41
CIN†	0.079 ± 0.006	—	$\mathbf{80.94 \pm 0.57}$	—	—
SUN†	0.083 ± 0.003	—	80.55 ± 0.55	—	—
GCN-AK$^{+\dagger}$	0.127 ± 0.004	$\mathbf{86.887 \pm 0.009}$	79.28 ± 0.101	$\mathbf{72.70 \pm 0.29}$	0.285 ± 0.000
GIN-AK†	0.094 ± 0.005	86.803 ± 0.044	0.783 ± 0.012	67.51 ± 0.21	0.274 ± 0.000
GIN-AK$^+$	0.082 ± 0.003	86.868 ± 0.028	77.37 ± 0.31	†72.39 ± 0.38	†0.293 ± 0.004
(Lit. results†)	†0.080 ± 0.001	†86.850 ± 0.057	†79.61 ± 1.19		
GIN	0.155 ± 0.005	85.692 ± 0.042	78.72 ± 0.54	59.55 ± 0.54	0.271 ± 0.003
(Lit. results†)	†0.163 ± 0.004	†85.732 ± 0.023	†78.81 ± 1.01	†59.82 ± 0.33	†0.268 ± 0.001
GIN+ELENE	0.092 ± 0.001	86.783 ± 0.044	78.92 ± 0.35	56.34 ± 0.06	0.277 ± 0.002
GIN+ELENE-L (ND)	0.083 ± 0.004	86.828 ± 0.002	78.26 ± 0.93	68.95 ± 0.25	$\mathbf{0.294 \pm 0.001}$
Best Result (ELENE / ELENE-L)	$\mathbf{0.079 \pm 0.003}$	86.828 ± 0.002	79.15 ± 1.45	68.95 ± 0.25	$\mathbf{0.294 \pm 0.001}$

Table 6.4: Memory and time performance on benchmark datasets. We report average epoch duration in seconds (s) and maximum memory consumption in gigabytes (GB) respectively. Dashed entries indicate executions that terminated due to running out of memory.

	ZINC		**PATTERN**		**MolHIV**		**CIFAR10**		**MolPCBA**	
GIN	6.02s	0.12GB	118.62s	1.42GB	14.88s	0.07GB	98.37s	0.90GB	223.13s	0.44GB
GIN-AK	9.76s	1.11GB	—	—	19.30s	0.64GB	283.93s	18.80GB	534.78s	3.80GB
GIN-AK^{+}	13.63s	1.68GB	—	—	25.47s	0.79GB	—	—	607.89s	3.83GB
GIN+ELENE	6.14s	0.13GB	90.15s	1.47GB	14.94s	0.07GB	120.21s	0.91GB	278.29s	0.46GB
+ELENE-L (ND)	10.23s	0.99GB	146.15s	7.80GB	42.53s	0.54GB	224.43s	10.72GB	451.12s	2.39GB
+ELENE-L (ED)	22.61s	2.85GB	—	—	32.28s	1.40GB	—	—	1025.58s	7.10GB

In ZINC, GIN+Elene-L (**ND**) requires 0.99GB compared to the 1.68GB of GIN-AK$^+$ while reaching comparable performance (0.083 ± 0.004 vs 0.082 ± 0.003, respectively). In MolHIV, GIN+Elene model requires only 70MB during training while outperforming the ROC of our reproduced run of GIN-AK$^+$, which required 790MB—an 11.3-fold reduction in memory usage. On PATTERN, we find that GIN+Elene-L (**ND**) achieves 99.95% of the performance of GIN-AK$^+$ while consuming 7.8GB of memory, compared to the 26.52GB from (Zhao et al. 2022)—a 3.4-fold reduction.
In summary, we find that Elene and Elene-L (**ND**) reach comparable performance to state-of-the-art methods while outperforming them in terms of time/memory efficiency. Using Elene encodings as node-features (**GIN**+Elene) only introduces minor overhead over the GIN baseline, while Elene-L (**ND**) achieves favorable memory performance compared to both GIN-AK and GIN-AK$^+$. Thus, our empirical results may also be interesting in constrained memory environments. Finally, we find that Elene-L (**ED**) incurs in additional memory costs, as expected by leveraging both node and edge embeddings.

6.4.5 Memory Scalability

Finally, we evaluate the scalability of Elene-L in a learning setting. We analyze the memory consumption as a function of the graph size and how it compares with other methods. For that, we follow a similar setting as (Mitton and Murray-Smith 2023) for SPEN: we design and implement a learning task on a large d-regular graph, and use it to explore memory consumption under different values of degree d and nodes n. With this setup, we train the model for 25 epochs to predict a constant variable so that both input tensors and gradient computations are kept in memory.
We evaluate both Elene-L (**ND**) and (**ED**), together with a GIN model without any Elene-L features, GIN-AK and GIN-AK$^+$ as baselines. For all Elene-L variants, we execute the benchmark with different values of $k \in \{1, 2, 3\}$.
Figure 6.7 shows our results when $d_{\texttt{max}} = 12$. As expected, the memory cost grows exponentially as a function of n. We observe that Elene-L (**ND**) with $k = 2$ can scale up to graphs with $10,000$ nodes with ego-network sub-graphs. Since all nodes have the same degree $d_{\texttt{max}} = 12$, at 2 hops we are guaranteed to find the root node, its 12 neighbors, and at least

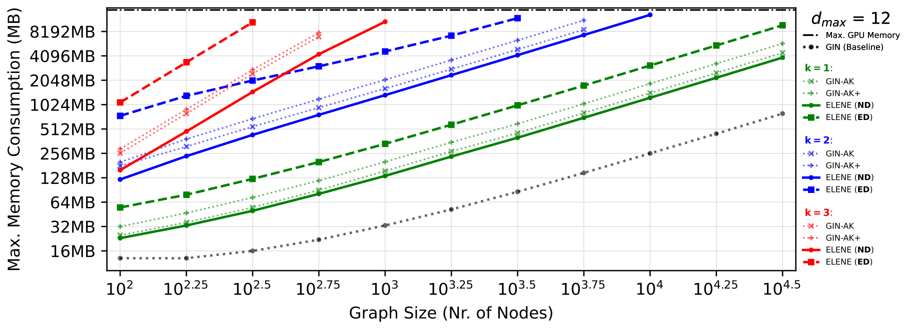


Figure 6.7: Memory scalability analysis of ELENE when $d_{\texttt{max}} = 12$. We include GIN (dotted line) and maximum GPU memory (dash-and-dotted line) as indicative lower and upper memory bounds. ELENE-L (**ND**, full lines) outperforms both GIN-AK, GIN-AK$^+$ (dotted lines) and ELENE-L (**ED**, dashed lines). Additionally, ELENE-L (**ND**) can encode all d-regular graphs in the benchmark when $k = 1$. As expected, memory consumption increases linearly with the number of nodes as $d_{\texttt{max}}$ is kept fixed.

one additional neighbor at 2 hops —or 14 total nodes. In practice, as the graphs are randomly generated, we find that each of the 2-depth subgraphs contains an average of 144.13 nodes (with the expected maximum at 145). Additionally, our experiments show that ELENE-L (**ND**) with $k = 1$ can scale up to graphs with $10^{4.5} = 31,623$ nodes with up to $d_{\texttt{max}} = 18$. Furthermore, despite requiring additional memory, the more expressive ELENE-L (**ED**) can nevertheless be used for $k = 1$ for graphs with up to $10^{4.5}$ nodes as well. We provide additional results for $d_{\texttt{max}} = 6$ and $d_{\texttt{max}} = 18$, as well as different graph density patterns, in Appendix C.2.2.

Recent methods like SPEN outperform global permutation-equivariant methods like PPGN. However, SPEN still struggles to process significantly smaller graphs, with $n \approx 1,000$ nodes and $k = 1$ depth ego-networks that contain 9 nodes, even with higher GPU memory as reported by Mitton and Murray-Smith (2023). Compared with the complexity of SPEN, which is $\mathcal{O}(n \cdot |\mathcal{V}_v^k|^2)$, ELENE-L can encode ego-networks with 16 times more nodes with comparable memory usage—while outperforming subgraph GNN baselines like GIN-AK and GIN-AK$^+$.

6.4.6 Experiments Summary

ELENE and ELENE-L consistently boost GNN performance on the four experimental benchmarks.

On Expressivity, Subsection 6.4.2 gives empirical support for Theorem 5, i.e. that ELENE-L (**ED**) can distinguish `SRG`s, *achieving 100% accuracy* on the challenging SR25 (Balcilar et al. 2021) dataset. Although SUN (Frasca et al. 2022) outperforms other models on Counting Substructures, ELENE and ELENE-L still improve baseline performance and match previous GIN-AK and GIN-AK$^+$ baselines respectively.

On Graph Properties, GIN+ELENE-L matches existing baselines, and a GIN-AK$^+$ model with ELENE-L *outperforms* previous state-of-the-art results on the `IsConnected` and `Diameter` tasks with -2.715 and -4.072 $\log_{10}$(MSE) respectively.

On h-Proximity, Subsection 6.4.3 validates Theorem 6, i.e., that ELENE-L (**ND**) is at least as expressive as SPNNs (Abboud, Dimitrov, and Ceylan 2022), as ELENE-L (**ND**) *outperforms SPNNs and Graphormers* at capturing *attributed structures*—that ELENE encodings alone *cannot capture.*

On Real World Graphs from Subsection 6.4.4, ELENE and ELENE-L reach state-of-the-art results. On ZINC, a GIN-AK$^+$ model with ELENE-L achieves 0.079 ± 0.003 MAE, matching CIN (Bodnar et al. 2021). A GIN+ELENE-L matches 99.95% of the performance of baselines on PATTERN while *consuming 3.4× less memory*, and GIN+ELENE reaches 0.1% less accuracy than GIN-AK$^+$ but does so using 1.47GB, compared to the 26.54GB reported for GIN-AK$^+$—*a 18.1× memory reduction.* To conclude this benchmark, we find that a GIN+ELENE-L *matches state-of-the-art results on MolPCBA* (0.294 ± 0.001 vs 0.293 ± 0.003 of GIN-AK$^+$ (Zhao et al. 2022)) without hyper-parameter tuning and while *consuming 37.60% less memory (2.39GB vs 3.83GB).*

Finally, Table 6.4 shows that ELENE and ELENE-L (**ND**) have favorable resource consumption compared to sub-graph GNN baselines. On Memory Scalability, Subsection 6.4.5 shows that ELENE-L can be used on d-regular graphs with more than 10^4 nodes where $d_{\texttt{max}} \in \{6, 12, 18\}$, validating the expected memory costs from Subsection 6.1.3 and outperforming the memory consumption of recent methods like SPEN (Mitton and Murray-Smith 2023).

6.5 Conclusions

We presented Elene, a principled edge-level ego-network encoding capturing the structural signals sufficient to distinguish 3-WL equivalent `SRG`s. We proposed two variants—Elene and Elene-L—and showed that Node-Centric and Edge-Centric representations exhibit different expressive power. To position our findings, we formally drew connections between Elene and recent Sub-Graph GNNs, Graph Transformers, and Shortest Path Neural Networks.
Empirically, we evaluated our methods on 10 different tasks, where the sparse Elene vectors improve performance on structural expressivity tasks. Our learnable Edge-Centric Elene-L variant boosts MP-GNN expressivity to reach 100% accuracy on the challenging SR25 dataset, while its Node-Centric counterpart improves over a strong baseline on the h-Proximity task and matches state-of-the-art results in several real-world graphs. Finally, we found our methods provide a trade-off between memory usage and structural expressivity, improving memory usage with up to $18.1\times$ lower memory costs compared to sub-graph GNN baselines.

Part III

Conclusions & Future Directions

Chapter 7

Conclusions

In the two previous parts of this thesis we have presented two facets of representation learning: textual emotion detection and structural graph encodings. As we initially discussed in Section 1.4, our end goal was to improve the understanding of existing limits, modelling approaches, and performance considerations on both domains—and eventually their combination.

In Chapter 3, we studied representations of text for the task of Emotion Detection, evaluating an array of statistical and neural representations in Section 3.1. In this way, we addressed research question **A1**: *what are the performance limits in textual Emotion Detection?* Although our findings pushed the state-of-the-art performance on the GoEmotions dataset and defined the first ever baseline for Vent (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2021b), the fine-tuning approach we presented is not new (Demszky et al. 2020; Lewis et al. 2020; Raffel et al. 2020). However, by sharing our systematic procedure and research artifacts, including our modelling library and visualization tools, our work remains extensible and compatible with novel architectures and alternative tasks (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2021a).

We also explored a different direction to answer research question **A2**: *are there differences in how writers express and readers perceive emotions in social media?* Namely, we focus our analysis on studying the *perspective* of the data that models are trained on. Do models trained on content produced by human *authors* learn to identify what *readers* would understand?

As natural language models grow larger and go on to be trained on larger, private data sources, the distinction between the perspectives of *readers* and *writers* might help identify shortcomings and biases. With our analysis and human-evaluation task, we released a novel dataset containing 2640 human-annotated samples to study emotion detection through the lens of readers and writers in a social network setting. This complements existing multi-perspective datasets on the Stanford Sentiment Treebank (SST) and Manually-Anotated Sub-Corpus (MASC), containing 40 annotated samples each (Buechel and Hahn 2017a,b). Finally, our emotion detection work spurred some follow-up research, particularly extending our results on Vent in the emerging field of guilt detection from text (Meque et al. 2023).

The other facet of our work has been graph representation learning. In the second part of the dissertation, we began by presenting a novel ego-network encoding algorithm dubbed IGEL and studying its relationship with the well-known Weisfeiler-Lehman (1-WL) algorithm. In Lemma 2, we have shown that IGEL is capable of distinguishing graphs that 1-WL cannot differentiate—and more expressive representations, such as those captured by MATLANG or Shortest Path Neural Networks on unattributed graphs. Subsection 5.3.2 also showed an upper-bound on distinguishability on Strongly Regular Graphs, which are known to be indistinguishable by the more powerful 3-WL test. Our theoretical results help connect ego-networks and the properties of the nodes within them—namely degrees and distances—with common expressivity measures of GNNs, as studied in (Arvind et al. 2020; Frasca et al. 2022; Morris et al. 2021; Papp and Wattenhofer 2022; Zhao et al. 2022).

In a practical setting, we showed that IGEL can be computed efficiently through Breadth-First Search, and produces permutation-equivariant representations. As a result of those properties, we suggested that IGEL may be used to efficiently generate features on unseen graphs. We executed an experimental benchmark to assess our theoretical claims in Section 5.4, evaluating IGEL on six experimental tasks on eleven datasets. Empirically, introducing IGEL as features matched or statistically improved the performance of MP-GNNs, sub-graph GNNs, spectral GNNs, and even graph-unaware multi-layer perceptrons—only by introducing IGEL encodings as features without architecture modifications.

We continued our analysis of expressivity in graph representations in Chapter 6, studying what structural attributes are necessary to distinguish challenging Strongly Regular Graphs. To address the limitations of previous methods, we presented Elene, an edge-aware ego-network encoding that extends the work on Igel and was shown to distinguish Strongly Regular Graphs, intuitively in Subsection 6.1.2 and formally in Theorem 3. We identified scenarios in which edge-level and sub-graph information improve the expressive power of models in terms of Weisfeiler-Lehman tests and logical expressivity. We studied the limitations of encoding methods like Igel on *attributed* settings. As a solution, we proposed a learnable variant of Elene dubbed Elene-L in Section 6.2 that could combine structural attributes from degrees, distances and edges with node and edge-attributes.

We provided an examination of the expressivity of Elene with and without edge-level signals, showing that introducing information across edges increases expressivity in Section 6.3. Furthermore, we connected Elene with Igel and Shortest Path Neural Networks, highlighting the relationship between ego-network encodings and recent GNN architectures. We implemented three empirical benchmarks to evaluate Elene and Elene-L on ten different tasks measuring expressivity, proximity and real-world performance. Overall, we found Elene signals outperform previous baselines on expressivity and proximity while reaching competitive performance on real world graphs, improving memory scalability with up to $18.1\times$ lower memory costs than state-of-the-art sub-graph GNNs.

With Igel and Elene we presented our line our research to improve the expressivity and predictive performance of graph neural networks through ego-network encodings. In doing so, both works incrementally addressed our research questions: *Is it possible to enhance graph neural networks through the incorporation of a set of predefined structural features,* **B1.** *in terms of expressive power?* and **B2.** *in terms of empirical performance?* Our results show that it is possible to design a set of features that efficiently improve the expressivity of any MP-GNN with Igel. We formally identified differences in expressivity between ego-network encodings depending on whether edge-level information is used. Our results on the learnable variants of Elene also indicated that the node-centric Elene-L reaches comparable performance to state-of-the-art sub-graph GNNs at lower time and memory costs. Edge-centric Elene-L, on the other hand, empirically validated our theoretical work by distinguishing all challenging Storngly Regular Graphs in the SR25 dataset—but at increased memory and computation costs and comparable or worse performance on empirical tasks. Our results on Elene highlight the differences in performance of raw ego-network encodings and their learnable variants, and how expressivity improvements and performance improvements can be independent. This result contrasts with recent research focusing on k-WL expressivity, which as we find is a subset of *all you need.*

7.1 Ethical Considerations and Societal Impact

Our main contributions on textual emotion detection leverage data collected from two social media platforms—Reddit and Vent. As a result, we believe that our analyses, particularly as they pertain to the differences in perspective between readers and writers, might have a broader interest for social research.

However, as we highlighted in Section 3.4, follow-up research must be carried on to understand the potential biases encoded in the GoEmotions and Vent data sets. Another concern is the potential ethical consideration arising from anonymisation issues that might may be surfaced on the Vent dataset. To address potential risks for data leakage where the original training data might be retrieved from the trained model, we provide access to our models behind a web interface or under request so that model weights are not readily accessible. Downstream work to deploy our techniques or their derivatives in user-facing products should follow ethic reviews to ensure that there is no potential for abuse.

Our main contributions with Igel and Elene are (a) a novel family of edge-aware features that can be used alone or during learning in MP-GNNs, with (b) a formal analysis of their expressivity that shows they can distinguish challenging `SRG`s, and (c) experimental results matching state-of-the-art learning models with favorable memory costs.

We do not foresee ethical implications of our theoretical findings. Our experimental results are competitive with state-of-the-art methods at a lower memory footprint, which may help solve tasks with limited memory budgets.

7.2 Limitations of our Work

In this section, we highlight limitations from our research and note opportunities for future work which we discuss further in Section 7.3.

On textual emotion detection, our work focused on modelling language through contextual language models like BERT. However, neural language representations have evolved rapidly in recent years. Novel architectures and modelling approaches have appeared and revolutionized the field, as we superficially introduced in Chapter 1.

In the future, it would be interesting to study the impact of larger foundation models, trained on larger corpora—and their implications to emotion detection in aspects as varied as multi-linguality, perspectives, and trade-offs between modelling performance and efficiency.

On graph representation learning, our work has focused on studying expressivity through the lens of ego-network graphs and their structural properties. Although we have connected our method with existing architectures, our encoding and learning approaches have only explored simple vector representation and embeddings. We believe that exploring ways to encode structural information—degrees, distances, and nodes directions—might help design simpler yet more expressive graph neural networks.

In both directions of our work, we addressed the independent research questions that we posed in Section 1.4. However, our original motivation was to merge both directions of research to answer a pending research question: *how can textual and graph representations be effectively combined?*

This initial motivation was driven by the industrial partnership described in Section 1.1. The vicissitudes of a pandemic, a company bankruptcy, and a change of employment while performing the research had an impact on the direction and scope of our work. However, the overall objective is still valuable, as highlighted by parallel work to connect emotion detection and graph representations in natural language applications (Joshi et al. 2022).

In the same manner, leveraging graph neural networks to improve web search engines is not novel. Retrospectively, it was the original use-case for the graph neural networkmodel (Scarselli et al. 2005). Recent works on session-based recommendation (Wang et al. 2020) and recommender systems (Liu, Ounis, and Macdonald 2022) show the interest in leveraging graph information in information retrieval tasks—sometimes forgoing the message-passing mechanism for efficiency, as in our ego-network encodings.

7.3 Future Directions

In Section 3.3, we studied how emotions collected on a social media setting were perceived differently between *readers* and *writers*. Evaluating models trained on writer-provided perspectives performed well on reader-provided annotations. However, we did not study whether previous messages from an author would improve the predictive power of the model. This modelling approach could be suited to recent generative large language models, where previous messages would be part of a 'user-prompt' modelling their profile and posting history to produce the next message. Furthermore, Vent data offers opportunities beyond our work, as it also provides timestamps and network links—meaning that it is possible to consider whether a given post is influenced by *recent messages* posted by other users connected to the author. Linking to our motivation in Section 1.5, we believe studying the connection between text and networks could provide insights on emotional communication in social graphs.

On graph representation learning, we believe that it would be valuable to further connect our methods with existing hierarchies beyond the k-WL tests. One area of interest would be to extend the analysis on MATLANG after introducing a transductive operation like e.g. matrix-vector products with one-hot encoded vectors. Another interesting direction is the study of GNN expressivity through the lens of logical expressions (Grohe 2021). We believe this framework to model expressivity is valuable as it connects graph computations with the kind of statements that can be *learned* by the models performing them.

The work on the h-proximity tasks from (Abboud, Dimitrov, and Ceylan 2022) that we extend in Chapter 6 can be understood as a first step in this direction. Intuitively, IGEL allows graph models to distinguish degrees and distances within the ego-network, so statements of the form '*how many of my k-hop neighbours have degree less or equal than d?*' can easily be implemented. ELENE and ELENE-L extend the same notion to the directions of edges, and their combination with node and edge attributes respectively. In the future, it would be valuable to explore the kinds of logical statements that our methods are *unable to express*, and their relationships to other expressivity hierarchies, if any. We believe this direction presents an opportunity to identify the learnable *semantics* of models with direct applications for interpretability.

Finally, we believe that another interesting area of work is to connect *expressivity* with *learning efficiency*: not only what computations can be captured by a GNN model, but also how many training data-points are required to do so. A first step in this direction would be to implement a generative benchmark of graph-level logical tasks such as '*how many nodes are red?*', '*how many blue nodes have a green edge?*', '*are there no green nodes with at most two red neighbours at k hops?*', or '*are all paths between red nodes composed of yellow edges?*' These synthetic tasks would be a way to measure the number of samples required to capture certain families of statements. A natural follow-up would be to connect the synthetic benchmarks with results on real-world tasks in which learning efficiency on empirical graph data sets.

Connecting the learning efficiency of graph models with their expressivity might help us grapple with Minsky's definition of artificial intelligence that opens Chapter 1. After all, *the science of making machines capable of performing tasks that would require intelligence if done by humans* must let us learn to generalize from limited data!

Part IV

Appendices

Appendix A

Appendix: Emotion Detection

In this appendix, we provide additional materials on four sub-sections. Appendix A.1 describes the datasets used in our analysis in further details while Appendix A.2 provides additional details about our experimental benchmark. We provide extended experimental results in Appendix A.3, and Appendix A.4 provides an overview of the open-source software methods used and published with our work, as presented in (Alvarez-Gonzalez, Kaltenbrunner, and Gómez 2021a).

A.1 Dataset Details

A.1.1 GoEmotions

The GoEmotions dataset is built from comments sampled from a Reddit dump from 2005 to 2019. Snippets are filtered to (a) reduce profanity and offensive or discriminating content, (b) keep a consistent comment length of under 30 tokens with a median of 12 tokens per comment and (c) balance sentiment, emotion and subreddit popularities. Proper names and religions are masked with special, replacing their occurrences in the text with `[NAME]` and `[RELIGION]` respectively.

A.1.2 Vent

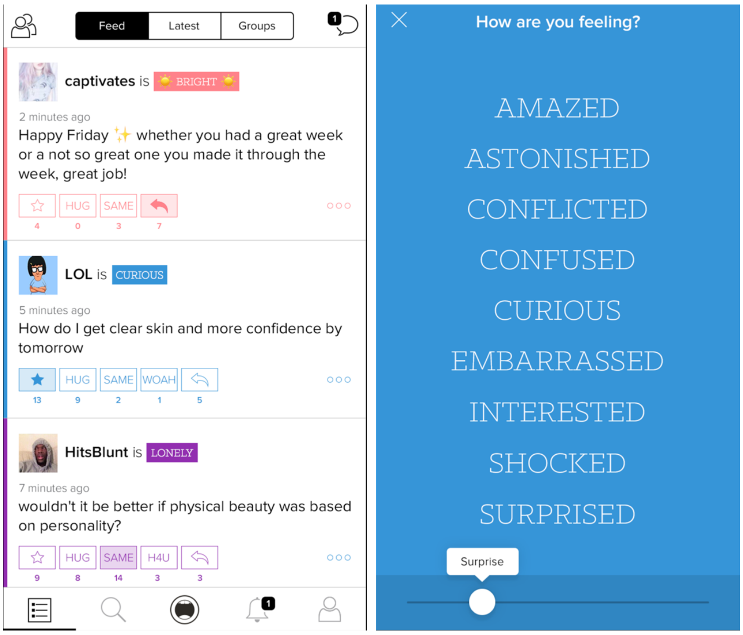
The Vent dataset is self-annotated by Vent users (writers) using the interfaces shown in Figure A.1. The distribution of comment lengths in number of tokens is shown in Figure A.2. We used 32 as the maximum number of tokens per Vent in alignment with the 75th percentile in the length distribution.

Filtering procedure

We filter the dataset to (a) contain months with sufficient volume and (b) only contain stable emotions that appear every month onwards from the chosen point.
In Figure A.3, we show the distribution of emotions over time in terms of the overall frequency of emotions from each of the 9 emotion categories. In (Lykousas et al. 2019), the authors report user activity peaks in April 2015, and slowly decreases afterwards.
We select the filter in July 2016 according to two criteria: first, we compute the number of valid distinct emotions over time (see main text for definition of a valid emotion). This is shown in Figure A.4 (blue line). We observe that after July 2016, the large majority of distinct emotions have been created. Second, to characterize their stability, we compute the relative

# examples	58,009
# emotions	27 + neutral
# unique raters	82
# raters / example	3 or 5
Marked unclear or difficult to label	1.6%
Labels per example	1: 83% 2: 15% 3: 2% 4+: .2%
# examples w/ 2+ raters agreeing on at least 1 label	54,263 (94%)
# examples w/ 3+ raters agreeing on at least 1 label	17,763 (31%)

Table A.1: General dataset and annotation statistics for the GoEmotions dataset. Taken from (Demszky et al. 2020).


Feed | Latest | Groups

captivates is BRIGHT

2 minutes ago

Happy Friday ✨ whether you had a great week or a not so great one you made it through the week, great job!

☆ 4 | HUG 0 | SAME 3 | ↩ 7

LOL is CURIOUS

5 minutes ago

How do I get clear skin and more confidence by tomorrow

★ 13 | HUG 9 | SAME 2 | WOAH 1 | ↩ 5

HitsBlunt is LONELY

7 minutes ago

wouldn't it be better if physical beauty was based on personality?

☆ 9 | HUG 8 | SAME 14 | H4U 3 | ↩ 3

How are you feeling?

AMAZED

ASTONISHED

CONFLICTED

CONFUSED

CURIOUS

EMBARRASSED

INTERESTED

SHOCKED

SURPRISED

Surprise

(a) Vent Feed (b) Emotion Picker

Figure A.1: Screenshots of the interface of Vent app (Lykousas et al. 2019).

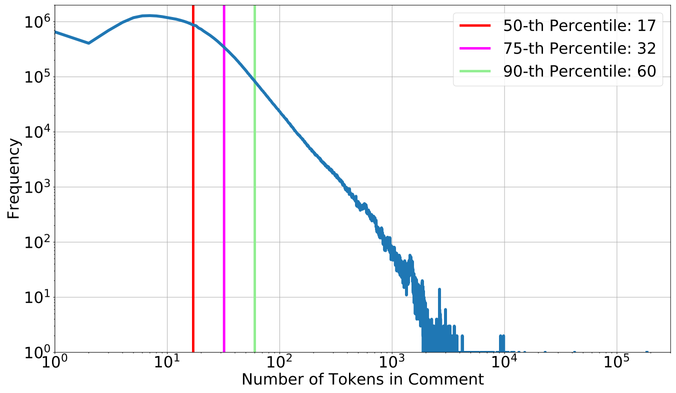


Figure A.2: Vent comment length distribution in number of tokens.

entropy between the emotion distributions of two consecutive months. This is shown in Figure A.4 (red lines). The emotion distributions consider the intersection (dashed red) or union (solid red) of emotions in each pair of consecutive months. We observe that after our cut-off in July 2016 both curves show that relative entropy takes regular values below 0.02, indicating that the use of emotions is stable after that month.

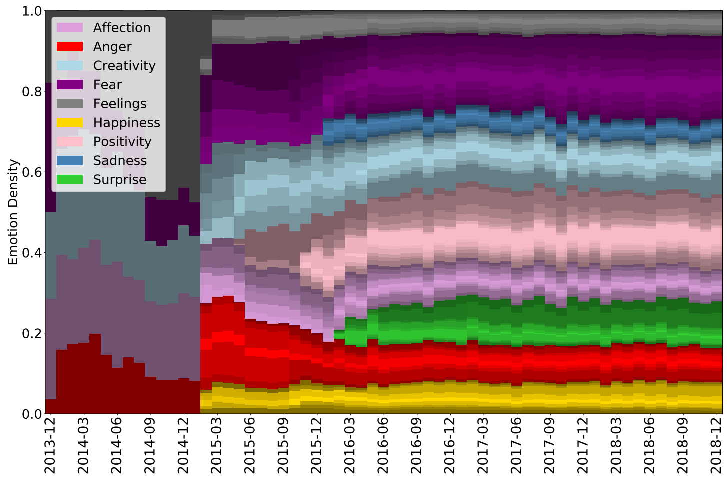


Figure A.3: Category-Level densities on a month-by-month basis. Each color represents an individual emotion, with its total size being the percentage of the emotion across all messages in that month. Densities are irregular initially, until the number of emotions in the app plateaus and all categories become defined.

Obscene Word Analysis

Unlike other works like GoEmotions (Demszky et al. 2020), we do not exclude content with foul language from the training process. Our rationale is that Vent is a dataset collected from a moderated social network. In this sense, we assume moderators remove toxic or unacceptable behaviour, while hypothesizing that foul language is a strong emotional signal that is necessary to capture emotional messages in certain categories. We note, however, that we do filter `nsfw` (not safe for work) content during the

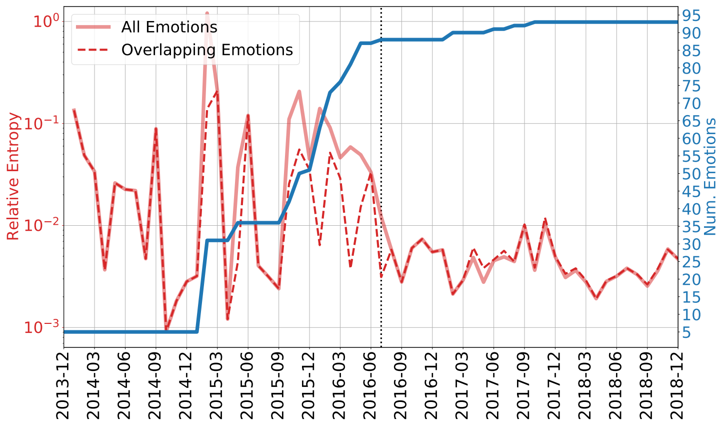


Figure A.4: Relative Entropies and Distinct Emotion Counts month over month. The relative entropy of *all* emotions includes peaks when new emotions are released, while overlapping emotions capture the stability of emotions that persist between two months. The number of emotions grows over time as new emotions are released and replaced. We find that the number of emotions and the relative entropies stabilize after 2016-07, represented by the dashed mark.

human annotation task to avoid exposing readers to inappropriate content. In order to justify the decision to keep obscene terms, we study whether there are significant differences in the degree of obscenity for each emotion and category. We flag whether or not a comment is obscene by detecting 763 words from two publicly available obscene [1] and bad word [2] lists. We exclude frequency analysis from our study: rather than accounting for obscene word frequencies, we use a binary indicator that signals whether an obscene word from either list is observed in a Vent comment.
We observe significant differences ($p < 10^{-20}$, using bootstrapped z-tests) in the percentage of vents containing obscene words between categories, estimated with bootstrapping on 100 independent runs with 10% of the dataset sampled with replacement.

[1] `https://github.com/RobertJGabriel/Google-profanity-words/blob/master/list.txt`

[2] `https://code.google.com/archive/p/badwordslist/downloads`

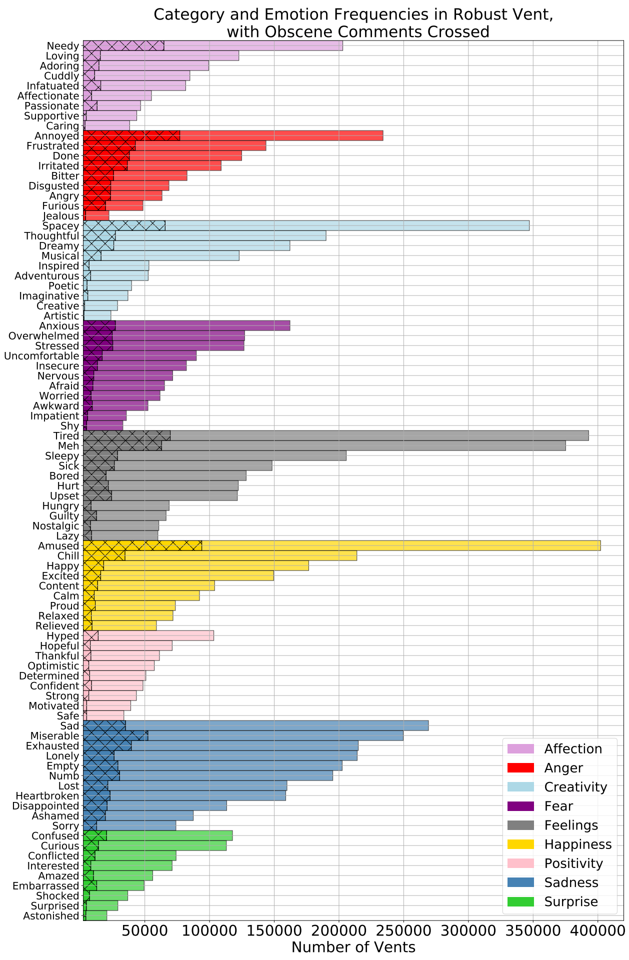


Figure A.5: Vent emotion labels and their frequencies after filtering for relevant emotions. Crossed darker regions represent the proportion of a given category that contains at least one obscene term. The colors showcase the emotion categories, which approximately align with Ekman's: *happiness, fear, sadness, surprise* and *anger* are present, while *disgust* is treated as an emotion under *anger*. There are four additional categories that match higher-level subjective states: *creativity, positivity, feelings* and *affection*.

We find that 10% of the comments in the Positive category contain obscene words (lowest) while 32% of Anger vents use obscene language (highest). Figure A.5 provides an overview of the number of snippets per emotion and category, and the proportion of obscene comments for each emotion.

A.2 Detailed Benchmark Design

A.2.1 Representation Approaches

We implement statistical and embedded representations. In particular, we employ 4 different algorithms:

1. **Bag-of-Words.** Each document is represented as a sparse vector containing the counts of every word. We limit the vocabulary size after filtering English stop words using Scikit-learn's list (Pedregosa et al. 2011) but without applying any transformations like stemming. Our vocabulary contains only the most frequent words, which we tune as a hyper-parameter shown in Table A.2 and Table A.3.
2. **TF-IDF.** Each document is represented as a sparse vector containing the normalized term-frequency divided by inverse document frequency. We apply the same vocabulary-building policy as on Bag-of-Words, including stop-word filtering and using the top most frequent words tuned as a hyper-parameter shown in Table A.2 and Table A.3.
3. **FastText.** We use a pre-trained unsupervised English FastText (Bojanowski et al. 2017b) model to embed the sequence of tokens in a sentence. We limit the length of the sentence as a hyper-parameter.
4. **BERT.** We use a pre-trained BERT (Devlin et al. 2019) English model to tokenise and embed the sequence of tokens in a sentence. Our design permits gradients to propagate when BERT is used as an input to a downstream neural model. We limit the length in total tokens and whether or not to propagate gradients (e.g. freeze BERT) as hyper-parameters shown in Table A.2 and Table A.3.

A.2.2 Modeling Methods

We employ five different learners:

1. **Naive Bayes.** We employ Scikit-learn (Pedregosa et al. 2011)'s implementation of Multinomial Naive Bayes in a one-vs-rest setting for multi-label classification.
2. **Logistic Regression.** Likewise, we use Scikit-learn's SGD-based Logistic Regression in a one-vs-rest setting for multi-label classification.
3. **Incremental Random Forests.** We use an extension to Scikit-learn's Random Forest classifier to incrementally build the forest in a mini-batch fashion, using the `incremental-trees` Python library (Jones 2019). Each mini-batch grows a fixed number of trees, up to a maximum number, for all batches.
4. **(Pooled) Neural Networks.** We implement a simple DNN architecture, applied to every embedded token in parallel with N output neurons, as many as there are classes. The final result is computed by applying a pooling function over the states, as shown in Figure 3.3a.
5. **(Bi)-LSTM.** We implement a stacked (Bi)-LSTM architecture that consumes an embedded sequence in a seq2seq manner, optionally processing in a single direction. The recurrent network captures inter-token relationships and produces N output neurons per token. The final result is computed by applying a pooling function over each token, as shown in Figure 3.3b.

A.2.3 Hyper-parameter Configuration

All the hyper-parameters used when tuning the algorithms used in our experiments are shown in Table A.2 and Table A.3. Each row shows a hyper-parameter affecting either the whole training process, the representation method, or the learning algorithm. Values in **bold** were chosen through Grid Search as the best performing values for that particular component and data set. Our grid search optimizes micro-F1 for each method, running on a Slurm cluster with 8GB Nvidia GPUs. Each configuration was executed once to identify candidates, with the final experiments executing the best performing configuration on 5 independent runs using different seeds. In the code samples, we provide the original `.json` files containing the exact run configurations for reproducibility, including execution files that contain the per-run evaluation metrics to trace the notebooks used to compute the values reported in the chapter.

Dataset	Component	Hyper-parameter	Values
GoEmotions	Global	Batch Size	100000 (statistical methods), 1024 (FastText), 64 (BERT)
	Bag-of-Words	Vocabulary Size	**5000**, 10000, 20000, 40000
	TF-IDF	Vocabulary Size	**5000**, 10000, 20000, 40000
	BERT	Freeze Model Max Length	True, **False** bert-base-cased, **bert-base-uncased** 25
	FastText	Model Max Length	Common Crawl English Model 25
	Log. Reg.	Epochs α Tolerance	1, 10, 50, **100** 0.00001, **0.0001**, 0.001, 0.01, 0.1, 1.0 0.001
	Naive Bayes	Smoothing Factor	1, **0.1**, 0.01, 0.001, ..., 1e-10
	Random Forest	Trees per Batch Max. Depth Max. Features Fraction Split Criterion	**1000**, 2000, 3000 3, 4, 5, 6, **7** **0.05**, 0.1, 0.2, 0.4 Entropy
	DNN Pool	Hidden Size Num. Layers Num. Epochs Learning Rate Epsilon Activation Pooling Function Optimizer	100 1, **2**, 3 **30**, 40, 50, 60 0.01, 0.001, **0.0001** 1e-5, **1e-6**, 1e-7 ELU, **Tanh** Attention, Mean, **Max** AdamW
	Bi-LSTM	Hidden Size Num. Layers Num. Epochs Learning Rate Epsilon Bidirectional Pooling Function Optimizer	100 1, **2** **30**, 40, 50, 60 **0.01**, 0.001, 0.0001 **1e-5**, 1e-6, 1e-7 **True**, False Attention, **Mean**, Max AdamW

Table A.2: Hyper-parameters used by the different representation and modelling algorithms tested for GoEmotions. Each hyper-parameter configuration is tested once to find candidate configurations with micro-F1 on the validation set as our objective. In **bold**, the best performing hyper-parameter value out of all configurations using a specific method.

A.2.4 Experimental Setting

We perform 5 independent experiment runs using different seeds for all underlying libraries used in our implementation. For every experiment configuration, we report the average performance across the 5 runs computed for each metric.

Dataset	Component	Hyper-parameter	Values
Vent	Global	Batch Size	100000 (statistical methods), 2048 (FastText), 64 (BERT)
	Bag-of-Words	Vocabulary Size	**5000**, 10000, 20000, 40000, 80000
	TF-IDF	Vocabulary Size	**5000**, 10000, 20000, 40000, 80000
	BERT	Freeze Model Max Length	True, **False** bert-base-cased, **bert-base-uncased** 40
	FastText	Model Max Length	Common Crawl English Model 40
	Log. Reg.	Epochs α Tolerance	1, 2, 10, **50**, 100 **0.00001**, 0.0001, 0.001, 0.01, 0.1, 1.0 0.001
	Naive Bayes	Smoothing Factor	1, 0.1, **0.01**, 0.001, ..., 1e-10
	Random Forest	Trees per Batch Max. Depth Max. Features Fraction Split Criterion	**66**, 125, 250, 500 4, **5** **0.05**, 0.1, 0.2, 0.4 Entropy
	DNN Pool	Hidden Size Num. Layers Num. Epochs Learning Rate Epsilon Activation Pooling Function Optimizer	**100**, 200 1, **2**, 3, 4, 5 1, 2, **3** **0.01**, 0.001, 0.0001 **1e-5**, 1e-6, 1e-7 **ELU**, Tanh **Attention**, Mean, Max AdamW
	Bi-LSTM	Hidden Size Num. Layers Num. Epochs Learning Rate Epsilon Pooling Function Bidirectional Optimizer	100, **200**, 400 1, **2**, 3 1, **2** 0.01, **0.001**, 0.0001 **1e-5**, 1e-6, 1e-7 **Attention**, Mean, Max **True**, False AdamW

Table A.3: Hyper-parameters used by the different representation and modelling algorithms tested for Vent. Each hyper-parameter configuration is tested once to find candidate configurations with micro-F1 on the validation set as our objective. In **bold**, the best performing hyper-parameter value out of all configurations using a specific method.

A.3 Extended Experiment Results

In this section, we extend the results of Subsection 3.2.2, reporting per-category performances beyond the results from Chapter 3 in Table A.4, and show the confusion matrix for all emotions in Figure A.6.

Category	Precision	Recall	F1-Score
Affection	**0.51**	0.48	0.49
Anger	0.42	0.37	0.40
Creativity	*0.25*	0.41	0.31
Fear	0.39	0.36	0.35
Feelings	0.33	0.45	0.38
Happiness	0.41	0.45	0.43
Positivity	0.29	*0.26*	*0.27*
Sadness	**0.51**	**0.52**	**0.51**
Surprise	0.30	0.35	0.33

Table A.4: Per-category performance of the best performing model (BERT / Bi-LSTM) on Vent. In **bold**, the best performing category for each metric; in *cursive*, the worst performing category.

A.3.1 Human Reader Evaluation

We provide additional details on the design of the HIT, experimental results, and our compensation structure to ensure that workers on our HITs receive a fair compensation.

Annotation Procedure

Workers are tasked to annotate the emotions for 10 comments, whose order is shuffled in every distinct session to avoid position biases. The annotation workflow starts with a set of instructions, shown in Figure A.7, emphasizing they have to match the author's (writer) emotion. Then the task starts showing texts, as illustrated in Figure A.8. Workers first select an emotion category and then choose the emotion most fitting the instructions. Upon selecting an emotion, a message prompts the worker on whether the prediction was correct or not at the emotion and category level, given the original Vent ground truth.
To ensure that workers submit quality work, and that the assessment is equal for all workers, we define approval rules based on their performance. For any submission, we expect that it is distinguishable from random choices, which means that it must contain correct predictions for at least

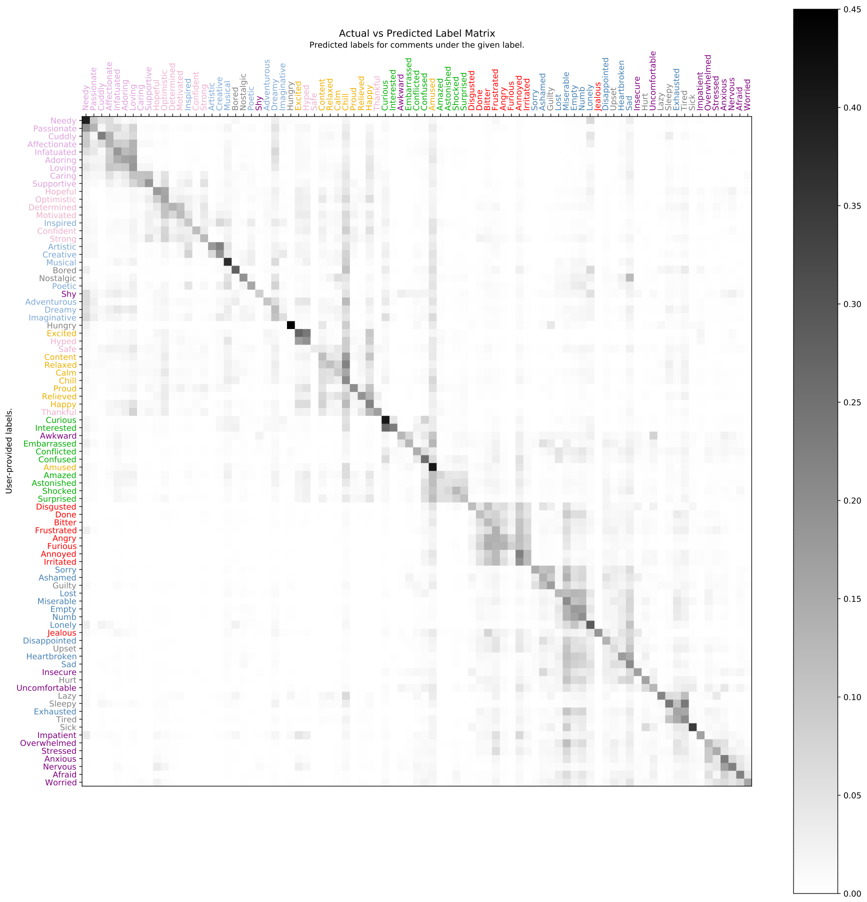
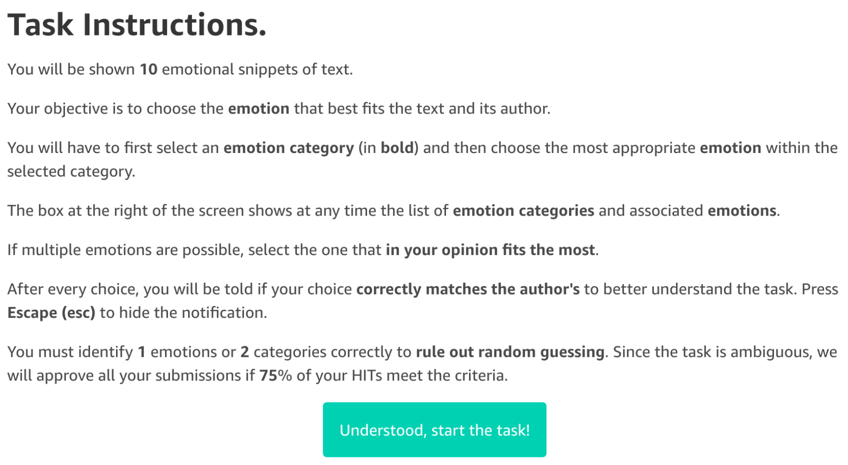


Figure A.6: Vent emotion confusion matrix for the best performing model (BERT + Bi-LSTM).

1 out of 88 emotions or 2 out of 9 emotion categories. We implement both constraints as an automated check of annotator quality, accepting or rejecting the provided annotations for every new submission. In order to account for the ambiguity of the task and avoid penalising quality workers[3], we approve all tasks submitted by workers that meet the qual-

[3]During our trial run (whose data we do not use in our final study), workers contacted



Task Instructions.

You will be shown **10** emotional snippets of text.

Your objective is to choose the **emotion** that best fits the text and its author.

You will have to first select an **emotion category** (in **bold**) and then choose the most appropriate **emotion** within the selected category.

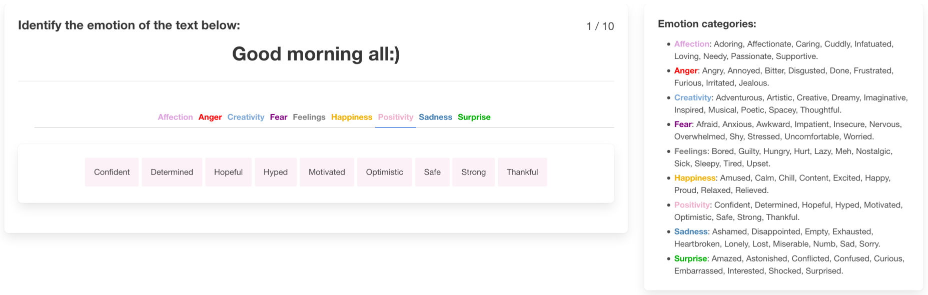
The box at the right of the screen shows at any time the list of **emotion categories** and associated **emotions**.

If multiple emotions are possible, select the one that **in your opinion fits the most**.

After every choice, you will be told if your choice **correctly matches the author's** to better understand the task. Press **Escape (esc)** to hide the notification.

You must identify **1** emotions or **2** categories correctly to **rule out random guessing**. Since the task is ambiguous, we will approve all your submissions if **75%** of your HITs meet the criteria.

Figure A.7: Instructions shown to the MTurk workers.



Identify the emotion of the text below:

1 / 10

Good morning all:)

Emotion categories:

- **Affection**: Adoring, Affectionate, Caring, Cuddly, Infatuated, Loving, Needy, Passionate, Supportive.
- **Anger**: Angry, Annoyed, Bitter, Disgusted, Done, Frustrated, Furious, Irritated, Jealous.
- **Creativity**: Adventurous, Artistic, Creative, Dreamy, Imaginative, Inspired, Musical, Poetic, Spacey, Thoughtful.
- **Fear**: Afraid, Anxious, Awkward, Impatient, Insecure, Nervous, Overwhelmed, Shy, Stressed, Uncomfortable, Worried.
- **Feelings**: Bored, Guilty, Hungry, Hurt, Lazy, Meh, Nostalgic, Sick, Sleepy, Tired, Upset.
- **Happiness**: Amused, Calm, Chill, Content, Excited, Happy, Proud, Relaxed, Relieved.
- **Positivity**: Confident, Determined, Hopeful, Hyped, Motivated, Optimistic, Safe, Strong, Thankful.
- **Sadness**: Ashamed, Disappointed, Empty, Exhausted, Heartbroken, Lonely, Lost, Miserable, Numb, Sad, Sorry.
- **Surprise**: Amazed, Astonished, Conflicted, Confused, Curious, Embarrassed, Interested, Shocked, Surprised.

Affection Anger Creativity Fear Feelings Happiness Positivity Sadness Surprise
Confident
Determined
Hopeful
Hyped
Motivated
Optimistic
Safe
Strong
Thankful

Figure A.8: Screenshot of the MTurk Annotation Tool. Workers must first select an emotion category, then choose the emotion within that category they find more appropriate.

ity criteria in 75% of their work. By the end of our 264 HIT batch, we approved 97.33% of the submissions sent by 84 different readers with an average emotion accuracy of 11.43% and an average category accuracy of 34.26%. The distribution of accuracy scores is shown in Figure A.9.

us about the ambiguity and meaninglessness of the emotions provided by Vent users and the impact of rejections in their future earnings on the platform.

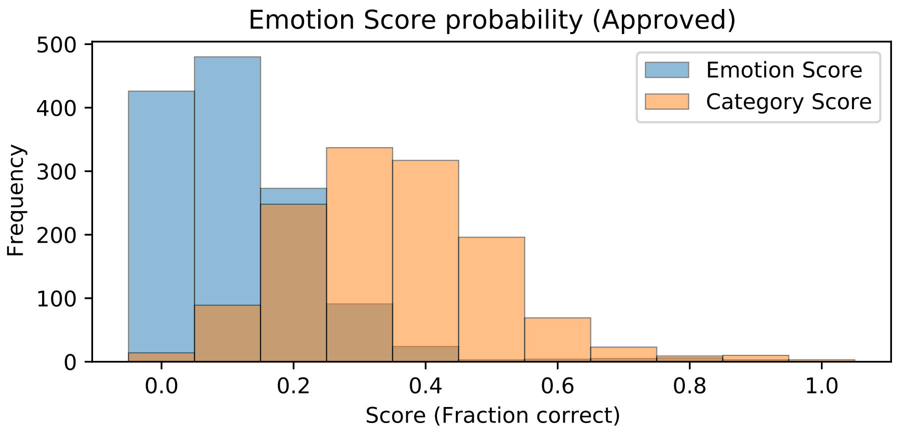


Figure A.9: Emotion and category accuracy scores for the readers, computed as the accuracy of the reader against the author-provided label on every task.

Inter-annotator Agreement

To validate that the emotion labels in Vent contain meaningful emotional information, we analyze the agreement between annotators (readers). The set of readers performing the annotations varies across snippets, so we focus on the number of readers that assign the same emotion and emotion category to any given snippet. For a given snippet, we consider the reader overlap as the total number of readers that agree with the most frequently assigned label. Since every snippet is shown to 5 different readers, the maximum level of agreement is 5, meaning that all readers agreed on the same judgement, while an agreement of 1 means that each reader chose a different label. The frequency of the overlaps observed across all snippets are shown in Figure A.10. At the emotion level (blue), we find an average agreement of 1.81 ($\pm$ 0.88) readers while at the category level (orange) we find that 2.95 ($\pm$ 1.03) readers agree on average with the most frequent label. These results show that the majority of annotators agree on the category labels, but specific emotions are hard to pin down.

Human vs. Model Performance

We provide detailed results at the category level of the best performing BERT + Bi-LSTM model and readers in Table A.5 and Table A.6 respectively. As highlighted in the chapter, we find that the precision of the model is lower across all categories while readers show consistently higher recall. On Affection and Fear, readers and the model achieve the same F1-score (0.44 and 0.42 respectively) while otherwise the model outperforms readers in all emotion categories except for Creativity and Surprise, which are the worst-performing categories for the model.

We also analyze the confusion matrices between readers and the model (constructed as per the main text). Figure A.11 shows that the model more closely predicts emotions given a particular author-provided label than the MTurk workers (the diagonal vector is more clearly defined). However, we observe that the confusion between negative (Anger, Fear, Feelings, and Sadness) and positive (Affection, Surprise, Creativity, Happiness, and

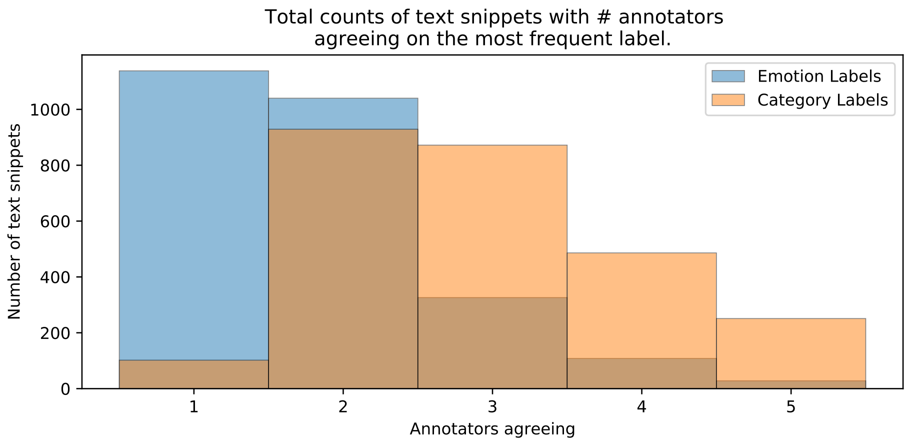


Figure A.10: Inter-annotator agreement at the emotion and category levels for the most frequent label. Readers rarely show total disagreement at the category level (average agreement 2.95 ± 1.03 out of 5), but struggle to exactly pin down the same specific emotions (average 1.81 ± 0.88 out of 5).

Emotion	**Prec**	**Rec**	**F1**	**Sup**
Affection	**0.47**	0.41	0.44	270
Anger	0.44	0.50	**0.46**	270
Creativity	*0.26*	0.50	*0.34*	300
Fear	0.45	0.40	0.42	330
Feelings	0.28	0.51	0.36	330
Happiness	0.29	0.57	0.38	270
Positivity	0.35	0.38	0.37	270
Sadness	0.35	**0.62**	0.44	330
Surprise	0.34	*0.34*	*0.34*	270

Table A.5: Category prediction results for our model, in terms of **Prec**ision, **Rec**all, **F1**-score and **Sup**port. In **bold**, the best performing category for each metric; in *cursive*, the worst performing category.

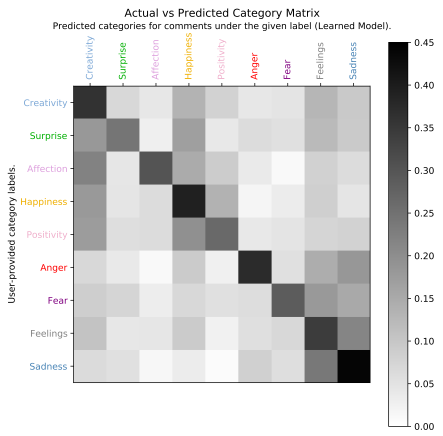
Emotion	**Prec**	**Rec**	**F1**	**Sup**
Affection	**0.31**	0.79	**0.44**	270
Anger	0.27	0.80	0.41	270
Creativity	*0.23*	*0.66*	*0.35*	300
Fear	0.30	*0.66*	0.42	330
Feelings	*0.23*	0.70	0.35	330
Happiness	0.25	0.69	0.37	270
Positivity	*0.23*	0.70	*0.34*	270
Sadness	0.27	**0.85**	0.41	330
Surprise	0.24	0.68	0.36	270

Table A.6: Category prediction results for the MTurk workers, in terms of **Prec**ision, **Rec**all, **F1**-score and **Sup**port. In **bold**, the best performing category for each metric; in *cursive*, the worst performing category.

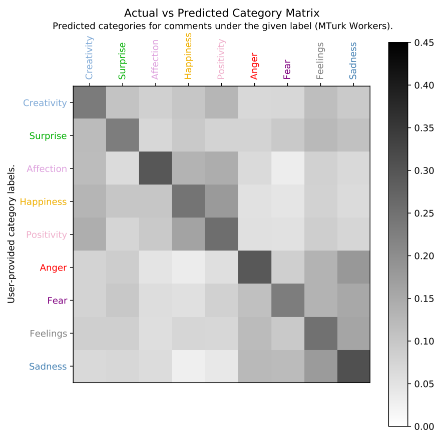
Positivity) categories is more apparent for workers.

To further understand the differences between readers and the model, we compute deltas between confusion matrices in Figure A.12. For an expected i-th emotion (row), red cells show when readers are more likely to predict the j-th emotion (column) than the model, while blue cells show the opposite case with the model having higher likelihood than readers.

To ensure the results are significant, we compare z-tested differences be-



(a) Confusion Matrix for the BERT + Bi-LSTM model.



(b) Confusion Matrix for the MTurk Workers.

Figure A.11: Comparison between the confusion matrices on Vent categories computed on the submitted MTurk batch.

tween model and readers over 10,000 bootstrapping runs with the sample size of the evaluation data set (2640 snippets).
Figure A.12 shows that the model is more likely to output Creativity or Happiness, while workers are more likely to predict Surprise, Affection and Positivity. We also find blue colors along the diagonals, which agrees with our results on the model having higher precision than the readers.

Worker Compensation Analysis

Finally, we discuss the annotator reward of our task, which we defined to ensure that MTurk workers receive fair compensation of at least $7.25, which is the minimum federal wage in the United States. Upon designing our task and annotation tool, we benchmark ourselves on a sample of 10 HITs using the Requester and Worker Sandbox. In doing so, we identify a minimum amount of time to show feedback after each example (4 seconds), and estimate that every task takes around 90 seconds.
With our first internal estimate, we sfchedule a test batch with 10% of the data and a reward of $0.20 per task, which we expect to translate

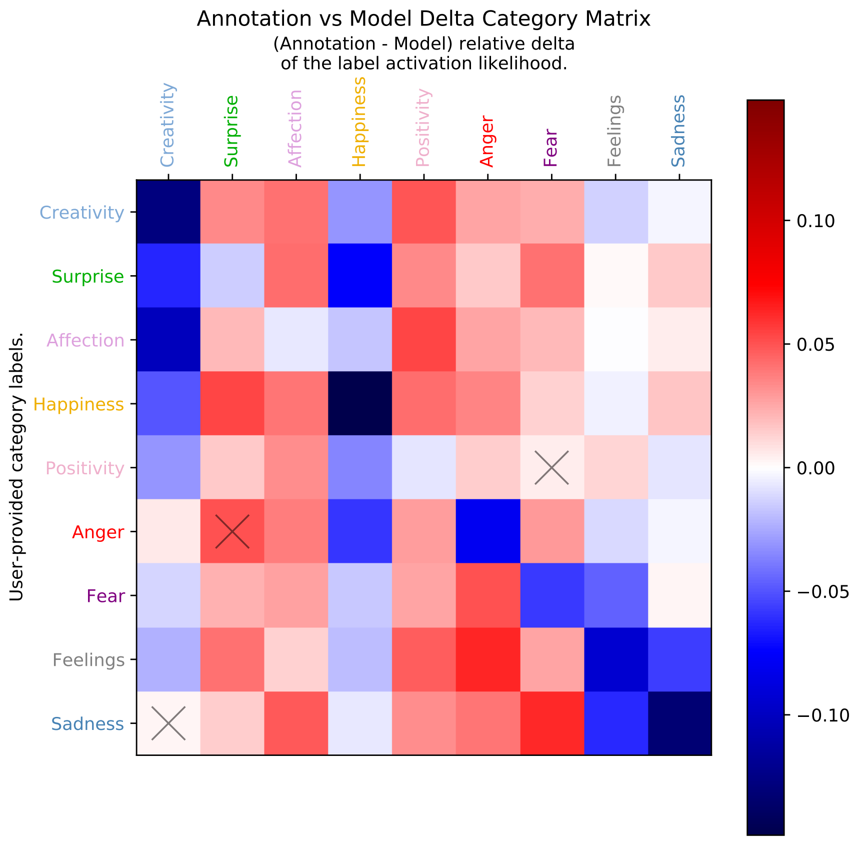


Figure A.12: Difference between Model and MTurk worker confusion matrices. Crossed-out cells indicate non-significant differences computed by running bootstrapped simulations on the annotations from the workers ($p = 0.001$).

into \$8 per hour. Our test batch helped us identify serious problems with our annotation tool, which allowed workers to submit incomplete tasks. Our acceptance rules flagged their submissions as invalid, which meant that workers could potentially get several rejections. Workers on MTurk depend on a high approval rate to access tasks, so they contacted us and promptly addressed the limitation.

Additionally, using data from this batch and input from workers, we observed that the task took longer to complete without prior context, and thus our reward was not enough to meet our goal. Given a median amount of time per task of around 130 seconds, we conservatively raise the reward per HIT to \$0.32. This brings our expected payout to \$8.86 per hour, which we use to submit our full batch.

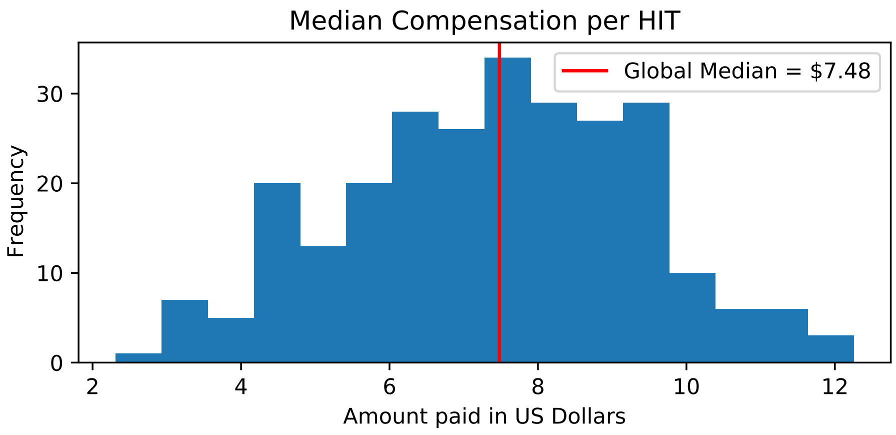


Figure A.13: Compensation per HIT. The median compensation per HIT is \$7.48, which means that we met our target of providing the US Federal minimum wage for workers engaging in our task.

In Figure A.13, we show the distribution of compensations per HIT and the overall median compensation per HIT (red vertical line), which is equivalent to \$7.48 per hour. We notice a large variability in the results, and observe that the final reward per task is lower than expected. This might be caused by the fact that despite text snippets in a HIT being chosen at random, some of the specific HITs remain more challenging than others. Additionally, it might be the case that some workers are more capable of performing the task than others, or that can trade off annotation quality for speed without bypassing our quality checks. Our results show that we achieve our goal of fairly compensating workers per HIT above minimum wage.

A.4 *Emotion-Core*: An Open Source Framework for Emotion Detection Research

We describe *Emotion-Core*, an Open-Source framework for training, evaluating, and showcasing textual Emotion Detection models. Our framework is composed of two components: *Emotion Classification* and *EmotionUI*, which allow researchers to easily extend and reuse existing emotion detection solutions. Our code is available and free to use for interested researchers.

A.4.1 High-Level Functionality and Purpose

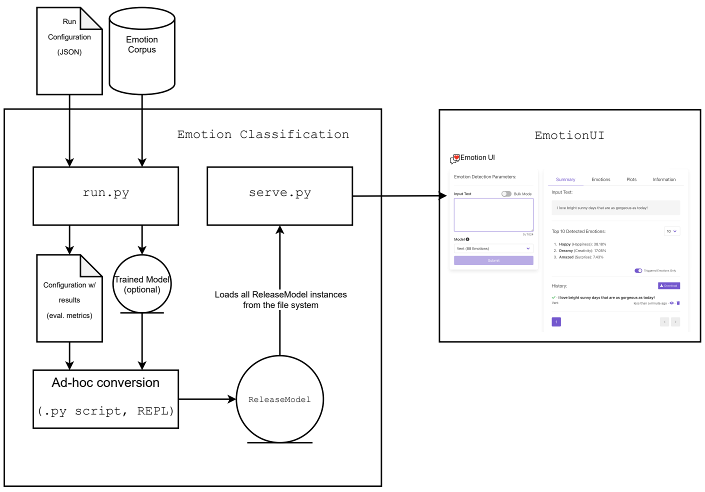
Given a text snippet, what are the possible emotions present in it, and how strongly are they expressed? Currently, there is no Open-Source framework to answer these questions, providing pre-trained models using large-scale data sets while being easily usable and extensible. In this contribution, we provide *Emotion-Core*, a collection of Open-Source tools to train, evaluate, deploy and showcase emotion detection models using a variety of datasets, modelling techniques, and evaluation approaches. It is composed of two separate sub-projects: *Emotion Classification* and *EmotionUI*, shown in Figure A.14.

— *Emotion Classification*: Modelling and Deployment Back-End

Emotion Classification is a Python project containing software tools to train, evaluate and deploy machine learning models for emotion detection. The functionality is split among two main programs:

- `run.py`: a single-argument program that executes an experiment configuration (declaring a deterministic 'run') from a JSON file, containing all the parameters to define how to perform model-training, evaluation and serialization.
- `serve.py`: a web-server that exposes release-ready research models through a REST API, so that emotion detection models can be explored using a simplified web interface such as *EmotionUI.*

Run configurations determine how to produce and evaluate an emotion detection model, with four categories of parameters:



Input Text:

I love bright sunny days that are as gorgeous as today!

Top 10 Detected Emotions:

1. **Happy** (Happiness): 38.18%
2. **Dreamy** (Creativity): 17.05%
3. **Amazed** (Surprise): 7.43%

History:

✓ · **I love bright sunny days that are as gorgeous as today!**
Vent
less than a minute ago

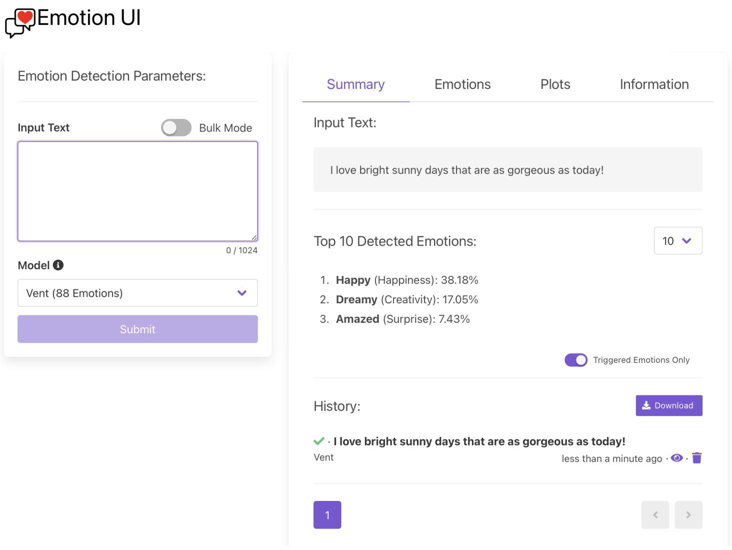
Figure A.14: *Emotion-Core* component flow chart. The diagram shows the inputs and outputs of the various systems in *EmotionUI*, from the run configuration and input data in `run.py`, the internal processing and output metrics, the preparation of a `ReleaseModel` instance for `serve.py`, and the common usage of *EmotionUI*.

1. **Data Loading.** Initial random seeds, split sizes for training, validation and test sets, or the name of a data column within the data set defining the different splits.

2. **Text Representations.** Define how to encode text — currently supports statistical document representations (Bag-of-Words, TF-IDF), word embeddings (FastText), or Transformer LMs (BERT).

3. **Modeling Approaches.** Declare the (streaming) modelling algorithm and hyper-parameters, with dependencies on the text representation. Includes support for traditional machine learning methods (Logistic Regression, Naïve Bayes, Random Forests) and Deep Learning approaches (Deep Neural Networks and Bi-LSTMs).

4. **Outputs and Serialization.** Specify file system paths to store model evaluation results, tuned classifier thresholds, and the trained model for deployment.


Emotion UI

Emotion Detection Parameters:

Input Text — Bulk Mode

0 / 1024

Model: Vent (88 Emotions)

Submit

Summary | Emotions | Plots | Information

Input Text:

I love bright sunny days that are as gorgeous as today!

Top 10 Detected Emotions: 10

1. **Happy** (Happiness): 38.18%
2. **Dreamy** (Creativity): 17.05%
3. **Amazed** (Surprise): 7.43%

Triggered Emotions Only

History: Download

✓ · **I love bright sunny days that are as gorgeous as today!**
Vent — less than a minute ago

1

Figure A.15: Screenshot of the public *EmotionUI*. Users can submit text to be classified using a model trained on different data sets, using a variety of document representations and modelling choices. The public release only exposes the best performing models on each data set.

— *EmotionUI*: Showcase and Visualization Interface

EmotionUI is a JavaScript project containing a single-page web application to showcase and explore emotion detection models trained with the *Emotion Classification* programs. It provides a user interface to submit text snippets both individually and in bulk, with tools to visualize the distribution of predicted emotions and emotion categories if available. The public interface is shown in Figure A.15.

A.4.2 Research Impact

Emotion-Core allows researchers to rapidly experiment with emotion detection data sets, document representations, and modelling approaches in a fully reproducible way. The platform was designed as a means to benchmark emotion detection models on two of the largest emotion classification corpora: GoEmotions (Demszky et al. 2020) and Vent (Lykousas et al. 2019). Currently, *Emotion-Core* only supports streaming multi-label modelling approaches, with an automatic evaluation module to ensure fair comparison across all methods. Within the *Emotion-Core* framework, modellers do not need to reimplement models for each data set. Instead, the experimental methods can be extended by providing data or new implementations matching simple interfaces:

- **Data formats.** Emotion data sets in *Emotion-Core* only require two columns: `text` and `emotion_index`. An experimenter can use a new data set by preprocessing the columns accordingly. Minimal support for managing data set splitting, as well as using already provided splits, is also provided.

- **Document Representation and Models.** Representation and model factories allows for centralized definitions of how to represent a document and train a model from it. Experimenters can redefine existing methods or use new ones by creating new `Extractor` and `Model` classes and registering them in their corresponding factories.

A.4.3 Limitations and Extended Applications

Emotion Classification currently provides users with ready-made recipes to train, evaluate and serialize models. However, deploying models is currently a manual endeavor, requiring the user to instantiate and serialize a `ReleaseModel` instance through an ad-hoc process (e.g. using an interactive Python REPL or a custom script). `ReleaseModel` instances can then be serialized to disk and automatically served by `serve.py` without additional work. As an open-source project, it is possible to add automated `ReleaseModel` support, saving users from doing manual exports combining the serialized results dictionary with the serialized model.
EmotionUI provides a web interface to evaluate and explore `ReleaseModel` instances served through the `serve.py` REST API. Currently, the host name of the REST API is not configurable in real-time, being set to either a local endpoint during debugging mode or a remote production environment in the public release. This limits the ease for experimenters to evaluate multiple versions at once using different environments.

Appendix B

Appendix: IGEL

In this appendix, we extend the content of Chapter 5 to provide additional information about the experimental setting in Section B.1. In Section B.2 and Section B.3, we respectively outline possible extensions to IGEL and describe the self-supervised setting used for link prediction experiments from Subsection 5.4.7.

B.1 Additional Experimental Details and Results

In this section we describe dataset details, hyper-parameters, and experiment details, and provide a pseudo-code of the Breadth-First Search implementation of IGEL.

B.1.1 Dataset Details.

We summarise all graph data sets used in the experimental validation of IGEL in Table B.1. To provide an overview of each data set and its usage, we indicate the average cardinality of the graphs in the data set (Avg. n), the average number of edges per graph (Avg. m), and the total number of graphs per dataset. We describe the task, loosely grouped in graph classification, non-isomorphism detection, graph regression, link prediction and vertex classification. We also provide the shape of the output for every problem, and describe the split regime for training / validation / test sets when relevant.

Table B.1: Overview of the graphs used in the experiments. We show the average number of vertices (Avg. n), edges (Avg. m), number of graphs, target task, output shape, and splits (when applicable).

	Avg. n	**Avg. m**	**Num. Graphs**	**Task**	**Output Shape**	**Splits (Train / Valid / Test)**
Enzymes	32.63	62.14	600	Multi-class Graph Class.	6 (multi-class probabilities)	9-fold / 1 fold (Graphs, Train / Eval)
Mutag	17.93	39.58	188	Binary Graph Class.	2 (binary class probabilities)	9-fold / 1 fold (Graphs, Train / Eval)
Proteins	39.06	72.82	1113	Binary Graph Class.	2 (binary class probabilities)	9-fold / 1 fold (Graphs, Train / Eval)
PTC	25.55	51.92	344	Binary Graph Class.	2 (binary class probabilities)	9-fold / 1 fold (Graphs, Train / Eval)
Graph8c	8.0	28.82	11117	Non-isomorphism Detection	N/A	N/A
EXP Classify	44.44	111.21	600	Binary Class. (pairwise graph distinguishability)	1 (non-isomorphic graph pair probability)	Graph pairs 400 / 100 / 100
SR25	25	300	15	Non-isomorphism Detection	N/A	N/A
RandomGraph	18.8	62.67	5000	Regression (Graphlet Counting)	1 (graphlet counts)	Graphs 1500 / 1000 / 2500
ArXiv ASTRO-PH	18722	198110	1	Binary Class. (Link Prediction)	1 (edge probability)	Randomly sampled edges 50% train / 50% test
Facebook	4039	88234	1	Binary Class. (Link Prediction)	1 (edge probability)	Randomly sampled edges 50% train / 50% test
PPI	2373	68342.4	24	Multi-label Vertex Class.	121 (binary class probabilities)	Graphs 20 / 2 / 2

B.1.2 Hyper-parameters and Experiment Details

Graph Level Experiments

We reproduce the benchmark of (Balcilar et al. 2021) without modifying model hyper-parameters for the tasks of Graph Classification, Graph Isomorphism Detection, and Graphlet Counting. For classification tasks, the 6 models in Table 5.2 are trained on binary / categorical cross-entropy objectives depending on the task. For Graph Isomorphism Detection, we train GNNs as binary classification models on the binary classification task on EXP (Abboud et al. 2021), and identify isomorphisms by counting the number of graph pairs for which randomly initialized MP-GNN models produce equivalent outputs on Graph8c[1][2]. For the graphlet counting regression task on the RandomGraph data set (Chen et al. 2020), we train models to minimize Mean Squared Error (MSE) on the normalized graphlet counts for five types of graphlets.

On all tasks, we experiment with $k \in \{1, 2\}$ and optionally introduce a preliminary linear transformation layer to reduce the dimensionality of Igel encodings. For every setup, we execute the same configuration 10 times with different seeds and compare runs introducing Igel or not by measuring whether differences on the target metric (e.g. accuracy or MSE) are statistically significant as shown in Table 5.1 and Table 5.2. In Table B.2, we provide the value of k that was used in our experimental results. Our results show that the choice of k depends on both the task and model type. We believe these results may be applicable to subgraph-based MP-GNNs, and will explore how different settings, graph sizes, and downstream models interact with k in future work.

Reproducibility– We provide an additional repository with our changes to the original benchmark, including our modelling scripts, metadata, and experimental results[3].

Vertex and Edge-level Experiments In this section we break down the best performing hyper-parameters on the Edge (link prediction) and Vertex-level (node classification) experiments.

[1]Simple 8 vertices graphs from: `http://users.cecs.anu.edu.au/~bdm/data/graphs.html`

[2]That is, models are not trained but simply initialized, following the approach of (Balcilar et al. 2021).

[3]`https://github.com/nur-ag/gnn-matlang`

Table B.2: Values of k used when introducing IGEL in the best reported configuration for graphlet counting and graph classification tasks. The table is broken down by graphlet types (upper section) and graph classification tasks on the TU Datasets (bottom section).

	Chebnet	GAT	GCN	GIN	GNNML3	Linear	MLP
Star	2	1	2	1	1	2	1
Tailed Triangle	1	1	1	1	2	1	1
Triangle	1	1	1	1	1	1	1
4-Cycle	2	1	1	1	1	1	1
Custom Graphlet	2	1	1	1	2	2	2
Enzymes	1	2	2	1	2	2	2
Mutag	1	1	1	1	1	1	2
Proteins	2	2	2	1	2	1	1
PTC	1	1	2	1	1	2	2

Link Prediction– The best configuration on the Facebook graph uses $k = 2$, learning $t = 256$ component vectors with $e = 10$ walks per node, each of length $s = 150$ and $p = 8$ negative samples per positive for the self-supervised negative sampling. On the arXiv citation graph, we find the best configuration with $k = 2$, $t = 256$, $e = 2$, $s = 100$ and $p = 9$.

Node Classification– In the node classification experiment, we analyze both encoding distances $k \in \{1, 2\}$. Other IGEL hyper-parameters are fixed after a small greedy search based on the best configurations in the link prediction experiments. For the MLP model, we perform greedy architecture search, including number of hidden units, activation functions and depth. Our results show scores averaged over five different seeded runs with the same configuration obtained from hyperparameter search.

The best performing hyperparameter configuration on the node classification is found with $k = 2$ on $t = 256$ length embedding vectors, concatenated with node features as the input layer for 1000 epochs in a 3-layer MLP using ELU activations with a learning rate of 0.005. Additionally, we apply 100 epoch patience for early stopping, monitoring the F1-score on the validation set.

Reproducibility– We provide a replication folder in the code repository for the exact configurations used to run the link prediction (edge-level) and node classification experiments[4].

[4]`https://github.com/nur-ag/IGEL`

B.1.3 IGEL through Breadth-First Search

The idea behind the IGEL encoding is to represent each vertex v by compactly encoding its corresponding ego-network $\mathcal{S}_v^k$ at depth k.

Algorithm B.1 IGEL Encoding (BFS).

```
Input: $v \in V, k \in \mathbb{N}$
 1: toVisit := [ ]                                   ▷ Queue of nodes to visit.
 2: degrees := { }                        ▷ Mapping of nodes to their degrees.
 3: distances := $\{v : 0\}$          ▷ Mapping of nodes to their distance to $v$
 4: while toVisit $\neq \emptyset$ do
 5:     $u$ := toVisit.dequeue()
 6:     currentDistance := distances[$u$]
 7:     currentDegree := 0
 8:
 9:     for $w \in u$.neighbors() do
10:         if $w \notin$ distances then
11:             distances[$w$] := currentDistance + 1
                                      ▷ $w$ is a new node 1-hop further from $v$.
12:         end if
13:         if distances[$w$] $\leq k$ then
14:             currentDegree := currentDegree + 1
                                             ▷ Count edges only within $k$-hops.
15:             if $w \notin$ degrees then   ▷ Enqueue if $w$ has not been visited.
16:                 toVisit.append($w$)
17:             end if
18:         end if
19:     end for
20:     degrees[$u$] := currentDegree
                    ▷ $u$ is now visited: we know its degree and distance to $v$.
21: end while
22: $e_v^k = \{\!\{$(distances[$u$], degrees[$u$]) $\forall\, u \in$ degrees.keys()$\}\!\}$
                                                       ▷ For each visited node:
23:                        ▷ Produce the multi-set of (distance, degree) pairs.
Output: $e_v^k : (\mathbb{N}, \mathbb{N}) \rightarrow \mathbb{N}$
```

The choice of encoding consists of a histogram of vertex degrees at distance $d \leq k$, for each vertex in $\mathcal{S}_v^k$. Essentially, IGEL runs a Breadth-First Traversal up to depth k, counting the number of times the same degree appears at distance $d \leq k$. The algorithm shown in Algorithm 5.1 showcases IGEL and its relationship to the 1-WL test. However, in a practical setting, it might be preferable to implement IGEL through Breadth-First Search (BFS). In Algorithm B.1, we show one such implementation that fits the time and space complexity described in Section 5.2.
Due to how we structure BFS to count degrees and distances in a single pass, each edge is processed twice—once for each node at end of the edge. It must be noted that when processing every $v \in V$, the time complexity is $\mathcal{O}(n \cdot \min(m, (d_{\mathtt{max}})^k))$. However, the BFS implementation is also embarrassingly parallel, which means that it can be distributed over p processors with $\mathcal{O}(n \cdot \min(m, (d_{\mathtt{max}})^k)/p)$ time complexity.

B.2 IGEL Beyond Unattributed Graphs.

In this appendix, we explore extensions to IGEL beyond unattributed graphs. In particular, we describe how minor modifications would allow IGEL to leverage label and attribute information, connecting IGEL with state-of-the-art MP-GNN representations.

B.2.1 Application to labelled graphs.

Labelled graphs are defined given $G = (V, E, \mathcal{L})$, where $\mathcal{L}_G(v) : V \mapsto \mathcal{L}$ is a mapping function assigning a label $L \in \mathcal{L}$ to each vertex. A standard way of applying the 1-WL test to labelled graphs is to replace $d_G(v)$ in the first step of Algorithm 4.1 with $\mathcal{L}_G(v)$, such that the initial coloring is given by node labels. Since IGEL retains a similar structure, the same modification can be introduced in Equation 5.1. Some applications might require both degree and label information, achievable through a mapping $C_G(v) : V \mapsto \mathbb{N}$ that combines $\mathcal{L}_G(v)$ and $d_G(v)$ given a label indexing function $I : \mathcal{L} \mapsto \{0, ..., |\mathcal{L}| - 1\}$:

$$C_G(v) = d_G(v) \cdot |\mathcal{L}| + I(\mathcal{L}_G(v))$$

Applying $\mathcal{L}_G(v)$ or $C_G(v)$ only requires adjusting the shape of the vector representation, whose cardinality will depend on the size of the label set (e.g. $|\mathcal{L}|$), or the combination of degrees and $\mathcal{L}$ given by $d_{\mathtt{max}} \times |\mathcal{L}|$.

B.2.2 Application to attributed graphs.

IGEL can also be applied to attributed graphs of the form $G = (V, E, \mathbf{X})$, where $\mathbf{X} \in \mathbb{R}^{n \times w}$. Following Algorithm 5.1, IGEL relies on two discrete functions to represent structural features—the path-length between vertices $l_G(u, v)$ and vertex degrees $d_G(v)$. However, in an attributed graph, it may be desirable to consider the attributes of a node besides degree to represent it with respect to its attributes and the attributes of its neighbours. To extend the representation to node attributes, $d_G(v)$ may be replaced by a bucketed similarity function $\phi(u, v) : (V \times V) \mapsto B$ applicable for v and any $u \in \mathcal{V}_v^k$, whose output is discretized into $b \in B \subseteq \mathbb{N}$ buckets $1 \leq b \leq |B|$. A straightforward implementation of $\phi_{\mathtt{cos}}$ is to compute the cosine similarity $(|\cdot|)$ between attribute vectors, and remaps the $[-1, 1]$ interval to discrete buckets by rounding:

$$\phi_{\mathtt{cos}}(u, v) : \left\lfloor \Big(|B| - 1 \Big) \cdot \frac{(\mathbf{X}_u | \cdot | \mathbf{X}_v) + 1}{2} \right\rceil$$

ϕ-IGEL is a generalization over the structural representation function in Algorithm 5.1 to take also source vertex v. Thus, unattributed and unlabelled IGEL can be understood as the case in which $\phi(u, v) = d_G(u)$, such that $|B| = d_{\mathtt{max}}$.
Furthermore, by introducing the bucketing transformation in ϕ-IGEL, we are de-facto providing a simple mechanism to control the size of IGEL encoding vectors. By implementing $\phi(u, v) = \lfloor d_G(u)/b \rfloor$ or introducing non-linear transformations such as $\phi(u, v) = \lfloor \log(d_G(u))/b \rfloor$ with $b \in \{1, ..., d_{\mathtt{max}}\}$, it is possible to compress $\text{IGEL}^k_{\mathtt{vec}}(v)$ into the $t = (k + 1) \times b$ denser components of a t-dimensional vector.

B.2.3 IGEL and Sub-graph MP-GNNs.

Another natural extension is to apply the distance-based aggregation schema to message passing GNNs. This can be achieved by expressing message passing in terms not just of immediate neighbours of v, but from nodes at every distance $1 \leq l \leq k$. Equation 4.1 then becomes:

$$\mu_v^{i,l} = \left\{\!\!\left\{ \textsc{Msg}_G^{i,l}(h_u^{i-1}) \mid \forall\, u \in \mathcal{V}_v^k \;\wedge\; l_G(u,v) = l \right\}\!\!\right\}$$

The update step will need to pool over the messages passed onto v:

$$\mathbf{x}_v^i = \textsc{Update}_G^i\left(\mu_v^{i,0}, \mu_v^{i,1}, ..., \mu_v^{i,k}, \mathbf{x}_v^{i-1}\right)$$

This formulation matches the general form of k-hop GNNs (Nikolentzos, Dasoulas, and Vazirgiannis 2020) presented in Subsection 4.3.3. Furthermore, introducing distance and degree signals in the message passing can yield models analogous to GNN-AK (Zhao et al. 2022)—which explicitly embed the distance of a neighbouring node when representing the ego-network root during message passing. As such, Igel directly connects the expressivity of k-hop GNNs with the 1-WL algorithm, and provides a formal framework to explore the expressivity of higher order MP-GNN architectures.

B.3 Self-supervised IGEL

We provide additional context for the application of Igel as a representational input for methods such as DeepWalk (Perozzi, Al-Rfou, and Skiena 2014). This section describes in detail how self-supervised Igel embeddings are learned. We also provide a qualitative analysis of Igel in the self-supervised setting using networks that are amenable for visualization. We focus on analyzing how k influences the learned representations.

B.3.1 IGEL & Self-supervised Node Representations.

Igel can be easily incorporated in standard node representation methods like DeepWalk (Perozzi, Al-Rfou, and Skiena 2014) or node2vec (Grover and Leskovec 2016). Due to its relative simplicity, integrating Igel only requires to replace the input to the embedding method so that Igel encodings be used rather than node identities. We provide an overview of

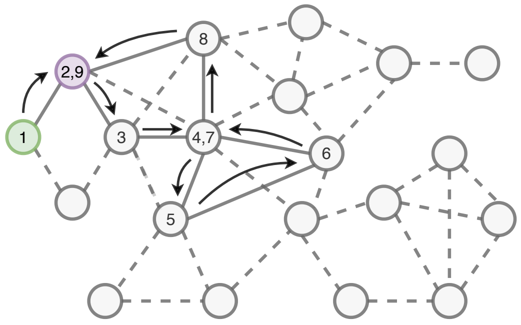


Figure B.1: Example of random walk starting on the green node and finishing on the purple node. Nodes contain the time-step when they were visited. The context of a given vertex will be any nodes whose time-steps are within the context window of that node.

the process of generating embeddings through DeepWalk, which involves (a) sampling random walks to capture the relationships between nodes, (b) training a negative-sampling based embedding model in the style of word2vec (Mikolov et al. 2013) on the random walks to embed random walk information in a compact latent space.

Distributional Sampling through Random Walks— First, we sample π random walks from each vertex in the graph. Walks are generated by selecting the next vertex uniformly from the neighbours of the current vertex. Figure B.1 illustrates a possible random walk of length 9 in the graph of Figure 5.1. By randomly sampling walks, we obtain traversed node sequences to use as inputs for the following negative sampling optimization objective.

Negative Sampling Optimization Objective— Given a random walk ω, defined as a sequence of nodes of length s, we define the context $\mathcal{C}(v, \omega)$ associated to the occurrence of vertex v in ω as the the sub-sequence in ω containing the nodes that appear close to v, including repetitions. Closeness is determined by p, the size of the positive context window, i.e., the context contains all nodes that appear at most p steps before/after the node within ω. In DeepWalk, a skip-gram objective learns to represent vertices appearing in similar contexts within random walks.

Given a node $v_c \in V$ in random walk ω, our task is to learn embeddings that assign high probability for nodes $v_o \in V$ appearing in the context

Table B.3: DeepWalk hyper-parameters.

Param.	**Description**
$\pi \in \mathbb{N}$	Random walks per node
$s \in \mathbb{N}$	Steps per Random Walk
$z \in \mathbb{N}$	# of neg. samples
$p \in \mathbb{N}$	Context window size

$\mathcal{C}(v_c, \omega)$, and lower probability for nodes not appearing in the context. Let $\sigma(\cdot)$ denote the logistic function. As we focus on the learned representation capturing these symmetric relationships, the probability of v_o being in $\mathcal{C}(v_c, \omega)$ is given by:

$$p\left(v_o \in \mathcal{C}(v_c, \omega)\right) = \sigma(\mathbf{e}_{v_c}^{\top} \cdot \mathbf{e}_{v_o}),$$

Table B.3 summarizes the hyper-parameters of DeepWalk. Our global objective function is the following negative sampling log-likelihood. For each of the π random walks and each vertex v_c at the center of a context in the random walk, we sum the term corresponding to the positive cases for vertices found in the context, and the expectation over z negative, randomly sampled, vertices.

Let $P_n(V)$ be a noise distribution from which the z negative samples are drawn, our task is to maximise (B.1) through gradient ascent:

$$\begin{aligned} l_{\mathrm{u}}(\mathbf{W}) = \sum_{j=1}^{w} \sum_{\substack{v_c \in \omega_j, \\ n_o \in \mathcal{C}(v_c, \omega_j)}} \Bigg[& \log \sigma(\mathbf{e}_{v_c}^{\top} \cdot \mathbf{e}_{n_o}) + \\ & \sum_{i=1}^{z} \mathbb{E}_{v_i \sim P_n(V)} \Big[\log \sigma(-\mathbf{e}_{v_c}^{\top} \cdot \mathbf{e}_{v_i}) \Big] \Bigg]. \end{aligned} \tag{B.1}$$

Defining the IGEL-*DeepWalk embedding function*— In DeepWalk, for every vertex $v \in V$, there is a corresponding t-dimensional embedding vector $e_v \in \mathbb{R}^t$. As such, one can represent the embedding function as a product between a one-hot encoded vector corresponding to the index of the vertex $\texttt{one-hot}_v \in \mathbb{B}^n$ where and an embedding matrix $\mathbf{E}_V \in \mathbb{R}^{n \times t}$:

$$\mathbf{e}_v = \texttt{one-hot}_v \cdot \mathbf{E}_V$$

Introducing IGEL requires modifying the shape of E_V to account for the shape of t_{vec}-dimensional $\text{IGEL}^k_{\texttt{vec}}(v)$ encoding vectors dependent on k and $d_{\texttt{max}}$, so that IGEL embeddings are computed as a weighted sum of embeddings corresponding to each (path length, degree) pair.
Let $\mathbf{E}^k_{\text{IGEL}} \in \mathbb{R}^{t_{vec} \times t_{emb}}$ define a structural embedding matrix with one embedding per (path length, degree) pair, a linear IGEL can be defined as:

$$\text{IGEL}^k_{\texttt{emb}}(v) = \text{IGEL}^k_{\texttt{vec}}(v) \cdot \mathbf{E}^k_{\text{IGEL}}$$

Since the definition of $\text{IGEL}^k_{\texttt{emb}}(v)$ is differentiable, it can be used as a drop-in replacement of $\mathbf{e}_v$ in Equation B.1.

Appendix C

Appendix: ELENE

In this appendix, we extend the text of Chapter 6. In Section C.1, we provide a pseudocode implementation of ELENE using Breadth First Search (BFS) that complements the algorithmic complexity analysis of Subsection 6.1.3. In Section C.2, we provide additional details about the experimental benchmark described in Section 6.4.

C.1 ELENE through BFS

This section showcases Algorithm C.1, a BFS implementation of the ELENE Encoding that spans edges until reaching the maximum encoding depth k given a node v in V. As noted in Subsection 6.1.3, this implementation can be trivially parallelized over p processors as the encoding of each $v \in V$ is independent of other nodes.

C.2 Benchmark Details

In this section, we provide an overview of the benchmark we execute when evaluating ELENE, including the variants of models we test, and descriptions of our code and compute environment. We summarize the datasets we use in Subsection C.2.1, provide additional experimental details in Subsection C.2.2 and include the best hyper-parameters found for our experiments in Subsection C.2.3.

Algorithm C.1 ELENE Node Encoding using BFS.

```
Input: $G = (V, E), v \in V, k : \mathbb{N}$
 1: distances := $\{v : 0\}$    ▷ Maps nodes to distances to $n$, $l_G(u, v)$.
 2: r_degrees := $\{v : (0, 0, 0)\}$    ▷ Maps of nodes to relative degrees, $d_{\mathcal{S}}^{(p)}(u|v)$.
 3: for (src, dst) in $G$.bfs_edges($n$, max_depth = $k$) do
 4:     if src $\notin$ distances then    ▷ NB: only one node $\notin$ distances.
 5:         dst, src := src, dst
 6:     end if
 7:     if dst $\notin$ distances then    ▷ dst is unknown, one-hop of src's distance.
 8:         distances[dst] := distances[src]
 9:     end if
10:     ▷ Compute distance deltas $\in \{-1, 0, 1\}$.
11:     dist_delta := distances[dst] − distances[src]
12:
13:     ▷ Access each relative degree counts.
14:     src_deg := r_degrees.get(src, [0, 0, 0])
15:     dst_deg := r_degrees.get(dst, [0, 0, 0])
16:
17:     ▷ Increment degree counts for each node and 'direction'.
18:     src_deg[dist_delta + 1]++    ▷ Map {-1, 0, 1} deltas to {0, 1, 2}.
19:     dst_deg[1 − dist_delta]++
20:
21:     ▷ Update relative degrees of src and dst.
22:     r_degrees[src] := src_deg
23:     r_degrees[dst] := dst_deg
24: end for
25:
26: ▷ For $u \in \mathcal{V}_v^k$, count Equation 6.1 quadruplets and return frequencies.
27: mapping := {}
28: for $u \in \mathcal{V}_v^k$ do
29:     quadruplet := $(l_G(u, v), d_{\mathcal{S}}^{(-1)}(u|v), d_G(u), d_{\mathcal{S}}^{(+1)}(u|v))$
30:     mapping[quadruplet]++
31: end for
Output: mapping
```

Benchmark Configuration. We build on top of the benchmark and implementation from (Zhao et al. 2022), introducing explicit ego-network attributes on top of their proposed evaluation framework for consistency.

All Elene results are reported by extending the node and edge attributes as input into a GIN (Xu et al. 2019) extended to support edge-level features when available (Hu et al. 2020b). In all experiments, we evaluate Elene-L on top of GINs with edge extensions (Hu et al. 2020b).

For all explicit ego-network attribute methods, we summarize the available hyper-parameters in Table C.1. For the implementation of Elene-L, we observed unstable training when using the sum pooling function during early stages of development. We found that training was stable using masked `Mean` pooling where the n node messages (or m for edge messages) in the ego-network sub-graph are averaged considering a binary mask for neighbours of the root node at a distance k or less. All our results are reported using `Mean` pooling, including our results on SR25, suggesting that this decision does not adversely impact the model expressivity expected from Section 6.3. The resulting implementation of Equation 6.4 is:

$$\mathbf{m}^t_{\texttt{out}}(v) = \Phi_{\texttt{out}}\left(\mathbf{x}^t_v \middle\| \sum_{u}^{\mathcal{V}^k_v} \frac{\mathbf{m}^t_{\texttt{nd}}(u|v)}{\texttt{size}(\mathcal{V}^k_v)} \middle\| \sum_{\langle u,w\rangle}^{\mathcal{E}^k_v} \frac{\mathbf{m}^t_{\texttt{ed}}(u,w|v)}{\texttt{size}(\mathcal{E}^k_v)}\right).$$

We use an analogous implementation of Equation 6.7. Finally, we describe the hyper-parameters implemented to control our models in Table C.1.

Tested Models. On the Expressivity tasks, ZINC and MolHIV, we evaluate all learnable variants (**ND** and **ED**), while on the remaining classification/regression benchmarks we only consider **(ND)** models due to reduced memory costs and limited computational bandwidth. Furthermore, in all Elene-L setups we test a reduced number of hyper-parameters due to computational constraints, unless specified otherwise, as follows: First, evaluate the differences of introducing Elene-L feature updates only before the first MP-GNN layer, or alternatively before each MP-GNN layer without weight sharing. Second, we evaluate different values of the maximum sub-graph distance to embed. We describe the hyper-parameters and modelling choices in detail in Subsection C.2.2.

Table C.1: Hyper-parameters controlling the behaviour of explicit ego-network attribute encodings. ELENE only relies on k, while ELENE-L has 5 additional configurable settings.

Parameter	**ELENE**	**ELENE-L**
Depth of Ego-Net (k)	{0, 1, 2}	{0, 1, 2, 3}
Embedding Type	Sparse	Dense, learned
Representation	Node-only	Edge-centric, Node-centric
Max. Encoded Degree	Set to $d_{\mathtt{max}}$ from the training dataset.	Set to $d_{\mathtt{max}}$ from the training dataset or 0 (ignore degree info).
Max. Encoded Distance	Equal to k	Set to k. Can be modified to control the sub-graph mean norm. factor.
Input Position	Before First layer.	Before First layer, before the first $\frac{L}{2}$ layers, before all L layers.

C.2.1 Dataset Details

We summarize the key aspects of the datasets we use to evaluate our proposed methods in Section 6.4. Table C.2 contains an overview of each benchmark and dataset, the objective being addressed, and high-level dataset statistics—namely number of graphs, and average number of nodes (n) and edges (m) per graph.

C.2.2 Detailed Experimental Summary

In this section, we summarize our experimental setup and training procedure, describing the hyper-parameters that we consider in each setting. For all the experiments described, ELENE encodings are evaluated by concatenating them with the node feature vectors and as part of the edge features when available, using the element-wise feature vector product following the same approach that IGEL used in Section 5.4. Unless otherwise stated, there is no additional hyper-parameter tuning.

Expressivity. See `expressivityDatasets.sh` for details.

—*EXP and SR25.* We evaluate ELENE on GIN and GIN-AK$^+$ for both data sets with $k \in \{0, 1, 2\}$. For ELENE-L, we evaluate all model variants for $k \in \{0, 1, 2\}$ with 8-dim embeddings for EXP and 32-dim embeddings for SR25. All models use $L = 4$ for EXP and $L = 2$ for SR25.
—*Counting Sub. and Graph Prop.* We evaluate ELENE on GIN and GIN-AK$^+$ for both data sets with $k \in \{0, 1, 2\}$. For ELENE-L, we evaluate all model variants for $k \in \{0, 1, 2\}$ with 16-dim embeddings. On the GraphProp dataset, we additionally try $k = 3$ after noticing expected positive results during early evaluation—as larger values of k enable the model to capture long-range dependencies. All models use $L = 3$ for Counting Sub. and $L = 6$ for Graph Prop.

Real World Graphs. See `benchmarkDatasets.sh` for details.

—*ZINC and MolHIV.* We evaluate ELENE on GIN and GIN-AK$^+$ with $k \in \{0, 1, 2\}$. For ELENE-L, we evaluate all variants for $k \in \{0, 1, 2, 3\}$ with 32-dim embeddings, using $L = 6$ on ZINC and $L = 2$ on MolHIV.

Table C.2: Dataset statistics.

Benchmark	Dataset	Objective	Tasks	Nr. of Graphs (Train / Valid / Test)	Avg. n	Avg. m
Expressivity	**EXP**	Distinguish 1-WL Equiv. graphs	2	1200	44.4	110.2
	SR25	Distinguish 3-WL Equiv. graphs	15	15	25	300
	CountingSub.	Count graph substructures	4	1500 / 1000 / 2500	18.8	62.6
	GraphProp.	Regress graph properties	3	5120 / 640 / 1280	19.5	101.1
Real World Graphs	**ZINC-12K**	Molecular prop. regression	1	10000 / 1000 / 1000	23.1	49.8
	CIFAR10	Multi-class class.	10	45000 / 5000 / 10000	117.6	1129.8
	PATTERN	Recognize subgraphs	2	10000 / 2000 / 2000	118.9	6079.8
	MolHIV	Binary class.	1	32901 / 4113 / 4113	25.5	54.1
	MolPCBA	Multi-label binary class.	128	350343 / 43793 / 43793	25.6	55.4
Proximity	***h*-Proximity**	Binary classification	4	9000	117.14	1484.82

—*PATTERN.* We evaluate ELENE on GIN with $k \in \{0, 1, 2\}$ and on GIN-AK$^+$ $k \in \{0, 1\}$. For ELENE-L, we evaluate all model variants for $k \in \{0, 1, 2, 3\}$ with 64-dim embeddings. Suspecting that degree information may nit play a salient role in the sub-graph patterns, we also evaluate the setting without degree information but found this slightly degrades performance compared to models that encode degree attributes. All models use $L = 6$.
—*CIFAR10 and MolPCBA.* We evaluate node-centric ELENE-L (**ND**) with $k \in \{1, 2, 3\}$. Due to computational constraints, we prioritize training with $k = 3$ given promising results in other tasks. On CIFAR, we discard uninformative degree information as graphs are $k = 8$ nearest neighbour graphs containing super-pixel information. We do not modify the architecture or hyper-parameters of the best-performing GNN-AK$^+$ model reported in (Zhao et al. 2022). Our results report average and standard deviations of the evaluation metric—Accuracy for CIFAR10, Average Precision (AP) for MolPCBA—collected from 3 independent runs.

***h*-Proximity.** See `proximityResults.sh` for details.

We evaluate node-centric ELENE-L without degree information, which matches the configuration of SPNNs. We do not tune any hyper-parameters, evaluating ELENE-L with $k \in \{3, 5\}$ fixing $L = 3$ and using 32-dim. embeddings. The first layer in the network embeds the color information, for which the model needs to appropriately learn to ignore irrelevant colors. Due to constrained computational resources, we only evaluate two maximum distances for ELENE-L, 3 and 5, sharing embedding weights and introducing ego-network signals before each of 3 GIN layers. We share ELENE-L embedding matrices across all layers, and set the maximum encoded degree $d = 0$ to only encode distance information. We report the mean and standard deviation of the binary classification accuracy computed across 10-folds over the dataset, following (Abboud, Dimitrov, and Ceylan 2022).

Memory Scalability.

We provide additional details and results from the memory scalability experiments in Subsection 6.4.5. Plots report the maximum memory consumption (y-axis) during training and inference on a dummy binary classification task where a single, randomly generated d-regular graph must be overfit as ‘positive’. Train, validation, and test sets are identical, and we iterate over the number of nodes in exponential increments of 0.25 between $10^2, 10^{2.25}, \dots, 10^{4.5}$ (x-axis). We supplement Subsection 6.4.5 with additional results when $d_{\texttt{max}} = 6$ and $d_{\texttt{max}} = 18$ in Figure C.1 and Figure C.2.

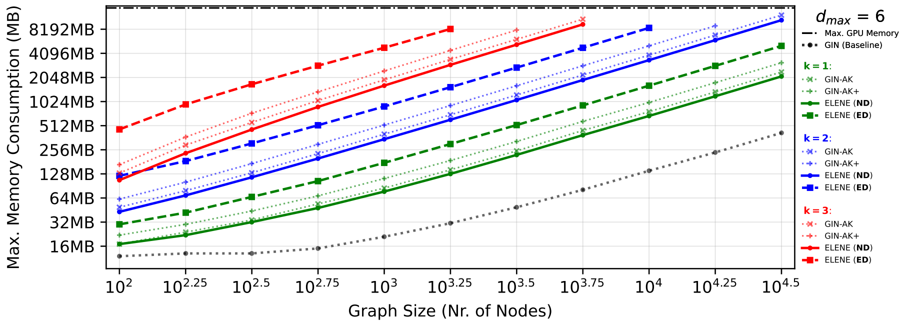


Figure C.1: Memory scalability analysis of ELENE with $d_{\texttt{max}} = 6$, produced as Figure 6.7. We include GIN (dotted line) and maximum GPU memory (dash-and-dotted line) as indicative lower and upper memory bounds. ELENE-L (**ND**, full lines) outperforms both GIN-AK, GIN-AK$^+$ (dotted lines) and ELENE-L (**ED**, dashed lines). Additionally, ELENE-L (**ND**) can encode all d-regular graphs in the benchmark when $k = 1$. As expected, memory consumption increases linearly with the number of nodes as $d_{\texttt{max}}$ is kept fixed.

Graph Density and Scalability.

We also provide extended memory scalability results by studying the impact of the density of the graph. We ran additional experiments on graphs where $N = 1000$, and evaluated the memory consumption as density increases as a function of the degree of nodes in the graph. We perform the same experiment in two settings: one where the degree distribution is

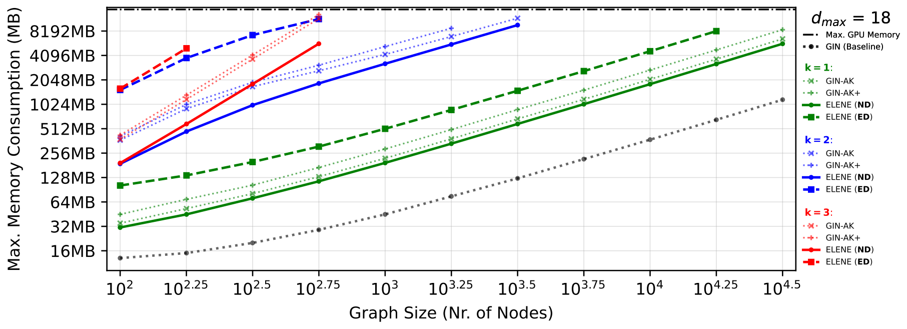


Figure C.2: Memory scalability analysis of ELENE with $d_{\mathtt{max}} = 18$. See caption of Figure C.1 for details.

regular (i.e., the graphs are d-regular, studying different values of d), and one where the distribution of degrees is irregular. In the irregular case, we study the case in which all nodes have *at least* degree d, but may have higher connectivity following the Barabási-Albert preferential attachment model (Barabási and Albert 1999).

— *Memory Consumption on Regular Density Graphs.* In Figure C.3, we compare GIN, GIN-AK, GIN-AK$^+$ and ELENE variants on at depths $k \in \{1, 2, 3\}$. Note that we could not include SPEN, as described in Subsection 6.4.5, due to reaching the maximum memory thresholds at $d_{\max} = 8$.

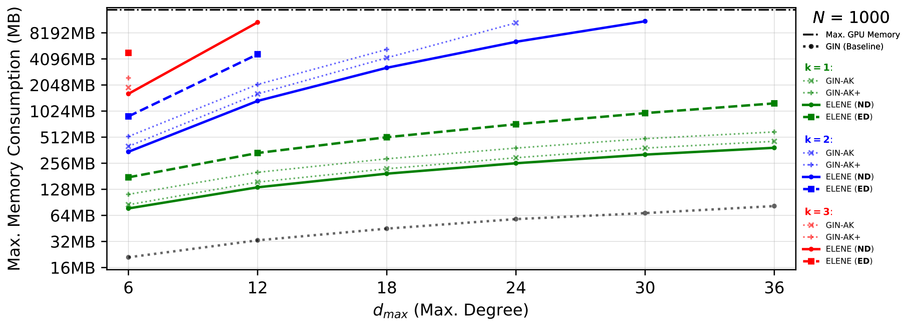


Figure C.3: Memory scalability analysis of ELENE with $N = 1000$ in function of increasing values of $d_{\max}$. See caption of Figure C.1 for details.

— *Memory Consumption on Irregular Density Graphs.* In Figure C.4, we repeat the analysis from Figure C.3 on graphs generated following the preferential attachment model where each node has at least m edges. We find that Elene-L (**ND**, full lines) outperforms both GIN-AK, GIN-AK$^+$ (dotted lines) and Elene-L (**ED**, dashed lines), matching Subsection 6.4.5 and results on regular connectivity patterns shown in Figure C.3.

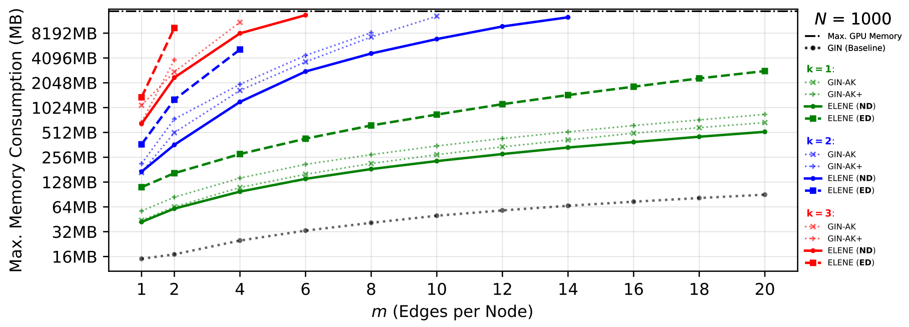


Figure C.4: Memory scalability analysis of Elene with $N = 1000$ in function of increasing values of $d_{\min}$ on random Barabási-Albert graphs. See caption of Figure C.1 for details.

C.2.3 Best Hyper-parameters

In this section we provide an overview of the best hyperparameters we find for Elene and Elene-L. For simplicity, we only report the best performing model, i.e., we do not distinguish between results enhancing a GIN or a GIN-AK$^+$ model. We group together hyper-parameters that are set at the dataset level (e.g. for the Counting Substructures or h-Proximity datasets), and report the hyper-parameters corresponding to the best performing models reported in Section 6.4.

In our summary, we include the best performing ego-network feature with (a) the ego-network depth — k, (b) the number of layers — L, (c) the embedding layer size for Elene-L and (d) the layer configuration. Layer configurations are either ‘First’ or ‘Full’ and follow the definition in Subsection C.2.2 and captured by Table C.1.

We summarise our findings in Table C.3. For datasets and tasks where multiple models achieve comparable performance (i.e. same performance metric with the reported significant digits), we break ties by reporting the model with the lowest memory footprint across the tie. In settings where ELENE and ELENE-L reach comparable performance, we describe this as part of the 'Layer Configuration' column.

Table C.3: Best hyper-parameters for the for the top performing models after introducing explicit ego-network attributes as shown in Section 6.4. We report the hyper-parameters corresponding to the best performing model by looking at the objective performance metric on each dataset, and resolve ties by selecting the model with the lowest memory footprint.

Benchmark	Dataset	Task	Ego-Net Feature	k-hops	L-Layers	Emb. Size (Elene-L)	Layer Config. (Elene-L)
Expr.	EXP		All Ego-Net Features Reach 100% Accuracy	1	4	32	First & Full
	SR25		GIN + Elene-L (ED)	1	2	32	First & Full
	Counting Sub.	Triangle	GIN-AK$^+$+Elene	2	3	16	Full
		Tailed Tri.	GIN-AK$^+$+Elene				
		Star	GIN + Elene-L (ND)				
		4-Cycle	GIN-AK$^+$+Elene				
	Graph Prop.	IsConn.	GIN-AK$^+$+Elene-L (ND)	3	6	16	First
		Diameter	GIN-AK$^+$+Elene-L (ND)				Full
		Radius	GIN-AK$^+$+Elene-L (ND)				Full
Real World Graphs	ZINC-12K		GIN+Elene-L (ND)	3	6	32	First (Elene), Full (Elene-L)
	CIFAR10		GIN+Elene-L (ND)	2	4	64	Full
	PATTERN		GIN+Elene-L (ND)	2	6	64	Full
	MolHIV		GIN+Elene	2	2	Elene, N/A	First
	MolPCBA		GIN+Elene-L (ND)	3	5	64	Full
Proximity	h-Proximity	$h = 3$	GIN + Elene-L (ND)	3	3	32	Full
		$h = 5$		5			
		$h = 8$		5			
		$h = 10$		5			

Bibliography

Abboud, Ralph, Radoslav Dimitrov, and İsmail İlkan Ceylan (2022). “Shortest Path Networks for Graph Property Prediction”. In: *Proceedings of the First Learning on Graphs Conference (LoG).* Oral presentation.

Abboud, Ralph et al. (2021). “The Surprising Power of Graph Neural Networks with Random Node Initialization”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 2112–2118.

Abdul-Mageed, Muhammad and Lyle Ungar (2017). “EmoNet: Fine-Grained Emotion Detection with Gated Recurrent Neural Networks”. In: *55th Annual Meeting of the Association for Computational Linguistics.*

Acheampong, Francisca Adoma, Chen Wenyu, and Henry Nunoo-Mensah (2020). “Text-based emotion detection: Advances, challenges, and opportunities”. In: *Engineering Reports* 2.7, e12189.

Agrawal, Sweta and Amit Awekar (2018). “Deep Learning for Detecting Cyberbullying Across Multiple Social Media Platforms”. In: *Advances in Information Retrieval.* Springer International Publishing, pp. 141–153.

Albert, Réka and Albert-László Barabási (2002). “Statistical mechanics of complex networks”. In: *Rev. Mod. Phys.* 74 (1), pp. 47–97.

Alon, Uri and Eran Yahav (2021). “On the Bottleneck of Graph Neural Networks and its Practical Implications”. In: *International Conference on Learning Representations.*

Alswaidan, Nourah and Mohamed Menai (2020). “A survey of state-of-the-art approaches for emotion recognition in text”. In: *Knowledge and Information Systems* 62.

Alvarez-Gonzalez, Nurudin, Andreas Kaltenbrunner, and Vicenç Gómez (2021a). “Emotion-Core: An Open Source framework for emotion detection research”. In: *Software Impacts* 10, p. 100179. ISSN: 2665-9638.

Alvarez-Gonzalez, Nurudin, Andreas Kaltenbrunner, and Vicenç Gómez (2021b). “Uncovering the Limits of Text-based Emotion Detection”. In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 2560–2583.

Alvarez-Gonzalez, Nurudin, Andreas Kaltenbrunner, and Vicenç Gómez (2022). “Beyond 1-WL with Local Ego-Network Encodings”. In: *The First Learning on Graphs Conference*.

Alvarez-Gonzalez, Nurudin, Andreas Kaltenbrunner, and Vicenç Gómez (2023a). “Beyond Weisfeiler—Lehman with Local Ego-Network Encodings”. In: *Machine Learning and Knowledge Extraction* 5.4, pp. 1234–1265.

Alvarez-Gonzalez, Nurudin, Andreas Kaltenbrunner, and Vicenç Gómez (2023b). “Improving Subgraph-GNNs via Edge-Level Ego-Network Encodings”. In: *Submitted to Transactions on Machine Learning Research*. Under review.

Arvind, V. et al. (2020). “On Weisfeiler-Leman invariance: Subgraph counts and related graph properties”. In: *Journal of Computer and System Sciences* 113, pp. 42–59.

Babai, Laszlo and Ludik Kucera (1979). “Canonical labelling of graphs in linear average time”. In: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, pp. 39–46.

Balcilar, Muhammet et al. (2021). “Breaking the Limits of Message Passing Graph Neural Networks”. In: *Proceedings of the 38th International Conference on Machine Learning (ICML)*.

Barabási, Albert-László and Réka Albert (1999). “Emergence of Scaling in Random Networks”. In: *Science* 286.5439, pp. 509–512.

Barceló, Pablo et al. (2020). “The Logical Expressiveness of Graph Neural Networks”. In: *International Conference on Learning Representations*.

Battaglia, Peter et al. (2016). “Interaction Networks for Learning about Objects, Relations and Physics”. In: *Advances in Neural Information Processing Systems*. Vol. 29.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). “Representation Learning: A Review and New Perspectives”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8, pp. 1798–1828.

Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent (2000). “A Neural Probabilistic Language Model”. In: *Advances in Neural Information Processing Systems.* Ed. by T. Leen, T. Dietterich, and V. Tresp. Vol. 13. MIT Press.

Bevilacqua, Beatrice et al. (2022). “Equivariant Subgraph Aggregation Networks”. In: *International Conference on Learning Representations.*

Bodnar, Cristian et al. (2021). “Weisfeiler and Lehman Go Cellular: CW Networks”. In: *Advances in Neural Information Processing Systems.* Vol. 34, pp. 2625–2640.

Bojanowski, Piotr et al. (2017a). “Enriching Word Vectors with Subword Information”. In: *Transactions of the Association for Computational Linguistics* 5.1, pp. 135–146.

Bojanowski, Piotr et al. (2017b). “Enriching Word Vectors with Subword Information”. In: *Transactions of the Association for Computational Linguistics* 5, pp. 135–146.

Bostan, Laura-Ana-Maria and Roman Klinger (2018). “An Analysis of Annotated Corpora for Emotion Classification in Text”. In: *27th International Conference on Computational Linguistics.*

Bouritsas, Giorgos et al. (2021). *Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting.*

Brassard-Gourdeau, Eloi and Richard Khoury (2019). “Subversive Toxicity Detection using Sentiment Information”. In: *Third Workshop on Abusive Language Online.*

Brijder, Robert et al. (2019). “On the Expressive Power of Query Languages for Matrices”. In: *ACM Trans. Database Syst.* 44.4.

Bronstein, Michael M. et al. (2021). “Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges”. In: *ArXiv* abs/2104.13478.

Brouwer, Andries E. and Hendrik Van Maldeghem (2022). *Strongly regular graphs.* Vol. 182. Cambridge University Press, p. 462. ISBN: 9781316512036.

Brown, Tom et al. (2020). “Language Models are Few-Shot Learners”. In: *Advances in Neural Information Processing Systems.* Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 1877–1901.

Bubeck, Sébastien et al. (2023). *Sparks of Artificial General Intelligence: Early experiments with GPT-4*. arXiv: `2303.12712 [cs.CL]`.

Buechel, Sven and Udo Hahn (2017a). "EmoBank: Studying the Impact of Annotation Perspective and Representation Format on Dimensional Emotion Analysis". In: *15th Conference of the European Chapter of the Association for Computational Linguistics*.

Buechel, Sven and Udo Hahn (2017b). "Readers vs. Writers vs. Texts: Coping with Different Perspectives of Text Understanding in Emotion Annotation". In: *11th Linguistic Annotation Workshop*, pp. 1–12.

Carlini, Nicholas et al. (2020). "Extracting Training Data from Large Language Models". In: *CoRR* abs/2012.07805. eprint: `2012.07805`.

Center, Pew Research (2016). *Nearly Eight-in-Ten Reddit Users Get News on the Site*.

Chauhan, Dushyant Singh et al. (2020). "Sentiment and Emotion help Sarcasm? A Multi-task Learning Framework for Multi-Modal Sarcasm, Sentiment and Emotion Analysis". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 4351–4360.

Chen, Zhengdao et al. (2020). "Can Graph Neural Networks Count Substructures?" In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 10383–10395.

Corso, Gabriele et al. (2020). "Principal Neighbourhood Aggregation for Graph Nets". In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. NIPS'20. Vancouver, BC, Canada: Curran Associates Inc. ISBN: 9781713829546.

Cowen, Alan and Dacher Keltner (2019). "What the face displays: Mapping 28 emotions conveyed by naturalistic expression". In: *American Psychologist* 75.

Cowen, Alan et al. (2018). "Mapping 24 Emotions Conveyed by Brief Human Vocalization". In: *American Psychologist*.

Cowen, Alan et al. (2019). "Mapping the Passions: Toward a High-Dimensional Taxonomy of Emotional Experience and Expression". In: *Psychological Science in the Public Interest* 20.1, pp. 69–90.

Cowen, Alan S. et al. (2020). "What music makes us feel: At least 13 dimensions organize subjective experiences associated with music across different cultures". In: *Proceedings of the National Academy of Sciences* 117.4, pp. 1924–1934.

Crowdflower (2016). *The Emotion in Text.* `https://www.figure-eight.com/data/sentiment-analysis-emotion-text/`.

Danescu-Niculescu-Mizil, Cristian et al. (2013). "No Country for Old Members: User Lifecycle and Linguistic Change in Online Communities". In: *Proceedings of the 22nd International Conference on World Wide Web.* WWW '13. Rio de Janeiro, Brazil: Association for Computing Machinery, pp. 307–318. ISBN: 9781450320351.

Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst (2016). "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems.* Vol. 29.

Demszky, Dorottya et al. (2020). "GoEmotions: A Dataset of Fine-Grained Emotions". In: *58th Annual Meeting of the Association for Computational Linguistics.*

Devlin, Jacob et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171–4186.

Duvenaud, David K et al. (2015). "Convolutional Networks on Graphs for Learning Molecular Fingerprints". In: *Advances in Neural Information Processing Systems.* Vol. 28.

Dwivedi, Vijay Prakash et al. (2020). "Benchmarking Graph Neural Networks". In: *arXiv preprint arXiv:2003.00982.*

Dwivedi, Vijay Prakash et al. (2022). "Graph Neural Networks with Learnable Structural and Positional Representations". In: *International Conference on Learning Representations.*

Ekman, Paul and Wallace V. Friesen (1971). "Constants across cultures in the face and emotion." In: *Journal of Personality and Social Psychology* 17.2, pp. 124–129.

El Ayadi, Moataz, Mohamed S. Kamel, and Fakhri Karray (2011). "Survey on speech emotion recognition: Features, classification schemes, and databases". In: *Pattern Recognition* 44.3, pp. 572–587. ISSN: 0031-3203.

Frasca, Fabrizio et al. (2022). "Understanding and Extending Subgraph GNNs by Rethinking Their Symmetries". In: *Advances in Neural Information Processing Systems.*

Fung, Pascale et al. (2018). "Towards Empathetic Human-Robot Interactions". In: *Computational Linguistics and Intelligent Text Processing*, pp. 173–193.

Gao, Hongyang, Zhengyang Wang, and Shuiwang Ji (2018). "Large-Scale Learnable Graph Convolutional Networks". In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* London, United Kingdom, pp. 1416–1424.

Gao, Leo et al. (2020). "The Pile: An 800GB Dataset of Diverse Text for Language Modeling". In: *arXiv preprint arXiv:2101.00027.*

Geerts, Floris (2021). "On the Expressive Power of Linear Algebra on Graphs". In: *Theory of Computing Systems* 65, pp. 1–61.

Gilmer, Justin et al. (2017). "Neural Message Passing for Quantum Chemistry". In: *Proceedings of the 34th International Conference on Machine Learning (ICML).* Vol. 70, pp. 1263–1272.

Goodfellow, Ian et al. (2014). "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems.* Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc.

Gori, M., G. Monfardini, and F. Scarselli (2005). "A new model for learning in graph domains". In: *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.* Vol. 2, 729–734 vol. 2.

Grohe, Martin (2017). *Descriptive Complexity, Canonisation, and Definable Graph Structure Theory.* Lecture Notes in Logic. Cambridge University Press.

Grohe, Martin (2021). "The Logic of Graph Neural Networks". In: *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science.* LICS '21. Rome, Italy: Association for Computing Machinery. ISBN: 9781665448956.

Grover, Aditya and Jure Leskovec (2016). "Node2Vec: Scalable Feature Learning for Networks". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 855–864.

Guerini, Marco and Jacopo Staiano (2015). "Deep Feelings: A Massive Cross-Lingual Study on the Relation between Emotions and Virality". In: *24th International Conference on World Wide Web.*

Gutteridge, Benjamin et al. (2023). "DRew: Dynamically Rewired Message Passing with Delay". In: *International Conference on Machine Learning.* PMLR, pp. 12252–12267.

Hamilton, Will, Zhitao Ying, and Jure Leskovec (2017). "Inductive Representation Learning on Large Graphs". In: *Advances in Neural Information Processing Systems 30.*

He, Kaiming et al. (2016). "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

Hedderich, Michael A. et al. (2021). "A Survey on Recent Approaches for Natural Language Processing in Low-Resource Scenarios". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Online: Association for Computational Linguistics, pp. 2545–2568.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long Short-Term Memory". In: *Neural Computation* 9.8, pp. 1735–1780.

Hoffmann, Jordan et al. (2022). *Training Compute-Optimal Large Language Models.* arXiv: `2203.15556 [cs.CL]`.

Hu, Weihua et al. (2020a). "Open Graph Benchmark: Datasets for Machine Learning on Graphs". In: *arXiv preprint arXiv:2005.00687.*

Hu, Weihua et al. (2020b). "Strategies for Pre-training Graph Neural Networks". In: *International Conference on Learning Representations.*

Huang, Ningyuan Teresa and Soledad Villar (2021). "A Short Tutorial on The Weisfeiler-Lehman Test And Its Variants". In: *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, pp. 8533–8537.

Husseini Orabi, Ahmed et al. (2018). "Deep Learning for Depression Detection of Twitter Users". In: *Fifth Workshop on Computational Linguistics and Clinical Psychology: From Keyboard to Clinic*, pp. 88–97.

Hutto, Clayton and Eric Gilbert (2014). "Vader: A parsimonious rule-based model for sentiment analysis of social media text". In: *7th International AAAI Conference on Web and Social Media.* Vol. 8. 1.

Jackson, Joshua Conrad et al. (2019). “Emotion semantics show both cultural variation and universal structure”. In: *Science* 366.6472, pp. 1517–1522.

Jones, Gareth (Garethjns) (2019). *IncrementalTrees: Online Tree Learners on top of Scikit-Learn.* `https://github.com/garethjns/IncrementalTrees`.

Joshi, Abhinav et al. (2022). “COGMEN: COntextualized GNN based Multimodal Emotion recognitioN”. In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Ed. by Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz. Seattle, United States: Association for Computational Linguistics, pp. 4148–4164.

Kim, Jinwoo et al. (2022). “Pure Transformers are Powerful Graph Learners”. In: *Advances in Neural Information Processing Systems.*

Kipf, Thomas N. and Max Welling (2017). “Semi-Supervised Classification with Graph Convolutional Networks”. In: *5th International Conference on Learning Representations, ICLR.*

Kirillov, Alexander et al. (2023). “Segment Anything”. In: *arXiv:2304.02643.*

Kolesnikov, Alexander et al. (2021). “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In.

Kondor, Risi Imre and John Lafferty (2002). “Diffusion kernels on graphs and other discrete structures”. In: *In Proceedings of the ICML*, pp. 315–322.

Kriege, Nils M. et al. (2018). “A Property Testing Framework for the Theoretical Expressivity of Graph Kernels”. In: *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18.* International Joint Conferences on Artificial Intelligence Organization, pp. 2348–2354.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems.* Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc.

Kumar, Shivani et al. (2023). “From Multilingual Complexity to Emotional Clarity: Leveraging Commonsense to Unveil Emotions in Code-Mixed

Dialogues". In: *The 2023 Conference on Empirical Methods in Natural Language Processing.*

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, p. 436.

Leskovec, Jure and Andrej Krevl (2014). *SNAP Datasets: Stanford Large Network Dataset Collection.*

Lewis, Mike et al. (2020). "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.* Online: Association for Computational Linguistics, pp. 7871–7880.

Li, Pan et al. (2020). "Distance Encoding: Design Provably More Powerful Neural Networks for Graph Representation Learning". In: *Advances in Neural Information Processing Systems.* Vol. 33, pp. 4465–4478.

Li, S. and W. Deng (2020). "Deep Facial Expression Recognition: A Survey". In: *IEEE Transactions on Affective Computing*, pp. 1–1.

Liu, Siwei, Iadh Ounis, and Craig Macdonald (2022). "An MLP-Based Algorithm for Efficient Contrastive Graph Recommendations". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval.* SIGIR '22. New York, NY, USA: Association for Computing Machinery, pp. 2431–2436. ISBN: 9781450387323.

Lykousas, Nikolaos et al. (2019). "Sharing Emotions at Scale: The Vent Dataset". In: *13th International AAAI Conference on Weblogs and Social Media* 13.01, pp. 611–619.

Malko, Anton et al. (2021). "Demonstrating the Reliability of Self-Annotated Emotion Data". In: *Proceedings of the Seventh Workshop on Computational Linguistics and Clinical Psychology: Improving Access.* Online: Association for Computational Linguistics, pp. 45–54.

Manning, Christopher D. and Hinrich Schütze (1999). *Foundations of Statistical Natural Language Processing.* Cambridge, Massachusetts: The MIT Press.

Maron, Haggai et al. (2019a). "Invariant and Equivariant Graph Networks". In: *International Conference on Learning Representations.*

Maron, Haggai et al. (2019b). "Provably Powerful Graph Networks". In: *Advances in Neural Information Processing Systems.* Vol. 32.

Meque, Abdul Gafar Manuel et al. (2023). “Machine learning-based guilt detection in text”. In: *Scientific Reports* 13.1, p. 11441.

Michel, Gaspard et al. (2023). “Path Neural Networks: Expressive and Accurate Graph Neural Networks”. In: *Proceedings of the 40th International Conference on Machine Learning (ICML).*

Mikolov, Tomas et al. (2013). “Distributed Representations of Words and Phrases and Their Compositionality”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’13. Lake Tahoe, Nevada: Curran Associates Inc., pp. 3111–3119.

Mitton, Joshua and Roderick Murray-Smith (2023). “Subgraph Permutation Equivariant Networks”. In: *Transactions on Machine Learning Research.*

Mohammad, Saif M (2021). “Sentiment analysis: Automatically detecting valence, emotions, and other affectual states from text”. In: *Emotion Measurement.* Elsevier, pp. 323–379.

Mohammad, Saif M, Svetlana Kiritchenko, and Xiaodan Zhu (2013). “NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets”. In: *Second Joint Conference on Lexical and Computational Semantics*, pp. 321–327.

Mohammad, Saif M and Peter Turney (2013). “Crowdsourcing a Word-Emotion Association Lexicon”. In: *Computational Intelligence* 29.

Mohammad, Saif M. et al. (2015). “Sentiment, Emotion, Purpose, and Style in Electoral Tweets”. In: *Information Processing and Management* 51.4, pp. 480–499.

Morris, Christopher et al. (2019). “Weisfeiler and Leman Go Neural: Higher-Order Graph Neural Networks”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01, pp. 4602–4609.

Morris, Christopher et al. (2020). “TUDataset: A collection of benchmark datasets for learning with graphs”. In: *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020).* arXiv: `2007.08663`.

Morris, Christopher et al. (2021). “Weisfeiler and Leman go Machine Learning: The Story so far”. Preprint. Weisfeiler and Leman go Machine Learning: The Story so far. arXiv: `2112.09992 [cs.LG]`.

Nielsen, Finn (2011). “A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs”. In: *ESWC2011 Workshop on ’Making*

Sense of Microposts'. Vol. 718. CEUR Workshop Proceedings, pp. 93–98.

Niepert, Mathias, Mohamed Ahmed, and Konstantin Kutzkov (2016). "Learning Convolutional Neural Networks for Graphs". In: *Proceedings of The 33rd International Conference on Machine Learning.* Vol. 48. New York, USA, pp. 2014–2023.

Nikolentzos, Giannis, George Dasoulas, and Michalis Vazirgiannis (2020). "k-hop graph neural networks". In: *Neural Networks* 130, pp. 195–205.

Noroozi, Fatemeh et al. (2018). "Survey on emotional body gesture recognition". In: *IEEE transactions on affective computing.*

Oono, Kenta and Taiji Suzuki (2022). "Graph neural networks exponentially lose expressive power for node classification". In: *International Conference on Learning Representations.*

Ortigosa, Alvaro, José M. Martín, and Rosa M. Carro (2014). "Sentiment analysis in Facebook and its application to e-learning". In: *Computers in Human Behavior* 31, pp. 527–541. ISSN: 0747-5632.

Ouyang, Long et al. (2022). "Training language models to follow instructions with human feedback". In: *Advances in Neural Information Processing Systems.* Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., pp. 27730–27744.

Papp, Pál András and Roger Wattenhofer (2022). "A Theoretical Comparison of Graph Neural Network Extensions". In: *Proceedings of the 39th International Conference on Machine Learning.* Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 17323–17345.

Papp, Pál András et al. (2021). "DropGNN: Random Dropouts Increase the Expressiveness of Graph Neural Networks". In: *35th Conference on Neural Information Processing Systems (NeurIPS).*

Pedregosa, F. et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Peng, Bo et al. (2023). *RWKV: Reinventing RNNs for the Transformer Era.*

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Process-*

ing (EMNLP). Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543.

Perozzi, Bryan, Rami Al-Rfou, and Steven Skiena (2014). "DeepWalk: Online Learning of Social Representations". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710.

Peters, Matthew E. et al. (2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, pp. 2227–2237.

Plutchik, Robert (1980). "A general psychoevolutionary theory of emotion". In: *Theories of Emotion*. Ed. by Robert Plutchik and Henry Kellerman. Academic Press, pp. 3–33.

Poria, Soujanya et al. (2019). "Emotion Recognition in Conversation: Research Challenges, Datasets, and Recent Advances". In: *IEEE Access* PP, pp. 1–1.

Ptaszynski, M. et al. (2009). "A System for Affect Analysis of Utterances in Japanese Supported with Web Mining". In: *Journal of Japan Society for Fuzzy Theory and Intelligent Informatics* 21, pp. 194–213.

Radford, Alec et al. (2018). "Improving language understanding by generative pre-training". In: *arXiv preprint*.

Radford, Alec et al. (2019). "Language Models are Unsupervised Multitask Learners". In: *arXiv preprint*.

Raffel, Colin et al. (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21.140, pp. 1–67.

Ramírez-Cifuentes, Diana et al. (2020). "Detection of Suicidal Ideation on Social Media: Multimodal, Relational, and Behavioral Analysis". In: *Journal of Medical Internet Research* 22.7.

Rampášek, Ladislav et al. (2022). "Recipe for a General, Powerful, Scalable Graph Transformer". In: *Advances in Neural Information Processing Systems* 35.

Rieck, Bastian, Christian Bock, and Karsten Borgwardt (2019). "A Persistent Weisfeiler–Lehman Procedure for Graph Classification". In: *Proceedings of the 36th International Conference on Machine Learning.*

Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 5448–5458.

Robertson, Stephen and Hugo Zaragoza (2009). “The Probabilistic Relevance Framework: BM25 and Beyond”. In: *Found. Trends Inf. Retr.* 3.4, pp. 333–389. ISSN: 1554-0669.

Rombach, Robin et al. (2021). *High-Resolution Image Synthesis with Latent Diffusion Models*. arXiv: 2112.10752 [cs.CV].

Russell, James (2003). “Core Affect and the Psychological Construction of Emotion”. In: *Psychological review* 110, pp. 145–72.

Samanta, Bidisha et al. (2020). “NEVAE: A Deep Generative Model for Molecular Graphs”. In: *Journal of Machine Learning Research* 21.114, pp. 1–33.

Scarselli, F. et al. (2005). “Graph neural networks for ranking Web pages”. In: *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI’05)*, pp. 666–672.

Scarselli, Franco et al. (2009). “The Graph Neural Network Model”. In: *IEEE Transactions on Neural Networks* 20.1, pp. 61–80.

Schwartz, H. Andrew et al. (2013). “Personality, Gender, and Age in the Language of Social Media: The Open-Vocabulary Approach”. In: *PLOS ONE* 8.9, pp. 1–16.

Scollon, Christie N. et al. (2004). “Emotions Across Cultures and Methods”. In: *Journal of Cross-Cultural Psychology* 35.3, pp. 304–326.

See, Abigail, Peter J. Liu, and Christopher D. Manning (2017). “Get To The Point: Summarization with Pointer-Generator Networks”. In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, pp. 1073–1083.

Shervashidze, Nino and Karsten Borgwardt (2009). “Fast subtree kernels on graphs”. In: *Advances in Neural Information Processing Systems*. Vol. 22.

Shervashidze, Nino et al. (2010). “Weisfeiler-Lehman Graph Kernels”. In: *Journal of Machine Learning Research* 1, pp. 1–48.

Shmueli, Boaz and Lun-Wei Ku (2019). “SocialNLP EmotionX 2019 Challenge Overview: Predicting Emotions in Spoken Dialogues and Chats”. In.

Silva, N. F. D., E. Hruschka, and Estevam R. Hruschka (2014). “Tweet sentiment analysis with classifier ensembles”. In: *Decis. Support Syst.* 66, pp. 170–179.

Staab, Robin et al. (2023). *Beyond Memorization: Violating Privacy Via Inference with Large Language Models.* arXiv: `2310.07298 [cs.AI]`.

Strapparava, Carlo and Rada Mihalcea (2007). “SemEval-2007 Task 14: Affective Text”. In: *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007).* Prague, Czech Republic: Association for Computational Linguistics, pp. 70–74.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). “Sequence to Sequence Learning with Neural Networks”. In: *Advances in Neural Information Processing Systems.* Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc.

Tai, Kai Sheng, Richard Socher, and Christopher D. Manning (2015). “Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks”. In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* Beijing, China: Association for Computational Linguistics, pp. 1556–1566.

Tausczik, Yla and James Pennebaker (2010). “The Psychological Meaning of Words: LIWC and Computerized Text Analysis Methods”. In: *Journal of Language and Social Psychology* 29, pp. 24–54.

Thelwall, Mike et al. (2010). “Sentiment Strength Detection in Short Informal Text”. In: *Journal of the American Society for Information Science and Technology* 61, pp. 2544–2558.

Togninalli, Matteo et al. (2019). “Wasserstein Weisfeiler–Lehman Graph Kernels”. In: *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 6436–6446.

Van Dam, Edwin R and Willem H Haemers (2003). “Which Graphs Are Determined by Their Spectrum?” In: *Linear Algebra and its Applications* 373, pp. 241–272.

Vaswani, Ashish et al. (2017). “Attention is All you Need”. In: *Advances in Neural Information Processing Systems.* Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc.

Veličković, Petar (2022). "Message passing all the way up". In: *ICLR 2022 Workshop on Geometrical and Topological Representation Learning.*

Veličković, Petar et al. (2018). "Graph Attention Networks". In: *International Conference on Learning Representations.*

Vignac, Clément, Andreas Loukas, and Pascal Frossard (2020). "Building powerful and equivariant graph neural networks with structural message-passing". In: *Advances in Neural Information Processing Systems.* Ed. by H. Larochelle et al. Vol. 33.

Vishwanathan, S. V. N. et al. (2010). "Graph Kernels". In: *J. Mach. Learn. Res.* 11, pp. 1201–1242. ISSN: 1532-4435.

Wang, Ziyang et al. (2020). "Global Context Enhanced Graph Neural Networks for Session-based Recommendation". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 169–178.

Wei, Jason et al. (2022). "Finetuned Language Models are Zero-Shot Learners". In.

Weisfeiler, B and A Leman (1968). "The Reduction of a Graph to Canonical Form and the Algebra which Appears Therein". In: *Nauchno-Technicheskaya Informatsia* 2(9), pp. 12–16.

Xu, Keyulu et al. (2019). "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations.*

Yang, Ze et al. (2019). "Read, Attend and Comment: A Deep Architecture for Automatic News Comment Generation". In: *Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).*

Ying, Chengxuan et al. (2021). "Do Transformers Really Perform Badly for Graph Representation?" In: *Thirty-Fifth Conference on Neural Information Processing Systems.*

You, Jiaxuan, Rex Ying, and Jure Leskovec (2019). "Position-aware Graph Neural Networks". In: *Proceedings of the 36th International Conference on Machine Learning.* Vol. 97. Proceedings of Machine Learning Research. Long Beach, California, USA: PMLR, pp. 7134–7143.

You, Jiaxuan, Rex Ying, and Jure Leskovec (2020). "Design Space for Graph Neural Networks". In: *NeurIPS.*

You, Jiaxuan et al. (2021). “Identity-aware graph neural networks”. In: *Thirty-Fifth AAAI Conference on Artificial Intelligence*. Vol. 35, pp. 10737–10745.

Yun, Seongjun et al. (2019). “Graph Transformer Networks”. In: *Advances in Neural Information Processing Systems*. Vol. 32.

Zhang, Bohang et al. (2023). “Rethinking the Expressive Power of GNNs via Graph Biconnectivity”. In: *International Conference on Learning Representations*.

Zhang, Muhan and Pan Li (2021). “Nested graph neural networks”. In: *Advances in Neural Information Processing Systems* 34.

Zhao, Lingxiao et al. (2022). “From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness”. In: *International Conference on Learning Representations*.