

A Study on Graph Learning: Methods, Explainability, Quantum Enhancement and Ethical Issues

Yanghepu Li

Doctor of Philosophy

A Study on Graph Learning:
Methods, Explainability, Quantum
Enhancement and Ethical Issues

(グラフ学習に関する研究—方法、説明可能性、量子強化、及び倫理問題—)

Yanghepu Li

Graduate School of Advanced Integrated Studies
in Human Survivability,
Kyoto University

March, 2026

Abstract

Graphs, also known as networks, model relations among objects across various domains including social studies, biology, chemistry, transportation, finance, knowledge systems and so on. Learning properties and structures from graphs is thus an important task. Today, the methods for learning graphs are shifting from classical algorithms to machine learning approaches and, more recently, to graph neural networks (GNNs). These advances have demonstrated significant progress in applications including recommendation systems, drug discovery, natural language processing, and even emerging areas such as quantum computing. Yet, graph-based intelligence today faces critical challenges. Graph-based models' application in high-stakes scenarios suffers from the opaque, black-box nature of modern graph learning approaches, the practical gap between theoretical potential of quantum graph methods and current quantum hardware limitations, and societal and ethical concerns regarding to transparency and trust.

This thesis provides a study on graph-based intelligence, integrating foundational concepts, methodological innovation, and implications. It begins by introducing the concept of graphs and the principles of modern graph learning, highlighting both the strengths and limitations of classical and contemporary methodologies. The discussion emphasizes that, while GNNs have enabled significant breakthroughs, their decision processes are often difficult to interpret, particularly in settings where model predictions must be scrutinized, verified, or justified. Building upon this observation, it addresses the challenge of explainability in graph learning and introduces a continuous representation of connectedness to improve instance-level explanations. Two proposed methods—Connected-CF² and CRoCExplainer—demonstrate how incorporating structural priors may yield more faithful and interpretable explanations compared with the state-of-the-art studies, supported by theoretical analysis and empirical evaluations. These approaches illustrate that enhancing explanations with structural reasoning improves clarity and usability, especially in scenarios where understanding why a model prioritizes certain substructures or relational patterns is critical for researchers and practitioners.

The thesis then turns to the emerging intersection between graph learning and quantum computation. After introducing the quantum principles relevant to machine learning, it examines Quantum Graph Neural Networks (QGNNs) and provides an analysis of the quantum hardware bottlenecks that presently constrain the practical performance of quantum machine learning. To address these limitations, the thesis provides an automatic method that contributes directly to the hardware-design of quantum computing. Specifically, it proposes an automated method for generating air-bridge layouts on superconducting quantum chips—an essential component for mitigating parasitic modes—thereby addressing a notable gap in today's quantum electronic design automation (EDA) toolchain and offering a practical step toward overcoming current hardware bottlenecks. By improving the physical routing and structural integrity of microwave transmission lines, this automated layout generation reduces unintended electromagnetic coupling and signal interference, which are known contributors to the degradation of qubit coherence. Consequently, the approach supports more stable quantum operations and provides a promising direction toward scalable quantum machine learning systems, where the theoretical advantages of quantum-enhanced graph models may eventually become realizable in practice.

Finally, the thesis examines ethical and societal dimensions of graph-based AI. With a review

on risks, challenges, and governance on AI, it presents an empirical study involving over 200 responses to assess public perceptions on AI transparency. The results highlight the importance of explainability and openness for establishing trust and responsible AI deployment. The survey findings suggest that transparency is broadly perceived as a foundational requirement for confidence in AI-assisted decision-making, indicating that the development of future graph-based systems must take into account not only technical performance, but also the expectations and concerns of the broader public and stakeholders involved.

As the conclusion, this work finds that advancing graph-based intelligence requires an integrated perspective—one that integrates methodological advances, explainability, quantum-inspired computation, and ethical responsibility. By bridging theoretical foundations, practical techniques, and societal considerations, the thesis provides a multidimensional framework for understanding the current landscape and future directions of graph-related AI studies. The findings underscore that responsible and trustworthy graph-based AI must be viewed not only as a technical objective but as a socio-technical endeavor shaped by interdisciplinary insights and collaborative development.

Contents

1	Introduction	1
1.1	What is Learning?	1
1.2	What is Machine Learning?	2
1.3	Learning Graphs	6
1.3.1	What is Graph?	6
1.3.2	Learning from Classic Graph Problems	8
1.3.3	Machine Learning for Graphs	9
1.4	Success and Issues Regarding Learning Graphs	10
1.4.1	Success of Learning Graphs	10
1.4.2	Issues and Challenges in Learning Graphs	11
1.5	Overview of this Thesis	12
2	Preliminaries	15
2.1	Graph and Graph-like Data	15
2.1.1	Definition and Examples	15
2.1.2	Graph Representation	19
2.1.3	The Specialty and Difficulty in Graph Data	21
2.2	Traditional Machine Learning for Graphs	22
2.3	Graph Learning	24
2.4	Graph Learning vs. Statistical Machine Learning on Graphs	29
2.5	Conclusion and Discussion	33
3	Explainability of Graph Learning	35
3.1	Introduction	35
3.2	Taxonomy of eXplainable AI (XAI) and Inductive Reasoning	37
3.3	Graph Explainability Problems (GEPs)	40
3.4	Connectedness - an Overlooked Property in GEPs	43
3.4.1	Approximations of Connectedness - Connected-CF ²	45
3.4.2	Improved Connectedness Representation: Continuous Representation of Connectedness (CRoC)	45
3.5	Explainer Framework with Connectedness and Improved Connectedness Repre- sentation	52
3.6	Conclusion of the Chapter	56

4	Graph and Quantum Computing	58
4.1	Introduction to Basic Quantum Computing	58
4.1.1	Quantum States	58
4.1.2	Quantum Gates and Quantum Circuits	59
4.1.3	Quantum Measurement	60
4.1.4	The Quantum Computing Workflow: From State Preparation to Measurement	62
4.2	Quantum Machine Learning	63
4.3	From Graph to Qubits: Quantum Graph Neural Network (QGNN)	65
4.4	The Bottleneck in Quantum Computing	66
4.4.1	Limitations of Quantum Hardware	66
4.4.2	Graph-related Algorithms in Automatic Construction of Air-bridge in Quantum Chip Layout	67
4.5	Conclusion of the Chapter	88
5	Risks, Challenges and Ethical Issues	89
5.1	Rapid Advancement of Modern AI	89
5.2	Risks and Challenges of Rapid-developing AI	92
5.3	Possible Solutions and AI Regulations to Address Risks and Challenges	96
5.4	Ethical Study about the Transparency of AI	97
5.4.1	Hypothesis Framework	98
5.4.2	Questionnaire and Workshop Design	98
5.4.3	Results and Discussion	99
5.4.4	Conclusion	111
5.5	Conclusion of the Chapter	111
6	Conclusion, Discussion, and Future Direction	112
6.1	Summary of Contributions	112
6.2	Discussion and Conclusion	113
6.3	Future Directions	115
	Acknowledgement	117
	Bibliography	118
	Appendices	130
A	Questionnaire on Transparency of AI	131
B	Workshop on Explainable AI	135

List of Figures

1.1	An illustration on learning. (Work by the author)	2
1.2	An illustration on a schematic architecture of a MLP consisting of three layers. (Work by the author)	5
1.3	Illustration of machine learning for image-based fruit recognition. (Work by the author)	6
1.4	An illustration of the Königsberg Problem. (Source: [10])	7
1.5	A showcase of the solution to 48 States in America Traveling Salesman Problem, provided by Cornell University Computational Optimization Open Textbook - Optimization Wiki from https://optimization.cbe.cornell.edu/index.php?title=File:48StatesTSP.png .	8
1.6	Structure overview of this thesis.	14
2.1	A graph example with 7 nodes and 6 edges. (Work by the author)	16
2.2	A molecule and the corresponding chemical graph, where hydrogens are omitted. (Work by the author)	16
2.3	An illustration of a social network, where nodes represent social accounts and edges model “follower” relationship. (Work by the author)	17
2.4	An abstract transportation network where edge weight represents the distance. (Work by the author)	17
2.5	An illustration of a heterophilic fraud detection graph, where blue nodes denote normal users while red nodes denote deceivers. (Work by the author)	18
2.6	Adjacency-list and adjacency-matrix representations of an undirected graph. [a] An undirected graph with 5 nodes and 7 edges. [b][c] The corresponding adjacency-list and adjacency-matrix representations of the graph in [a], respectively.	20
2.7	A binary pixel art provided by http://www.uff.br/cdme/ .	21
2.8	An illustration of machine learns for graphs, where colors denote labels. Red and green are two known labels, while gray denotes unknown labels. (source: [44])	23
2.9	A showcase of feature extraction of a social network. A node (e.g., a user account in social media) is transformed into a tensor consisting of structural information extracted from the graph (degree, centrality, etc.) and personal information (e.g., age, gender, interests, etc.). (Adjusted from [44] by the author)	24
2.10	Framework of mol-infer proposed by Zhu, et al. [48], where stage 2 is based on human knowledge.	25
2.11	Visualization of a GNN framework. (Adjusted from [53] by the author)	26

2.12	An illustration of a Graph Neural Network (GNN) for molecular toxicity prediction. (Work by the author)	28
2.13	Illustration of the structure of our graph learning model used in Section 2.4. (Work by the author)	31
3.1	An illustration to showcase the significance of eXplainable AI (XAI). The figures of Nobel prize winners are provided by [69]; the figures of protein and workflow are from [68], and the figures of instruments are from Japan Synchrotron Radiation Research Institute (the left one) and Institute for Protein Research, the University of Osaka (the right one).	36
3.2	An illustration of XAI approaches taxonomy.	38
3.3	An illustration of graph learning to infer task-specific properties (e.g., mutagenicity).	40
3.4	Illustration of the aim in GEP: identifying the graph components or patterns (NO_2 in this case) that most strongly contribute to the model’s prediction.	40
3.5	An illustration of GNNExplainer ([84]). For a GNN model predicting a person’s hobby on social networks (left), GNNExplainer tries to give corresponding explanations based on one’s social circles (right).	42
3.6	An illustration of CF-GNNExplainer ([85]).	42
3.7	Illustration of the limitations of purely factual and purely counterfactual reasoning in graph explanations, and the benefit of combining both of them ([95]).	43
3.8	An illustration on the limitations for explanations given by existing models that do not consider connectedness.	44
3.9	CRoC values for some graphs with the same number of nodes. (©2025 IEEE. Reprinted with permission from [30]).	46
3.10	An illustration of the examples in the above Remark 2.	47
3.11	An illustration on Algorithm 1 working against a given graph (notice that we do not really need to re-index nodes after sorting). (©2025 IEEE. Reprinted with permission from [30]).	48
3.12	A comparison between the value of Eq. 3.3 and the output of Algorithm 1 for 1,000 random graphs. It can be observed that the latter is usually bounded by the former and three times of it. (©2025 IEEE. Reprinted with permission from [30]).	48
3.13	Correlations between CRoC (calculated from Algorithm 1) and the number of connected components. Red circles represent the symmetric graphs, whereas blue triangles denote the asymmetric graphs. The total number of nodes of the graph is fixed to 100. (©2025 IEEE. Reprinted with permission from [30]).	49
3.14	Correlations between CRoC (calculated from Algorithm 1) and the ratio of the sizes of the two components. The total number of nodes of the graph is fixed to 100. (©2025 IEEE. Reprinted with permission from [30]).	50
3.15	Correlations between CRoC (calculated by Algorithm 1) and the number of components for random Erdős–Rényi networks. (©2025 IEEE. Reprinted with permission from [30]).	50
3.16	Correlations between CRoC (calculated by Algorithm 1) and the variance of component sizes for random Erdős–Rényi networks. (©2025 IEEE. Reprinted with permission from [30]).	51

3.17	An example of the explanations (in bold edges) tested on the Mutag ₀ , BA-shapes, and Tree-cycles datasets. While CF ² produces a more disconnected subgraph, CRoCExplainer tends to find the connected one. (©2025 IEEE. Reprinted with permission from [30]).	55
3.18	Some failure cases given by CRoCExplainer for Tree-Cycles.	56
4.1	Visualization and illustration of qubit states in Bloch Sphere. (Work by the author)	60
4.2	A parameterized quantum circuit (PQC) (explained in Section 4.2) consisting of alternating layers of single-qubit rotation gates $R_y(\theta_i)$ and entangling CNOT gates arranged in a ladder-like topology given IBM Qiskit in https://quantum.cloud.ibm.com/docs/en/api/qiskit/circuit_library . Such variational structures are commonly used in quantum machine learning models and variational quantum algorithms, where the trainable parameters $\{\theta_i\}$ encode learnable transformations and the CNOT layers generate entanglement across qubits. The circuit depth and entanglement pattern together determine the expressive power of the quantum model.	62
4.3	Visualization of a QNN. The input state ρ_{in} is prepared using the operation U_{data} , followed by VQC $U(\theta)$ and measurement operation.	64
4.4	An illustration of QGNN.	65
4.5	The coplanar waveguide (CPW). [a] A 3D illustration of a CPW (yellow strips) and air-bridge (white); [b] A CPW in GDSII format; [c] A close-up of a section of a CPW, with the CPW path in red, the CPW channel in green and four sets of lines of the CPW in blue. (©2024 IEEE. Reprinted with permission from [31]).	68
4.6	Components in the GDSII file format are represented by a sequential array of points. This pad is represented by $[a, b, c, d, e, f, g, h, i, j, k, l]$, where $a, \dots, l$ represents actual coordinates in the layout. (©2024 IEEE. Reprinted with permission from [31]).	70
4.7	EDA process for quantum processor design, with our contribution in red. (©2024 IEEE. Reprinted with permission from [31]).	70
4.8	Air-bridge level information. [a] the location (coordinates) of the geometric center of the air-bridge, [b] the direction angle of the length of the air-bridge, [c] size information of air-bridge central body (red arrows) and air-bridge pad (black arrows). (©2024 IEEE. Reprinted with permission from [31]).	72
4.9	Visualization of the parameter structs. [a], [b], [c] correspond to point_parameter , arc_parameter , and center_parameter , respectively. (©2024 IEEE. Reprinted with permission from [31]).	76
4.10	Visualization of arcs, sorted_arcs, and channels. [a] arcs (red), the input of Algorithm 4; [b] sorted_arcs (red), the output of Algorithm 4; [c] CPW channels computed by Algorithm 5, including arcs (green) and lines (blue). (©2024 IEEE. Reprinted with permission from [31]).	76
4.11	Special structures. [a] a boundary line of a CPW; [b] a no-arc polygon; [c] a CPW intersection. (©2024 IEEE. Reprinted with permission from [31]).	79

4.12	Two types of air-bridge: [a] Separated air-bridges and a close-up of pad size information (in black arrows) and central body size information (in red arrows); [b] Full package air-bridges and a close-up of single small window size information (in red arrows) and pad size information (in black arrows). (©2024 IEEE. Reprinted with permission from [31]).	80
4.13	An exaggerated example of air-bridges with inaccurate locations. This phenomenon is commonly encountered when manually placed. The e_i and l_i in Eq. 4.10 refer to the length of the green arrow and the length of the blue arrow. (©2024 IEEE. Reprinted with permission from [31]).	81
4.14	An example of overlap: air-bridges may erroneously overlap with components such as control lines, or other air-bridges. This phenomenon is commonly encountered at coupled CPWs. The o_i and $s_i/2$ in Eq. 4.11 refer to the green area and the blue area in the second air-bridge. (©2024 IEEE. Reprinted with permission from [31]).	82
4.15	An example of air-bridges on control lines. The a_{ik} and c_i in Eq. 4.12 refer to the length of the green arrow and the length of the blue arrow. (©2024 IEEE. Reprinted with permission from [31]).	82
4.16	Accuracy and Overlap in terms of the number of points in different quantum chips. (©2024 IEEE. Reprinted with permission from [31]).	84
4.17	A 4-qubit layout provided by KQCCircuits [156] with air-bridges designed by our parametric approach. [a] Original layout. [b] Separated air-bridges. [c] Full package air-bridges. [d] Foldable visualization layers enabled by our approach. (©2024 IEEE. Reprinted with permission from [31]).	84
4.18	Details of debugging layers. [a] point type layer, [b] arc layer, [c] center layer, and [d] channel layer. (©2024 IEEE. Reprinted with permission from [31]).	85
4.19	SEM images of the separated and full package air-bridges in the fabricated surface-13 quantum chips. (©2024 IEEE. Reprinted with permission from [31]).	87
5.1	An example of drawing a singing cat generated by Generative AI.	91
5.2	An illustration on knowledge distillation. (Work by the author)	91
5.3	A schematic diagram of a user-friendly AI development platform based on distillation. (Work by the author)	92
5.4	Aspects of responsible AI given in [164].	93
5.5	Framework of AI transparency hypotheses.	99
5.6	Gender distribution of survey respondents (N=206).	101
5.7	Age distribution of survey respondents (N=206).	101
5.8	Education distribution of survey respondents (N=206).	102
5.9	Occupation distribution of survey respondents (N=206).	102
5.10	AI exposure frequency of respondents (N=206).	103
5.11	Self-assessed knowledge towards AI of respondents (N=206).	103
5.12	AI knowledge coverage in respondents.	104
5.13	AI application exposure to respondents, where multiple choices are allowed. . . .	105
5.14	Correlations between the multi-item scale questions.	105
5.15	Pearson correlations between questions (N=206).	106

5.16	Multiple Linear Regression model summary for dependent variable y_{explain} and predictors $x_{\text{develop}}, x_{\text{reliability}}$	107
5.17	ANOVA results of Multiple Linear Regression model for dependent variable y_{explain} and predictors $x_{\text{develop}}, x_{\text{reliability}}$	108
5.18	Coefficients of Multiple Linear Regression model for dependent variable y_{explain} and predictors $x_{\text{develop}}, x_{\text{reliability}}$	108
5.19	Multiple Linear Regression model summary for dependent variable y_{open} and predictors $x_{\text{reuse}}, x_{\text{transfer}}$	108
5.20	ANOVA results of Multiple Linear Regression model for dependent variable y_{open} and predictors $x_{\text{reuse}}, x_{\text{transfer}}$	109
5.21	Coefficients of Multiple Linear Regression model for dependent variable y_{open} and predictors $x_{\text{reuse}}, x_{\text{transfer}}$	109
6.1	An illustration of homophilic and heterophilic graphs. [a] A homophilic social network, where blue edges represent friendship relations. In such networks, connected (adjacent) nodes tend to share similar attributes, reflecting the common assumption that “friends are alike.” [b] A heterophilic social network that encodes negative or conflicting relationships. Black edges denote strongly adversarial ties, while brown edges indicate past disputes. In contrast to panel [a], adjacent nodes in heterophilic graphs are more likely to be dissimilar. This highlights the fundamentally different structural assumptions underlying the two types of graphs.	114

List of Tables

2.1	List representation vs. matrix representation of a graph.	21
2.2	Details of some datasets in graph, images and text data.	22
2.3	Statistics of QM9 dataset used in experiments.	29
2.4	Results of GraphSAGE in learning QM9.	30
2.5	Properties used in the experiments.	31
2.6	Results for MU prediction.	31
2.7	Results for ALPHA prediction.	32
2.8	Results for HOMO prediction.	32
2.9	Results for LUMO prediction.	32
2.10	Results for GAP prediction.	32
2.11	Results for U0 prediction.	32
2.12	Results for CV prediction.	32
2.13	Results for memory used (Unit: Gb).	32
2.14	Results for time used (Unit: second).	32
2.15	Results for ALPHA prediction.	33
2.16	Results for LUMO prediction.	33
2.17	Results for U0 prediction.	33
2.18	Results for CV prediction.	33
3.1	Categorization of representative GEP methods by reasoning type.	41
3.2	Comparison of connectedness approximations.	51
3.3	Datasets used in the empirical study. (©2025 IEEE. Reprinted with permission from [30]).	53
3.4	Detail on the trained basic GCN. (©2025 IEEE. Reprinted with permission from [30]).	54
3.5	Comparison of different methods, where Acc , Pre , Re and F₁ denote the dataset-level metrics Accuracy, Precision, Recall and F ₁ , respectively, in %. (©2025 IEEE. Reprinted with permission from [30]).	55
3.6	Running time(s) comparison between CRoCEXplainer and Connected-CF ² on different datasets. (©2025 IEEE. Reprinted with permission from [30]).	56
4.1	Research problems addressed in this work. (©2024 IEEE. Reprinted with permission from [31]).	72

4.2	Details of open-source quantum chips from KQCCircuits [156]. Chips are sorted in order of the number of points on each chip. Chips 2 and 3 (with 0 qubits) contain only resonators and control lines. (©2024 IEEE. Reprinted with permission from [31]).	80
4.3	Performance metrics achieved by our approach for quantum chips in the dataset. n_p and n_c denote the number of points and number of centers in the corresponding layouts, respectively. (©2024 IEEE. Reprinted with permission from [31]).	83
4.4	Algorithm running time complexity. (©2024 IEEE. Reprinted with permission from [31]).	85
5.1	Geographical distribution of valid responses (by Country).	100
5.2	Mapping between questionnaire items and model variables.	105
5.3	Mean and standard deviation of results from Q20 (N=202).	110

Chapter 1

Introduction

1.1 What is Learning?

The core of this thesis lies in the concept of learning graphs. Before delving into this central topic, it is essential to first address a more fundamental question: *What is learning?* Clarifying this notion provides a foundation for understanding the essence and motivation of the present research.

Learning by human is a complex and multifaceted concept, defined and interpreted differently across disciplines such as behaviorism, constructivism, cognitivism, social constructivism, and neuroscience. For instance, the following are representative definitions of learning used in humanity [1].

- Learning is the process of acquiring knowledge or skill through study, experience, or training.
- Learning is experience that brings about a relatively permanent change in behavior.
- Learning is a change in neural function as a consequence of experience.
- Learning is the cognitive process of acquiring skill or knowledge.
- Learning is an increase in the amount of response rules and concepts in the memory of an intelligent system.

Zhao considers that learning is the process to find good information [2]. Similarly, according to Smith's definition [3], human learning is often not fully under our control and is wrapped up with the environments we inhabit and the relationships we make. It involves encountering signals from the senses; attending to them; looking for connections and meanings; and framing them so that we may act. This definition concludes four key factors of learning:

- (1) How to process signals (sensing).
- (2) How to react.
- (3) Looking for connections and meanings.
- (4) How to frame/conclude signals into future actions.

We illustrate the above process through an example how a child learns what an apple is (see Figure 1.1). When a child first encounters an apple, he has no prior knowledge of what it is. The child begins by processing sensory signals such as the shape, size, and color of the apple. Then, he may respond by touching it, smelling it, tasting it, or asking his parents about it. In seeking connections and meaning, the child discovers that it tastes sweet, and learns from others that it is a type of fruits called "apple," and that it is considered healthy. Later, when the child sees the fruit again or hears the word "apple,"

this prior experience allows him to associate the concept with the familiar red, sweet fruit—framing the experience into an actionable understanding. This sequence of perception, reaction, association, and conceptualization exemplifies the very process of learning.

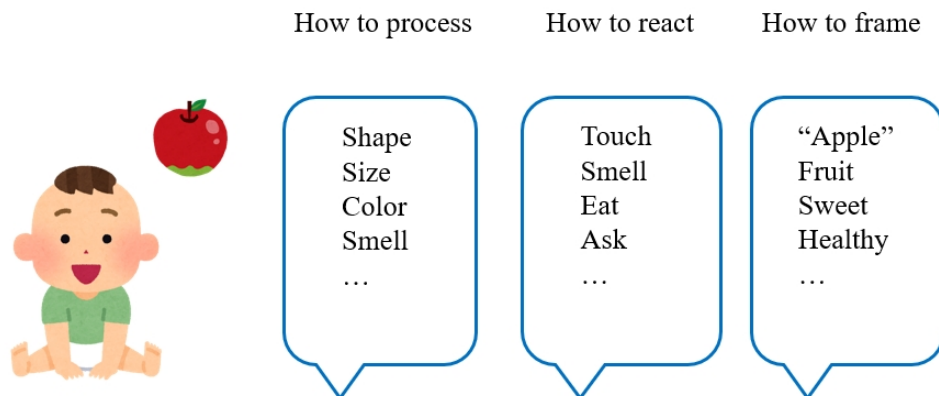


Figure 1.1: An illustration on learning. (Work by the author)

The story of the tomato’s spread in the U.S.A. provides another example demonstrating that learning—particularly collective learning—is a dynamic and evolving process. Tomatoes, once considered as “Poison Apples”, made many people afraid of them for more than 200 years. In 1820, Robert Gibbon Johnson publicly ate a basket of tomatoes before 2,000 people who believed it was poisonous. Johnson had proved that the “poison apple,” as the tomato was known at the time, was not poisonous after all [4]. And so, as the story goes, the tomato-growing industry in the United States was born.

This story has been told in numerous articles and books. There is no clear evidence showing it happened. However, this story defied prevailing misconceptions and symbolized the transformative power of individual learning in reshaping collective beliefs. Let us discuss it in more details.

The tomato’s reputation had long been shaped by early classifications such as that of the Italian herbalist Pietro Andrea Mattioli, who in 1544 described it as a nightshade and mandrake—plants associated with toxicity and sensuality [5]. This categorization led Europeans to regard the tomato with suspicion for centuries. Over time, however, through repeated individual experimentation and cultural adaptation, this perception gradually shifted.

Today, tomato has become the most widely consumed vegetable in the world [6]. Its journey from a feared “poison apple” to a global staple illustrates how individual acts of inquiry and reinterpretation can accumulate to transform collective understanding. This process exemplifies the inherently dynamic nature of learning, in which personal experiences continuously interact with and reshape shared knowledge, thereby driving the broader development of human societies.

Here, we have discussed what human learning is, the four core activities related to learning, and the significance of learning itself. This can be generalized to machine learning.

1.2 What is Machine Learning?

Machine learning (ML) can be understood as a way for computers to *learn from experience*, much like humans do. When a child repeatedly sees apples in different shapes and colors, he gradually learns to recognize an apple even in new situations. Machine learning wants to do the same: given samples, we want the machine to automatically detect patterns of the samples, and use those patterns to make predictions or decisions when new data is given.

Understanding how humans learn sheds light on why machine learning has become such a central tool in modern science and technology. Human learning is powerful—we observe, reason, and adapt based

on experiences. However, the scale and complexity of today's information far exceed what individuals or even groups of people can process reliably in a short time. Modern societies generate enormous amounts of data every second: medical records, sensor readings, financial transactions, social interactions, and images or videos captured by countless devices. Much of the data contains interesting patterns, but these patterns may be difficult for humans to understand or learn. Let us demonstrate it with an example.

In general, a machine learning task aims to learn a mapping from input data to output targets based on example pairs. Let

- $\mathcal{X}$ denote the *input space*,
- $\mathcal{Y}$ denote the *output space*, and
- $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N \subseteq \mathcal{X} \times \mathcal{Y}$ denote a training dataset of N examples.

A machine learning model is a parameterized function

$$f_{\theta} : \mathcal{X} \rightarrow \mathcal{Y},$$

where θ represents the set of **learnable parameters** (weights).

The goal of learning is to find a parameter value θ^* such that it minimizes the expected discrepancy between the model's predictions and the true outputs called **ground truth**. This is typically formulated as minimizing the empirical loss:

$$\theta^* = \arg \min_{\theta} \mathcal{L}(\mathcal{D}; \theta) = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \ell(f_{\theta}(x_i), y_i),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_{\geq 0}$ is a **loss function** that measures prediction error. Learning is performed by finding the parameters θ (e.g., using gradient-based optimization) so that the trained model generalizes well to unseen inputs from $\mathcal{X}$.

A simple machine learning method is **linear regression**, one of the earliest and most fundamental learning models. Linear regression tries to find a straight-line relation from the input data. For instance, suppose we want to predict a person's weight from his height. We first collect a dataset of a number of individuals, each with their height and weight. This is the sensing process. Next, let us assume

$$\text{weight} \approx a \cdot \text{height}^b$$

or equivalently

$$\log(\text{weight}) \approx b \cdot \log(\text{height}) + \log(a),$$

for some constants $a, b > 0$.

At this point, the learning process begins. We treat the true weights in the dataset as the ground truth, denoted as y_i , and use the model to generate predictions $\hat{y}_i$ from the corresponding heights. A commonly used loss function is the **mean squared error (MSE)**:

$$\mathcal{L}(a, b) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2.$$

The goal of learning is to find the parameters a and b that **minimize** this loss function—that is, achieve the best fit between the predicted values and the observed reality. It is well-known that the **Body Mass Index (BMI)** $\frac{\text{weight}}{\text{height}^2}$ was decided in this way because $b \approx 2$ (see [7]).

While linear regression is powerful for capturing simple linear relationships, many real-world problems require learning more complex, nonlinear patterns. A foundational model that extends this capability is the **perceptron**, later generalized to the **multi-layer perceptron (MLP)**. The perceptron takes an input

vector $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, computes a weighted sum including a bias parameter $b \in \mathbb{R}$, and passes it through a nonlinear activation function $\sigma(\cdot) : \mathbb{R} \rightarrow \mathbb{R}$, i.e.,

$$y = \sigma \left(\sum_{i=1}^n w_i x_i + b \right),$$

where w_i and b are again the **parameters to be learned**. The activation function σ is critical in enabling the model to express nonlinear relationships. A widely used activation is the **Rectified Linear Unit (ReLU)**, defined as $\sigma(z) = \max(0, z)$, which filters out negative values and accelerates training. A variation **LeakyReLU** allows a small slope for negative inputs to mitigate the inactive neurons:

$$\sigma(z) = \begin{cases} z, & z \geq 0, \\ \alpha z, & z < 0, \end{cases}$$

for a small constant $\alpha > 0$. When multiple perceptrons are stacked into **layers**, they form an MLP capable of approximating highly complex functions.

For example, if we want to predict a student's academic performance based on multiple features such as *study hours*, *sleep time*, and *course difficulty*, an MLP can model nonlinear effects such as the diminishing return on study time or thresholds where insufficient sleep sharply degrades performance. Mathematically, the MLP learns a function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$ such that

$$\text{score} \approx f(\text{study hours}, \text{sleep time}, \text{difficulty}),$$

where $f(\cdot)$ is represented by compositions of perceptrons and activation functions, with a **matrix of learnable parameters** optimized during training. Using the MLP illustrated in Figure 1.2, let the input be represented as a 3-dimensional vector

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{R}^3,$$

where x_1 , x_2 , and x_3 correspond to *study hours*, *sleep time*, and *course difficulty*, respectively. We construct a three-layer MLP with one hidden layer of dimension 4. Let

$$\mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \in \mathbb{R}^4$$

be the bias of the hidden layer, and $c \in \mathbb{R}$ the bias of the output layer. Then the hidden-layer representation and the final prediction are given by

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = \sigma(W_1 \mathbf{x} + \mathbf{b}), \quad \hat{y} = W_2^\top \mathbf{h} + c,$$

where $W_1 \in \mathbb{R}^{4 \times 3}$ and $W_2 \in \mathbb{R}^4$ are learnable weight parameters, $\mathbf{b}$ and c are learnable bias parameters, h_1, h_2, h_3 and h_4 are latent variables, and $\sigma(\cdot)$ is a nonlinear activation function such as ReLU. Overall, the MLP learns a function

$$\hat{y} = f(\mathbf{x}) = W_2^\top \sigma(W_1 \mathbf{x} + \mathbf{b}) + c.$$

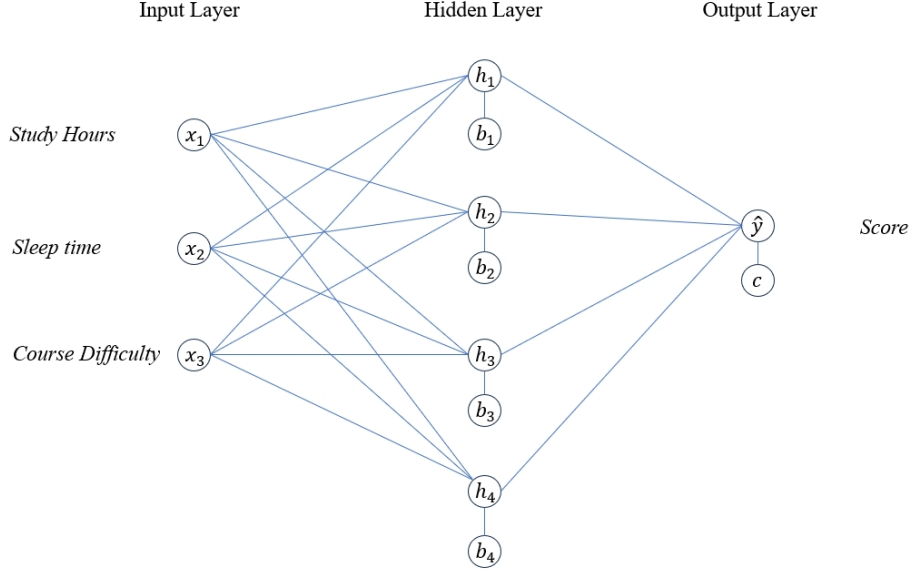


Figure 1.2: An illustration on a schematic architecture of a MLP consisting of three layers. (Work by the author)

During training, the predicted score $\hat{y}_i$ is compared with the ground-truth score y_i using a loss function such as the mean squared error,

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2,$$

and the parameters W_1 , W_2 , $\mathbf{b}$, and c can be optimized via gradient descent [8] to minimize $\mathcal{L}$.

A more advanced example of machine learning is image recognition, which closely parallels how a child learns to recognize different fruits. In this scenario, we want to learn from a dataset of fruit images. Instead of simple numerical inputs such as height and weight, each image is represented as a two-dimensional grid of pixel values. Similar to an MLP, where information is passed layer by layer through learnable weight matrices and activation functions, image recognition commonly employs **convolutional neural networks (CNNs)** [9]. A convolutional layer can be viewed as a specialized extension of the perceptron/MLP concept: rather than connecting every input pixel to every neuron, convolution applies small learnable filters that slide across the image to detect meaningful local patterns such as edges, colors, and shapes. This significantly reduces the number of parameters while preserving spatial structure. For brevity, the detailed mathematical expressions are not presented here.

When provided with a collection of labeled fruit images, the machine first processes each image using convolutional layers and activation functions such as ReLU, then learns hierarchical visual representations that capture increasingly complex features. After training, the model can function as a **fruit classifier**, capable of identifying different fruits in images, even if it has never seen before. Here, the parameters to be found—i.e., learned—are represented by a **matrix of learnable filters and weights**.

Mathematically, a convolutional neural network learns a mapping from an image input $\mathbf{X}$, represented as a two-dimensional pixel matrix, to a fruit category among possible classes (e.g., five classes of apple, pear, banana, watermelon, strawberry, and kiwi):

$$\text{fruit} \approx f(\mathbf{X}),$$

where $f(\cdot)$ is composed of stacked convolutional layers and activation functions. During training, the learnable filters and weight matrices are optimized to minimize the classification error across all labeled fruit images (see Figure 1.3).

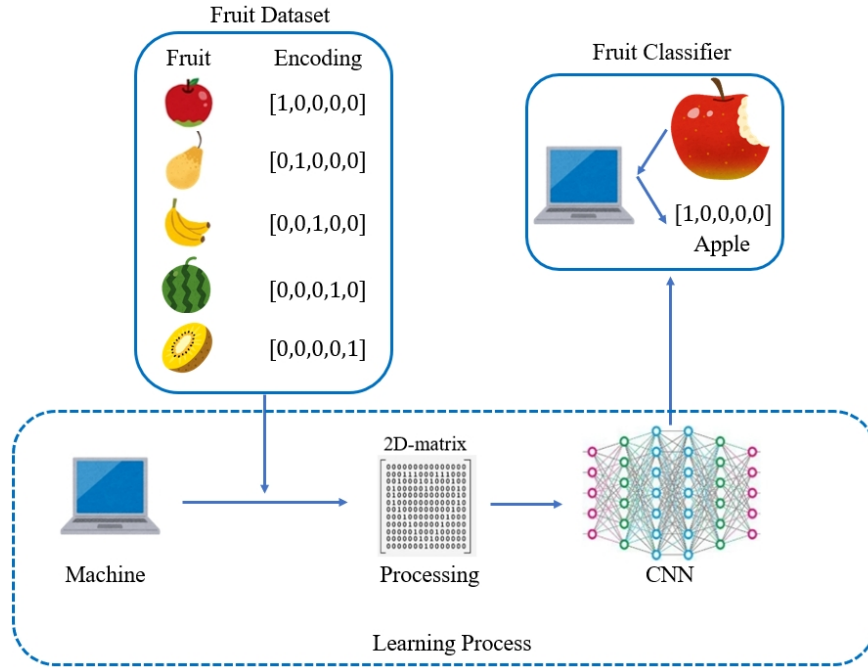


Figure 1.3: Illustration of machine learning for image-based fruit recognition. (Work by the author)

In summary, machine learning can be summarized as a mathematical process that first, assuming a relation $y = f_{\theta}(x)$ between an observed variable x and a property y with parameter θ , tries to find an optimal θ such that f_{θ} explains the known dataset of x and y , i.e., $\hat{y} = f_{\theta}(x) \approx y$, then uses f_{θ} to predict y for unknown x .

Machine learning has achieved remarkable success across an exceptionally wide range of real-world applications. Everyday technologies such as facial recognition on smartphones, recommendation systems on streaming platforms, and voice assistants like Siri are powered by models that learn from vast amounts of data. These achievements demonstrate the transformative impact of machine learning: by uncovering patterns too complex or subtle for manual analysis, it has become an indispensable tool for solving problems at scale, improving efficiency, and enabling new forms of automation and intelligence.

Although many classical examples of machine learning involve numerical data (such as heights and weights) or grid-structured data (such as images), modern data increasingly comes in the form of **relations** and **interactions**. Social networks connect people, transportation systems link locations, molecules consist of atoms bonded together, and knowledge bases encode relations between concepts. These structures usually are difficult to model as simple tables or images—they are best described as graphs, where nodes represent entities and edges capture their connections.

Next, we focus on graphs—the central research subject of this thesis—and then examine how graphs can be learned, classic graph problems along with the development of learning methodologies on graphs.

1.3 Learning Graphs

1.3.1 What is Graph?

The word “graph” dates back to the emergence of the **Seven Bridges of Königsberg Problem**, a historically notable problem in mathematics. According to the description of Wikipedia, “The city of Königsberg in Prussia (now Kaliningrad, Russia) was set on both sides of the Pregel River, and included two large islands—Kneiphof and Lomse—which were connected to each other, and to the two mainland portions of the city—Altstadt and Vorstadt—by seven bridges. The problem was to devise a walk through

the city that would cross each of those bridges once and only once” [10]. Figure 1.4 illustrates the problem in detail: [a] shows the actual layout of the seven bridges in Königsberg provided by https://commons.wikimedia.org/wiki/File:Bridges_of_Konigsberg.png. [b] represents an abstract representation of [a] provided by https://commons.wikimedia.org/wiki/File:7_bridges.svg. [c] depicts a graph derived from [b], where lands and islands are modeled by nodes and bridges by edges, adjusted from https://commons.wikimedia.org/wiki/File:K%C3%B6nigsberg_graph.svg.

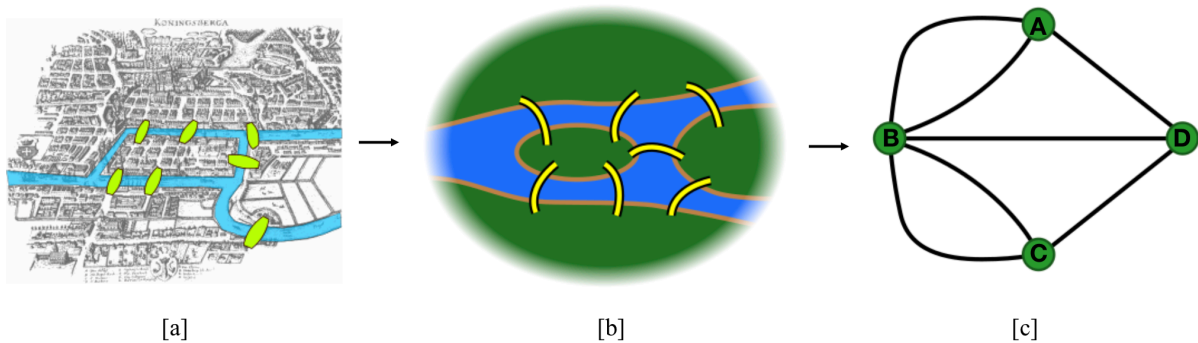


Figure 1.4: An illustration of the Königsberg Problem. (Source: [10])

To address this problem, how would a reader approach? A naive instinct is to draw various possible routes in Figure 1.4 [a], in an attempt to identify a feasible walk. This was what the tourists visiting Königsberg did at that time. This approach resembles the way one might navigate a maze. Such behavior can hardly be regarded as learning, as it merely involves trial and error without uncovering underlying principles or developing a framework for future reasoning.

Let’s discuss how Leonhard Euler, one of the greatest mathematicians in history, learned and solved this problem. Euler noticed that a solution, if exists, does not depend on the length of bridge and distance between islands. He reformulated the problem in abstract terms, consisting of land masses and the bridges connecting them (see Figure 1.4 [b]). And he pointed out that the choice of route inside each land mass is insignificant and the only important thing of a route here is the sequence of bridges crossed. In modern terms, one replaces each land mass with an abstract “vertex” or node, and each bridge with an abstract connection, an “edge,” which only serves to record a pair of vertices (land masses) is connected by that bridge. The resulting mathematical structure is a **graph** (see Figure 1.4 [c]).

He then made a crucial observation in this context: If such a walk exists, then, except for the starting and ending points of the walk, every time one enters a vertex (via one bridge), one must also leave it (via another). In other words, the times one enters a non-terminal vertex must equal the times one leaves it [11]. Notably, the number of edges incident to a vertex is called the degree of a node. Therefore, we learned that is an important **feature** of graph. In the following thesis, more features of graphs will be discussed and explored in learning graphs from different perspectives.

In honor of Euler, an Euler path is a walk through a graph that traverses each edge exactly once, while an Euler tour is an Euler path that begins and ends at the same vertex. For a graph to contain an Euler tour, every vertex must have an even degree (an even number of incident edges); for a graph to contain an Euler path but not an Euler tour, exactly two vertices must have odd degrees.

Applying this reasoning to the Königsberg bridge problem, each of the four land masses (vertices) is connected by an odd number of bridges—one by five bridges and the remaining three by three bridges each. Since all four vertices have odd degrees, and at most two could serve as the endpoints of an Euler path, it follows that no Euler path or Euler tour exists in this graph. Hence, it is impossible to devise a walk that crosses each bridge exactly once.

Reflecting on the process of Euler’s learning about graphs, he first identified the essential elements of the problem—abstracting it into a graph structure—to process the outside signals. He then uncovered the

underlying connections—the intrinsic properties of graphs to demonstrate the impossibility of a feasible solution. In doing so, Euler not only resolved the specific problem but also established the conceptual foundations for subsequent research on graph-related problems. This can be considered an example how human learns properties and structures from graphs. Building upon this, we next introduce other examples of classic graph problems and methodologies for learning graphs.

1.3.2 Learning from Classic Graph Problems

Before discussing machine learning for graphs, it is useful to review several classical graph problems that illustrate both the expressive power of graphs and the computational challenges they present. Here, we introduce two classic graph problems, **Traveling Salesman Problem (TSP)** and **Graph Partitioning Problem**, along with their corresponding approaches. These problems not only demonstrate the diversity of graph-based tasks but also correspond to real-world applications such as logistics optimization and functional-group detection in networks ¹.

Travelling Salesman Problem (TSP)

The Traveling Salesman Problem is one of the most well-known combinatorial optimization problems regarding graphs. Given a set of cities (vertices or nodes) connected by routes (links or edges), the goal is to find a shortest tour that visits each city exactly once and returns to the starting point (see a showcase in Figure 1.5). TSP appears in various logistics and routing applications, such as delivery planning, circuit board design, and robot path optimization.

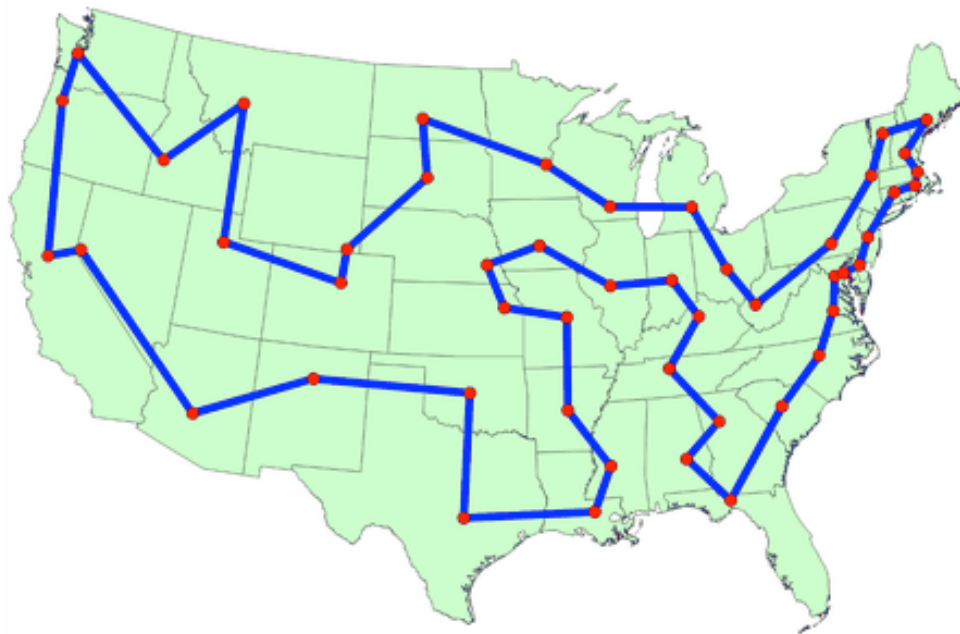


Figure 1.5: A showcase of the solution to 48 States in America Traveling Salesman Problem, provided by Cornell University Computational Optimization Open Textbook - Optimization Wiki from <https://optimization.cbe.cornell.edu/index.php?title=File:48StatesTSP.png>.

Because TSP is NP-hard ², solving it is become computationally expensive as the number of cities grows [12]. Classical approaches to TSP include:

¹Generally, we say a network is a graph with real-world meaning attached. A graph becomes a network when its nodes and edges represent something meaningful, not just abstract elements.

²In computational complexity theory, NP problems form an important class of decision problems whose solutions can be verified in polynomial time, although no polynomial-time algorithm is currently known to solve them. NP-hard refers to opti-

- Exact algorithms ³: They include naive enumeration method, e.g., dynamic programming (Held-Karp algorithm) [13] and branch-and-bound methods [14]. These guarantee optimal solutions but scale poorly.
- Heuristics ⁴: They include nearest-neighbor heuristics [15, 16], genetic algorithms [17, 18, 19], etc. These approaches find good (but not necessarily optimal) solutions efficiently for large cases.
- Approximation algorithms ⁵: They provide theoretical guarantees on how close the solution is to optimal. They may be for certain special cases (e.g., metric TSP [20]).

These traditional methods have been extensively studied and form the foundation of many modern optimization systems. One clear success is in solving route planning problems—for example, how delivery companies like **FedEx**, **UPS**, or **Amazon** figure out the most efficient way for their trucks to deliver hundreds of packages in a day.

Graph Partitioning Problem

The graph partitioning problem aims to divide a graph into several groups (or clusters) such that the number of edges crossing between different groups is minimized, while keeping group sizes balanced. This problem arises in applications such as **detecting functional groups** in biological networks, **clustering communities** in social networks, **parallel computing load balancing**, and **identifying modular structures** in molecular or chemical systems, and circuit board design.

Classical approaches include greedy algorithms [21], flow-based algorithms [22], etc. These methods are efficient and interpretable but often rely on hand-crafted heuristics and problem-specific design choices. In the next subsection, we will see how ML methods can help us to do the task automatically.

1.3.3 Machine Learning for Graphs

While classical algorithms and heuristics have achieved substantial success for TSP, graph partitioning, and many other graph problems, they fundamentally rely on the knowledge and insight of researchers or experts. Solving these problems requires human to carefully observe the structure of the graph, understand the underlying patterns, and design problem-specific rules or strategies. The computer, in this context, primarily serves as **a tool to implement and execute** the solutions devised by human. Even the most sophisticated heuristics or exact algorithms are ultimately shaped by human reasoning and expertise.

Machine learning–based approaches, in contrast, offer a fundamentally different perspective. Instead of requiring manually crafted rules, ML models can **learn patterns and strategies directly from known data**, discovering solutions that may be difficult for humans to anticipate. This approach can improve scalability, adaptability, and performance across diverse and complex graph structures. In the next section, we will explore how machine learning for graphs differs from these traditional methods and examine the advantages and limitations of each paradigm.

In Section 1.2, we discussed what machine learning is and provided basic examples using numeric data and images. However, machine learning on graphs is fundamentally different from these examples.

mization problems whose decision variants (e.g., is there a tour with length shorter than 10 km?) are at least as difficult as the hardest NP problems.

³An exact algorithm is a computational method that always finds the optimal solution to a problem. For example, in the Travelling Salesman Problem, an exact algorithm would find the shortest possible tour, although it may require very long computation time for large instances.

⁴A heuristic is a practical approach designed to find a good or acceptable solution quickly, without guaranteeing optimality. Heuristics are often used for very complex problems where exact algorithms are too slow, such as routing or scheduling large networks.

⁵An approximation algorithm is an algorithm that efficiently finds a solution close to the optimal one, with a provable bound on how close it can be from the optimum. This provides a balance between solution quality and computational efficiency for hard problems.

Unlike numeric tables or images, which can be easily represented as vectors or matrices, graphs are more complex structures and are not straightforward to encode.

Moreover, problems on graphs seeking to infer unknown or future information about graph elements are more various and complicated. These can be classified at multiple levels:

- **Node-level prediction:** Examples include recommending a product to a user in a user–item graph or predicting a protein’s function in a biological interaction network.
- **Edge-level prediction:** Examples include forecasting whether two users will form a friendship (link prediction).
- **Graph-level prediction:** Examples include determining whether a molecule will be soluble or toxic based on its entire structural graph.

Traditionally, machine learning for graphs heavily relied on carefully designing features from the graph—for example, computing *node degrees*, *centralities*, or *other handcrafted measures*⁶—and then representing these features in vector form so that standard ML algorithms could be applied.

More recently, a new approach called graph learning has emerged. With no handcrafted features, graph learning methods leverage the graph’s structural information directly, often using a process known as message passing [23] to propagate information along edges between nodes. More details about these methods and their principles will be discussed in Chapter 2.

1.4 Success and Issues Regarding Learning Graphs

Learning graphs has achieved several noticeable successes in real-world applications—some of which we might already encounter in daily life without realizing it.

1.4.1 Success of Learning Graphs

Smart Route Planning and Optimization

One clear success is—for example, how delivery companies like **FedEx**, **UPS**, or **Amazon** [24] figure out the most efficient way for their trucks to deliver hundreds of packages in a day. Instead of relying only on traditional mathematical formulas, modern systems now use graph learning to “learn” from past delivery patterns, traffic data, and road maps. Each city or delivery stop is treated as a **node**, and each road between them as a **connection**. With this approach, algorithms can learn to make better predictions about which routes save the most time or fuel. Companies like **Google Maps** and **Amazon Logistics** have explored such methods to make route planning faster and smarter.

Recommendation Systems

Another area where graph learning shines is **recommendation systems**, the kind that suggest what to watch, buy, or read next. Think about **Netflix** recommending a new show, **Spotify** suggesting songs, or **Amazon** showing products you might like [25]. Behind the scenes, these systems treat users and items as nodes in a big network—for example, you’re connected to the movies you’ve watched or the books you’ve purchased, and to other people with similar tastes. Graph learning helps uncover patterns within this web of connections.

For instance, if many people who liked *The Lord of the Rings* also liked *Game of Thrones*, the system can recommend *Game of Thrones* to someone who just watched *The Lord of the Rings*. What’s powerful here is that the graph model doesn’t just look at direct connections; it can trace through multiple layers

⁶These are features of graphs, which will be further explained in Chapter 2.

of relationships—such as “friends of friends” or “people who like what people like you like.” This helps create highly personal and accurate recommendations.

Discovering New Materials and Medicines

A more scientific but equally practical success is in **chemistry and drug discovery**. Molecules can be represented as graphs—atoms are nodes, and chemical bonds are edges. Traditionally, chemists described molecules with fixed “fingerprints,” but graph learning allows computers to learn these fingerprints automatically. This helps scientists predict how a molecule might behave—whether it could become a good medicine, or if it might be toxic—much faster than traditional experiments. For example, companies like **DeepMind** use machine learning to speed up the search for new drugs or materials [26]. During the COVID-19 pandemic, such models were used to screen large chemical databases for molecules that might bind effectively to viral proteins [27]. This kind of learning helps researchers test millions of possibilities in days rather than months, saving both time and cost.

1.4.2 Issues and Challenges in Learning Graphs

While learning graphs has achieved various and great success, it also faces practical problems that affect how widely and safely it can be used. In this thesis, I organize these difficulties into three main themes, which correspond to Chapters 3, 4, and 5: (1) the inner black-box nature and lack of explainability, (2) the high computational cost and scalability issues with the potential for quantum enhancement, and (3) ethical concerns. Let us see them briefly.

Explainability and the Black-Box Nature of Learning Models

Although machine learning systems have achieved remarkable success, many of them still operate as **black boxes**: they make predictions, but we often do not know *why*. This becomes a serious issue in important areas such as **drug discovery**, **medical diagnosis**, **material design**, and **financial decision-making**, where the reasoning behind a prediction matters just as much as the prediction itself. For example, if a model suggests a new chemical compound as a potential drug, researchers must understand *which part* of the molecule is important for its effect or side-effect. Without explanation, it is difficult to trust the model or use its output safely.

This lack of interpretability has motivated a growing field of research that aims to make complex models more transparent even if we do not know how it was trained. These approaches attempt to identify which input features, regions, or patterns are most responsible for a model’s decision, and to ensure that the reasoning aligns with scientific or ethical expectations.

Chapter 3 will introduce this challenge in detail. It will illustrate why explainability matters, what kinds of risks arise when models behave as black boxes, and how current methods attempt to open this black box. Moreover, we will introduce our contributions in this field, by bringing in a new quantitative metric of connectedness of the graph, which has been long overlooked in previous researches.

Computational Limits and the Search for Better Efficiency

Modern machine learning models require enormous computational resources. As datasets grow larger and models become deeper, the cost of training and running these systems increases rapidly. For example, large social networks, transportation networks, and molecular datasets can contain millions of elements, making it extremely demanding for classical computers to process them efficiently.

This scalability issue has become a bottleneck for further progress. In recent years, researchers have explored various paths to overcome it. One promising direction is quantum computing, which offers fundamentally different ways of performing calculations. In theory, quantum algorithms can solve certain problems much faster than classical ones. For example, Shor’s algorithm [28] shows that quantum

computers could factor large numbers exponentially faster, and similar ideas might eventually accelerate optimization or search tasks in machine learning.

However, today’s quantum devices are still in limited conditions. They suffer from noise, short coherence times, and high error rates, which means that the powerful algorithms predicted in theory cannot yet be fully implemented. To bridge this gap, significant engineering work is needed to improve quantum hardware and reduce errors—an effort often referred to as quantum electronic design automation (quantum EDA).

Chapter 4 will introduce basic quantum computing preliminaries, connection between graph and quantum systems, quantum enhancement on graph learning, and potentials and limitations of existing quantum devices. We will introduce our contribution to quantum EDA, which automates the air-bridge construction in superconducting quantum device layouts to effectively and efficiently suppress parasitic mode in circuits. This task is closely related to learning graphs and represents a meaningful step toward enabling high-quality quantum devices capable of supporting practical quantum computation.

Ethical Challenges and the Responsibility Crisis in Machine Learning

Today’s AI is mostly based on machine learning. While technical progress continues rapidly, ethical and social considerations often lag behind. This imbalance has created what many researchers call an “Artificial Intelligence (AI) responsibility gap”—a situation in which AI systems make or influence decisions, but it is unclear who is accountable when something goes wrong.

For example, some unexpected situations may happen:

- A recommendation system might unintentionally amplify misinformation.
- A credit scoring model might produce biased decisions for certain groups.
- A medical AI system might make a harmful suggestion due to biased training data.
- Some have used open-source AI models to conduct financial fraud, causing significant losses.

In many such cases, responsibility is difficult to assign. Is the problem caused by the data, the developers, the users, or the system itself? Who is supposed to be responsible? The lack of clear guidelines makes it difficult to regulate or safely deploy AI in sensitive environments.

These concerns have led to increasing interest in **AI ethics**, including fairness, transparency, privacy protection, and governance frameworks. As AI becomes more widely used, it is important to identify potential risks early and ensure that these systems serve society in a safe and responsible manner.

Chapter 5 reviews current risks, challenges, existing principles, and regulations from learning on graphs to general AI. Moreover, based on my works during the overseas knight internship (mushashugyo in the program of GSAIS) and project-based research (PBR, respectively), we proposed our own hypothesis regarding AI transparency and designed a questionnaire to collect and analyze relevant data. The results are thoroughly examined and presented in Chapter 5, providing valuable quantitative evidence and contributions to an area where empirical research on the ethics of AI transparency has been largely lacking.

1.5 Overview of this Thesis

Chapter 1 has presented the motivation, background, and general introduction of this thesis. The remainder of the thesis is organized as follows.

Chapter 2 lays out the mathematical foundations of graphs, traditional machine learning on graphs, and modern graph learning methods. This chapter establishes the essential concepts needed for the subsequent chapters. We begin by examining the structure of graphs and explaining why their characteristics

pose unique challenges for standard machine learning techniques. We then review how traditional machine learning processes graph data, with particular attention to methods for extracting graph features and converting them into vectorized representations. After highlighting the limitations of these approaches, we introduce graph learning, a more recent paradigm that leverages the intrinsic structural information of graphs rather than relying on handcrafted features. To illustrate the differences between the two approaches, we additionally conduct an original comparative experiment that demonstrates their respective strengths and weaknesses.

Chapter 3 addresses the challenge of explainability in graph learning, focusing on the black-box nature of modern graph models. We begin with a review of existing explainability approaches and reflect on their underlying inductive reasoning principles. We then identify a key limitation in current literature: the insufficient treatment of a fundamental graph property—connectedness—within explainer frameworks. To address this gap, we propose two novel methods, Connected-CF2 and CRoCExplainer, which explicitly incorporate connectedness into the explanation process. Part of the works have been published as [29] and [30]. Through theoretical analysis and empirical evaluation, we show that these models provide a complementary perspective and enrich the ongoing research on explainability in graph learning.

Chapter 4 explores the potential of quantum computing for learning on graphs. We begin by introducing the fundamental concepts of quantum computation that are relevant for understanding quantum machine learning. Following this, we review existing connections between graph structures and quantum models, highlighting quantum neural networks as a representative example of this intersection. We then examine the current bottlenecks in quantum hardware, which significantly constrain the practical performance of quantum machine learning systems. These limitations motivate the pursuit of improvements at the hardware and design level. Building on this motivation, we present a case study in which graph-based techniques are applied to quantum electronic design automation (quantum EDA), demonstrating how structural insights can enhance quantum device development [31]. This case study illustrates the substantial untapped potential in this area and suggests promising directions for future research.

Chapter 5 examines the broader risks, challenges, and regulatory considerations of AI, extending the discussion from graph-based learning to general AI systems. We begin by reviewing major ethical concerns and existing governance frameworks, focusing on issues such as transparency, accountability, and public trust. Building on these foundations, we introduce our own hypothesis concerning AI transparency and develop a questionnaire-based study to gather empirical data on public perceptions and expectations. The collected responses are analyzed in detail, providing quantitative insights into an area where empirical research remains limited. The findings presented in this chapter offer meaningful evidence and contribute to the growing discourse on the ethical implications of AI transparency.

Finally, **Chapter 6** concludes the thesis, summarizing key findings and highlighting potential directions for future research. Figure 1.6 visualizes the structure overview.

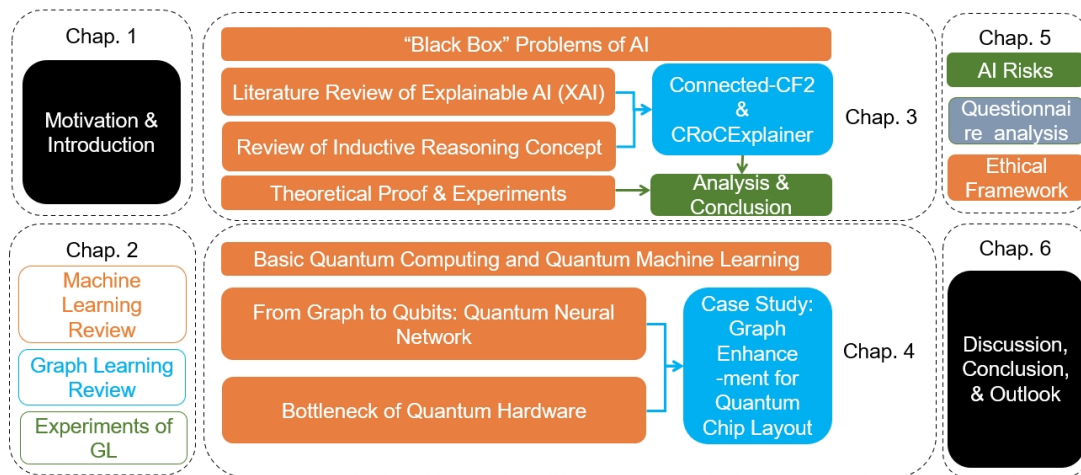


Figure 1.6: Structure overview of this thesis.

Chapter 2

Preliminaries

In Chapter 1, we provided a brief introduction to graphs and the learning methodologies associated with them. In this chapter, we begin by presenting the formal mathematical definitions of graphs. We then illustrate these concepts through a series of examples, highlighting the distinctive properties of graph structures and explaining why they are well suited for representing complex information in modern applications. This discussion establishes the foundational concepts required for the chapters that follow. At the same time, we also examine the difficulties and computational challenges that arise precisely because of the unique characteristics of graph data.

Subsequently, we discuss how traditional machine learning methods operate with graphs. The focus here is on understanding how classical approaches process graph inputs, especially the techniques used to extract structural features and transform them into vectorized representations suitable for learning algorithms.

Traditional machine learning on graphs also suffers from its limitation, i.e., highly dependent of feature-extraction design and technique. To handle this, we introduce graph learning, a more recent paradigm that leverages the intrinsic structural information of graphs rather than relying on handcrafted features. To illustrate the differences between the two approaches, we additionally conduct a comparative experiment that demonstrates their respective strengths and weaknesses.

2.1 Graph and Graph-like Data

2.1.1 Definition and Examples

A graph (see Figure 2.1 for an illustration) is a mathematical structure used to represent a set of objects and the connections between them:

- The objects are called vertices (or nodes); and
- The connections between them are called edges (or links).

Mathematically, we define a graph as $G = (V, E)$, where V is the set of vertices representing objects, and E is the set of edges representing connections between pair of vertices.

Figure 1.4[c] in the Seven Bridges Problems (introduced in Chapter 1) can be seen as a simple graph, where lands (or islands) are represented by vertices and bridges are represented by edges. Moreover, graphs can be extended to represent various objects. For example, graph can be employed to represent molecules (see Figure 2.2 for an illustration), where atoms are expressed as nodes and chemical bonds are modeled as edges.

Note that the edges in these two cases (Figure 1.4[c] and Figure 2.2) only represent the connection between two nodes, i.e., the edge does not contain a direction from one node to the other. These graphs are called **undirected graph**. Formally, an **undirected graph** is a graph in which the direction of the

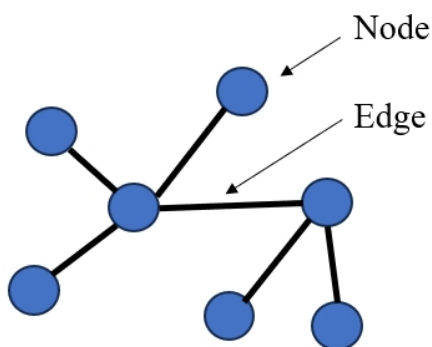


Figure 2.1: A graph example with 7 nodes and 6 edges. (Work by the author)

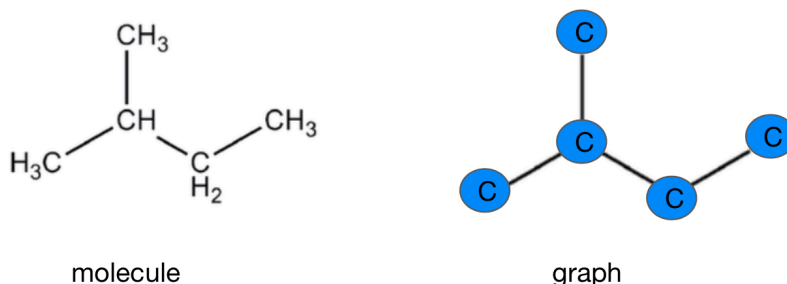


Figure 2.2: A molecule and the corresponding chemical graph, where hydrogens are omitted. (Work by the author)

edges is not defined. So if an edge exists between node u and v , then there is a path from node u to v and vice versa.

In contrast, Figure 2.3 gives an example that edges have directions. Figure 2.3 shows a simple social network, where each node represents a person's social account (e.g., X, etc.), and the directed edges depict a “follower” relationship. For example, “Kathy” follows “James” and “James” follows “Steven.” Unlike undirected connections, this type of relationship is **inherently one-way**: the fact that Kathy follows James does not imply that James follows Kathy. In other words, the relationship lacks symmetry, and the direction of the edge carries essential information about how users interact within the network. We denote this graph as **directed graph**. **Directed graph** is a graph connected by directed edges, where each edge has a specific direction from one node to the other node.

In the examples above, edges simply represent the existence of some connection. But are all connections the same? In many real-world situations, different connections may carry different strengths or properties. For instance, in a transportation network, two locations may be connected by a road, but this connection also has a **distance** associated with it. Such additional information cannot be ignored, as distance fundamentally influences how the network is used.

To represent this kind of information, we use a **weighted graph**. In a weighted graph, each edge is associated a numerical value—called a *weight*—which typically reflects the cost, distance, strength, or capacity of the connection. These weights allow the graph to encode richer relationships. For example, in a transportation network, shorter roads may have smaller weights, while longer or slower routes have larger weights (see Figure 2.4). Weighted graphs therefore provide a more expressive framework for modeling systems in which not all connections are equal.

Note that all above examples except Figure 2.3 are **connected graphs**, in which there exists a path between every pair of vertices. It is obvious that there are some graphs are not connected. For example,

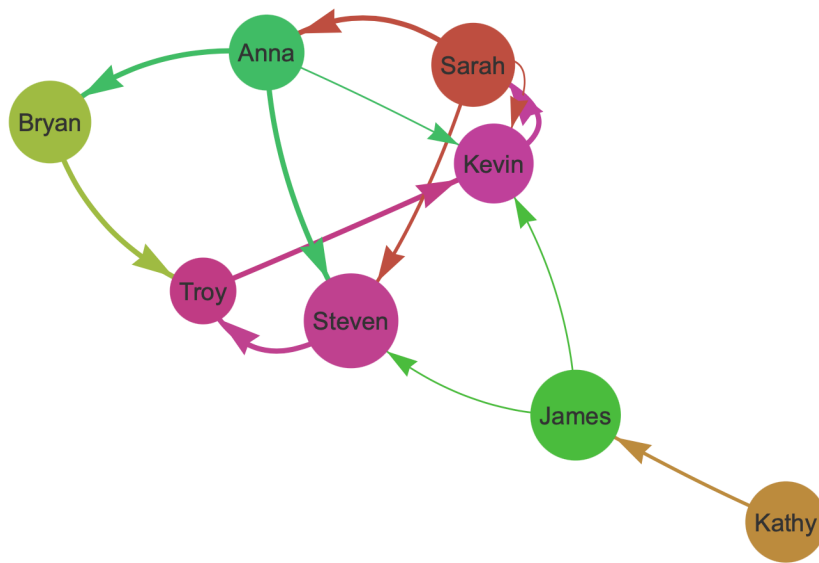


Figure 2.3: An illustration of a social network, where nodes represent social accounts and edges model “follower” relationship. (Work by the author)

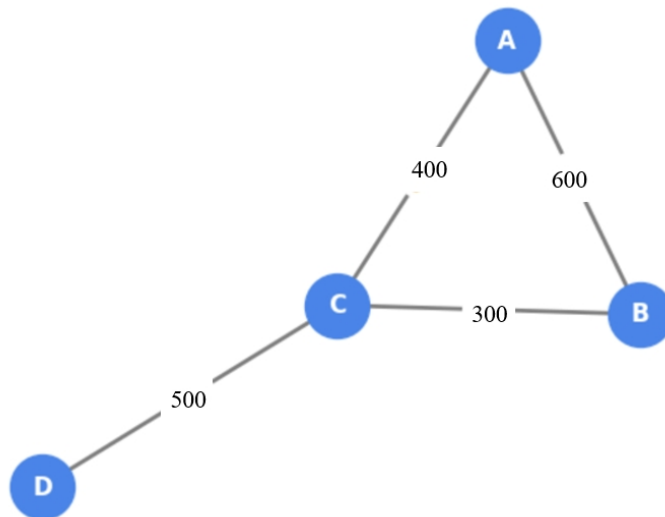


Figure 2.4: An abstract transportation network where edge weight represents the distance. (Work by the author)

in Figure 2.4, if there were an accident that destroy the road between C and D, it makes D disconnected to the other part of the graph. A (connected) component of G is a maximal connected subgraph of G . The connectedness of graph is also an important property of graph, which will be further explored in Chapter 3.

Transportation networks, social networks, etc., are graphs in which connected nodes tend to have similar features, labels, or properties. For instance, the “friend” relationship in a social network usually indicates they share some interests or hobbies. We generally denote this graph **homophilic graph**. In contrast, if a graph’s connected nodes tend to have dissimilar features or labels, it is called a **heterophilic graph**. A classic example is fraud detection networks (see Figure 2.5). Fraudulent users often connect to legitimate users to camouflage themselves. Thus, edges often connect fraudulent and non-fraudulent accounts.

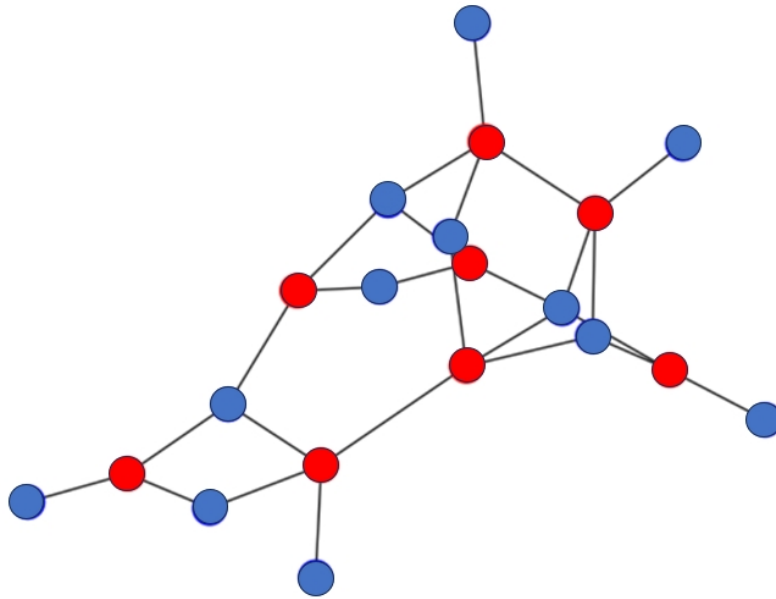


Figure 2.5: An illustration of a heterophilic fraud detection graph, where blue nodes denote normal users while red nodes denote deceivers. (Work by the author)

The aforementioned examples are from transportation networks, social networks and chemical graphs. The capacity of graphs to simulate multiple, various and cross-domain scenarios inherently stems from the universal abstraction of vertices and edges, which can model diverse relationships. Vertices can represent entities like social accounts, atoms, websites, etc, while edges can be defined flexibly by social connections, chemical bonds, hyperlinks accordingly. This “entity-relation” duality enables graphs to fundamentally adapt to different real-world scenarios. Currently, graphs have become indispensable in computer science, data analysis, biology, social sciences, etc.

On the other hand, the wide range of use cases and diverse categorization schemes also highlight the inherent complexity of graph-structured data. In real-world scenarios, a network often requires careful construction and may contain a large amount of intricate information, potentially fitting into multiple graph categories at the same time. This further illustrates the challenges involved in building and processing graph data, and explains why applying machine learning to graphs presents difficulties that are quite different from those encountered with traditional numerical or image-based data.

2.1.2 Graph Representation

In Subsection 2.1.1, we have introduced graphs with multiple instances and provided various examples across different real-world scenarios to illustrate the taxonomy of graphs, highlighting their multiplicity and complexity. In this subsection, we will delve into more detailed notations, preliminaries, and representations of graphs. Specifically, we will explore how graphs are represented, modeled, and stored in computers. Additionally, we will also mention the challenges associated with graph data in the field of computer science.

Graph Notations and Preliminaries

We review the fundamental concepts and notations used throughout the thesis. Unless otherwise stated, all definitions are restricted to *simple undirected graphs*, meaning that (i) each pair of nodes is connected by at most one edge, (ii) edges carry no orientation, and (iii) no self-loops are allowed. More general graph types, such as directed or weighted graphs, are beyond the scope of these preliminaries.

Graphs and Adjacency Matrices. Consider a graph by $G = (V, E)$, where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Because directly manipulating the node and edge sets can be cumbersome, it is common to use the **adjacency matrix** to represent a graph in an algebraically convenient form.

Given an ordering of the nodes $v_1, \dots, v_{|V|}$, the adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$ is defined as

$$A[i, j] = \begin{cases} 1, & \text{if } (v_i, v_j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

For a simple undirected graph, A is symmetric, i.e., $A[i, j] = A[j, i]$. This representation enables convenient algebraic manipulation of structural properties of the graph.

Node Degree. A fundamental structural quantity in a graph is the **degree** of a node. For a node $u \in V$, its degree is the number of edges incident to it:

$$\deg(u) = \sum_{v \in V} A[u, v].$$

Although directed graphs allow distinctions between in-degree and out-degree, this thesis focuses on simple undirected graphs, and therefore uses the above definition. The degree of a node is often interpreted as a coarse measure of its connectivity or potential influence in the network.

Degree Matrix. The **degree matrix** $D \in \mathbb{R}^{|V| \times |V|}$ provides a matrix-form representation of node degrees. It is defined as a diagonal matrix:

$$D[i, i] = \deg(v_i), \quad D[i, j] = 0 \quad \text{for } i \neq j.$$

This compactly encodes the local connectivity of each node and serves as a key component in many graph operators.

Graph Laplacian. The **graph Laplacian** is a central object in spectral graph theory and plays an important role in various graph learning tasks. For an undirected graph, the (combinatorial) Laplacian is defined as

$$L = D - A,$$

where D is the degree matrix and A is the adjacency matrix. Given the ordered nodes $v_1, \dots, v_{|V|}$, we may write $L = (L[i, j])$ explicitly as

$$L[i, j] = \begin{cases} \deg(v_i), & \text{if } i = j, \\ -1, & \text{if } i \neq j \text{ and } (v_i, v_j) \in E, \\ 0, & \text{otherwise.} \end{cases}$$

The Laplacian captures essential structural characteristics of the graph, and its eigenvalues and eigenvectors underpin applications in spectral clustering, graph signal processing, and theoretical analysis. A deeper discussion of its properties is provided in Section 3.4.

Graph Centrality Measures. Beyond quantities such as degrees, graphs exhibit more complex structural patterns that require richer metrics. **Graph centrality** [32] measures quantify the relative importance of nodes in the network. While the mathematical definitions are omitted here, several widely used examples include degree centrality, closeness centrality, etc. These centrality measures offer complementary perspectives on the roles nodes play within a graph, providing structural insights that extend beyond simple degree statistics.

Graph Representation in Computer Science

There are two common ways to represent graphs with computer: adjacency lists and adjacency matrices. Both of them can be applied to represent directed and undirected graphs. And they have their own advantages and disadvantages. We have introduced adjacency-matrix representation in last section. Here, we mainly discuss adjacency-list representation.

Adjacency-list Representation. The adjacency-list representation of a graph $G = (V, E)$ is composed of an array Arr of $|V|$ lists, one for each vertex in V . For a node $u \in V$, the adjacency list $\text{Arr}[u]$ consists of a list of all the neighbors of node u in graph G , i.e., there is an edge $(u, v) \in E$. Figure 2.6[b] is an adjacency-list representation of an undirected graph. For example, $\text{Arr}[1] = \{2, 5\}$ stores the edges that are incident to node 1, i.e., $(1, 2), (1, 5) \in E$.

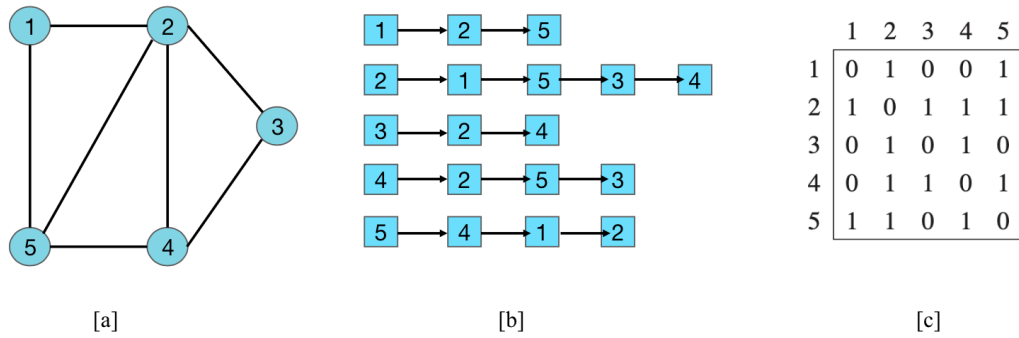


Figure 2.6: Adjacency-list and adjacency-matrix representations of an undirected graph. [a] An undirected graph with 5 nodes and 7 edges. [b][c] The corresponding adjacency-list and adjacency-matrix representations of the graph in [a], respectively.

Note that a crucial difference between adjacency-list and adjacency-matrix is that the list representation has the advantage of requiring only $O(|V| + |E|)$ memory, whereas the matrix representation requires $O(|V|^2)$ memory, regardless of the number of edges in the graph. This difference becomes significant when the graph is large and sparse, meaning $|E|$ is much smaller than $|V|^2$. In such cases, the matrix representation becomes far more memory-intensive than the list representation. Moreover, Table 2.1 demonstrates more properties of aforementioned two representations. Besides memory efficiency, the adjacency list representation has advantages when visiting a node's neighbors, i.e., the nodes directly connected to it, with a time complexity of $O(\deg(v))$. In contrast, the adjacency matrix representation requires traversing all nodes to identify neighbors. However, the matrix representation is more efficient for checking whether an edge (u, v) exists in the graph, as it can achieve this in $O(1)$ time, whereas the list representation requires traversing the adjacency array of u or v , resulting in $O(\deg(v))$ time complexity. Another significant difference is that the matrix representation is more suitable for matrix-based computations due to its fixed dimensions. On the other hand, the list representation, with its unevenly sized

arrays for different nodes, is challenging to apply in mathematical calculations.

Table 2.1: List representation vs. matrix representation of a graph.

	Matrix	List
Memory	$O(V ^2)$	$O(V + E)$
Edge Search	$O(1)$	$O(\deg(v))$
Visit neighbors	$O(V)$	$O(\deg(v))$

The two aforementioned representation methods illustrate the flexibility of graph data, allowing one to choose the appropriate graph representation based on the characteristics of the graph in question, i.e., the type of graph, or the specific graph operations frequently performed. This flexibility is a significant advantage of graphs; however, it also introduces greater complexity, as it requires careful and deliberate selection of graph modeling approaches. This is one of the key challenges in working with graph data. In the following section, we will delve into the unique characteristics of graph data, particularly its non-Euclidean nature, and compare it with Euclidean data, such as images and videos, to introduce the challenges of graph modeling and the acquisition of graph data.

2.1.3 The Specialty and Difficulty in Graph Data

Euclidean vs. Non-Euclidean

Euclidean data generally refers to the kind of data with a regular geometric structure, which can be embedded in a Euclidean space, where relationships between data points are in Euclidean geometry. Such data can be represented in a fixed-dimensional coordinate system (e.g., 2D grids, 3D volumes, or 1D sequences), and more significantly, distances or similarities can be computed using classical metrics like the Euclidean distance (L2-norm). Figure 2.7 demonstrates an example of a binary art, using a 35×35 matrix to represent a digital art, where 0 is used to represent black and 1 is used to represent white. Such properties enable Euclidean data to be formalized into specific genres of Convolutional neural network (CNNs) [33, 34, 35].

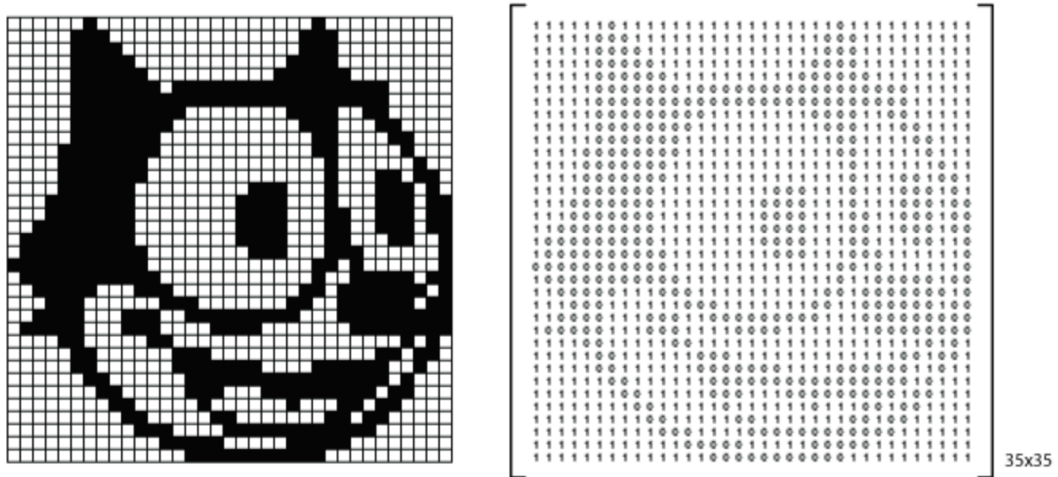


Figure 2.7: A binary pixel art provided by <http://www.uff.br/cdme/>.

Non-Euclidean data, such as the social network [36] or a genomic regulatory network [37], do not have such underlying Euclidean structure and properties including global parameterization, common system of coordinates, vector space structure or shift-invariance, making traditional deep learning architectures

like CNNs not well defined for them [38]. Graph data inherently possess non-Euclidean structures with dynamic topologies and complex relationships, intriguing extreme challenges compared to grid-base images or text data. Therefore, a graph structure requires more delicate design for its representation and corresponding learning.

Bottlenecks in Graph Data Acquisition

The scarcity of large-scale, high-quality and various graph datasets stems from technical, ethical, and domain-specific barriers, compared with the abundance of image/text data.

Table 2.2: Details of some datasets in graph, images and text data.

Data Type	Representative Datasets	Size(GB)	Key Features
Graph Data	Open Graph Benchmark[39]	50-2500	Largest subset: 244M nodes and 1.8B edges [40]
	GDB-13 [41]	50-100	Domain-specific (biomedical)
Image Data	LAION-5B [42]	240,000	5B image-text pairs, multi-model
Text Data	RedPajama-Data-1T [43]	1,200,000	1.2T tokens from CommonCrawl, arXiv, etc.

Table 2.2 shows details of some common used open-source datasets in graph, image and text data. Generally, open-source graph datasets are orders of magnitude smaller than image/text counterparts. For instance, LAION-5B [42] is about tens of times larger than the largest open-source graph dataset (OGB [39]). One key factor is that images and texts can be easily generated via cameras, web scraping, or AI generating. However, graph data modeling require domain-specific tools. Moreover, the annotation costs of graph are significantly higher than of images and text. For example, graph labels (e.g., molecular properties) demand expert knowledge, whereas image/text labels can be crowdsourced. Such limitations lead to the bottlenecks in graph data acquisition, following with graph research, representation or learning.

2.2 Traditional Machine Learning for Graphs

In Chapter 1, we introduced examples such as weight prediction from height and image recognition, both of which can be naturally expressed within the standard machine-learning formulation $y \approx f_\theta$. Their inputs—numerical vectors for height and weight, or pixel grids for images—can be directly encoded as fixed-size tensors, making them compatible with classical learning pipelines.

However, **graph data** cannot be applied to this framework in such a straightforward way. Graphs carry rich *structural information*: nodes may have different degrees, edges may contain attributes, and the number and arrangement of nodes vary across samples. As a result, we cannot simply stack graphs into tensors¹, the way we do with images or numerical features. This irregularity requires a dedicated representation and additional notation, which we introduce next.

A graph, in the context of machine learning, is typically denoted by

$$G = (V, E, \mathbf{X}, \mathbf{A}),$$

where V is the set of nodes, E is the set of edges, $\mathbf{X} \in \mathbb{R}^{|V| \times d}$ is the node-feature matrix, and $\mathbf{A} \in \{0, 1\}^{|V| \times |V|}$ is the adjacency matrix encoding the connectivity structure. Unlike images or tabular features:

- the size of V varies from graph to graph,
- the adjacency structure $\mathbf{A}$ introduces combinatorial complexity, and

¹In the context of artificial intelligence (AI) and machine learning, a tensor is a multi-dimensional array used as the fundamental data structure to store and manipulate data.

- learning must consider both node attributes and their pairwise relationships.

Therefore, traditional machine-learning models cannot directly process graphs in their raw form. Handling graph data requires specialized architectures or algorithms designed to respect permutation invariance, variable graph sizes, and structural dependencies—topics that the following sections will introduce in detail.

Traditional machine learning approaches on graphs usually begin by extracting some statistics or features based on heuristic functions or domain knowledge, then use these features as input to a standard machine learning classifier (e.g., logistic regression).

For example, suppose we have an input graph $G = (V, E)$ and a subset $\hat{G} \subseteq G$ that we know the labels (red or green) of nodes in $\hat{G}$, we want to figure out the labels of unknown nodes in $G \setminus \hat{G}$ (see Figure 2.8). This is a classic **node classification task**.

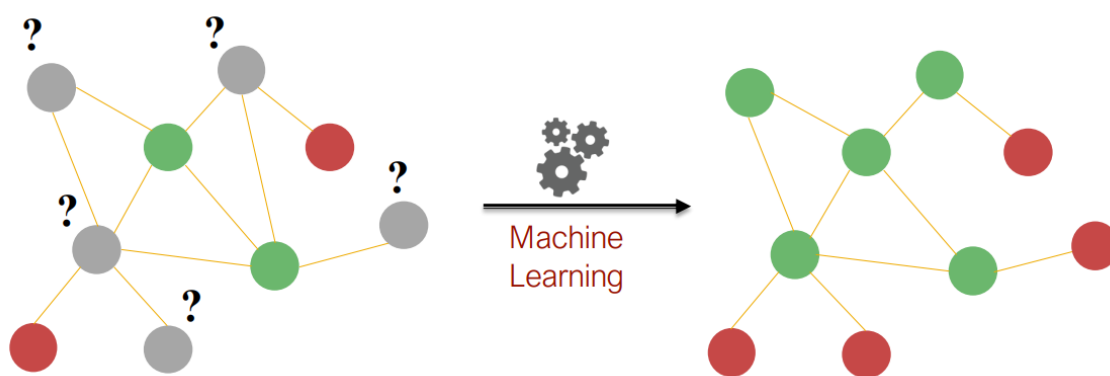


Figure 2.8: An illustration of machine learns for graphs, where colors denote labels. Red and green are two known labels, while gray denotes unknown labels. (source: [44])

Traditional machine learning approaches on graphs typically address problems like node classification by first transforming the graph into a set of manually designed features. Instead of operating directly on the graph’s complex topology, researchers extract structural statistics—such as node degree, centrality measures, or clustering coefficients—and combine them with any available node attributes to form a feature tensor (illustrated in Figure 2.9). This tensor is then fed into a standard machine learning model (e.g., logistic regression or a Support Vector Machine (SVM)) to perform prediction.

Another common and important research topic among graph-object researches is molecule analysis, the process of studying the physical and chemical properties of molecules. Traditional methods for molecule analysis are experimental techniques, including spectroscopy, chromatography, microscopy, etc. [45]. They usually require laboratory personnel with specialized knowledge, highly sophisticated facilities, and enormous time and efforts [45]. Density Functional Theory (DFT) is a quantum mechanical approach for calculating the electronic structure of many-body systems, such as atoms, molecules, and solids [46], [47]. It is a very popular and widely used method in computational chemistry and physics, as it allows for the calculation of various properties of these systems, including their energies, structures, and spectra. However, DFT has some limitations, including some inaccuracies due to approximations to the true density functional, difficulties in detecting strongly correlated systems, and high computational cost on large systems [46], [47].

With the rapid advancing of machine learning algorithms, molecule analysis has achieved great progress by applying machine learning models. Unlike traditional experimental approaches and DFT which mainly focus on single-molecule analysis, traditional machine learning methods excel at employing the dataset of known molecules to predict properties for unknown molecules. This is highly efficient and promising. Notice that a traditional machine learning model on molecule analysis usually requires prior knowledge of extracting features from molecules and delicate design of structures of the model. For

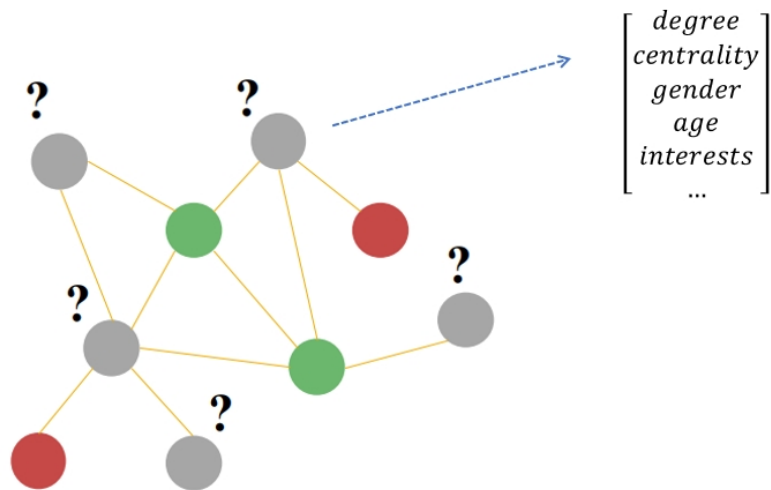


Figure 2.9: A showcase of feature extraction of a social network. A node (e.g., a user account in social media) is transformed into a tensor consisting of structural information extracted from the graph (degree, centrality, etc.) and personal information (e.g., age, gender, interests, etc.). (Adjusted from [44] by the author)

example, Zhu, et al., designed a traditional machine learning model, mol-infer, to predict properties with prior knowledge and the Mixed-Integer Linear Programming (MILP) for designing molecules with given properties [48] (see Figure 2.10).

While this strategy simplifies the learning process by avoiding direct manipulation of the graph structure, it also comes with clear limitations. The model can only exploit the information explicitly encoded by the handcrafted features, meaning much of the relational and topological information in the graph remains unused. Moreover, the final predictive performance is heavily dependent on the quality and completeness of the feature extraction process, placing significant burden on domain expertise and heuristic design.

In next section, we will introduce **Graph Learning (GL)**, a latest kind of machine learning techniques on graphs, is being applied to handle above challenges. Graph learning usually has better capacity to feature and represent graphs without pre-processing graph data, and they can reach equal or better performances, especially when the training dataset is large enough.

2.3 Graph Learning

Graph learning refers to a special kind of ML methods to learn graphs that can directly utilize graph topology—i.e., the relationships among nodes—when producing predictions or representations. Unlike traditional machine learning pipelines that require handcrafted features derived from the graph, graph learning methods aim to automatically capture structural patterns by learning representations that reflect both node attributes and connectivity.

Graph learning emerged to bridge these gaps by **automating feature extraction** while preserving relational inductive bias. Early graph learning interpreted graph signals as **smooth functions** over the graph topology, with methods like Dong et al. [49] and Kalofolias [50]. These models enforced smoothness in the spectral domain, learning graph structures that minimize total variation. However, these approaches are fundamentally limited by their strong reliance on smoothness assumptions, which restrict their ability to model irregular, heterogeneous, or non-smooth relational patterns commonly found in real-world graphs.

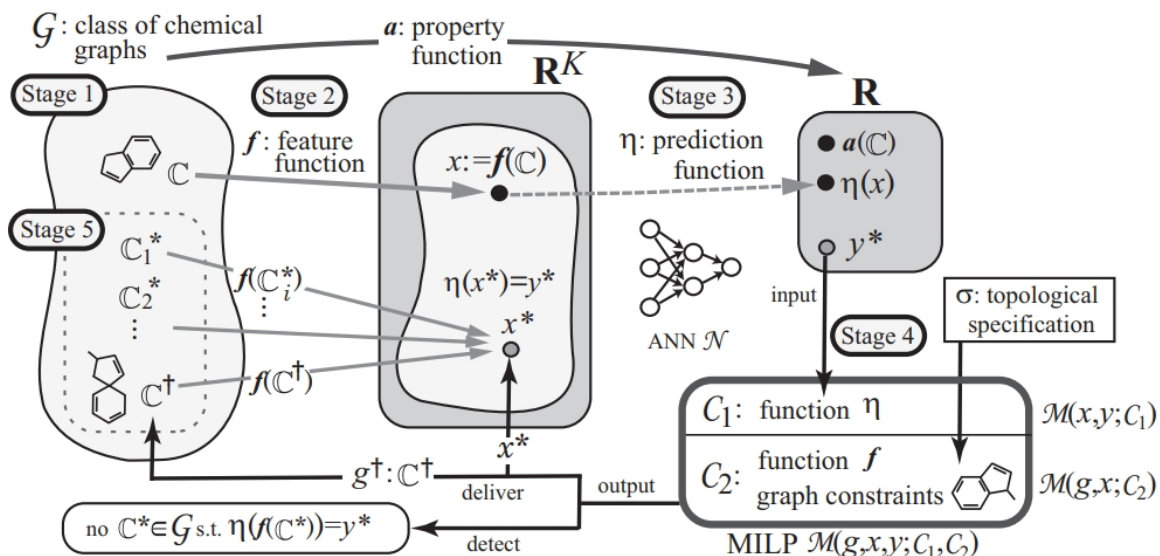


Figure 2.10: Framework of mol-infer proposed by Zhu, et al. [48], where stage 2 is based on human knowledge.

A significant milestone in the evolution of graph learning was the emergence of representation learning methods, that directly mapped nodes or graphs into low-dimensional embeddings² while preserving network proximity. **DeepWalk** [51] and **node2vec** [52] are two well-known methods.

DeepWalk treats truncated random walks on graphs analogously to sentences in natural language and trains node embeddings using the so called Skip-Gram model. This allows nodes with similar structural or positional roles to acquire similar embeddings. Node2vec extends this idea by introducing a biased random-walk strategy that smoothly interpolates between breadth-first and depth-first exploration. This flexibility enables the model to capture both community-based similarity and structural equivalence in the learned representations.

These methods established the paradigm of latent representation learning for graphs, where tasks such as node classification, clustering, and link prediction can be solved using simple downstream classifiers operating directly on embeddings. More importantly, these embedding techniques laid the conceptual foundation for **Graph Neural Networks** (GNNs), which implement representation learning through explicit **message passing** over graph neighborhoods.

Before presenting specific GNN architectures, we first introduce the basic notations and preliminaries that describe how message passing operates as a learnable, layer-wise process.

Given a graph $G = (V, E)$, for each node $v \in V$, we denote its neighbors by

$$\mathcal{N}(v) = \{u \in V : (u, v) \in E\}.$$

Assume that each node is equipped with an input feature vector $\mathbf{x}_v \in \mathbb{R}^d$, where d is the feature dimension. These initial node features serve as the starting point for graph neural networks, and we write

$$\mathbf{h}_v^{(0)} = \mathbf{x}_v.$$

In analogy to a standard neural network that learns a mapping

$$f_\theta : \mathcal{X} \rightarrow \mathcal{Y},$$

²In the context of AI, embedding refers to techniques for mapping complex data (such as words, images, or audio) to numerical vectors in some metric space. This process transforms data into a format that can mathematically process and compare.

a GNN learns a mapping

$$f_{\theta}(G, X) = H,$$

where $X = \{\mathbf{x}_v\}_{v \in V}$ denotes the input node features, $H = \{\mathbf{h}_v^{(K)}\}_{v \in V}$ denotes the output node representations after K layers of message passing, and $\theta = \{\theta^{(0)}, \theta^{(1)}, \dots, \theta^{(K-1)}\}$ is the collection of *learnable parameters* of all layers. In most GNNs, each layer k has its own weight matrix (or matrices), which we denote by $W^{(k)}$ and include in $\theta^{(k)}$.

A GNN maintains a (hidden) representation $\mathbf{h}_v^{(k)}$ for every node $v \in V$ at each layer index $k \geq 0$. For each k , node v first aggregates information from its neighborhood and then updates its representation. A general message-passing step can be expressed as

$$\mathbf{m}_v^{(k)} = \text{AGGREGATE}(\{\mathbf{h}_u^{(k)} : u \in \mathcal{N}(v)\}),$$

$$\mathbf{h}_v^{(k+1)} = \text{UPDATE}(\mathbf{h}_v^{(k)}, \mathbf{m}_v^{(k)}; \theta^{(k)}).$$

Here, AGGREGATE describes how node v collects information from its neighbors (for example, by taking a sum, mean, or attention-weighted combination), while UPDATE specifies how the current representation of v is combined with the aggregated message and transformed using learnable parameters. A common concrete form of the update rule is

$$\mathbf{h}_v^{(k+1)} = \sigma\left(W^{(k)} \cdot \text{COMBINE}(\mathbf{h}_v^{(k)}, \mathbf{m}_v^{(k)})\right),$$

where $W^{(k)}$ is a learnable weight matrix of the k -th layer, $\sigma(\cdot)$ is a nonlinear activation function (e.g., ReLU), and COMBINE typically concatenates or linearly mixes $\mathbf{h}_v^{(k)}$ and $\mathbf{m}_v^{(k)}$.

In this formulation, the index k plays the same role as the layer index in a multilayer perceptron (MLP): $k = 0$ corresponds to the input features $\mathbf{h}_v^{(0)} = \mathbf{x}_v$, and each application of the pair

(AGGREGATE, UPDATE)

with its own parameter set $\theta^{(k)}$ constitutes one GNN layer. Stacking multiple layers ($k = 0, 1, \dots, K-1$) allows node embeddings $\mathbf{h}_v^{(K)}$ to incorporate information from progressively larger neighborhoods, i.e., from nodes that are multiple hops away in the graph (see Figure 2.11).

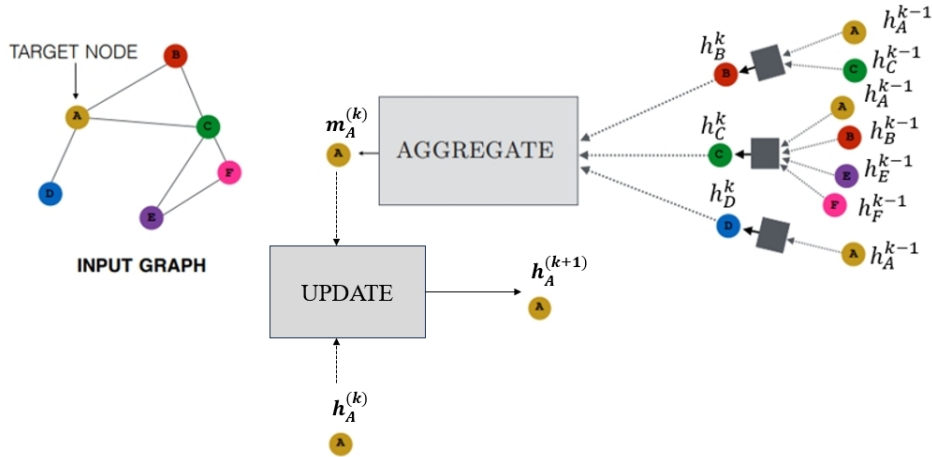


Figure 2.11: Visualization of a GNN framework. (Adjusted from [53] by the author)

Among the many formulations of graph neural networks, two foundational architectures—Graph Convolutional Network (GCN) [54] and GraphSAGE [55]—have become widely adopted due to their conceptual clarity and empirical effectiveness. Both can be understood as specific instantiations of the message-passing framework described earlier, where each GNN layer indexed by k applies a pair of operations: AGGREGATE and UPDATE, parameterized by learnable weight matrices $W^{(k)}$.

The Graph Convolutional Network (GCN) performs neighborhood aggregation using a normalized mean of node features. At layer k , each node v aggregates the features of its neighbors according to

$$\mathbf{m}_v^{(k)} = \sum_{u \in \mathcal{N}(v) \cup \{v\}} \frac{1}{\sqrt{d_u d_v}} \mathbf{h}_u^{(k)}, \quad (2.1)$$

where d_v is the degree of node v including a self-loop. The node representation is then updated by applying a linear transformation followed by a nonlinear activation:

$$\mathbf{h}_v^{(k+1)} = \sigma\left(W^{(k)} \cdot \mathbf{m}_v^{(k)}\right), \quad (2.2)$$

where $W^{(k)}$ is the learnable weight matrix of layer k . By collecting the hidden features of all nodes into a matrix $\mathbf{H}^{(k)} \in \mathbb{R}^{n \times d_k}$, the GCN update can be written compactly in matrix form as

$$\mathbf{H}^{(k+1)} = \sigma\left(\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2} \mathbf{H}^{(k)} W^{(k)}\right),$$

where $\tilde{A} = A + I$ and $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$. In this formulation, each GCN layer corresponds to one iteration of message passing, and stacking multiple layers increases the receptive field of node representations.

In contrast, GraphSAGE was introduced for inductive learning, emphasizing flexible neighborhood aggregation applicable to unseen graphs. At each layer, the AGGREGATE operation computes the mean representation of the neighbors of node v :

$$\mathbf{m}_v^{(k)} = \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k)}.$$

The UPDATE step then forms the next representation by concatenating the node's own information with the aggregated neighborhood signal and transforming the result:

$$\mathbf{h}_v^{(k+1)} = \sigma\left(W^{(k)} \cdot \text{concat}(\mathbf{h}_v^{(k)}, \mathbf{m}_v^{(k)})\right),$$

where $W^{(k)}$ is the layer-specific learnable parameter matrix. This formulation highlights the central modeling philosophy of GraphSAGE: node representations are formed by combining a node's intrinsic attributes with aggregated relational information, enabling inductive generalization beyond the observed training graph.

Both GCN and GraphSAGE instantiate the general message-passing principle using architecture-specific aggregation and update rules, where the layer index k corresponds directly to the depth of a neural network model. Therefore, stacking multiple layers allows nodes to incorporate increasingly rich structural context, reflecting the intuition that relational meaning emerges from progressively expanded neighborhoods.

Note that in many graph learning tasks, the objective is not only to learn node-level representations but also to compute a representation for an entire graph (e.g., for graph classification, regression, or graph-level prediction). To obtain a fixed-size embedding for the whole graph, a READOUT operation is applied after the final message-passing layer. The READOUT function maps the set of final node representations to a single graph representation:

$$\mathbf{h}_G = \text{READOUT}(\{\mathbf{h}_v^{(K)} : v \in V\}),$$

where $\mathbf{h}_v^{(K)}$ is the representation of node v after K layers of message passing and $\mathbf{h}_G$ denotes the resulting graph-level embedding. The READOUT function is typically a permutation-invariant operator,

ensuring that the output does not depend on the ordering of nodes. Common READOUT operations include element-wise summation, mean pooling, max pooling, or more advanced attention-based pooling strategies:

$$\mathbf{h}_G = \sum_{v \in V} \mathbf{h}_v^{(K)}, \quad \mathbf{h}_G = \frac{1}{|V|} \sum_{v \in V} \mathbf{h}_v^{(K)}, \quad \mathbf{h}_G = \max_{v \in V} \mathbf{h}_v^{(K)}.$$

In this formulation, the READOUT operator plays a role analogous to global pooling in convolutional neural networks, compressing spatially distributed representations into a compact vector usable for subsequent prediction layers. When combined with a final output layer, a GNN for graph-level prediction can be written as

$$\hat{y} = f_{\theta_{\text{out}}}(\mathbf{h}_G),$$

where θ_{out} represents learnable parameters (e.g., in a fully connected or softmax output layer).

To make these operations more intuitive, consider one common task of predicting the toxicity of a molecule. A molecule is treated as a graph where atoms serve as nodes and chemical bonds as edges. Each atom initially carries features describing its element type or valence. During message passing, an atom collects information from the atoms bonded to it (the AGGREGATE step), summarizing the local chemical environment. Then, in the UPDATE step, the atom refines its own embedding by combining its previous state with this neighborhood message. After several rounds of this process, each atom may encode higher-level structural information, such as functional groups or aromatic rings. Finally, a graph-level READOUT operation aggregates node embeddings into a global representation used to predict the molecule’s toxicity.

Figure 2.12 depicts the whole process. The input is a molecular graph where each node represents an atom, and its initial features $\mathbf{h}^0$ encode physical and chemical properties such as atom type, valence, mass, and melting point, etc. Through multiple layers of message passing, each node updates its hidden representation $\mathbf{h}^{(k)}$ by aggregating information from neighboring atoms. After the final GNN layer, the resulting node embeddings $\mathbf{h}^{(K)}$ are aggregated using a sum-based readout function to produce a graph-level embedding, which is subsequently used to predict the molecule’s toxicity.

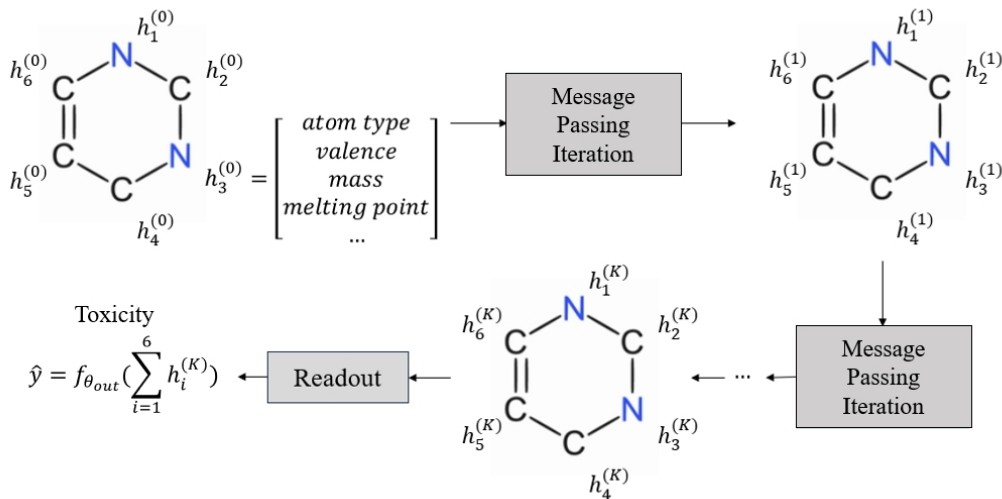


Figure 2.12: An illustration of a Graph Neural Network (GNN) for molecular toxicity prediction. (Work by the author)

GNNs have demonstrated remarkable success in a variety of real-world domains, significantly enhancing performance compared to traditional methods. In **drug discovery**, for instance, by integrating domain knowledge, models like RG-MPNN incorporate pharmacophore structures into GNNs to further improve both predictive accuracy and interpretability in chemical property prediction [56]. In **traffic**

forecasting, temporal GNNs such as the T-GCN model combine graph convolution with sequential modeling (e.g., GRU) to capture both spatial dependencies across road networks and temporal evolution of traffic flow, yielding state-of-the-art prediction results [57]. In **recommender systems**, GNNs are used to model high-order collaborative signals in social interaction graphs, embedding user-item relationships through multi-hop neighborhood aggregation to improve recommendation accuracy beyond conventional collaborative filtering [58]. These examples illustrate how GNNs leverage relational structure to deliver powerful predictive models across chemistry, transportation, and recommendation.

Despite their huge success, GNN models face challenges:

- **Over-smoothing**: With increasing layers, embeddings converge to indistinguishable vectors (particularly for GCN) [59].
- **Expressivity ceiling**: Message passing is provably limited (equivalent to the 1-Weisfeiler-Lehman test) [60].
- **Locality bias**: Models primarily aggregate local information, struggling with long-range dependencies and global graph patterns.

More recently, researchers sought to overcome the **limitations of locality and over-smoothing in GNNs** by adapting transformer-style encoder-decoder models to graphs: Graph-BERT, Graphormer, etc. These models represent the latest evolution, bridging GNNs with large-scale transformer architectures to achieve state-of-the-art performance across molecular property prediction, knowledge graphs, and social network analysis. For detail, we refer the readers to [61, 62].

2.4 Graph Learning vs. Statistical Machine Learning on Graphs

In order to further understand and evaluate the capacity of graph learning models, we compose a model with GraphSAGE and conduct experiments for molecular property prediction task on QM9 [63, 64]. The comparable baseline in the experiments is a state-of-the-art but traditional machine learning model, mol-infer [48] introduced in Section 2.2. Note that mol-infer [48] model requires prior knowledge for molecules, whereas GraphSAGE does not.

QM9 dataset contains real molecules with the ground-truth ³ on features and properties, which is important for us to evaluate the accuracy of the models. Concrete statistics are shown in Table 2.3.

Table 2.3: Statistics of QM9 dataset used in experiments.

Datasets	#graph	#avg node	#avg edge	#property	#feature
QM9	130,831	18	37.3	19	11

Our prediction model is built upon the GraphSAGE architecture described in the previous section, where node embeddings are updated through iterative neighborhood aggregation and transformation. Specifically, at each GraphSAGE layer k , node representations are computed as

$$\mathbf{h}_v^{(k+1)} = \sigma \left(W^{(k)} \cdot \text{concat}(\mathbf{h}_v^{(k)}, \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(k)}) \right),$$

where $W^{(k)}$ is the learnable weight matrix that maps the concatenated features into the next hidden representation. By stacking multiple GraphSAGE layers, the model progressively integrates multihop structural and chemical information across the molecular graph.

³Ground-truth is factual information obtained through direct observation or measurement, as opposed to being inferred.

To determine the appropriate architecture for the model, we conducted a grid search over two key hyperparameters: (i) the number of GraphSAGE layers, and (ii) the dimensionality of the hidden representation in each layer. We conducted experiments on the QM9 dataset, which contains approximately 130k small organic molecules. A fixed test set of 1,000 molecules is held out for evaluation, while the remaining molecules are used for training. We predict four molecular properties: polarizability (α), LUMO energy, internal energy at 0 K (U0), and heat capacity (CV). Table 2.4 summarizes the experimental results, where the vertical axis denotes hidden layer dimensions, the horizontal axis denotes the number of hidden layers, and the table elements report the mean adjusted R^2 values across all predicted molecular properties. The best-performing configurations are indicated in **bold**. Balancing predictive performance and parameter complexity, we choose 6 GraphSAGE layers with a hidden dimension of 64.

Table 2.4: Results of GraphSAGE in learning QM9.

	1	2	3	4	5	6	7	8	9	10
16	0.865	0.941	0.964	0.965	0.968	0.963	0.969	0.970	0.971	0.970
32	0.883	0.965	0.971	0.976	0.979	0.980	0.982	0.983	0.983	0.982
64	0.889	0.969	0.979	0.983	0.985	0.986	0.987	0.986	-3.685	-0.354
128	0.878	0.769	0.718	-0.722	0.983	0.983	-15.125	0.981	0.294	-17.374

In our implementation, Adam [65] is used as the optimizer, and mean squared error is employed as the loss function:

$$\text{Loss} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2,$$

where n is the number of training samples, y_i represents ground truth values, and $\hat{y}_i$ denotes model predictions. We apply Leaky ReLU as the activation function. For the graph-level prediction layer, we apply sum pooling as the READOUT operator. Hyperparameters are set as learning rate 0.001 and weight decay 0.0001. To ensure a fair comparison across experiments, we keep all structural settings unchanged except the hyperparameters varied via grid search.

The resulting model architecture is illustrated in Figure 2.13. Each GraphSAGE layer transforms node features from dimension m to n , denoted as (m, n) in the figure. Here, n_{features} , n_{hidden} , and $n_{\text{properties}}$ refer to the dimensionality of initial node features, hidden node embeddings, and predicted molecular properties, respectively.

We conducted two experiments: (i) the property prediction task on large dataset (up to the whole QM9 dataset); (ii) the property prediction task on small dataset (a small subset of QM9).

In (i), we randomly selected QM9 dataset into six groups: 0.3% QM9, 20% QM9, 40% QM9, 60% QM9, 80% QM9, and 100% QM9. This grouping method is used to ensure that each group has approximately the same increment. Here, $\alpha\%$ QM9 indicates that this group accounts for $\alpha\%$ of the overall QM9 dataset. Note that the QM9 has 130,831 graphs. Except the 0.3% QM9 only has around four hundred graphs, other groups have more than 20,000 graphs, which satisfies the purpose that we want to test the ability of our model on large datasets. Moreover, the size of each group scales up very fast from 0.3% QM9 to 100% QM9, which enables us to observe how the size of the dataset affects our model’s capacity. The ratio between training and test set is 8 : 2 in each group.

We use adjusted R^2 value [66], a less biased estimator of the R^2 value, to evaluate the results of this task. Values R^2 and adjusted R^2 can be computed as follows:

$$R^2 = 1 - \frac{\sum_i^n (y_i - f_i)^2}{\sum_i^n (y_i - \bar{y})^2}, \quad (2.3)$$

where y_i is the ground-truth, $\hat{y}_i$ is the predicted value and $\bar{y}$ is the mean of the ground truth values, $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$.

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(N - 1)}{N - p - 1}, \quad (2.4)$$

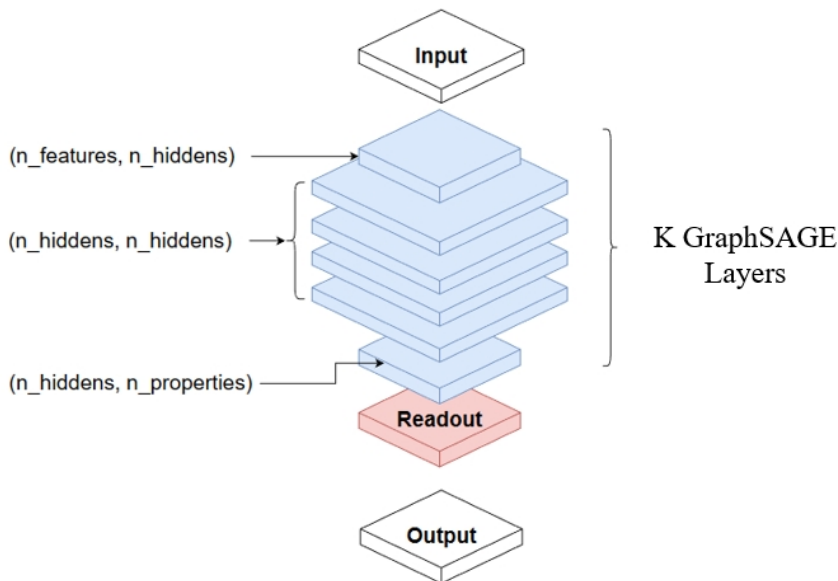


Figure 2.13: Illustration of the structure of our graph learning model used in Section 2.4. (Work by the author)

where N is the total sample size, p is the number of independent variable.

We choose seven properties in order to align with mol-infer as prediction targets (see Table 2.5), which are also predicted by mol-infer [48], and compare the results with mol-infer [48].

Table 2.5: Properties used in the experiments.

Property	Denotation	Description	Unit
μ	MU	Dipole moment	D
α	ALPHA	Isotropic polarizability	a_0^3
ϵ_{HOMO}	HOMO	Highest occupied molecular orbital energy	eV
ϵ_{LUMO}	LUMO	Lowest unoccupied molecular orbital energy	eV
$\Delta\epsilon$	GAP	Gap between ϵ_{HOMO} and ϵ_{LUMO}	eV
U_0	u0	Internal energy at 0K	eV
c_v	cv	Heat capacity at 298.15K	$\frac{cal}{mol\ K}$

Tables 2.6, 2.7, 2.8, 2.9, 2.10, 2.11, and 2.12 report the results of (i) and Tables 2.13 and 2.14 report the memory and time used in (i). We can reach some observations from the results: (1) graph learning model behaves better on the large dataset (20% QM9 and bigger datasets); (2) graph learning model has stronger capacity as the size of dataset scales up; (3) when the size of dataset reaches some big threshold, the capacity of the model will not get an obvious increase even if we double the size; and (4) graph learning model is more efficient on both time and space.

Table 2.6: Results for MU prediction.

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	0.5466	0.6422	0.6462	0.6591	0.6597	0.6463
our model	0.4398	0.5816	0.681	0.7167	0.7096	0.7329

In (ii), we aim to evaluate the capacity of our model on the small dataset. To fulfill this, we randomly extract four subsets from QM9 with the number of data to be 2,000, 4,000, 8,000, and 16,000. Let the

Table 2.7: Results for ALPHA prediction.

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	0.6722	0.9783	0.9798	0.977	0.9748	0.9827
our model	0.7377	0.9855	0.9904	0.9869	0.9803	0.993

Table 2.8: Results for HOMO prediction.

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	-1.6136	0.8146	0.8501	0.8398	0.8359	0.8398
our model	0.424	0.8156	0.9018	0.8996	0.8646	0.9004

Table 2.9: Results for LUMO prediction.

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	0.594	0.9162	0.9266	0.9276	0.9252	0.9346
our model	0.504	0.9698	0.9738	0.9762	0.9787	0.9808

Table 2.10: Results for GAP prediction.

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	0.3684	0.8789	0.8753	0.8929	0.8863	0.8954
our model	0.3683	0.86	0.9135	0.9236	0.9237	0.942

Table 2.11: Results for U0 prediction.

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	0.8435	0.9909	0.9987	0.999	0.9993	0.9996
our model	0.3683	0.9993	0.9996	0.9996	0.9997	0.9997

Table 2.12: Results for CV prediction.

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	0.8211	0.9732	0.9782	0.9751	0.9772	0.9786
our model	0.7053	0.9837	0.9905	0.9889	0.989	0.9889

Table 2.13: Results for memory used (Unit: Gb).

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	0.13	2.82	10.26	15.5	25.22	36.62
our model	4.736	5.506	5.568	5.824	6.432	6.848

Table 2.14: Results for time used (Unit: second).

	0.3% QM9	20% QM9	40% QM9	60% QM9	80% QM9	100% QM9
mol-infer	51.4	5074.3	11947.2	14875.1	19172.3	24127
our model	2.2	64.8	167.3	225	323	409.6

ratio of training and test set still be 8 : 2, others remain the same setup with (i). Here, we only predict ALPHA, LUMO, U0 and CV properties since they are also reported in mol-infer.

Results can be seen from Tables 2.15, 2.16, 2.17 and 2.18. In a nutshell, graph learning model cannot outperform the traditional machine learning model mol-infer [48] for small datasets.

Table 2.15: Results for ALPHA prediction.

	2k	4k	8k	16k
mol-infer	0.913	0.963	0.979	0.981
our model	0.08638	0.23056	0.4647	0.72672

Table 2.16: Results for LUMO prediction.

	2k	4k	8k	16k
mol-infer	0.777	0.852	0.893	0.93
our model	0.427	0.542	0.813	0.727

Table 2.17: Results for U0 prediction.

	2k	4k	8k	16k
mol-infer	0.944	0.963	0.984	0.992
our model	-0.089	0.536	0.53	0.72

Table 2.18: Results for CV prediction.

	2k	4k	8k	16k
mol-infer	0.923	0.957	0.978	0.977
our model	0.351	0.490	0.674	0.776

Based on the results, we can conclude some observations about graph learning methods: (1) graph learning methods usually perform better on larger datasets; (2) graph learning models’ performances on small datasets are unstable, not as good as traditional machine learning models with prior knowledge; (3) graph learning approaches excel at handling graph-structure data on time and space compared with traditional machine learning models. As for the baseline, an important observation is that the performances of traditional machine learning models are highly dependent on the prior knowledge. With appropriate and excellent prior knowledge, traditional machine learning models can reach extraordinary results.

2.5 Conclusion and Discussion

The comparative study presented in this chapter highlights the distinct strengths and limitations of traditional machine learning methods on graphs and modern graph learning approaches. The experimental results consistently indicate that graph learning models demonstrate superior scalability and performance as the size of the dataset increases. By directly operating on graph-structured data and learning representations in an end-to-end manner, graph learning is able to efficiently exploit complex relational patterns that are difficult to capture through handcrafted features. This advantage becomes increasingly pronounced in large-scale settings, making graph learning particularly well suited for the contemporary data landscape characterized by high volume, high dimensionality, and complex structural dependencies.

At the same time, the results also reveal that traditional machine learning approaches retain notable advantages in small-data regimes. Owing to their reliance on feature extraction guided by prior domain

knowledge, these methods can achieve competitive performance even when training data are limited. More importantly, the use of explicit, human-designed features provides a natural basis for interpretability: model predictions can often be directly related to specific extracted features, enabling a degree of transparency and causal reasoning. In contrast, graph learning models primarily rely on learned representations derived from graph structure and node or edge attributes, without incorporating explicit prior assumptions about which features are semantically or causally meaningful. As a result, while graph learning excels in predictive power, it often lacks an inherent interpretability foundation, making the reasoning behind its predictions more difficult to understand.

This limitation in explainability is particularly critical in high-stakes application domains, such as healthcare, finance, and scientific discovery, where trust, accountability, and error diagnosis are as important as raw predictive accuracy. The observations in this chapter therefore motivate the need for systematic research on explainability in graph learning. Understanding *why* a graph learning model makes a particular prediction, and how structural and feature-level information contribute to its decision, is essential for bridging the gap between performance and reliability. These considerations naturally lead to the focus of the next chapter, which investigates graph explainability methods aimed at providing principled and faithful explanations for graph learning models.

Chapter 3

Explainability of Graph Learning

Graph Neural Networks (GNNs) have achieved remarkable success across a wide range of relational learning tasks, including molecular property prediction, recommender systems, traffic forecasting, and fraud detection. Their ability to encode both node features and graph topology has made them the dominant paradigm for learning from structured relational data. However, despite their predictive power, many GNN models operate as black boxes: they provide accurate outputs but offer little visibility into the reasoning underpinning those outputs. This opacity undermines user trust and limits the deployment of GNNs in high-stakes domains—such as drug discovery, healthcare, and financial risk analysis—where accountability and explainability are essential.

Explainable AI (XAI) has therefore emerged as a promising direction for addressing this gap. XAI seeks to make model behavior comprehensible to humans by exposing the principles, evidence, or structures behind predictions. In Section 3.2, we examine the motivations and potential benefits of XAI, particularly in settings where model transparency is crucial. We further introduce major taxonomies that organize the diverse landscape of explainability methods, and we discuss explainability through the lens of inductive reasoning—clarifying how explanation strategies reflect assumptions about what constitutes meaningful evidence or reasoning patterns.

Building on this general foundation, the remainder of the chapter focuses on explainability for graph-structured data. Graph Explainability Problems (GEPs) aim to identify the subgraphs, components, or patterns driving a GNN’s prediction. In Section 3.3, we present the mathematical formulation of GEPs, illustrate their use cases, and review representative methods—covering factual, counterfactual, and hybrid approaches. Section 3.3 highlights both the progress and the limitations of current instance-level explainers.

One limitation stands out across the literature: existing GEP methods rarely account for the connectedness of explanatory subgraphs. Many methods allow explanations to fragment into disconnected pieces, which harms explainability and misaligns with how meaningful structures appear in real-world graphs (e.g., functional groups in molecules, coherent motifs in semantic or social graphs). In Section 3.4, we analyze this issue in depth and present our contributions from two publications. We introduce principled measures for representing and evaluating connectedness, and we integrate these notions into explainer frameworks that enforce or quantify structural coherence. Through theoretical analysis and empirical evaluation, we demonstrate that incorporating connectedness yields explanations that are more faithful, more compact, and more aligned with human intuition.

3.1 Introduction

The rapid proliferation of GNNs has brought remarkable success across domains such as drug discovery, transportation network prediction, and recommend systems. Yet despite these advances, many GNN models still operate as **black boxes**—they output predictions without revealing why those predictions

uncertain of whether its defects have been addressed and the public uneasy, for understandable reasons, about the technology’s safety. How can society adopt a technology that is as transformative as the internet when its most important decisions are made in an impenetrable black box?

From the perspective of avoiding bias, it is also significant to endeavor to unfold AI’s black-box nature. AI models are now widely used in financial credit scoring and automated trading, yet such models may inadvertently encode biased decision-making patterns—for example, assigning systematically higher loan-rejection probabilities to certain demographic groups. When these systems operate as black boxes without interpretable reasoning, such biases become even more difficult to detect, potentially leading to hidden, large-scale, and systematic discrimination with severe societal and economic consequences. Regulation calls for AI to maintain fairness and accountability [73], putting institutions in the context of having to choose between complex, possibly more accurate models and simple, unambiguous ones which are legally sound, while preserving customer’s confidence.

This aforementioned examples indicate why **explainability research** matters for AI: it seeks to ensure that model predictions are accompanied by human-understandable reasons, restoring transparency and accountability to automated decision-making.

Addressing this challenge is essential, and eXplainable AI (XAI) has emerged as a crucial direction for the future development of trustworthy AI systems in general. These use cases—which include drug discovery, self-driving cars, financial services, and oncology—all center on a single mandate: For AI to be effectively integrated, trusted, and responsibly operationalized in sensitive real-world applications, it is preferred that AI to be comprehensible to the humans who develop, use, regulate, and are impacted by the decisions it makes. This is the fundamental goal of XAI.

XAI encompasses methods and techniques designed to make the outputs, inner workings, and behaviors of complex AI models transparent and interpretable to humans. It aims to answer questions like [74, 75]:

- Why did the model make this specific prediction/decision?
- What features/structures in the input data were most influential?
- How would the output change if specific inputs changed?
- What are the model’s inherent limitations and potential biases?

It is important to note that all the examples discussed above involve **high-stakes scenarios**, whether related to human health, safety, or economic security. There are also application domains in which the explainability of AI models is less critical; for instance, in recommend systems, convenience and predictive performance often take priority. Such systems tolerate occasional errors, and users typically care more about the quality of the recommended items than about the underlying reasons for each prediction.

This contrast highlights an emerging trend in the development of AI: **the requirements placed on AI models vary across application contexts**, with the need for explainability depending strongly on the stakes and the nature of the task.

This thesis focuses specifically on explainability challenges in high-explainability-demand settings, such as molecular analysis and drug design. Other application domains with lower explainability requirements fall outside the scope of this work.

3.2 Taxonomy of eXplainable AI (XAI) and Inductive Reasoning

As XAI continues to expand across fields—from computer science to law and ethics—the number of methods, perspectives, and technical strategies has grown rapidly. This diversity, while valuable, also makes it difficult to compare approaches, evaluate their suitability for different applications, or identify

methodological gaps. A taxonomy therefore becomes essential: it provides a structured language for organizing explanation strategies, understanding their assumptions, and guiding both practitioners in selecting appropriate tools and researchers in formulating new questions.

Following Adadi et al. [76], we adopt a simple but widely used classification that distinguishes **what part of the model is being explained** and **how tightly the explanation method depends on the underlying model** (see Figure 3.2).

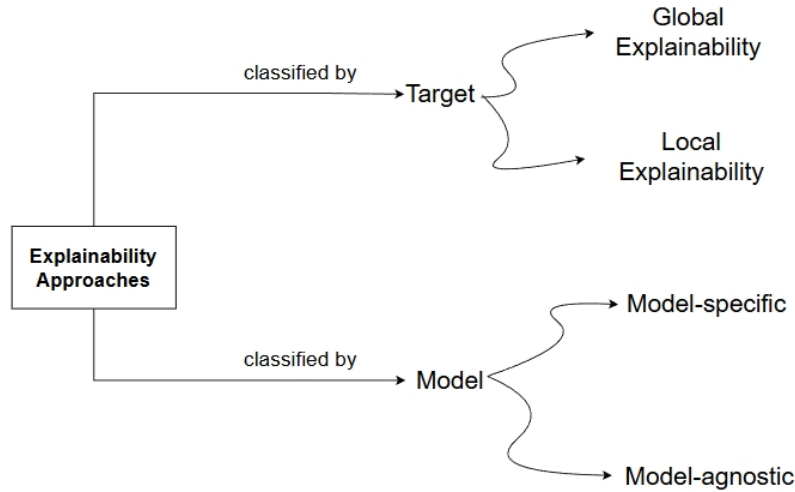


Figure 3.2: An illustration of XAI approaches taxonomy.

First, explanations may target either the **global behavior** of a system or **local decisions** on individual instances.

- *Global explanations* seek high-level understanding of a model’s overall logic—useful for global-level analyses or system auditing, but often hard to obtain for complex models [77, 78, 79, 80].
- *Local explanations* instead clarify why a model produced a specific prediction, which is generally more feasible and more relevant in practice; much of contemporary XAI, including methods reviewed later, falls into this category [74, 81, 82, 83].

Second, explanations techniques can be **model-specific** or **model-agnostic**.

- *Model-specific* methods leverage internal architectures (e.g., rule sets or additive models) but limit the choice of models one may use [79, 78].
- *Model-agnostic* methods, by contrast, treat the model as a black box and generate explanations through post-hoc analysis, making them broadly applicable and dominant in modern XAI practice [84, 85].

Together, these two dimensions—**target (global vs. local)** and **dependency (specific vs. agnostic)**—provide a practical map of the XAI landscape. However, they describe *what* a method explains, not *why* or *how* we reason about explanations. To deepen the discussion, the next section examines XAI methods from the perspective of **inductive reasoning**, addressing the fundamental question: *on what logical basis do explanation methods justify their claims?*

Reasoning - the act of drawing conclusions from the available information—is central to human cognition, and learning, decision-making, predicting, and problem-solving. Further, reasoning becomes the

underlying foundation and assumption of explainability techniques. Although deductive reasoning has been emphasized in logic (inference of necessary truths from premises), a great deal of ordinary and scientific thinking is based on inductive reasoning [86, 87]. Inductive reasoning is about determining general patterns or rules (or probable outcomes) based on specific observations or premises, so it deals with uncertainty and probability rather than certitude of the conclusion. We are especially interested in inductive reasoning, in the precise articulation of its constitutive subprocesses, factual reasoning and counterfactual reasoning. We will also briefly introduce the broader taxonomy of reasoning types.

Inductive reasoning allows us to extend generalizations beyond samples we already have. It is a form of induction that concludes something about what holds in general, what will hold in the future, or what is a cause of what based solely on patterns or co-occurrences in a sample of small or specific instances. Its conclusions can be more or less probable, but not certain, e.g., one sees many sheep and all of them are white; so the probable conclusion may be that “All sheep are white” - a conclusion falsifiable by further evidence.

Inductive reasoning is often based on representativeness, similarity and covariation detection, and prior knowledge and is prone to cognitive bias [88], and it is vulnerable to cognitive biases (e.g., availability, confirmation bias) and relies critically on the quality and representativeness of the sampled instances. Yet inductive inference is after all an extremely useful and necessary instrument for our thinking about, summarizing, and extending over what is unknown. At times it is at best an optional tool, rare even, especially when trying to address highly complex phenomena or systems.

Factual reasoning, closely related to causal reasoning, is a primary mode of inductive reasoning where conclusions are drawn about relationships or causes based on actual events, evidence, and patterns observed in the real world.

Factual reasoning centers on “what caused what” or “what is associated with what” based on available observation. It answers the question: “Given what happened, what factors were likely involved?” [89, 90]. Factual reasoning involves observing covariation, temporal contiguity, spatial contiguity, contingency, and integrating this with background knowledge to infer causal structures. For example, a doctor diagnosing an illness based on symptoms and test results; an individual assuming rain is likely because they see dark clouds and recall past associations.

Counterfactual reasoning is a robust and powerful mode of thought that is closely linked to inductive reasoning. It is the process of virtually recreating what an alternative to actual past or current events might have been, or known as the “what if” scenarios. Counterfactual Reasoning about what the present or past would have been like if an antecedent event or state of affairs (or, perhaps, action) had been different (e.g., “If only I had caught the bus, I wouldn’t have been late”) [91]. It involves mentally “undoing” an event, comparing the counterfactual scenario to reality, and deriving inferences. While induction primarily generalizes from observed data, counterfactual reasoning often explores the boundaries of inferred relationships by considering deviations. It helps refine causal models generated inductively by testing their sensitivity to changes. Conclusions about “what would have happened” are themselves inductive inferences based on mental simulation of an altered world.

Besides inductive reasoning, factual reasoning, and counterfactual reasoning, there exist other fundamental reasoning types:

- **Deductive Reasoning:** Deriving logically certain conclusions from premises (e.g., “If P then Q; P; Therefore Q”) [92]. However, plenty of things in life do not have premises.
- **Analogical Reasoning:** Drawing inferences based on mapping relational structures from a well-known domain (source) to a less familiar domain (target). It shares similarities with induction by generalization but focuses on structural parallels [93].

However, these forms of reasoning fall outside the scope of this thesis and will not be examined further. In the next section, we turn to explainability research in graph-based AI models—i.e., graph explainability research—building on the preliminaries established above.

3.3 Graph Explainability Problems (GEPs)

In Chapter 2, we introduced the general research problem addressed by graph learning methods—namely, learning to map a graph-structured input to a target property. For example, in molecular toxicity prediction, a molecule is first represented as a graph of atoms and bonds, and a given GNN model processes this structure to output properties such as toxicity or mutagenicity (see Figure 3.3). This illustrates the typical setting of graph learning: given a graph and a task-specific model, the goal is to infer meaningful attributes or behaviors of the underlying system.

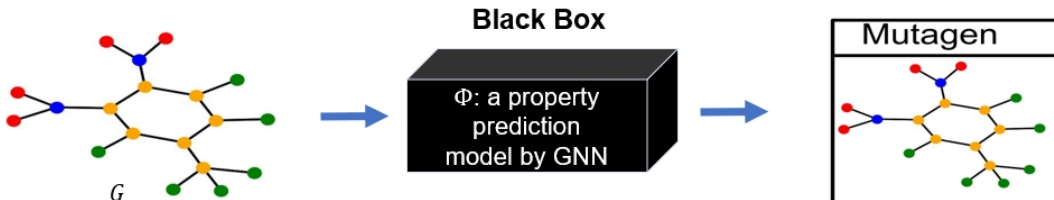


Figure 3.3: An illustration of graph learning to infer task-specific properties (e.g., mutagenicity).

While graph learning models focus on predicting properties from graph-structured inputs, Graph Explainability Problems (GEPs) concentrate on uncovering *why* a model produces a particular output. Their goal is to illuminate the black-box reasoning of GNNs by identifying the substructures, features, or graph components that drive a given prediction (see Figure 3.4). By revealing these underlying decision patterns, GEPs provide users with greater transparency and confidence, which is essential for deploying graph-based AI in high-stakes domains such as drug discovery, material design, and fraud detection.

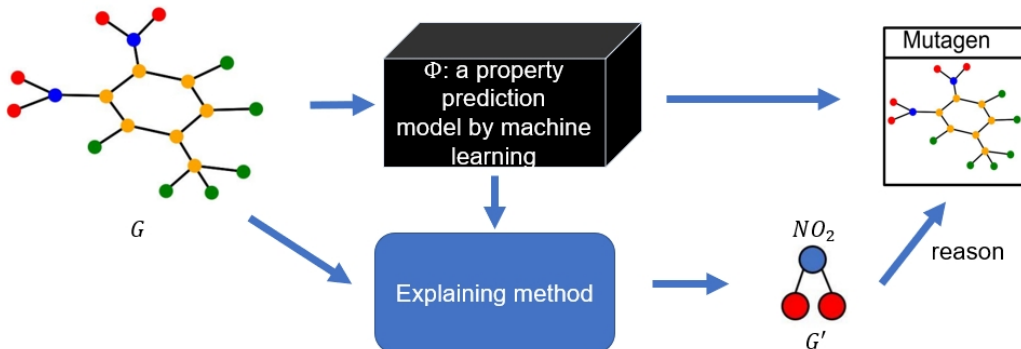


Figure 3.4: Illustration of the aim in GEP: identifying the graph components or patterns (NO_2 in this case) that most strongly contribute to the model’s prediction.

Formally, given a GNN model $f_\theta : \mathcal{G} \rightarrow \mathcal{Y}$, where $\mathcal{G}$ is the set of graphs and $\mathcal{Y}$ is the output space (e.g., class labels, regression targets). The goal of graph explainability is to uncover a simpler form of “reasoning” that accounts for the model’s prediction $f_\theta(G)$ on an input graph $G \in \mathcal{G}$. In practice, such reasoning is often expressed through *substructures* of the original graph, most commonly subgraphs (although subsets of node or edge features could also be used; for clarity, we focus on subgraphs in this chapter).

An *instance-level explainer* (local explanation in XAI) seeks a small, task-relevant subgraph

$$G' \subseteq G$$

such that the model’s output on this subgraph closely approximates the original prediction:

$$f_\theta(G') \approx f_\theta(G).$$

Here, G' serves as the explanatory evidence—a minimal or salient structure the model relies on for that specific prediction.

Alternatively, a *model-level explainer* (global explanation in XAI) aims to construct a simpler surrogate model

$$f'_{\theta'} : \mathcal{G} \rightarrow \mathcal{Y},$$

such as a linear model, a rule-based system, or a probabilistic model, that approximates the original GNN across all possible graphs:

$$f'_{\theta'}(G) \approx f_{\theta}(G) \quad \text{for all } G \in \mathcal{G}.$$

This approach captures the broader decision logic of the model rather than a single prediction, offering a global, human-interpretable summary of how f_{θ} behaves.

Although model-level explainers offer the appealing goal of capturing a GNN’s global decision logic, constructing such surrogate models is fundamentally challenging. Modern GNNs often contain deep architectures, nonlinear message-passing operations, and high-dimensional representations, which together encode complex decision boundaries that are difficult to approximate with any simple, human-interpretable model such as a linear regressor or a rule set. When the surrogate model is made more expressive in an attempt to match the GNN’s behavior, it inevitably becomes increasingly opaque itself, leading to the undesirable situation of “explaining one black box with another”. As a result of this inherent tension between accuracy and interpretability, most contemporary research in graph explainability focuses on *instance-level explainers*, which aim to identify salient subgraphs responsible for individual predictions rather than reconstruct the entire global reasoning process of the model.

Following the formalization of instance-level explainers, we review several representative approaches that have shaped the development of graph explainability. These methods can be broadly understood through their reasoning styles—factual, counterfactual, or combinations thereof—and their strategies for selecting explanatory subgraphs. Table 3.1 summarizes the categorization of the methods discussed in this subsection.

Table 3.1: Categorization of representative GEP methods by reasoning type.

Method	Factual	Counterfactual	Duplicate (Factual & Counterfactual)
GNNExplainer [84]	✓	–	–
CF-GNNExplainer [85]	–	✓	–
SubgraphX [94]	✓	–	–
CF ² [95]	✓	✓	✓

GNNExplainer [84] marks one of the earliest general frameworks for instance-level graph explanations. Given a trained GNN and an input graph, it learns continuous masks over edges and node features that maximize the mutual information between the model’s prediction and a masked version of the graph. The learned sparse mask highlights a compact subgraph that preserves the original prediction, thereby offering a factual explanation (Figure 3.5). Despite its foundational role, GNNExplainer may exhibit instability due to non-convex optimization, and its dense mask parameterization can limit scalability on large graphs.

While GNNExplainer identifies structures supporting the current prediction, CF-GNNExplainer [85] introduces a counterfactual perspective. Instead of searching for subgraphs that reproduce the prediction, it seeks minimal graph perturbations—often realized as sparse edge deletions—that cause the model to change its prediction (see Figure 3.6). Starting from the original graph, the method performs matrix sparsification to search for a minimal set of edge removals (shown as dashed red edges) that alter the model’s prediction. The resulting sparsified structure yields an optimal counterfactual example that highlights which edges are necessary for the original prediction. This provides explanations of the form: “if these edges were removed, the prediction would differ.” Such explanations are valuable for model recourse

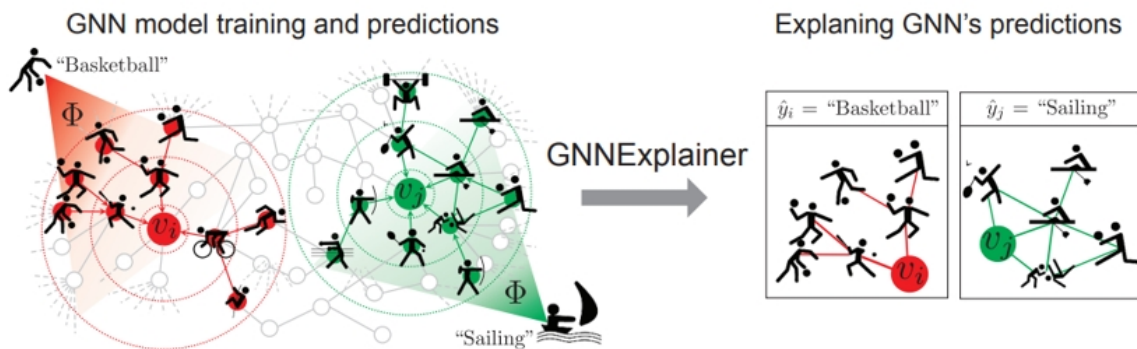


Figure 3.5: An illustration of GNNExplainer ([84]). For a GNN model predicting a person’s hobby on social networks (left), GNNExplainer tries to give corresponding explanations based on one’s social circles (right).

and sensitivity analysis. However, the method is typically restricted to deletion-based modifications and solves a difficult sparsity-constrained optimization problem, which can be computationally demanding.

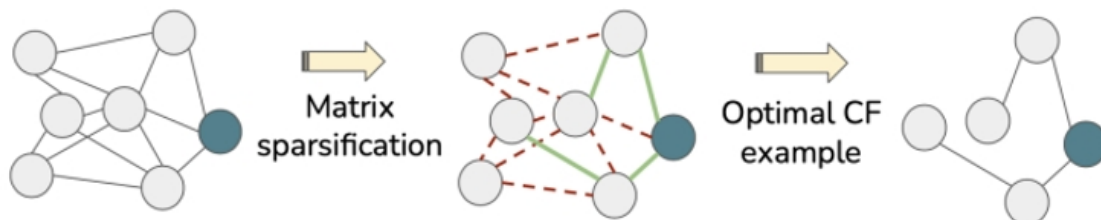


Figure 3.6: An illustration of CF-GNNExplainer ([85]).

SubgraphX [94] reframes the explanation task as a discrete subgraph search problem. By employing Monte Carlo tree search (MCTS), the method explores candidate subgraphs in the **connected subgraph search field** and evaluates them using a Shapley-value-based [96] score that measures each subgraph’s marginal contribution to the prediction. This approach yields explanations that are inherently discrete and human-interpretable, aligning closely with structural motifs such as functional groups in chemistry. However, the combinatorial explosion of subgraph candidates and the computational demands of MCTS can limit its applicability on large graphs or complex models.

CF² [95] provides a unified view of graph explanations by integrating both factual reasoning and counterfactual reasoning. It formulates an optimization objective that balances the requirement that explanations maintain sufficient and necessary. Figure 3.7 illustrates the motivation behind the proposed CF² explanation framework. As shown in Figure 3.7(a), factual reasoning identifies a large subgraph that preserves the original model prediction; however, the resulting explanation often contains redundant atoms and bonds, making the reasoning difficult to interpret. Conversely, Figure 3.7(b) demonstrates counterfactual reasoning, which searches for minimal structural changes that alter the prediction. Although counterfactual explanations reveal sensitive components, they may provide insufficient insight when the identified subgraph is overly small or fragmented. To address these limitations, Figure 3.7(c) presents CF², which integrates both perspectives to achieve a balanced explanation that is simultaneously sufficient and necessary. By combining factual and counterfactual reasoning objectives, CF² produces concise yet faithful subgraphs that more accurately reflect the model’s decision-making process. In this way, CF² bridges factual and counterfactual approaches while also advancing the methodology for evaluating explanation quality.

The methods reviewed above demonstrate the richness of current graph explainability research, span-

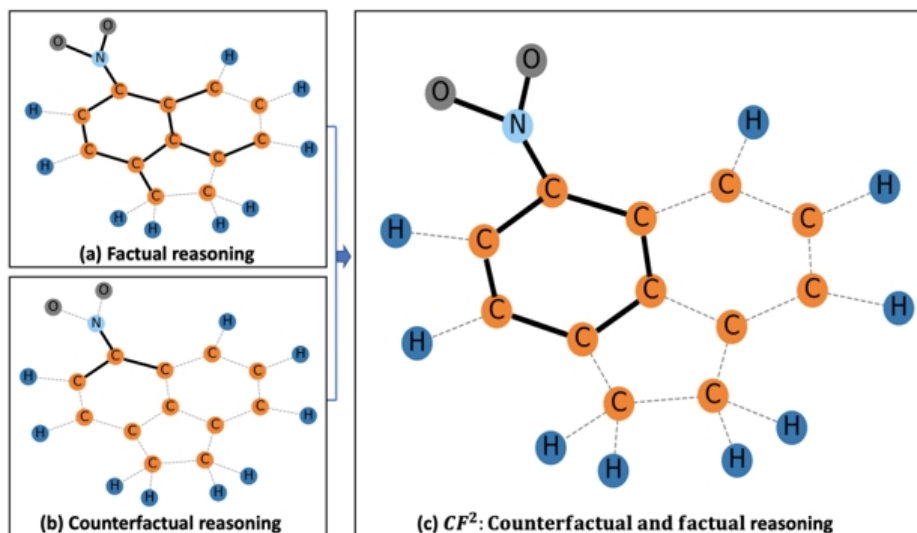


Figure 3.7: Illustration of the limitations of purely factual and purely counterfactual reasoning in graph explanations, and the benefit of combining both of them ([95]).

ning factual, counterfactual, and hybrid reasoning paradigms. Techniques such as GNNExplainer and SubgraphX emphasize identifying substructures that preserve the original prediction, while CF-GNNExplainer highlights minimal perturbations that alter it. CF² further unifies these perspectives by jointly enforcing sufficiency and necessity. Despite their conceptual differences, all of these approaches share a common objective: to extract meaningful explanatory subgraphs that reveal the decision-making behavior of complex GNN models.

However, an important aspect remains underexplored across these methods: the *connectedness* of the generated explanations. Most explainers do not explicitly enforce that the explanatory subgraph forms a connected component. As a result, explanations may fragment into several disconnected pieces, complicating interpretation and reducing their correspondence to real-world structural motifs—particularly in domains such as chemistry, where explainability is often inseparable from the connectedness of functional groups. This observation motivates the next section, where we examine the role of connectedness in graph explanations and discuss why maintaining structural coherence is essential for producing faithful and human-understandable rationales.

3.4 Connectedness - an Overlooked Property in GEPs

A largely overlooked yet fundamental property of graph explanations is the *connectedness* of the identified subgraph. While instance-level explainers aim to extract a subset of nodes and edges that account for a model’s prediction, most existing methods do not enforce that this subset forms a connected component. As a consequence, many explanations produced by factual, counterfactual, or hybrid approaches fragment into several disconnected pieces. Such explanations may still approximate the model’s output, but they often fail to provide a coherent or human-meaningful rationale.

Taking an explanation example given by CF², which gives two disconnected parts as the reason, it is prone to ask questions like “why the explanation is disconnected?”, “which part in the explanation matters most?”, and “does the disconnectedness count in the explanation?” (see Figure 3.8). Due to the absence of consideration in connectedness of explanations, CF² cannot give corresponding illustrations when those questions arise. The same dilemma occurs in GNNExplainer, CF-GNNExplainer, etc., as well. This limits the performance of such approaches.

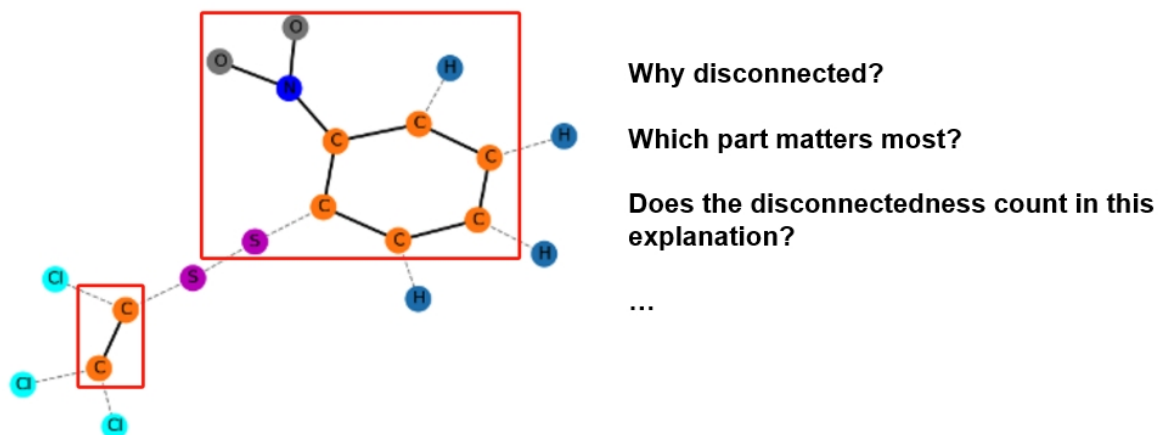


Figure 3.8: An illustration on the limitations for explanations given by existing models that do not consider connectedness.

By contrast, SubgraphX [94] considers the connectedness by searching explanations from the field of connected subgraphs as the reasoning. Unfortunately, it constrains the explanations to be connected; hence is insufficient in flexibility of explanations. The limitation of enforcing connectedness becomes particularly evident in applications where the true explanatory factors are naturally *disconnected*. For example, in semantic or dependency graphs derived from natural language, the core components of a long sentence—such as the subject and the main verb—may be separated by intervening modifiers, adverbial clauses, or auxiliary structures. Although these elements are semantically central to the sentence, they may not form a connected subgraph due to the complexity and hierarchical nature of syntactic constructions. Under the connected-subgraph constraint of SubgraphX, the explainer is forced to include multiple irrelevant intermediate nodes and edges merely to maintain connectivity. As a result, the final explanation becomes inflated with redundant linguistic elements, reducing both explainability and faithfulness. Moreover, it has exponential time complexity; hence is limited to finding very small connected subgraphs.

Lacking sufficient consideration on the connectedness of the reasoning subgraphs arises because that the connectedness of a graph is difficult to optimize within the framework of graph learning, since it is defined as a *discrete* number of the (connected) components of the graph, whereas graph learning uses *continuous* optimization method (e.g., gradient descent).

In order to deeply discuss the connectedness in GEPs, we give following foundations and preliminaries in Chapter 2. Because the graph is undirected, both the adjacency matrix A and the degree matrix D are symmetric, and hence the graph Laplacian L is also real and symmetric. This guarantees that L has a full set of real eigenvalues. Moreover, it can be shown that the Laplacian is positive semidefinite, implying that all of its eigenvalues are nonnegative [53]. A direct consequence of the definition is that the constant vector $[1, \dots, 1]^T$ lies in the nullspace of L , i.e.,

$$L \cdot [1, \dots, 1]^T = 0.$$

Therefore, 0 is always an eigenvalue of the Laplacian. The spectrum of L encodes a wide range of graph characteristics, including connectedness, cluster structure, and diffusion behavior, and plays an essential role in spectral clustering, graph signal processing, and theoretical analysis.

Theorem 1 serves as the theoretical foundation of this chapter. It highlights the intrinsic relationship between the (discrete) connectedness of a graph and its Laplacian matrix.

Theorem 1 (see [97, 98, 99]). *For any simple undirected graph G , the multiplicity of eigenvalue 0 of $L(G)$ is equal to the number K of the components of G .* $\square$

3.4.1 Approximations of Connectedness - Connected-CF²

Based on Theorem 1, we proposed Connected-CF²¹ with an optimizable approximation to the connected state of graphs.

For an input graph G , we want the explanation to be connected, i.e., the subgraph G' has only one connected component. According to Theorem 1, the geometric multiplicity of eigenvalue 0 for the Laplacian matrix $L = L(G')$ is 1. To get the geometric multiplicity, we need to calculate the dimension of the solution space in the linear system $(L - 0 \times I)\eta = L\eta = \mathbf{0}$, where I is the n -dimensional identity vector, and n is the number of nodes in G' . Due to the rank-nullity theorem, the dimension of $\{\eta | L\eta = \mathbf{0}\} \subseteq \mathbb{R}^n$ is $n - \text{rank}(L)$. As a result, the dimension of $\{\eta | L\eta = \mathbf{0}\}$ is 1 if and only if $\text{rank}(L) = n - 1$.

To approach this, we introduced an approximation of graph connectedness as

$$\det(L(G')) + \exp(-\theta \sum_{i,j=1}^n C_{ij}^2), \quad (3.1)$$

where $\det(\cdot)$ denotes the determinant of a matrix and C_{ij} denotes the **first minor** (i.e., the determinant of the submatrix by subtracting the i -th row and j -th column from the original matrix) of $L(G')$. When Eq. (3.1) converges to zero, meaning that $\det(L(G')) = 0$ and $\exp(-\theta \sum_{i,j=1}^n C_{ij}^2) \rightarrow 0$ (note that both terms are non-negative), it follows that $L(G')$ has determinant of 0 and at least one non-zero first minor. Note that the hyperparameter θ is set to be $\theta \gg 1$. This implies $\text{rank}(L(G')) = n - 1$. From Theorem 1, we can conclude that G' is connected. To compute (3.1), we need to calculate one determinant of dimension n and n^2 determinants of dimension $n - 1$. Thus, the computational complexity is $O(n^3 + n^2 \times (n - 1)^3) = O(n^5)$, which is a critical limitation of [29].

3.4.2 Improved Connectedness Representation: Continuous Representation of Connectedness (CRoC)

Although Connected-CF² proposed an approximation of graph connectedness, it suffers from the limitation that it cannot represent the situation where graphs are not connected (or only partially connected). Furthermore, its high computational complexity constrains its employment.

To solve this, we propose an improved continuous connectedness CRoC for graphs². Given a simple undirected graph G with K components, re-index the nodes so that nodes $v_1, v_2, \dots, v_{k_1}$ induce a component of the largest size k_1 , nodes $v_{k_1+1}, \dots, v_{k_1+k_2}$ induce a component of the second largest size k_2 ($k_2 \leq k_1$), and so on. The last component containing v_n is thus the smallest component. Let $e_1 = (1, \dots, 1, 0, \dots, 0)^T \in \{0, 1\}^n$ where the first k_1 elements are 1s and other elements are 0s. Vector e_1 is called the *indicator vector* of the first component. Similarly we can define the indicator vector $e_2 = (e_{2i}) \in \{0, 1\}^n$ for the second component so that $e_{2i} = 1$ if and only if $k_1 + 1 \leq i \leq k_1 + k_2$, and so on. The indicator vector $e_K \in \{0, 1\}^n$ is associated with the last component. We have the next theorem.

Theorem 2. *The set $\{e_1, e_2, \dots, e_K\}$ of indicator vectors as defined above is an orthogonal basis of the eigenspace corresponding to the zero eigenvalue of $L(G)$.*

Proof. It is obvious that $e_i \neq \mathbf{0}$ and $e_i^T e_j = 0$ for any $i \neq j$. Moreover, for any $i = 1, 2, \dots, K$, $L(G)e_i = (D(G) - A(G))e_i = \mathbf{0}$. Since K is the number of components of G , Theorem 1 shows that the multiplicity of zero eigenvalue of $L(G)$ is K . This implies that the eigenspace corresponding to zero eigenvalue of $L(G)$ is of dimension K . Therefore, the set $\{e_1, \dots, e_K\}$ is an orthogonal basis of the eigenspace corresponding to the zero eigenvalue of $L(G)$. $\square$

Let $e_1, \dots, e_K$ be defined as above. We define a vector

$$u(G) = \sum_{i=1}^K i \times e_i. \quad (3.2)$$

¹Part of this subsection has been published as [29]. ©Springer Nature. Reprinted with permission.

²Part of this subsection has been published as [30]. ©2025 IEEE. Reprinted with permission.

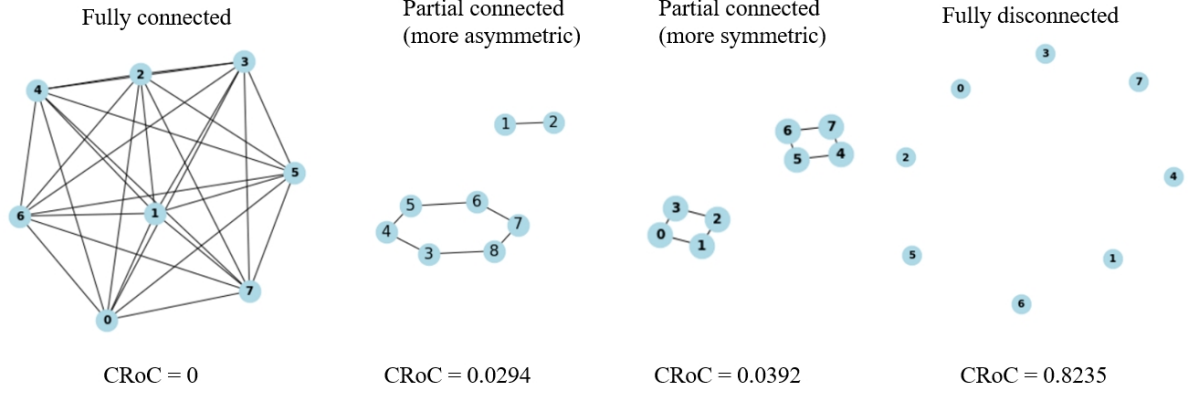


Figure 3.9: CRoC values for some graphs with the same number of nodes. (©2025 IEEE. Reprinted with permission from [30]).

Notice that $u(G) \in \mathbb{R}^n$ is uniquely decided by G . In fact, letting $u(G) = (u_1, \dots, u_n)^T$, we have $u_1 = \dots = u_{k_1} = 1, u_{k_1+1} = \dots = u_{k_1+k_2} = 2, \dots, u_n = K$. Let $\bar{u} = \frac{1}{n} \sum_{i=1}^n u_i$ be the average of u_i . We define

$$\text{CRoC}(G) = \frac{4 \sum_{i=1}^n (u_i - \bar{u})^2}{\sum_{i=1}^n i^2} \quad (3.3)$$

to be the CRoC of G (see Figure 3.9 for some examples). Notice that, like the variance, CRoC estimates how far the vector $u(G)$ is from a mean vector. It is easy to see that $\text{CRoC}(G) = 0$ if and only if G is connected (i.e., $K = 1$). Moreover, we can prove the following theorems.

Theorem 3. For two n -nodes graphs G_1 and G_2 with K_1 size- d_1 and K_2 size- d_2 components, respectively, $K_1 > K_2$ implies $\text{CRoC}(G_1) > \text{CRoC}(G_2)$.

Proof. Let $u(G_1) = (u_1, u_2, \dots, u_n)^T$, where $n = d_1 K_1$. By definition, $u_1 = \dots = u_{d_1} = 1, u_{d_1+1} = \dots = u_{2d_1} = 2, \dots, u_{n-d_1+1} = \dots = u_n = K_1$. Therefore,

$$\bar{u}(G_1) = \frac{d_1(1 + 2 + \dots + K_1)}{n} = \frac{K_1 + 1}{2}.$$

Hence

$$\text{CRoC}(G_1) = \frac{4d_1 \sum_{i=1}^{K_1} (i - \frac{K_1+1}{2})^2}{\sum_{i=1}^n i^2} = \frac{2(K_1^2 - 1)}{2n^2 + 3n + 1}.$$

Analogously, $\text{CRoC}(G_2) = \frac{2(K_2^2 - 1)}{2n^2 + 3n + 1} < \text{CRoC}(G_1)$. $\square$

Remark 1. Notice that $\text{CRoC}(G_i) \rightarrow 1$ when $K_i \approx n$ and $n \rightarrow \infty$ for the graphs considered by Theorem 3.

Remark 2. Theorem 3 shows graphs containing more components with similar symmetry exhibit larger CRoC values. However, the converse may not hold, i.e., a larger CRoC may not imply more components. For instance, consider two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where $V_1 = V_2 = \{v_1, v_2, v_3, v_4\}$, $E_1 = \{(v_1, v_2), (v_3, v_4)\}$, and $E_2 = \{(v_2, v_3), (v_3, v_4)\}$. It is easy to verify that $\frac{2}{15} = \text{CRoC}(G_1) > \text{CRoC}(G_2) = \frac{1}{10}$ but $K_1 = K_2$. This example (see Figure 3.10) illustrates that, as expected, CRoC can quantify component size asymmetry (variance) in graphs with identical node counts and component counts, in addition to evaluating the connectedness.

The proof for Theorem 3 also shows the next theorem.

Theorem 4. For two graphs G_1 and G_2 with K_1 size- d_1 and K_2 size- d_2 components, respectively, $K_1 = K_2$ and $d_1 < d_2$ implies $\text{CRoC}(G_1) > \text{CRoC}(G_2)$. $\square$

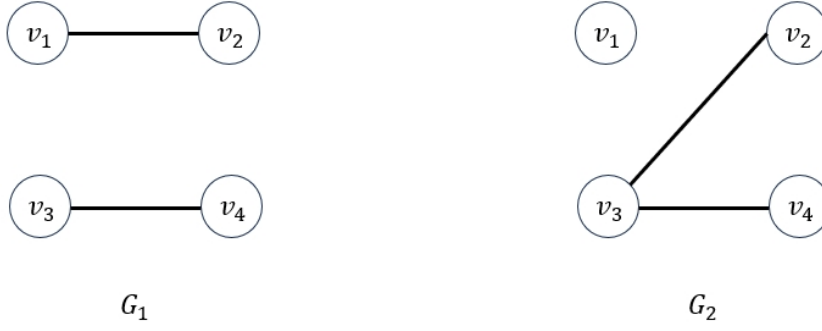


Figure 3.10: An illustration of the examples in the above Remark 2.

Although the above theorems clarify the definition of CRoC and some intrinsic properties, calculating Eq. 3.3 requires discretely constructing the basis mentioned in Theorem 2, which is not directly applicable in the context of continuous optimization. Therefore, we propose Algorithm 1 to calculate CRoC from a set of orthogonal eigenvectors corresponding to the eigenvalue 0 for $L(G)$, where the ∞ -norm $\|\cdot\|_\infty$ and the 1-norm $\|\cdot\|_1$ calculate the maximum absolute value among elements and the sum of the absolute values of the elements of a vector, respectively.

Algorithm 1 Finding CRoC(G)

Input: A simple undirected graph G

- 1: Calculate the Laplacian matrix $L(G)$ of G .
- 2: Calculate a set $S = \{e_1, \dots, e_k\}$ of the orthonormal eigenvectors corresponding to eigenvalue 0 for $L(G)$ (see the next remark).
- 3: $e_i = e_i / \|e_i\|_\infty$ for all i .
- 4: Sort S in nondecreasing order of $\|e_i\|_1$.

Output: The value calculated by Eq. 3.2 and Eq. 3.3.

Remark. For Step 2, we use `numpy.linalg.eigh()` from the NumPy (<https://numpy.org/>) package, where eigenvalues and eigenvectors are computed using LAPACK [100]. Figure 3.11 illustrates Algorithm 1 for a given graph. Notice that the time complexity of Algorithm 1 is $O(n^3)$, dominated by Step 2.

We note that the eigenspace corresponding to eigenvalue 0 for $L(G)$ may have more than one orthonormal basis. Therefore Step 2 in Algorithm 1 may find an orthogonal basis different from the one shown in Theorem 2. Nevertheless, it is interesting to observe from empirical studies that Algorithm 1 could find the preferred basis in many cases. Figure 3.12 compares the value of Eq. 3.3 based on the indicator basis and the output by Algorithm 1 for 1,000 random graphs of 20 nodes generated by the Erdős–Rényi model [101, 102]. The probability p for an edge generation follows a normal distribution with mean $\mu_p = 0.03$ and standard deviation $\sigma_p = 0.01$, i.e., $p \sim \mathcal{N}(\mu_p, \sigma_p^2)$. It can be observed that in most cases, the output of Algorithm 1 is within the value of Eq. 3.3 and three times of that value.

To further demonstrate that CRoC (here and thereafter, CRoC refers to the results calculated from Algorithm 1) shares similar properties shown by Theorem 3, we conduct several empirical studies.

Firstly, we construct graphs of 100 nodes and increase the number K of the components from 1 to 100 in a symmetric manner such that the sizes of components differ the minimum. For $K = 2$, each component contains 50 nodes. For $K = 3$, the three components contain 33, 33, and 34 nodes, respectively. And so on. For $K = 100$, every component contains a single node. Red circles in Figure 3.13 show the results. It can be observed that CRoC increases as the number of components increases.

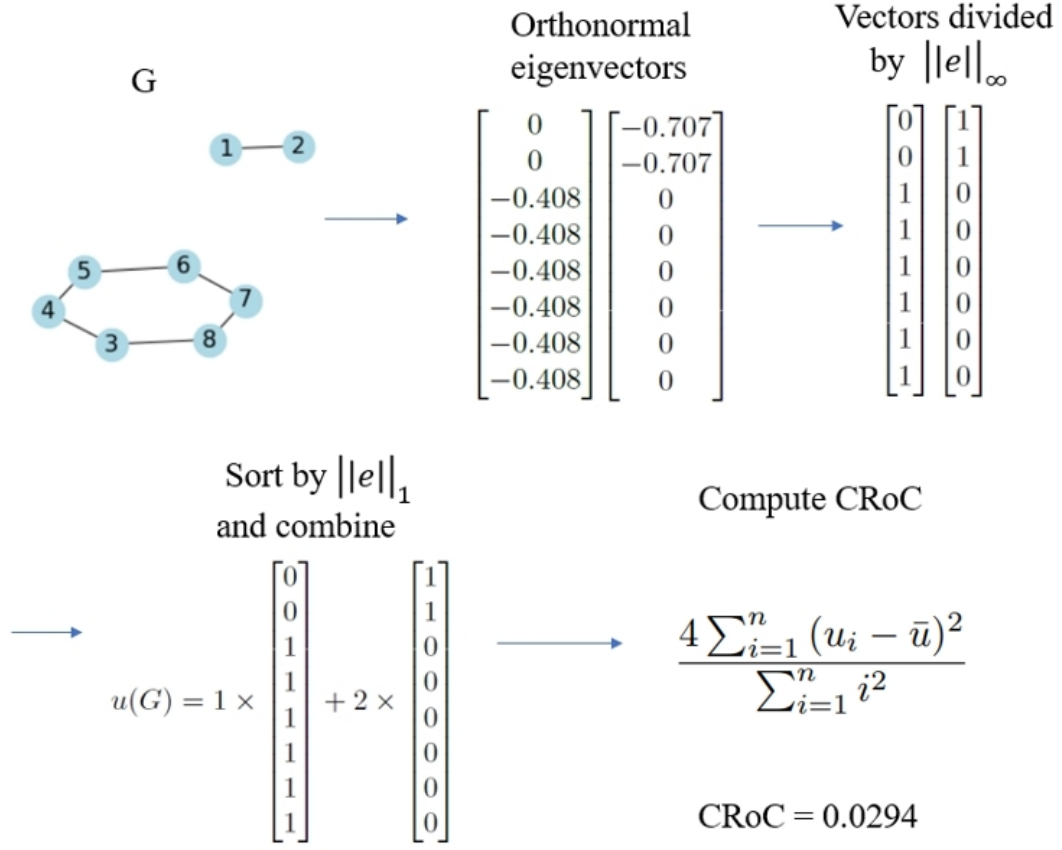


Figure 3.11: An illustration on Algorithm 1 working against a given graph (notice that we do not really need to re-index nodes after sorting). (©2025 IEEE. Reprinted with permission from [30]).

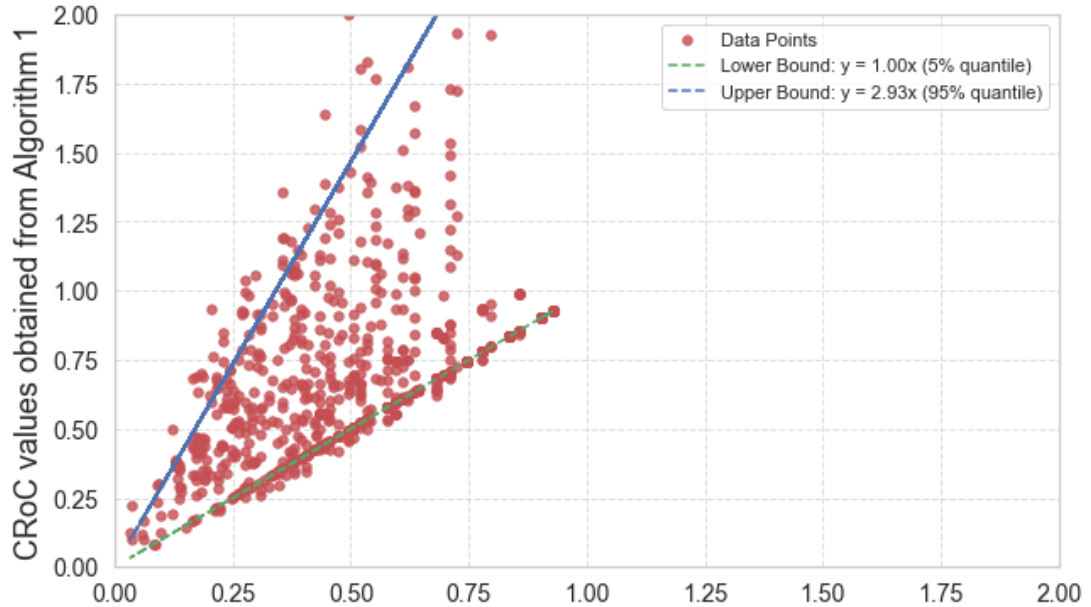


Figure 3.12: A comparison between the value of Eq. 3.3 and the output of Algorithm 1 for 1,000 random graphs. It can be observed that the latter is usually bounded by the former and three times of it. (©2025 IEEE. Reprinted with permission from [30]).

Next, we construct graphs containing 100 nodes and increase the number K of components from 1 to 100 in the “most asymmetric” manner. For $K = 2$, one component contains 99 nodes, and the other is an isolated node. For $K = 3$, one component contains 98 nodes, and the other two are isolated nodes. And so on. For $K = 100$, each component contains a single node. Blue triangles in Figure 3.13 show that CRoC increases as the number K of components increases as well.

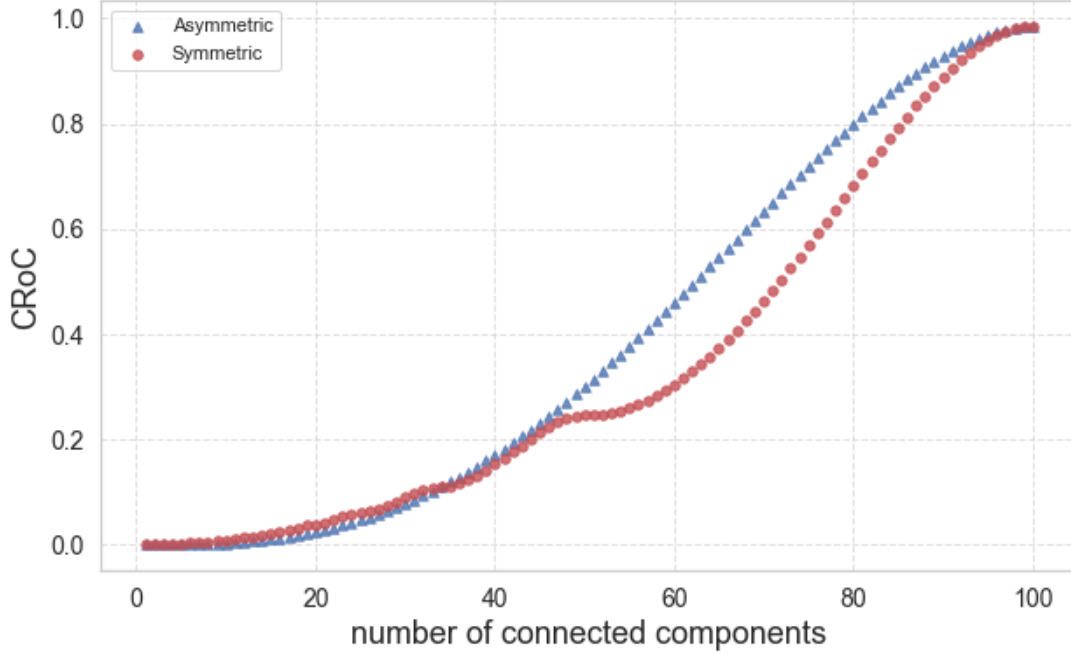


Figure 3.13: Correlations between CRoC (calculated from Algorithm 1) and the number of connected components. Red circles represent the symmetric graphs, whereas blue triangles denote the asymmetric graphs. The total number of nodes of the graph is fixed to 100. (©2025 IEEE. Reprinted with permission from [30]).

Then, we construct a graph with 100 nodes and 2 connected components. We systematically vary the ratio r of the sizes of the two components, starting from the most asymmetric case $r = 1/99$ and gradually increasing the symmetry until it reaches the most symmetric case $r = 1$. This allows us to explore how CRoC changes with respect to the symmetry between components. Figure 3.14 highlights that, for this fixed number of components, the CRoC increases as the symmetry between the components increases. Notably, a comparison between Figure 3.13 and Figure 3.14 shows that although both the component count and symmetry of components influence the CRoC, the effect of the component count to CRoC is far more pronounced.

Finally, to empirically study CRoC in real-world scenarios, we use one hundred random networks, all with 100 nodes and generated by the Erdős–Rényi model [101, 102]. The probability p for an edge generation follows a normal distribution with mean $\mu_p = 0.03$ and standard deviation $\sigma_p = 0.01$, i.e., $p \sim \mathcal{N}(\mu_p, \sigma_p^2)$. For each network, we calculate its CRoC value by Algorithm 1, the number K of components, and the variance in its component sizes defined by $\text{Var} = \frac{\sum_{i=1}^K (n_i - \bar{n})^2}{K}$ where n_i is the size of component i and $\bar{n}$ is their mean value. Figure 3.15 and Figure 3.16 show the results of this study, including the relationship between CRoC and the number of components, as well as between CRoC and the variance of component sizes, respectively. The results suggest that graphs with a greater number of components generally exhibit higher CRoC values. Additionally, increased level of symmetry among components tends to result in larger CRoC values. This suggests we may use it in the explainer (see the next section).

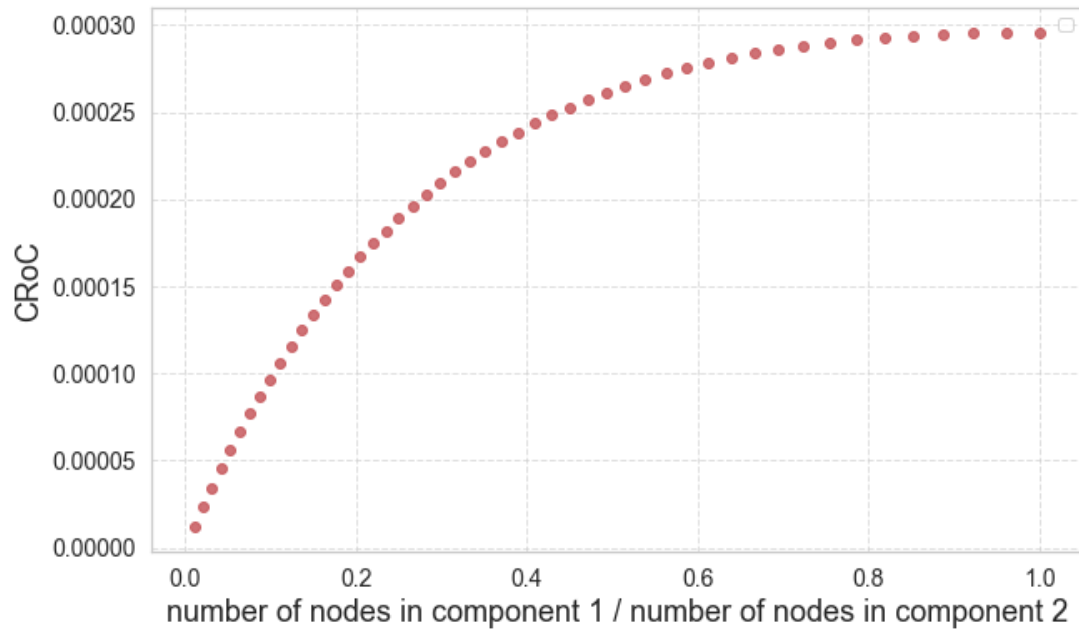


Figure 3.14: Correlations between CRoC (calculated from Algorithm 1) and the ratio of the sizes of the two components. The total number of nodes of the graph is fixed to 100. (©2025 IEEE. Reprinted with permission from [30]).

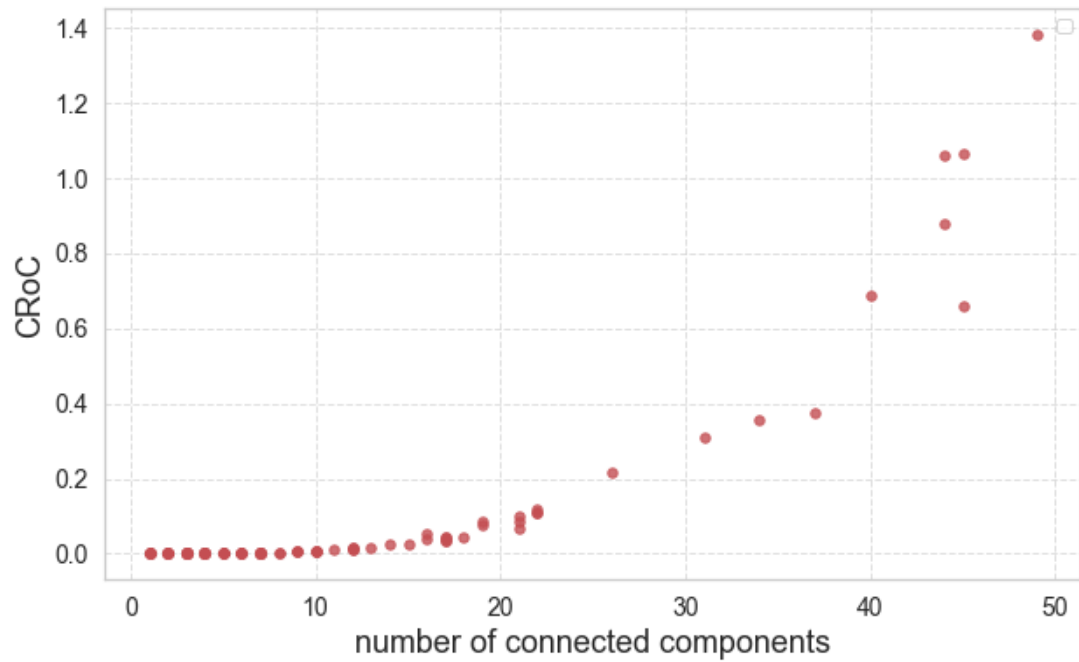


Figure 3.15: Correlations between CRoC (calculated by Algorithm 1) and the number of components for random Erdős–Rényi networks. (©2025 IEEE. Reprinted with permission from [30]).

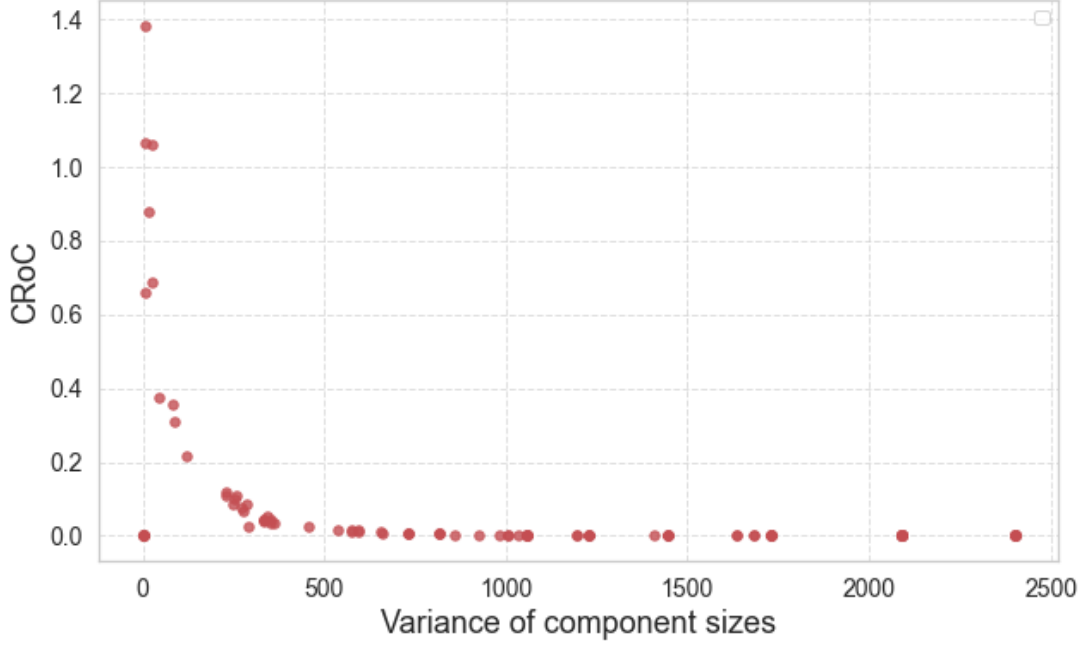


Figure 3.16: Correlations between CRoC (calculated by Algorithm 1) and the variance of component sizes for random Erdős–Rényi networks. (©2025 IEEE. Reprinted with permission from [30]).

To end, we compare the CRoC metric obtained from Algorithm 1 with several existing approximations for graph connectedness.

Approximation from Eq. (3.1) [29] can only evaluate whether a graph is connected with $O(n^5)$, which can be achieved by CRoC at zero value with a significantly lower complexity of $O(n^3)$.

Another approximation is a variant from heat kernel [103] to approximate the number of components in graphs by

$$\sum_{i=1}^n \exp\left(-\frac{\lambda_i^2}{\sigma^2}\right), \quad (3.4)$$

where λ_i are the eigenvalues of the Laplacian matrix $L(G')$, and σ is a small positive parameter controlling the sensitivity to eigenvalues near zero. When σ is sufficiently small, (3.4) is only sensitive to near-zero eigenvalues, since if $\lambda_i = 0$, $\exp(-\frac{\lambda_i^2}{\sigma^2}) = 1$; otherwise, $\exp(-\frac{\lambda_i^2}{\sigma^2})$ is close to 0. Therefore, if σ is carefully selected,

$$\sum_{i=1}^n \exp\left(-\frac{\lambda_i^2}{\sigma^2}\right) \approx \text{Number of Components}.$$

Note that (3.4) requires computing eigenvalues, which has the same computational complexity $O(n^3)$ as CRoC. However, CRoC offers additional capabilities, such as evaluating the asymmetry between different components.

Table 3.2: Comparison of connectedness approximations.

Method	Connectedness	Component Symmetry	Complexity
Connected-CF2 [29]	✓	–	$O(n^5)$
Heat Kernel Variant [103]	✓	–	$O(n^3)$
CRoC [30]	✓	✓	$O(n^3)$

In summary, compared to existing approximations, CRoC not only surpasses their computational efficiency but also provides enhanced capabilities (see Tabel 3.2). Therefore, we consider CRoC is a more versatile and powerful tool for analyzing graph structures, and will use it in the proposed explainer framework.

3.5 Explainer Framework with Connectedness and Improved Connectedness Representation

In this section, we will combine previous representations into explainer framework for GEPs to give explanations.

Given a GNN model $f_\theta : \mathcal{G} \rightarrow \mathcal{Y}$ and a graph $G \in \mathcal{G}$, we consider an *explanation* for them as a subgraph G' of G such that $f_\theta(G') \approx f(G)$. Following Tan et al. [104], we also handle an explanation by an edge mask matrix $M \in \{0, 1\}^{n \times n}$, i.e., the spanning subgraph G_M of G with adjacency matrix $A(G_M) = M \odot A(G)$ is an explanation for f_θ and G , where $\odot$ denotes element-wise multiplication.

An exact definition of $f_\theta(G') \approx f(G)$ is called a *reasoning*. In the literature, factual reasoning and counterfactual reasoning are widely used. The former tries to find *sufficient* explanations, whereas the latter tries to find *necessary* explanations. More precisely, factual reasoning finds an edge mask matrix M such that $f_\theta(G_M)$ is as close as possible to $f(G)$, whereas counterfactual reasoning finds M such that $f_\theta(G_{\bar{M}})$ is as far as possible from $f(G)$, where $\bar{M} = \mathbf{1}^{n \times n} - M$ denotes the edge mask matrix inverse to M . Both have their pros and cons. While most explainer models use one of them, Tan et al. [104] reported that combining both excels others that use only one. Following their framework, we also use both reasonings.

We want to find compact, effective and connected explanations. Following [104], we define the compactness of an explanation M by $\|M\|_0$, i.e., the number of non-zero values in M . To define the effectiveness, we need more details on GNN. In general, a GNN model outputs a feature $\hat{y} \in \mathcal{Y}$ that maximizes the probability P_θ for a given graph G , i.e.,

$$\hat{y} = \underset{y \in \mathcal{Y}}{\arg \max} \{P_\theta(y|A(G))\},$$

where P_θ can be obtained by a softmax layer before the output. Tan et al. [104] calls M an effective factual reasoning if

$$P_\theta(\hat{y}|A(G_M)) > \max_{y \in \mathcal{Y}, y \neq \hat{y}} P_\theta(y|A(G_M)),$$

and an effective counterfactual reasoning if

$$P_\theta(\hat{y}|A(G_{\bar{M}})) < \min_{y \in \mathcal{Y}, y \neq \hat{y}} P_\theta(y|A(G_{\bar{M}})).$$

For estimating the connectedness, we use the output of Algorithm 1 applied to G_M , denoted by $\text{CRoC}(M)$. Ideally we want to find an M^* that solves the following problem.

$$\begin{aligned} (P) \quad & \text{minimize} \quad \|M\|_0 \\ & \text{s.t., } P_\theta(\hat{y}|A(G_M)) > \max_{y \in \mathcal{Y}, y \neq \hat{y}} P_\theta(y|A(G_M)), \\ & P_\theta(\hat{y}|A(G_{\bar{M}})) < \min_{y \in \mathcal{Y}, y \neq \hat{y}} P_\theta(y|A(G_{\bar{M}})), \\ & \text{CRoC}(M) = 0. \end{aligned}$$

For [29], the CRoC term is its corresponding approximation as Eq. 3.1.

This optimization problem is difficult to solve due to its combinatorial nature [104]. We relax it to a continuous optimization problem instead. For the objective function, we relax the domain of the elements of M from $\{0, 1\}$ to $[0, 1]$. Accordingly, we substitute the L_0 -norm $\|M\|_0$ of M with the differential $\|M\|_2$.

Compared with the L_1 -norm applied by Tan et al. [104], which is non-differentiable at zero, L_2 -norm is differentiable everywhere. For the reasoning constraints, we follow [104] to relax them with

$$L_f = \text{ReLU}(\zeta - (P_\theta(\hat{y}|A(G_M)) - \max_{y \in \mathcal{Y}, y \neq \hat{y}} P_\theta(y|A(G_M))))$$

and

$$L_c = \text{ReLU}(\zeta - (\min_{y \in \mathcal{Y}, y \neq \hat{y}} P_\theta(y|A(G_{\bar{M}})) - P_\theta(\hat{y}|A(G_{\bar{M}})))),$$

respectively, where ζ is a pre-defined value (we used 0.5 in the empirical study). For the connectedness, CRoC is differential when M is relaxed to real-valued. Finally, we consider the Lagrangian Relaxation problem (P) as the follows:

$$(P') \text{ minimize } \|M\|_2 + \lambda(\alpha L_f + (1 - \alpha)L_c) + \beta \text{CRoC}(M)$$

Parameters λ and β adjust the trade-off between effectiveness and connectedness. Parameter α adjusts the trade-off between factual and counterfactual reasoning. To get an integer solution M^* to (P) from a relaxed solution M to (P'), we simply round the elements to 1 if it is greater or equal to 1/2, otherwise 0.

We remark that the proposed framework can adapt to node classification in a straightforward manner.

Empirical Studies

In this section, we first introduce the datasets, followed by an overview of the baseline methods. Then we describe the experimental setup, and finally we present the experimental results along with discussion.

Datasets and Baselines

We use two synthetic datasets and one real-world dataset, all of which contain ground-truth labels. The synthetic datasets Tree-Cycles and BA-shapes were provided by [84]. The real-world dataset Mutag₀ is a subset of the Mutag dataset [105] extracted by Tan et al. [104]. It contains only mutagenic compounds with benzene-NO₂ and non-mutagenic compounds without benzene-NO₂. See Table 3.3 for the detail.

Table 3.3: Datasets used in the empirical study. (©2025 IEEE. Reprinted with permission from [30]).

Datasets	#graph	#avg node	#avg edge	#class	#feature	task
BA-shapes	1	700	4100	4	-	node
Tree-Cycles	1	871	1950	2	-	node
Mutag ₀	2301	31.74	32.54	2	14	graph

For the baselines, we want them to find explanations of subgraphs and do not have “critical” conditions on datasets, e.g., PGExplainer [106] requires extra motif information to generate reasoning. In addition, SubgraphX [94] is not included in the baselines due to its prohibitive computational cost. As the result, the following models are chosen.

- **GNNExplainer** [84]: It finds explanations by maximizing the mutual information.
- **CF-GNNExplainer** [85]: It is an extended version of GNNExplainer with counterfactual reasoning.
- **CF²** [104]: It considers both factual reasoning and counterfactual reasoning.
- **Connected-CF²** [29]: It adds connectedness consideration to CF².

Setup

The experiments consist of two phases: (1) training a fixed baseline GNN model for classification, and (2) solving the optimization problem to find corresponding explanations. For the baseline model, we use Graph Convolutional Network (GCN) [107]. GCN is a popular GNN with the aggregate and update rule presented in Eqs. (2.1) and (2.2). We use a fixed GCN as the base GNN model, which consists of three convolutional layers of hidden dimension 16. After each layer, except for the final one, we apply the ReLU activation function, and a softmax layer is used before the final classification. The learning rate is set to 0.001, with the training set comprising 80% of the total dataset. Table 3.4 provides detailed information on the base GCN model used in the experiments.

Table 3.4: Detail on the trained basic GCN. (©2025 IEEE. Reprinted with permission from [30]).

Datasets	BA-Shapes	Tree-Cycles	Mutag ₀
Epochs	3000	3000	1000
Accuracy (in %)	97.86	98.29	98.05

In the explanation phase, we keep all hyperparameters in CRoCExplainer consistent across datasets. The value of λ is set to 500 for BA-Shapes and Tree-Cycles, and 1000 for the Mutag₀ dataset. We set α to 0.6 to prioritize factual reasoning over counterfactual reasoning, and β is set to 100.

Results and Discussion

Since datasets Tree-Cycles, BA-Shapes, and Mutag₀ all contain ground-truth labels, we use common classification metrics Accuracy, Precision, Recall, and F₁ scores to evaluate the explanations on the test datasets for different methods. Accuracy, Precision, Recall and F₁ are defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}},$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}},$$

here, TP, TN, FP and FN correspond to an edge both in the given explanation and the ground-truth, an edge neither in the explanation nor in the ground-truth, an edge in the explanation while not in the ground-truth, and an edge not in the explanation while in the ground-truth, respectively. Note that Accuracy, Precision, Recall, and F₁ are computed separately for each individual explanation, we extend these metrics in **bold** form to evaluate model performance across entire datasets. Specifically, we calculate dataset-level aggregates by taking the mean values across all instances:

$$\mathbf{Accuracy} = \frac{1}{|\mathcal{F}|} \sum_{p \in \mathcal{F}} \text{Accuracy}(e),$$

$$\mathbf{Precision} = \frac{1}{|\mathcal{F}|} \sum_{p \in \mathcal{F}} \text{Precision}(e),$$

$$\mathbf{Recall} = \frac{1}{|\mathcal{F}|} \sum_{p \in \mathcal{F}} \text{Recall}(e),$$

$$\mathbf{F}_1 = \frac{1}{|\mathcal{F}|} \sum_{p \in \mathcal{F}} F_1(e),$$

where $p \in \mathcal{F}$ denotes an explanation generated by the Explainer for a graph G in the target dataset D , and $|\mathcal{F}|$ represents the total number of explanations. This aggregation strategy provides a unified performance measure across all input instances.

Table 3.5 presents the results of the explanations for each dataset in comparison to the ground truth. Across all datasets, CRoCExplainer demonstrates high *Accuracy* and F_1 scores, highlighting its strong and robust explanatory capability. Notably, in Tree-Cycles dataset, CRoCExplainer outperforms other methods. Moreover, CRoCExplainer often achieves a higher *Recall* value, indicating its tendency to provide explanations that align closely with ground-truth data (see Figure 3.17). This performance results from CRoCExplainer’s consideration of connectedness in its explanations.

Table 3.5: Comparison of different methods, where **Acc**, **Pre**, **Re** and **F₁** denote the dataset-level metrics Accuracy, Precision, Recall and F₁, respectively, in %. (©2025 IEEE. Reprinted with permission from [30]).

Models	BA-Shapes				Tree-Cycles				Mutag ₀			
	<i>Acc</i> %	<i>Pr</i> %	<i>Re</i> %	<i>F₁</i> %	<i>Acc</i> %	<i>Pr</i> %	<i>Re</i> %	<i>F₁</i> %	<i>Acc</i> %	<i>Pr</i> %	<i>Re</i> %	<i>F₁</i> %
GNNExplainer	95.25	60.08	60.08	60.08	92.78	68.06	68.06	68.06	96.96	59.71	85.17	68.85
CF-GNNExplainer	94.39	67.19	54.11	56.79	90.27	87.40	47.45	59.10	96.91	66.09	39.46	47.39
CF ²	96.37	73.15	68.18	66.61	93.26	84.92	73.84	75.69	97.34	65.28	88.59	72.56
Connected-CF ²	95.22	56.73	81.51	62.00	93.76	68.71	97.45	78.60	96.54	49.87	96.30	63.45
CRoCExplainer	94.99	58.33	80.14	62.28	93.79	69.37	96.53	78.62	96.53	49.7	95.37	63.18

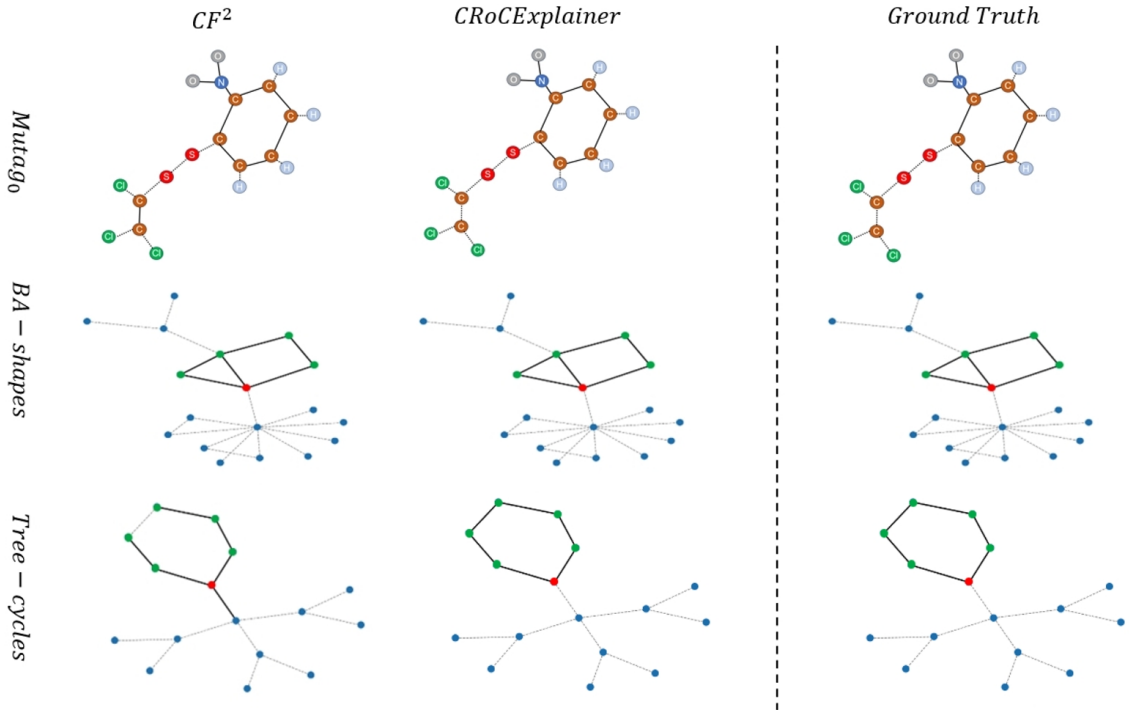


Figure 3.17: An example of the explanations (in bold edges) tested on the Mutag₀, BA-shapes, and Tree-cycles datasets. While CF² produces a more disconnected subgraph, CRoCExplainer tends to find the connected one. (©2025 IEEE. Reprinted with permission from [30]).

Note that incorporating connectedness-related considerations, such as those in CRoCExplainer, does not strictly guarantee that the resulting explanation is fully connected. In practice, the explanation emerges from a trade-off among multiple penalty terms in the optimization objective. As a result, suboptimal or

undesired structures may still arise. Figure 3.18 illustrates several failure cases produced by CRoCExplainer, including an explanation that is not connected. Our setting introduces a quantifiable connectedness penalty, which provides a principled basis for analyzing failure cases in a more fine-grained manner. Moreover, the explicit formulation of this penalty allows flexible control via hyperparameters, offering the potential to adapt explanation behavior to different future requirements.

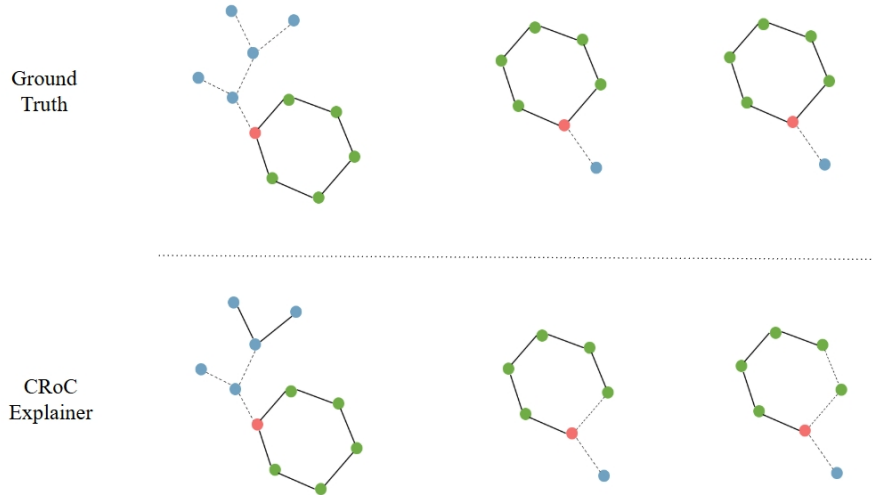


Figure 3.18: Some failure cases given by CRoCExplainer for Tree-Cycles.

Finally, we provide a running time comparison between CRoCExplainer and Connected-CF². The experiments were conducted on the same machine with an AMD Ryzen 7 5800H processor, 32 GB DDR4 RAM and an NVIDIA GeForce RTX 3060 GPU. As shown in Table 3.6, CRoCExplainer significantly reduces computational complexity compared to Connected-CF².

Datasets	BA-Shapes	Tree-Cycles	Mutag ₀
Connected-CF ²	13572.9	1330.4	1647.7
CRoCExplainer	3799.7	1051.1	480.7

Table 3.6: Running time(s) comparison between CRoCExplainer and Connected-CF² on different datasets. (©2025 IEEE. Reprinted with permission from [30]).

In this section, we introduce a novel metric CRoC to measure the connectedness in a graph, which can be optimized using continuous optimization methods. Based on this metric, we propose CRoCExplainer to generate explanations for GNN predictions. Empirical results demonstrate that CRoCExplainer provides highly effective explanations with greater efficiency, highlighting the effectiveness of the CRoC metric.

3.6 Conclusion of the Chapter

This chapter investigated the problem of explainability in graph learning and examined why interpretability is particularly essential in applications involving high-stakes decisions, such as scientific discovery, drug design, security analysis, and safety-critical infrastructure. In these domains, understanding *why* a graph neural network (GNN) arrives at a particular prediction can be as important as the prediction itself; without clarity regarding the underlying reasoning process, it becomes difficult for practitioners to validate model outputs, ensure reliability, or build trust in real-world deployments.

A central focus of this chapter was the role of graph structural properties in explanation quality, especially the often-overlooked notion of connectedness. Many existing explanation methods primarily emphasize node-level or edge-level relevance but frequently overlook whether the resulting explanations form coherent and meaningful substructures. By explicitly incorporating connectedness into the explanation process, the proposed methods—Connected-CF² and CRoCExplainer—are able to produce explanations that are both more faithful to the model’s decision process and more interpretable to human analysts. This structural perspective highlights the importance of reasoning not only about which components of a graph matter but also how they relate and interact as a whole.

Looking forward, the work in this chapter suggests that graph explainability research may benefit from increasingly fine-grained and structurally aware approaches. Graphs encode rich patterns that extend beyond individual nodes or edges—such as motifs, cycles, communities, and hierarchical organization—offering many directions for developing explanation methods capable of capturing higher-level relational semantics. As graph learning continues to expand into more complex and sensitive areas, designing explainability techniques that reflect the unique structural characteristics of graph data may become essential for improving transparency, usability, and trust in graph-based AI systems.

Chapter 4

Graph and Quantum Computing

The growing interest in quantum computing has opened new possibilities for enhancing machine learning models, including those designed for graph-structured data. Quantum effects such as superposition, entanglement, and high-dimensional Hilbert space representations offer conceptual advantages that may extend the expressive capacity of classical Graph Neural Networks (GNNs). To situate these developments within the context of graph learning, this chapter provides a structured overview of the foundations, methods, and current limitations of quantum computation as they relate to graphs.

We begin in Section 4.1 with the essential principles of quantum computing, introducing qubits, quantum gates, and measurement in a manner focused on their computational relevance. Section 4.2 then presents the basic ideas of Quantum Machine Learning (QML), explaining how quantum circuits can be trained and how classical data can be embedded into quantum states. Building on these foundations, Section 4.3 introduces Quantum Graph Neural Networks (QGNNs) and explains how graph topology can be mapped onto qubits and processed through parameterized quantum circuits. Section 4.4 discusses the practical bottlenecks of today’s quantum hardware, highlighting the gap between theoretical potential and real-world feasibility, and presenting our contribution to quantum-level design automation for superconducting chips. Finally, Section 4.5 concludes.

4.1 Introduction to Basic Quantum Computing

4.1.1 Quantum States

Quantum computers differ fundamentally from classical ones by replacing binary bits with qubits (a qubit or quantum bit is a basic unit of quantum information). While a classical bit must be definitively 0 or 1, a qubit exists in a quantum superposition of two fundamental quantum states:

- The $|0\rangle$ state, represented mathematically as $\begin{pmatrix} 1 \\ 0 \end{pmatrix}$.
- The $|1\rangle$ state, represented as $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

These correspond physically to the ground state and excited state, respectively, within a two-level quantum device. Unlike a classical bit’s definite state, the general state of a single qubit $|\psi\rangle$ is a linear combination of $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where, $\alpha, \beta \in \mathbb{C}$ are complex numbers called probability amplitudes. They are constrained by the normalization condition $|\alpha|^2 + |\beta|^2 = 1$, ensuring the total probability of finding the qubit in either $|0\rangle$ or $|1\rangle$ upon measurement equals 1. This quantum state has an alternative parameterization using angles:

$$|\psi\rangle = e^{i\gamma} \left(\cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi} \sin\left(\frac{\theta}{2}\right)|1\rangle \right),$$

where θ, ϕ are real numbers with $0 \leq \theta \leq \pi$ and $0 \leq \phi < 2\pi$, and γ is a global phase with no observable effect.

An n -qubit quantum register occupies a 2^n -dimensional Hilbert space¹ obtained through the tensor product of single-qubit spaces. Its most general state is:

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} \alpha_x |x\rangle, \quad (4.1)$$

where $|x\rangle$ represents computational basis states (e.g., $|00 \cdots 0\rangle, |00 \cdots 1\rangle, \dots, |11 \cdots 1\rangle$), and $\alpha_x \in \mathbb{C}$ are complex probability amplitudes satisfying the normalization condition $\sum_x |\alpha_x|^2 = 1$.

The tensor product $\psi_1 \otimes \psi_2$ formally combines quantum states from separate systems, with compact notations like $|\psi_1 \psi_2\rangle$ used for multi-qubit states. This efficient representation avoids explicit listing of all 2^n basis states. However, when the composite state cannot be decomposed into individual tensor products:

$$|\psi\rangle \neq |\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots \otimes |\phi_n\rangle, \quad (4.2)$$

the system exhibits quantum entanglement. A quintessential example is the Bell state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle), \quad (4.3)$$

which violates separability since no coefficients $a, b, c, d \in \mathbb{C}$ can satisfy $|\Phi^+\rangle = (a|0\rangle + b|1\rangle) \otimes (c|0\rangle + d|1\rangle)$. The Bell state demonstrates that quantum systems can occupy states where:

$$\begin{array}{ccc} \text{joint system} & & \text{product of individual states} \\ |\psi\rangle & \neq & |\psi_1\rangle \otimes |\psi_2\rangle \end{array}$$

This non-separability is the mathematical signature of entanglement, creating uniquely quantum correlations that form the basis for quantum computing advantages. Separately, amplitude encoding provides a method to embed classical data into quantum states. For a vector $\mathbf{x} \in \mathbb{R}^N$ where $N = 2^n$:

$$|\mathbf{x}\rangle = \frac{1}{\|\mathbf{x}\|_2} \sum_{i=1}^N x_i |i\rangle, \quad (4.4)$$

the Euclidean norm $\|\mathbf{x}\|_2 = \sqrt{\sum_i x_i^2}$ ensures quantum state normalization. This enables the representation of classical information through quantum probability amplitudes.

4.1.2 Quantum Gates and Quantum Circuits

Quantum operations on qubits are mathematically represented by **unitary matrices**, universally termed quantum gates. These gates physically alter qubit states while preserving quantum coherence. Single-qubit gates operate on the Bloch Sphere (see Figure 4.1), with fundamental operations including:

- **Pauli Gates:**

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad (4.5)$$

performing π -radian rotations around respective axes.

- **Arbitrary Rotation Gates:**

$$R_x(\theta) = e^{-i\theta X/2}, R_y(\theta) = e^{-i\theta Y/2}, R_z(\theta) = e^{-i\theta Z/2} \quad (4.6)$$

enabling parameterized rotations by angle θ .

¹A Hilbert space is a complete vector space with an inner product, which generalizes Euclidean space to infinite dimensions. In quantum mechanics, a Hilbert space is a complex vector space that serves as the mathematical space for all possible states of a quantum system.

- **Hadamard Gate:**

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad (4.7)$$

generating superposition states from computational basis states.

- **Controlled-NOT (CNOT) Gate:**

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (4.8)$$

entangles qubits by flipping the target qubit conditional on the control qubit's state.

- **SWAP Gate:**

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4.9)$$

exchanges quantum states between two qubits.

These gates constitute a universal set: when combined, they can approximate any quantum operation with arbitrary precision. Quantum algorithms are executed via quantum circuits—structured sequences of these gates applied to qubit registers, enabling state evolution toward computational solutions. The CNOT gate's entangling capability is particularly crucial, serving as the primary mechanism for generating quantum entanglement in algorithmic processes.

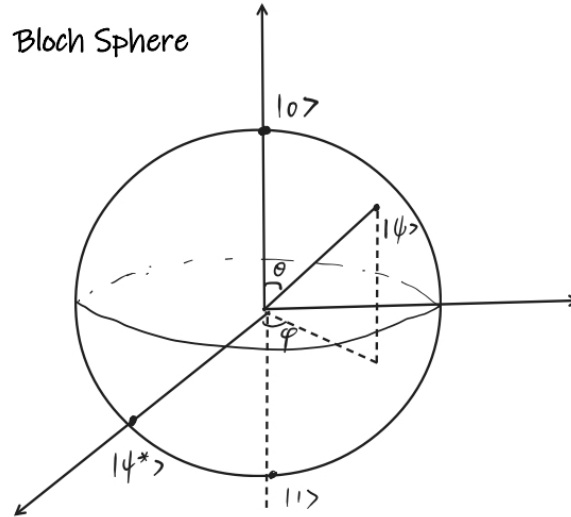


Figure 4.1: Visualization and illustration of qubit states in Bloch Sphere. (Work by the author)

4.1.3 Quantum Measurement

Quantum measurements extract classical information from evolved quantum states, as unitary gate operations alone do not produce observable outputs. When a measurement occurs, a qubit's superposition

state collapses probabilistically to a computational basis state. For example, performing a standard computational basis measurement (equivalent to measuring the Pauli-Z observable) on state $\psi = \alpha|0\rangle + \beta|1\rangle$ yields outcome 0 with probability $|\alpha|^2$ and outcome 1 with probability $|\beta|^2$.

The mathematical formalism for this projective measurement uses the Hermitian operator:

$$\sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1|,$$

which has eigenvalue $\lambda_0 = 1$ for eigenvector $|0\rangle$ and $\lambda_1 = -1$ for $|1\rangle$, giving the outcome–probability mapping:

$$P(\lambda_0) = |\alpha|^2, \quad P(\lambda_1) = |\beta|^2.$$

Observables as Measurement Operators. More generally, a quantum measurement is defined by an *observable* O , which is any Hermitian operator whose eigenvalues correspond to the possible measurement outcomes. Common examples include the Pauli matrices X , Y , and Z , or tensor products of these operators across multiple qubits. Observables play a crucial role in quantum computation—particularly quantum machine learning—because they determine what information about the quantum state is extracted and how measurement results are interpreted.

Expectation Values and the Trace Operator. Rather than relying on a single probabilistic outcome, quantum algorithms typically use the *expectation value* of an observable as the final computational result. For a quantum state ρ , the **expectation value** of observable O is computed as:

$$\mathbb{E}[O] = \text{Tr}(O\rho),$$

where $\text{Tr}(\cdot)$ denotes the matrix trace. This trace expression provides a deterministic quantity derived from the underlying probability distribution and allows quantum models to output continuous values. In variational quantum algorithms and QML, the expectation value becomes the model’s prediction used to compute loss functions and gradients.

Measurement of Multi-Qubit Registers. For multi-qubit systems, measurement samples binary strings from the quantum state’s probability distribution. A single instance of measuring the state $|\psi\rangle = \sum_x \alpha_x |x\rangle$ yields string $x \in \{0, 1\}^N$ with probability $|\alpha_x|^2$. Due to the inherently probabilistic nature of quantum mechanics:

- A single measurement provides limited state information.
- Repeated measurements (“shots”) are required for statistical accuracy.
- The squared amplitudes are estimated experimentally as

$$|\alpha_x|^2 \approx \frac{\text{Count}(x)}{N_{\text{shots}}}.$$

Thus, quantum algorithms must carefully balance unitary evolution—used to create interference and entanglement—with measurement procedures that extract meaningful classical information. In QML, this extraction typically takes the form of expectation values of chosen observables, linking the quantum state evolution to the model’s final prediction.

4.1.4 The Quantum Computing Workflow: From State Preparation to Measurement

The preceding subsections introduced the three fundamental components of a quantum computation: quantum states, quantum gates, and quantum measurements. We now summarize how these pieces fit together to form a complete computational process. A quantum algorithm can be viewed as a three-stage pipeline:

1. **State Preparation** (Input Encoding)
2. **Unitary Evolution** (Quantum Gates and Circuits)
3. **Measurement and Readout** (Classical Output)

1. State Preparation. Every quantum computation begins by initializing qubits into a known quantum state. Most commonly, all qubits are initialized to the computational basis state $|0 \cdots 0\rangle$, though more expressive encodings—such as amplitude encoding or basis embedding—may be used when classical data must be injected into the quantum device. This initialization defines the starting point of the quantum computation in an exponentially large Hilbert space.

2. Unitary Evolution via Quantum Gates. Once prepared, the quantum state undergoes a sequence of unitary transformations implemented by quantum gates. These gates manipulate the probability amplitudes and phases of the quantum state, enabling the construction of interference, entanglement, and other non-classical phenomena essential for quantum speedups. The full sequence of gates constitutes a *quantum circuit* (see Figure 4.2), and the circuit’s design determines the algorithmic behavior. Because each gate is reversible and norm-preserving, the evolution process is deterministic at the level of quantum amplitudes, even though its measurement outcomes will be probabilistic.

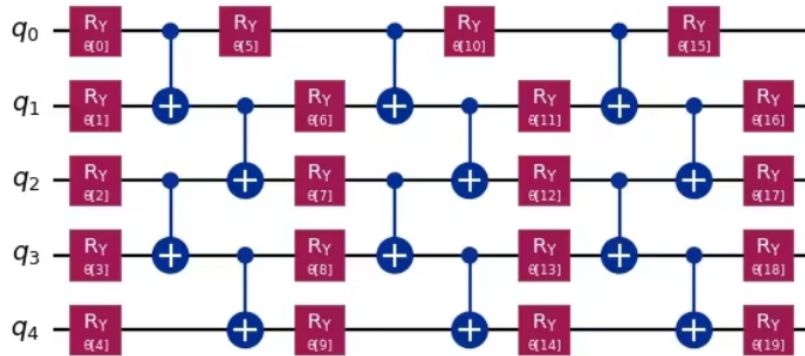
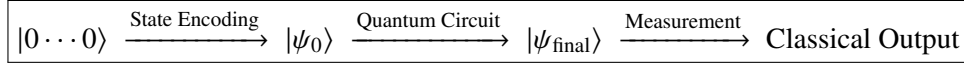


Figure 4.2: A parameterized quantum circuit (PQC) (explained in Section 4.2) consisting of alternating layers of single-qubit rotation gates $R_Y(\theta_i)$ and entangling CNOT gates arranged in a ladder-like topology given IBM Qiskit in https://quantum.cloud.ibm.com/docs/en/api/qiskit/circuit_library. Such variational structures are commonly used in quantum machine learning models and variational quantum algorithms, where the trainable parameters $\{\theta_i\}$ encode learnable transformations and the CNOT layers generate entanglement across qubits. The circuit depth and entanglement pattern together determine the expressive power of the quantum model.

3. Measurement and Classical Output. After the state has been transformed by the quantum circuit, a measurement extracts classical information from the final quantum state. Measurement collapses the superposition, sampling outcomes according to the squared magnitudes of the amplitudes. In practice, quantum algorithms repeat the execution multiple times to estimate the underlying probability distribution accurately. This conversion from a quantum probability distribution to classical data produces the final computational result, such as a bitstring, a distribution, or a derived expectation value.

Putting the Stages Together. Thus, a quantum computation proceeds as:



This workflow highlights the distinct yet tightly interconnected roles played by state preparation, unitary evolution, and measurement. Understanding this pipeline is essential for analyzing both quantum algorithms and quantum machine learning models, where each stage imposes unique constraints on expressivity, noise tolerance, and computational feasibility.

4.2 Quantum Machine Learning

Quantum computation has emerged as a promising approach to circumvent classical computational barriers in machine learning. We witness the rise of Quantum Machine Learning (QML) methods with arguably theoretical advantages on regression and classification tasks [108, 109, 110, 111]. These techniques have its remarkable application from the study of quantum systems where quantum computers can effectively solve information by the virtue of ability to naturally represents the quantum states [112].

In general, Quantum Machine Learning (QML) refers to the study of machine learning models that leverage quantum-mechanical principles—such as superposition, interference, and entanglement—to process information and perform learning tasks. A typical QML model consists of three essential components: an input encoding scheme, a parameterized quantum circuit (PQC), and a measurement-driven output map.

Input Encoding. The input to a QML model is classical data, usually represented as a vector $\mathbf{x} \in \mathbb{R}^d$ or a set of graph- or image-based features. To operate on a quantum device, this information must be embedded into a quantum state $\rho_{\mathbf{x}}$ through an encoding procedure such as amplitude encoding, basis encoding, or angle encoding. This state preparation step determines how classical information is geometrically positioned in the quantum Hilbert space, greatly influencing the model’s expressive capacity.

Learnable Parameters. The learnable parameters of a QML model are typically the rotation angles or gate parameters of a parameterized quantum circuit. Denoted by $\boldsymbol{\theta} = (\theta_1, \theta_2, \dots, \theta_m)$, these parameters define a unitary transformation $U(\boldsymbol{\theta})$ acting on the encoded input state. Training a QML model consists of optimizing $\boldsymbol{\theta}$ to minimize a loss function in the same spirit as classical neural network training.

Measurement and Output. After the circuit acts on the encoded quantum state, measurements are performed on selected qubits. These measurements produce classical outcomes whose expectation values form the model’s prediction. Formally, the output is given by

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}[O] = \text{Tr}[O U(\boldsymbol{\theta}) \rho_{\mathbf{x}} U(\boldsymbol{\theta})^\dagger],$$

where O is an observable (e.g., Pauli operator), $\text{Tr}(\cdot)$ denotes the matrix trace, and the expectation is estimated via repeated measurements. This expectation value can be mapped to a continuous label (regression), a probability (classification), or a discrete decision.

Putting the Stages Together. Thus, a QML model can be viewed as a quantum analogue of a classical parametric model:

$$\mathbf{x} \xrightarrow{\text{encoding}} \rho_{\mathbf{x}} \xrightarrow{U(\theta)} U(\theta)\rho_{\mathbf{x}}U(\theta)^\dagger \xrightarrow{\text{measurement}} f_\theta(\mathbf{x})$$

This pipeline formalizes QML as a learnable function class where parameters correspond to quantum gate operations, and the model’s hypothesis space is determined by the structure and depth of the underlying quantum circuit. The combination of high-dimensional Hilbert space representations and tunable quantum dynamics provides QML with its distinctive expressive power.

Quantum Neural Networks (QNNs) (see Figure 4.3) stand out as one of the most common examples, which use quantum principles, such as superposition and entanglement to handle more complex and bigger data. This combination of probing of many computational pathways at once and generating highly entangled states in qubit pairs may confer efficiency advantages to QNNs compared to classical NNs [113, 114].

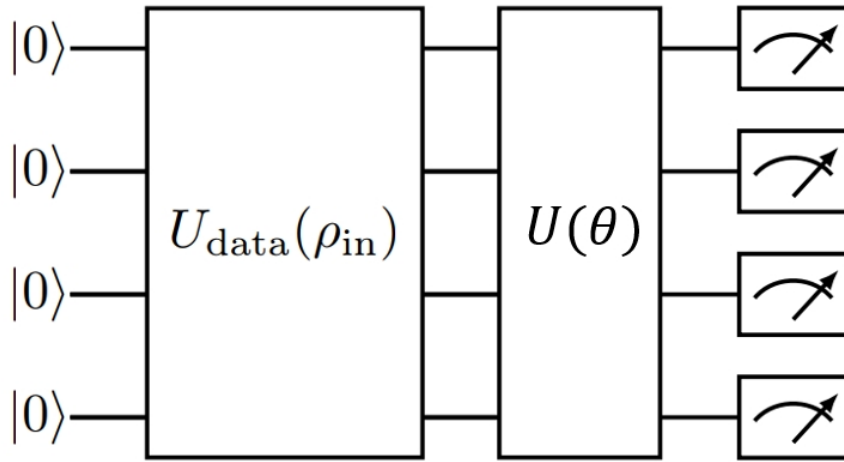


Figure 4.3: Visualization of a QNN. The input state ρ_{in} is prepared using the operation U_{data} , followed by VQC $U(\theta)$ and measurement operation.

But its impact is not easy to realize in practice. Today’s quantum systems still operate under significant hardware limitations, such as short coherence times—the limited duration during which qubits remain in well-defined quantum states—and restricted physical connectivity between qubits. While fully fault-tolerant quantum computers remain a long-term goal, Noisy Intermediate-Scale Quantum (NISQ) devices have enabled pragmatic progress through approaches such as Variational Quantum Circuits (VQCs) [115, 116, 117].

A VQC consists of a parameterized quantum circuit in which certain gates contain tunable rotation angles. These parameters are optimized by a classical optimizer in a hybrid quantum–classical loop. Specifically, the quantum processor prepares a state, applies the parameterized circuit, measures the expectation values of chosen observables, and the classical processor updates the parameters to minimize a task-specific loss function. This architecture closely resembles classical neural networks—where trainable gates play the role of weights—and is expressive enough to model complex functions while remaining compatible with current hardware constraints.

Moreover, PQCs form the foundation of quantum neural network (QNN) implementations on available devices, making them central to many real-world demonstrations of quantum machine learning [109, 118]. Their flexibility and hardware efficiency also make VQCs the standard building block for quantum graph learning models, including the QGNN variants discussed later in this chapter.

However, two critical challenges limit the development of QNNs. First, quantum frameworks are limited in their ability to approximate the high-degree nonlinear transformations that allow classical deep learning models to be expressive [119]. Second, QNN training suffers from the “barren plateau” phenomenon (i.e. exponentially smallness of a gradient as a function of the problem size) that in practice gradients almost completely vanish as the dimension of the system is large, stalling the convergence of optimization despite theoretical optimization landscape benefits [120]. Hybrid quantum-classical strategies are being developed by researchers to overcome some of these challenges as we march towards useful quantum advantage for machine learning.

4.3 From Graph to Qubits: Quantum Graph Neural Network (QGNN)

Verdon, et al. (2019) [121] proposed Quantum Graph Neural Networks (QGNNs), interpreting a graph $G = (V, E)$ as a networked quantum system. Each node $v \in V$ is associated with a quantum subsystem and its own Hilbert space H_v , while the global state of all nodes lives in the tensor product space $H_V = \bigotimes_{v \in V} H_v$.

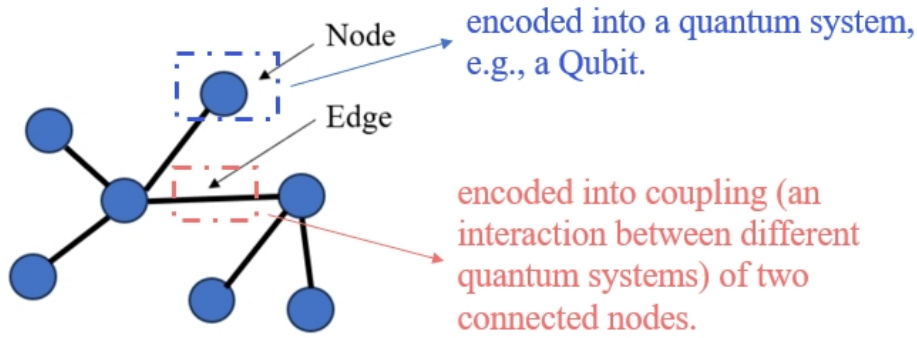


Figure 4.4: An illustration of QGNN.

Crucially, the connections in the graph dictate which quantum subsystems may interact: two node states can only exchange information if they are joined by an edge. This property allows quantum dynamics—such as entangling operations defined by the Hamiltonian—to directly mirror the structure of the underlying graph [122, 123]. As a result, QGNNs offer a principled way to combine quantum computation with relational learning, providing a promising direction for modeling complex interconnected systems. By wrapping the graph topology up in the Hamiltonian describing the quantum dynamics of the system, QGNNs build a more complex data embedding—generating features that may not be available to classical approaches [124].

QGNNs arise from combining the structural strengths of GNNs with the expressive power of quantum computation. While classical GNNs have achieved impressive performance, their representational capacity is ultimately constrained by classical architectures—for example, many GNNs are limited by the Weisfeiler–Lehman (1-WL) test [60]. QGNNs attempt to overcome these limitations by using quantum states and quantum circuits to encode and process graph-structured information. The quantum state space grows exponentially with the number of qubits, giving QGNNs access to richer feature representations that may be inaccessible to classical models.

At a high level, a QGNN embeds a graph into a quantum system and uses a PQC to learn properties of that graph. One common approach is to construct a Hamiltonian² whose matrix representation

²The Hamiltonian operator is the quantum mechanical operator representing the total energy of a system, which is the sum of its kinetic and potential energies.

corresponds to the graph’s adjacency or Laplacian matrix. Because the Hamiltonian can naturally encode adjacency relations, applying the corresponding unitary evolution allows the quantum circuit to reflect the structure of the graph. Although calculating the exact unitary may be intractable, practical approximations—such as Trotter–Suzuki decomposition [125, 126]—enable efficient implementation on near-term quantum hardware. In this way, QGNNs operate as VQCs whose learnable parameters are optimized to match tasks such as node classification, graph classification, or regression.

For example, Mernyei et al. [127] demonstrated that QGNNs possess expressive power strictly beyond the classical 1-WL test, owing to their inherently quantum properties. In their framework, quantum states and unitary evolutions enable QGNNs to capture higher-order correlations and global structural information that 1-WL—and thus standard message-passing GNNs—fundamentally fail to distinguish. This enhanced expressivity arises from the ability of quantum systems to represent superpositions of graph configurations and to encode interactions through entanglement, allowing the model to process multiple structural patterns simultaneously. Their results provide one of the rigorous theoretical justifications for the potential of quantum graph learning, and they suggest that QGNNs may overcome well-known limitations of classical GNN architectures in tasks that require distinguishing complex graph symmetries and non-local dependencies.

4.4 The Bottleneck in Quantum Computing

4.4.1 Limitations of Quantum Hardware

In the previous sections, we introduced the fundamentals of quantum computation, outlined the principles of quantum machine learning (QML), and described the theoretical advantages of Quantum Graph Neural Networks (QGNNs). These models promise expressive capabilities beyond classical architectures, due to quantum superposition, entanglement, and the exponentially large Hilbert space available to quantum systems.

However, despite their theoretical appeal, the practical performance of QML and QGNN models remains far below that of classical machine learning. Current demonstrations are largely experimental, small-scale, and sensitive to noise. This gap between theory and practice is primarily caused by severe limitations of today’s quantum hardware. A consensus across recent studies [115, 128, 129] shows that quantum computers are still far from the size, stability, and reliability required to support large-scale learning tasks.

In this section, we summarize the major hardware bottlenecks that constrain practical quantum machine learning, focusing on three key dimensions: (1) the limited number of qubits, (2) noise and the lack of fault-tolerant error correction, and (3) the difficulty designing of quantum hardware and chip design.

Limited Number of Qubits One of the most fundamental constraints of present-day quantum computers is the limited number of available qubits. While theoretical advantages of QML stem from the exponentially large Hilbert space of 2^n -dimensional quantum states, current hardware offers only tens to a few hundred physical qubits [115, 128]. Moreover, only a fraction of these qubits are sufficiently coherent and controllable to participate in computation. Because logical qubits require thousands of physical qubits for fault tolerance [130], today’s devices fall several orders of magnitude short of what is needed for scalable learning tasks. As a result, QML and QGNN models can only be tested on extremely small datasets and toy graph instances, restricting their real-world applicability.

Noise and the Absence of Fault-Tolerant Error Correction

Quantum systems are inherently fragile. Qubits rapidly lose coherence due to environmental interactions, and multi-qubit gates introduce non-negligible noise. Typical two-qubit gate error rates remain between 10^{-3} and 10^{-2} on leading NISQ platforms [115, 131], causing errors to accumulate quickly in deep

circuits. Full quantum error correction requires thousands of physical qubits per logical qubit [130], far beyond current capabilities.

For QML and QGNNs, these limitations are particularly damaging: variational quantum circuits suffer from barren plateaus [132], where noise further suppresses gradients and makes training unstable. Measurement noise also degrades estimation of expectation values, limiting the precision of QML predictions. As a result, the gap between theoretical expressivity and practical performance remains substantial.

Challenges in Quantum Hardware and Chip Design

The physical realization of quantum processors presents another major bottleneck. Superconducting, trapped-ion, and photonic architectures each face fundamental engineering challenges, including layout constraints, cross-talk, parasitic modes, and limited qubit connectivity [133, 134]. These hardware characteristics heavily restrict how quantum circuits can be compiled and executed. For example, many architectures only support entangling gates between physically adjacent qubits, forcing complex routing or SWAP operations that further increase noise [135].

Designing scalable quantum chips remains a significant open challenge: qubit fabrication, wiring density, cryogenic control, and electromagnetic interference are all limiting factors. These hardware-level difficulties form a critical bottleneck preventing QML and QGNN models from achieving their theoretical potential. Even essential components such as air-bridges on superconducting chips lack automated design pipelines, and small layout decisions can have substantial effects on device performance.

Motivated by these practical constraints, our work aims to contribute to the hardware design side of quantum computing. In particular, we propose an automated method for generating air-bridge layouts on superconducting quantum chips, addressing a gap in the current quantum electronic design automation (EDA) toolchain. This method, detailed in Section 4.4.2, demonstrates how our algorithms can support quantum hardware development and help mitigate some of the layout challenges that currently limit device scalability and reliability.

4.4.2 Graph-related Algorithms in Automatic Construction of Air-bridge in Quantum Chip Layout

Practical quest for quantum advantage is more and more dedicated as a means to cope with imperfections of NISQ (noisy intermediate-scale quantum) processors. Uncontrolled electromagnetic interactions, mainly crosstalk and parasitic coupling, are a fundamental issue in superconducting quantum circuit designs. These phenomena are major contributors to qubit decoherence and operational infidelity and place severe limitations on circuit depth and complexity. These hardware-level defects importantly translate into the performance of quantum algorithms performed on these devices. Algorithms requiring complex entanglement structures, such as Quantum Graph Neural Networks (QGNNs), are particularly vulnerable; hardware-level noise can greatly amplify algorithmic limitations such as the barren plateau problem which can suffocate trainability far before scaling rewards are realized. Dealing with the physical sources of crosstalk is thus not a pure hardware optimization but a necessary step to allow to carry out future application-layer computation.

Passive mitigation methods, such as frequency detuning, are limited by scaling. Air-bridge structures provide a known active technique to Proton-Exchange Membrane (PEM) parasitic surface modes and ground plane interactions. Manual insertion of them in complex, multi-qubit designs, however, is still arduous, error-prone, and a source of delay for device rapid iteration and scaling. In 2024, we [31]³ addressed this fabrication challenge by presenting and demonstrating an end-to-end solution for automatically generating custom air-bridge networks at the same time as the rest of the chip design. Our approach seeks to systematically suppress cross-talk pathways and parasitic capacitance to increase qubit coherence times and reduce cross-coupling between qubits. The following contents, coming from [31], provide a

³Part of this subsection has been published as [31]. ©2024 IEEE. Reprinted with permission.

brilliant example and exploration in practical physical quantum chip-level to address the limitations of unsolved quantum computing errors.

Introduction

Superconducting quantum processors are one of the leading candidates in the pursuit of building a scalable, fault-tolerant quantum computer [136]. In such devices, coplanar waveguides (CPW), which consist of a central conducting strip and pair of lateral ground planes on a dielectric substrate (see Figure 4.5[a]), play a crucial role in coherently interfacing superconducting qubits and resonators [137]. Due to their design flexibility and broad operating frequency range, CPWs serve various functions, including qubit readout, control and coupling. However, integrating circuit elements by CPWs involves curvature and discontinuities of the central conductors and corresponding ground planes, resulting in undesired parasitic modes which can alter couplings and act as qubit loss channels [138, 139].

An efficient and effective way to suppress parasitic modes in superconducting CPWs involves the use of air-bridges [140, 141, 142] (see Figure 4.5[a]). These air-bridges equalize the potential between the two isolated ground planes, and suppress the excitation of parasitic slot line modes. Meanwhile, they also reroute the return currents to reduce unwanted cross-talk between the readout and control lines. The manufacture and application of air-bridges have been explored in a variety of ways, including different fabrication processes and different superconducting materials [143, 144, 142, 145, 146]. For example, Chen et al. give a procedure for fabricating air-bridges using re-flowed photoresist as a scaffold [143], while Dunsworth et al. [144] use a scaffolding technique based on another dielectric (silicon dioxide). Full package air-bridges, with their tunnel-like structure and almost completely shield the CPW lines by covering them in a continuous ground plane[144], have also been manufactured and applied in more complex chip architectures, such as those based on flip-chip [147, 148, 149, 150] design and through-silicon vias [151, 152]. As a critical elements in multi-qubit quantum processors, air-bridges have been experimentally demonstrated to introduce minimal capacitive loss and serve as a crucial tool in enhancing circuit flexibility while effectively mitigating microwave and flux control crosstalk [143, 146].

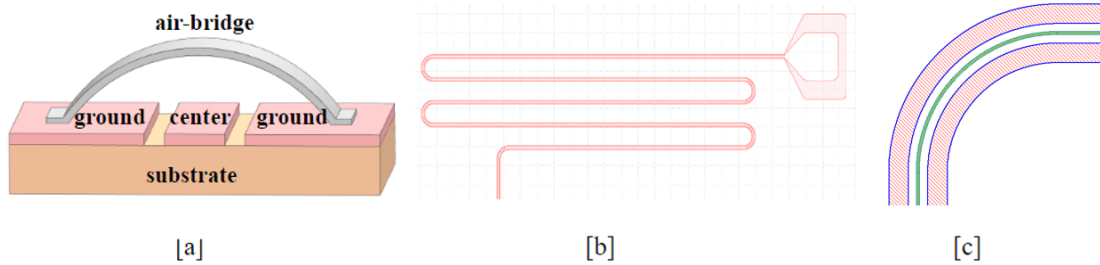


Figure 4.5: The coplanar waveguide (CPW). [a] A 3D illustration of a CPW (yellow strips) and air-bridge (white); [b] A CPW in GDSII format; [c] A close-up of a section of a CPW, with the CPW path in red, the CPW channel in green and four sets of lines of the CPW in blue. (©2024 IEEE. Reprinted with permission from [31]).

Currently, the design of most superconducting quantum chips, particularly the design and placement of air-bridges on qubits' control/readout lines, is primarily performed manually using various drawing software, such as the widely used Klayout (open-source) and L-edit (commercial). As quantum processors increase in size, this process becomes progressively more time-consuming and error-prone. It is crucial to align the placement of air-bridges with the center line of the CPW (CPW channel), to ensure that they are situated at the absolute center of the CPW. Therefore, channel information is important for the placement of the air-bridges. Recently, auxiliary design software such as Qiskit Metal [153] has emerged, allowing users to specify CPW channel information directly in the front end. The subsequent laying out of CPW signal lines and air-bridges is then automated. These methods, however, still require manual specification

of CPW channel information based on predetermined patterns and imposes restrictions, such as limiting channels to 90-degree turns, significantly limiting design freedom.

Alternatively, automatic routing algorithms, such as topological router [154] or the A* algorithm [155], can be used to provide channel information, given a chip layout containing other components such as qubits and bonding pads (but excluding CPW signal lines). However, these approaches are still in their infancy, and their effectiveness and reliability are limited. As a result, these methods currently fail to effectively address routing and air-bridge construction issues for medium-scale chips.

Furthermore, for manually plotted layouts or layouts obtained from many other sources, CPW channels are not pre-specified. In fact, for consistency with fabrication requirements, the central channel (a line with no width) is not included on quantum chip layouts. Considering the fact that existing methods cannot adequately meet practical needs, CPW routing and air-bridge construction are almost always performed manually. To date, a universal solution to air-bridge construction has not been systematically provided.

Regardless of the layout generation method, GDSII (Graphic Design System II)—a universal standard format for data exchange among different drawing software and between designers and fabricators—is widely employed in superconducting quantum chip development. Therefore, we propose a general method for air-bridge construction directly using this file format. While channel information is important for the automated generation of air-bridges, extracting this information directly from the GDSII format chip layout (e.g. Figure 4.5[b]) requires considerable effort. Firstly, due to their geometry, CPWs appear as four sets of lines in the layout (see Figure 4.5[c]). This results in a fourfold increase in the number of data points under analysis. Secondly, as illustrated in Figure 4.6, the GDSII file represents patterns by sequential point arrays, with polygons formed by the closed graphs generated by connecting each point in the array in order. This file format has three defining characteristics: (1) low data discrepancy: all components in quantum chip such as resonators, qubits, control/readout lines, etc., are represented by point arrays; (2) high data complexity: at resolutions utilized for fabrication (a database unit of $0.001\mu\text{m}$), polygons representing arcs typically comprise thousands of points; for example, the CPW in Figure 4.5[b] contains over 4,000 points; (3) multi-data-representations: the data representation of a polygon is not unique; for instance, the triangular polygon described by point array ABC can also be described by BCA or CAB . These properties lead to **three difficulties**: (1) distinguishing CPWs from other components on the quantum chip; (2) computational challenges in processing large amounts of data; (3) matching the sequence of points in each polygon (which may begin at an arbitrary point in the CPW) to the physical trajectory of the CPW in the graphical layout, i.e., by identifying the start and end-points.

Contributions.

In this section, we propose a parametric EDA method for coplanar waveguide channel recognition and air-bridge generation in quantum chips. Given a quantum chip layout that comprises components requiring air-bridges, such as qubit control/readout lines, and presented in GDSII or other similar file formats with hundreds of polygons depicted by sequential point arrays, our method can automatically recognize the CPW channels and construct air-bridges in seconds. This represents an acceleration of over $10,000\times$ compared to manual air-bridge construction, and generates air-bridges in the required shapes and locations. To further evaluate our method, we propose two metrics, **accuracy** and **overlap**, along with a reference value, coverage. These metrics are practical for testing the effectiveness of air-bridges on quantum chips and were chosen to be useful for designers.

We conduct experiments on a dataset consisting of open-source quantum chip layouts by KQCCircuits [156] and report the results to both demonstrate the effectiveness of our approach and serve as a benchmark for future works. The results highlight our method’s robustness, accuracy and efficiency. In addition, our method also enables a clear and informative visualization and log system to facilitate debugging and evaluation. Our approach provides a convenient, plug-and-play API for automated air-bridge design, which can be easily combined with other EDA modules (see Figure 4.7). It allows for rapid air-bridge construction in situations where CPW channels are not pre-specified, and paves the way for a more flexible, automated and modular approach to superconducting quantum processor EDA.

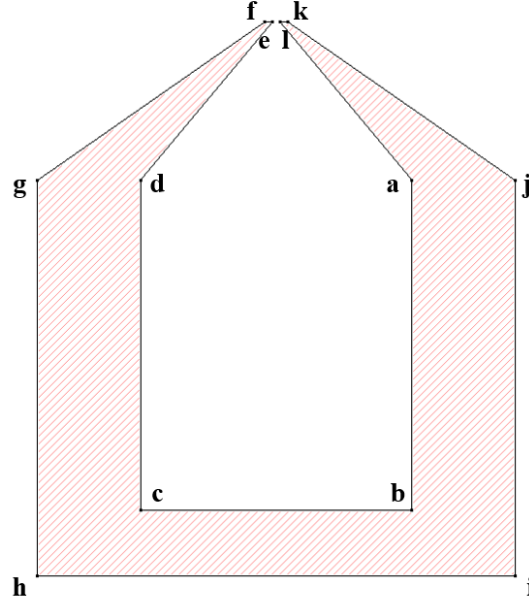


Figure 4.6: Components in the GDSII file format are represented by a sequential array of points. This pad is represented by $[a, b, c, d, e, f, g, h, i, j, k, l]$, where $a, \dots, l$ represents actual coordinates in the layout. (©2024 IEEE. Reprinted with permission from [31]).



Figure 4.7: EDA process for quantum processor design, with our contribution in red. (©2024 IEEE. Reprinted with permission from [31]).

Preliminaries And Notation

In this section, we introduce preliminary information and notation at three levels: (1) The layout level, which mainly concerns elements requiring air-bridges in superconducting quantum chips; (2) The CPW Level, which includes the CPW path and the CPW channel; and (3) The air-bridge level, detailing the information required for air-bridge generation. Here, for the sake of simplicity and without compromising generality, we take the separated air-bridge as an illustrative example. The full-package air-bridge can be described in a similar way.

Layout Level Let $C = \{c_1, c_2, \dots, c_n\}$ be the set of components, which are of different types, such as qubits' control/readout lines and associated pads in the layout. As mentioned earlier, electrical components in a quantum chip layout are represented in GDSII format by a sequential point array. Therefore, any component in the layout can be expressed as $c_i = [p_1, p_2, \dots, p_k]$, where p_i is a point described by its coordinates. Note that, due to the inherent characteristics of GDSII, it is difficult to discern the specific component type directly from its point set c_i .

CPW Level As air-bridges are typically laid out on components with CPW structure, e.g., the control/readout lines, the concepts of CPW path (Figure 4.5[c] in red) and CPW channel (Figure 4.5[c] in green) are important. A CPW path is the composition of two parallel shapes (or *polygons* in GDSII's language), as shown by the red shapes in Figure 4.5[c], which correspond to the gaps between the ground and the center (yellow strips in Figure 4.5[a]). We use $\rho = [p_1, p_2, \dots, p_m] \in C$ to describe a CPW path, where p_i denotes a point. Let $Path = \{\rho_1, \rho_2, \dots, \rho_n\}$ represents the set of CPW paths. Note that $Path$ is a subset of C .

The CPW channel (the green line in Figure 4.5[c]) is the center line of the CPW path, which is also the center of the central strip (Figure 4.5[a]) in the CPW. We denote the set of CPW channels by $\pi = \{\pi_i = (x_i(t), y_i(t)) : i = 1, 2, \dots, n\}$, where π_i is the channel of CPW path ρ_i , and t is a distance parameter. Although CPW channels are not physically manufactured on quantum chips, they are particularly important for specifying the locations of air-bridges, which should be positioned along the channels (see Figure 4.12).

Air-bridge Level As depicted in Figure 4.8, we specify (one 2D layer of) a separated air-bridge by $b := (u_b, \theta_b, \beta_b)$, where $u_b := (x_{u_b}, y_{u_b})$ are the coordinates of the geometric center of the air-bridge, θ_b is the direction angle of the length of the air-bridge, and β_b contains size information that describes the air-bridge's length, width, etc. In other words, an air-bridge in the layout can be precisely specified by its location, direction, and size information (see Figure 4.8).

Problem Formulation

We consider a quantum chip layout with a set of components $C = \{c_1, c_2, \dots, c_n\}$ and specified air-bridge size information β and placement interval I , i.e., the distance between two adjacent air-bridges. This work addresses three key problems. The first is to identify the set of CPW paths from the set of components C :

Given $C = \{c_1, c_2, \dots, c_n\}$, identify the set of CPW paths

$$Path = \{\rho_1, \dots, \rho_m\} \subseteq C = \{c_1, c_2, \dots, c_n\}.$$

The second problem is to compute the CPW channel of each CPW path: Given a set of CPW paths $Path = \{\rho_1, \dots, \rho_m\}$, where $\rho_i = [u_{i1}, \dots, u_{ik}]$, compute the set of CPW channels

$$\pi = \{\pi_i = (x_i(t), y_i(t)) : i = 1, 2, \dots, m\},$$

where π_i is the CPW channel of ρ_i , and t is a distance parameter.

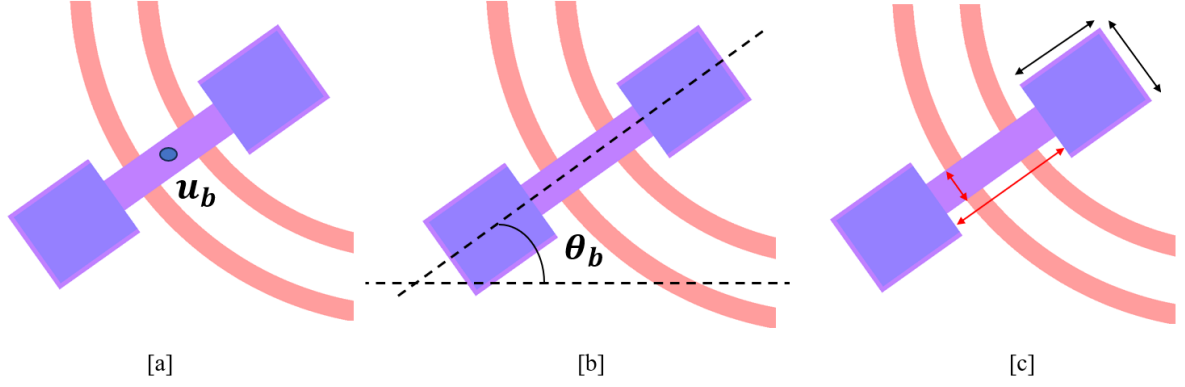


Figure 4.8: Air-bridge level information. [a] the location (coordinates) of the geometric center of the air-bridge, [b] the direction angle of the length of the air-bridge, [c] size information of air-bridge central body (red arrows) and air-bridge pad (black arrows). (©2024 IEEE. Reprinted with permission from [31]).

The third problem is to find the largest air-bridge set $B := \{B_1, \dots, B_m\}$, where B_i is the set of separated air-bridges generated along channel $\pi_i \in \pi$. To be more precise, denote the number of elements in set $\cdot$ by $|\cdot|$, the distance between points A and B by $|AB|$, a point u on channel π_i by $u \in \pi_i$, the angle of contingency of the channel π_i at point u by $\Theta(u, \pi_i)$, and assume $m = |\text{Path}| = |\pi|$. Then, our goal is to solve the following optimization problem:

$$\begin{aligned}
 \max \quad & \sum_{i=1}^m |B_i| \\
 \text{s.t.} \quad & u_{b_j} \in \pi_i, b_j \in B_i \\
 & \tan(\theta_{b_j}) \cdot \tan(\Theta(u_{b_j}, \pi_i)) = -1, b_j \in B_i \\
 & |u_{b_j} u_{b_{j-1}}| = I, b_j, b_{j-1} \in B_i \\
 & \beta_{b_j} = \beta, b_j \in B_i \\
 & i = 1, 2, \dots, m
 \end{aligned}$$

This problem possesses an **optimal substructure**, aiming to construct as many separated air-bridges as possible on channels while satisfying the required intervals, directions, and shapes. In the above formulation, the objective function is sum of the number of air-bridges on each channel π_i , with each air-bridge placed such that (i) its geometric center lies on the corresponding CPW channel π_i (see Figure 4.8[a]), (ii) its length slope is perpendicular to π_i 's tangent slope (see Figure 4.8[b]), (iii) the distance between adjacent air-bridges is I , and (iv) size information is given by β (see Figure 4.8[c]).

Table 4.1 qualitatively explains the above three problems what this research aims to solve:

Problem	Target
Problem 1	Identify all CPW paths from components in the layout
Problem 2	Compute CPW channels from CPW paths
Problem 3	Construct as many air-bridges as possible

Table 4.1: Research problems addressed in this work. (©2024 IEEE. Reprinted with permission from [31]).

The Framework And Implementation

Our method is based on a workflow consisting of the following components: **Classification and Parameterization**, which classifies points into pre-defined categories and assigns simplified parameters to them preserving the geometric information; **Detection**, which identifies CPW paths in the layout and deduces CPW channels; **Special Structure Identification**, which handles special cases that may occur in certain chip designs; **Air-bridge Construction**, which places air-bridges at appropriate locations automatically; **Visualization and Log**, which provides comprehensive visualization information and a log system for debugging and evaluation.

The input of the workflow consists of a set of components, denoted as $C = \{c_1, c_2, \dots, c_n\}$, where each c_m is equipped with a sequential array of points $[v_{m1}, v_{m2}, \dots, v_{mk}]$. For each component c_m along with its point array, Algorithm 1 processes them by assigning a `PointType` to each point. This information enable us to easily deduce parameters `arc_parameter` and `center_parameter` to c_m . Based on these, Algorithm 2 then decides whether the component c_m is a CPW path. If yes, Algorithm 2 also generates an `arc_array` that corresponds to the center of each `center_parameter` in c_m . Next, Algorithm 3 takes a CPW path ρ_m with its arcs and centers as input, and produces a sequential arc array, `sorted_arcs`, which represents the CPW sequence from start to finish. Algorithm 4 then processes a CPW path ρ_m , using the `sorted_arcs` obtained from Algorithm 3 and `arc_array` obtained from Algorithm 2, to output the corresponding CPW channel. Algorithm 1 and 2 combined tackle Problem 1, and Algorithm 3 and 4 combined address Problem 2. Lastly, a simple greedy algorithm is employed to generate as many required air-bridges as possible on CPW channels to solve Problem 3.

Classification and Parameterization For each component c_m , this step classifies its points into three categories according to their local geometry, thereby imposing structural information and decreasing the data complexity of the layout input. As mentioned in Section I, arcs in a CPW path usually follow a standard circulant arc track and are represented by an extremely dense set of points. To reduce the vast amount of data in subsequent processing, we first classify these points in the layout into three categories: CORNER, LINEPOINT, and ARCPOINT, depending on their local geometric structure (see Algorithm 2). Here, LINEPOINT and ARCPOINT denote points within straight lines and arcs, respectively, while CORNER denotes all other points, i.e. those turning points between different straight lines, between different arcs, or between a straight line and an arc.

For each point in the layout, we assign a new `point_parameter` structure:

```
1 point_parameter {  
2     PointType;  
3     angle;  
4     equality;  
5 }
```

Here, `PointType` is an enum which can take values corresponding to CORNER, LINEPOINT or ARCPOINT; `angle` is the exterior angle at this point; and `equality` describes whether the point is equidistant between its two neighbours (see Figure 4.9[a]).

After assigning `point_parameter` structures to points, we can extract higher-level geometric information (i.e., whether a sequence of points corresponds to a line or an arc) in the following way. If a series of ARCPOINTS exist between two CORNERS, we extract an arc from the ARCPOINTS; if LINEPOINTS exist or no points exists between two CORNERS, we extract a straight line. The layout can thus be represented by lines, arcs and CORNERS. Algorithm 2 decreases the data complexity significantly, since ARCPOINTS and LINEPOINTS can then be omitted after extraction.

While a straight line can easily be represented by two points, slightly more work is required to express the geometric properties of arcs which, in the case of CPW paths, are directed partial circumferences

Algorithm 2 PointType Assignment

Input: A component $c_m \in C$ with its sequential point array $[v_1, v_2, \dots, v_n]$

```
1:  $\theta_1 \leftarrow \arccos(\frac{\overrightarrow{v_n v_1} \overrightarrow{v_1 v_2}}{||\overrightarrow{v_n v_1}|| ||\overrightarrow{v_1 v_2}||})$ 
2:  $e_1 \leftarrow ||\overrightarrow{v_n v_1}|| = ||\overrightarrow{v_1 v_2}||$ 
3: for  $i \leftarrow 2$  to  $n - 1$  do
4:    $\theta_i \leftarrow \arccos(\frac{\overrightarrow{v_{i-1} v_i} \overrightarrow{v_i v_{i+1}}}{||\overrightarrow{v_{i-1} v_i}|| ||\overrightarrow{v_i v_{i+1}}||})$ 
5:    $e_i \leftarrow ||\overrightarrow{v_{i-1} v_i}|| = ||\overrightarrow{v_i v_{i+1}}||$ 
6: end for
7:  $\theta_n \leftarrow \arccos(\frac{\overrightarrow{v_{n-1} v_n} \overrightarrow{v_n v_1}}{||\overrightarrow{v_{n-1} v_n}|| ||\overrightarrow{v_n v_1}||})$ 
8:  $e_n \leftarrow ||\overrightarrow{v_{n-1} v_n}|| = ||\overrightarrow{v_n v_1}||$ 
9: for  $i \leftarrow 1$  to  $n$  do
10:  if  $\theta_i = 0$  then
11:    assign  $v_i$  to LINEPOINT
12:    continue
13:  end if
14:  if  $e_i$  is false then
15:    assign  $v_i$  to CORNER
16:    continue
17:  end if
18:  if  $\theta_{i-1} = \theta_i$  and  $\theta_i = \theta_{i+1}$  then
19:    assign  $v_i$  to ARCPOINT
20:  else
21:    assign  $v_i$  to CORNER
22:  end if
23: end for
```

Output: PointType of each point in c_m

of circles. To do so, we define a new structure, **arc_parameter**, which can be used to capture the salient features of a set of contiguous ARCPOINTS corresponding to a single arc:

```

1 arc_parameter {
2     direction;
3     radius;
4     angle;
5     startPoint;
6     midPoint;
7     endPoint;
8     center;
9 }

```

Here, *direction* indicates whether the arc is directed clockwise or counter-clockwise; *startPoint*, *midPoint* and *endPoint* are the coordinates of the start, middle, and end points of the arc; *radius* and *angle* denote the radius of the corresponding circle and angle subtended by the arc, respectively; and *center* contains the coordinates of the center of the circle (see Figure 4.9[b]).

Due to its geometry, a CPW path is in fact associated with four arcs sharing a common center point (see Figure 4.9[b]). Therefore, even though **arc_parameter** can capture the properties of single arc, it is insufficient for CPW path identification. To overcome this issue, we propose a further data structure **center_parameter**, which describes multiple **arc_parameter** structures sharing a common center (see Figure 4.9[c]).

```

1 center_parameter {
2     center;
3     arcs;
4     channel_radius;
5     direction;
6     startPoint;
7     midPoint;
8     endPoint;
9 }

```

Here, *center* contains the coordinates of the common center point, *arcs* is an array of **arc_parameter** structs; *channel_radius* is the radius from the center to the CPW channel; *direction* is the direction consistent with the real CPW geometric direction, i.e., from one end to the other; and *startPoint*, *midPoint*, *endPoint* are the start, middle and end points of the channel, respectively.

Note that if multiple arcs have the same center, only one **center_parameter** will be initiated, and will store the information of all related arcs in *arcs*. This will then be used to compute *channel_radius*/*startPoint*/*midPoint*/*endPoint* later. **Center_parameter** will be used in **Detection** for deciding whether the arcs sharing a common center correspond to a CPW path.

Let us review the whole **Parameterization** process (see Figure 4.9). We first classify points in each component, assigning them a *PointType* of CORNER, LINEPOINT or ARCPOINT, and storing this information in **point_parameter**. Next, we use *PointType* to extract arc-level information, which is stored in **arc_parameter**. Then, arc-level information is combined to give center-level information, stored in **center_parameter**. Note that all **point_parameter**, **arc_parameter** and **center_parameter** structs can be computed in linear (in the number of points) time after Algorithm 2 has been run.

Detection Based on the parameterized **arc_parameters** and **center_parameters** of component c_m , this step of **Detection** is to detect CPW paths and determine their corresponding channels. Not all arcs described by **arc_parameter** and **center_parameter** may belong to a CPW path, as they may belong

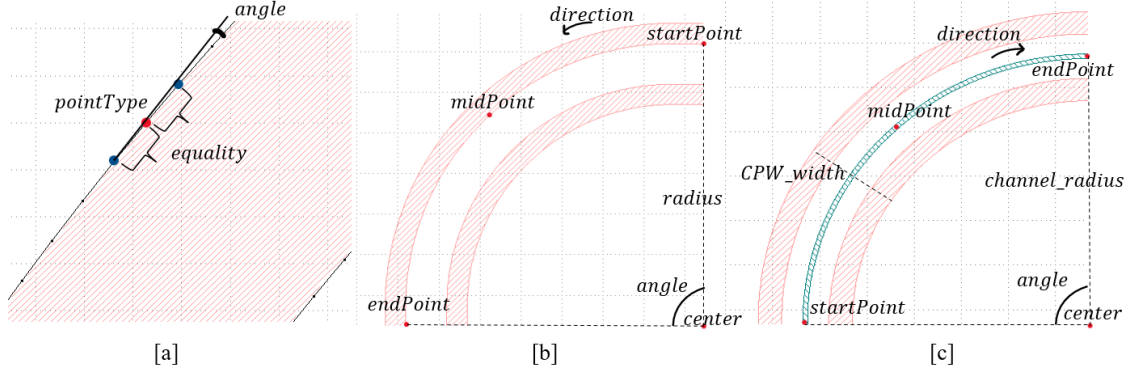


Figure 4.9: Visualization of the parameter structs. [a], [b], [c] correspond to `point_parameter`, `arc_parameter`, and `center_parameter`, respectively. (©2024 IEEE. Reprinted with permission from [31]).

to other components in the layout, e.g., pads. We therefore make use of the CPW double line property – a center corresponding to a CPW path must have four arcs, while centers on other structures do not possess this property – to filter out components other than CPW paths (see Algorithm 3).

The arcs and centers corresponding to CPW paths detected by Algorithm 3 may be deduced in random order. Thus, in order to extract the channel of a complete CPW path in the layout, e.g., control line of the qubit, we must first sort the sequence of CPW path parameters in order from start to finish (with respect to the actual CPW). More precisely, we use Algorithm 4 to find the **sorted** and **non-repetitive** arc sequences (see Figure 4.10[b]). Here, the **sorted** means that the sequence of arcs follows the actual sequence corresponding to the geometry of the CPW path, and the **non-repetitive** means that each arc in the sorted sequences has a unique center point. Note that, due to the symmetry of CPWs, in Algorithm 4 we only need to search the first half of the array of arcs (Line 6) for the start arc. This reduces the time complexity and ensures no integer overflow in Line 16.

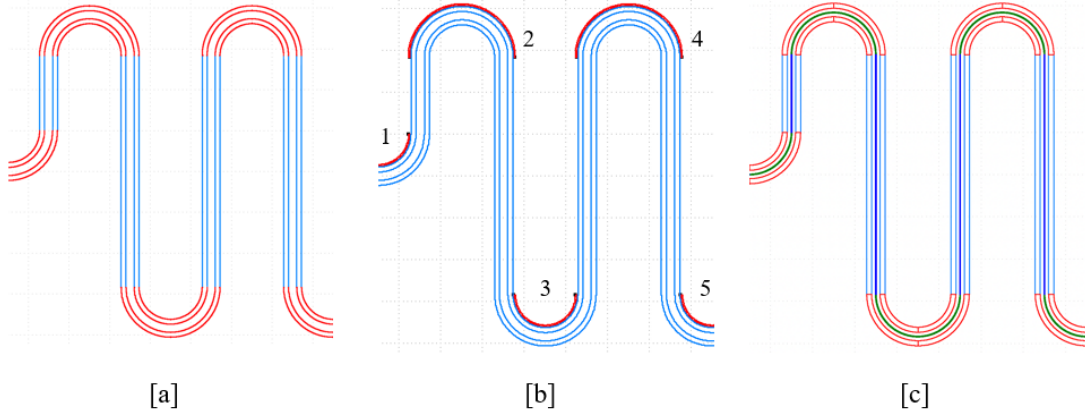


Figure 4.10: Visualization of arcs, sorted_arcs, and channels. [a] arcs (red), the input of Algorithm 4; [b] sorted_arcs (red), the output of Algorithm 4; [c] CPW channels computed by Algorithm 5, including arcs (green) and lines (blue). (©2024 IEEE. Reprinted with permission from [31]).

From the output of Algorithm 4 – the sorted arc sequences for each CPW path – we can then compute the CPW channels corresponding to the arcs in the CPW using Algorithm 5 and the geometric information stored in `center_parameter`. The CPW channels output from Algorithm 5 can be seen in Figure 4.10[c].

Algorithm 3 CPW Path Detection

Input: A component c_m with (1) its array of arcs $A = \{a_1, \dots, a_k\}$ represented by **arc_parameter**, (2) its array of centers $CT = \{ct_1, \dots, ct_n\}$ represented by **center_parameter**

```
1: CPW_Path_Flag  $\leftarrow$  False
2: for  $i \leftarrow 1$  to  $n$  do
3:   count  $\leftarrow 0$ 
4:   index  $\leftarrow \{\}$ 
5:   arc_array  $A_i \leftarrow \{\}$ 
6:   for  $j \leftarrow 1$  to  $k$  do
7:     if  $ct_i$  is the center of  $a_j$  then
8:       count  $\leftarrow$  count + 1
9:       append  $j$  to index
10:    end if
11:  end for
12:  if count = 4 then
13:    CPW_Path_Flag  $\leftarrow$  True
14:    for  $j$  in index do
15:      append  $a_j$  to  $A_i$ 
16:    end for
17:  else
18:    for  $j$  in index do
19:      delete  $a_j$  from  $A$ 
20:    end for
21:  end if
22: end for
```

Output: CPW_Path_Flag of c_m ;

arc_array $A_i = \{a_{i1}, a_{i2}, a_{i3}, a_{i4}\}$ appended to corresponding ct_i 's **center_parameter**, $i = 1, \dots, n$, if CPW_Path_Flag = True

Algorithm 4 CPW Path Arcs Sorting

Input: A CPW path ρ_m with (1) its array of arcs $A = \{a_1, \dots, a_n\}$ represented by **arc_parameter**, (2) arcs' directions obtained from corresponding **arc_parameter** $D = \{d_1, \dots, d_n\}$, where d_i is the direction of a_i (clockwise or counter-clockwise), (3) arcs' centers obtained from corresponding **arc_parameter** $CT = \{ct_1, \dots, ct_n\}$, where ct_i is the center of a_i

```
1: index  $\leftarrow 0$ 
2: sorted_arcs  $S \leftarrow \{\}$ 
3: if  $d_1 \neq d_n$  and  $ct_1 = ct_n$  then
4:   index  $\leftarrow 1$ 
5: else
6:   for  $i \leftarrow 2$  to  $\frac{n}{2}$  do
7:     if  $d_{i-1} \neq d_i$  and  $ct_{i-1} = ct_i$  then
8:       index  $\leftarrow i$ 
9:       break
10:    end if
11:  end for
12: end if
13: if index = 0 then
14:   return  $S$ 
15: else
16:   for  $i \leftarrow$  index to index +  $\frac{n}{4}$  do
17:     append  $a_i$  to  $S$ 
18:   end for
19:   return  $S$ 
20: end if
```

Output: **sorted_arcs** S of ρ_m

Algorithm 5 CPW Channel Computation

Input: A CPW path ρ_m with (1) the **sorted_arcs** $S = \{s_1, s_2, \dots, s_k\}$ obtained from Algorithm 3, (2) **sorted_arcs** S 's centers obtained from corresponding **arc_parameter** $CT = \{ct_1, ct_2, \dots, ct_k\}$, where ct_i is the center of s_i , (3) **arc_array** $A_i = \{a_{i1}, a_{i2}, a_{i3}, a_{i4}\}$ corresponding to ct_i in CT obtained from Algorithm 2

```
1: for  $i \leftarrow 1$  to  $k$  do
2:   for  $j \leftarrow 1$  to 4 do
3:      $r_{ij} \leftarrow$  radius of  $a_{ij}$ 
4:   end for
5:    $r_i \leftarrow \frac{r_{i1} + r_{i2} + r_{i3} + r_{i4}}{4}$ 
6:    $s \leftarrow$  startPoint of  $s_i$ 
7:    $e \leftarrow$  endPoint of  $s_i$ 
8:    $start_i \leftarrow$  the point at distance  $r_i$  from  $ct_k$  to  $s$ 
9:    $end_i \leftarrow$  the point at distance  $r_i$  from  $ct_k$  to  $e$ 
10: end for
11: for  $i \leftarrow 1$  to  $k - 1$  do
12:   The CPW channel  $\pi_m$  includes the arc from point  $start_i$  to point  $end_i$  on a circle with center  $ct_i$  and radius  $r_i$  and includes the line segment from  $end_i$  to  $start_{i+1}$ 
13: end for
14: The CPW channel  $\pi_m$  includes the arc from point  $start_k$  to point  $end_k$  on a circle with center  $ct_k$  and radius  $r_k$ 
```

Output: The CPW channel π_m

Special Structure Identification To be flexible enough to deal with a variety of complex chip layouts, we must be able to handle certain special structures.

Boundary Lines.

In **Detection**, we extract CPW channels based on sorted arcs. However, the channels corresponding to CPW boundary lines (shown in Figure 4.11[a]) are not located within arcs, which means the previous steps cannot deal with this scenario. Instead, we detect four CORNERS described in **Parameterization** that determine four parallel lines extending outward from the terminal point (i.e., `endPoint`) of the end arcs, and extract the CPW channel using these four CORNERS and the terminal points of the arcs.

No-arc Polygons.

Identification of CPWs based on arcs does not work for no-arc polygons (see Figure 4.11[b]). We handle this scenario by detecting four straight parallel segments.

Intersections.

Some components in quantum chips involve intersections between CPW paths (see Figure 4.11[c]). These intersections make CPW paths more complex, and no longer correspond to simple paths traversing from a start point to an end. Previous steps extract one CPW channel from one CPW path. If an intersection is involved, we instead extract multi-channels by determining whether the tangent directions of the terminal points of adjacent arcs (i.e., the `endPoint` of one arc and `startPoint` of the next) are collinear.

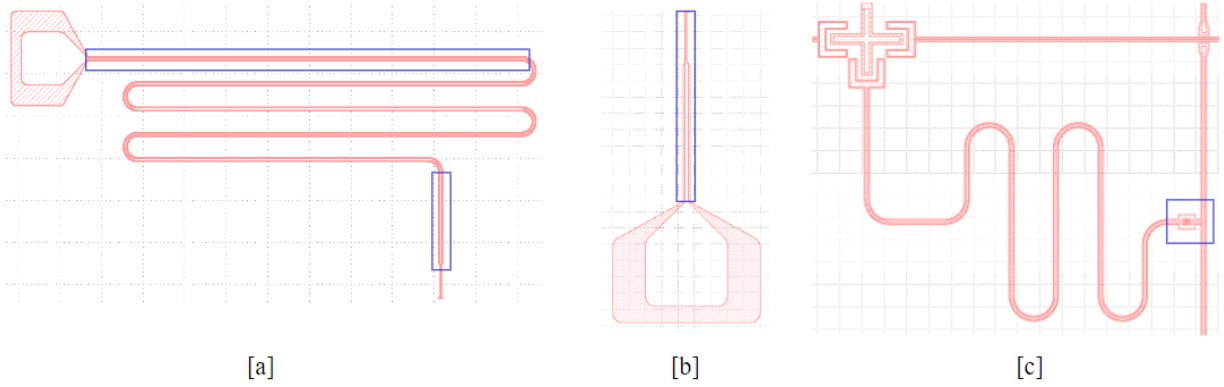


Figure 4.11: Special structures. [a] a boundary line of a CPW; [b] a no-arc polygon; [c] a CPW intersection. (©2024 IEEE. Reprinted with permission from [31]).

Air-bridge Construction Once the CPW channels have been obtained in **Detection**, we solve the optimization **Problem 4.4.2** described in Section III with a greedy algorithm, and use the solution to place air-bridges of various shapes along the CPW channels. Figure 4.12 illustrates the construction of two types of air-bridge in this manner: separated and full package air-bridges, which are commonly adopted in a variety of chip architectures, such as planar, flip-chip [147, 148, 149, 150] and those based on through-silicon vias [151, 152]. Note that for full package air-bridges, their location and direction follow the CPW channel. The position of the specific structure—small window—in full package air-bridges (refer to Figure 4.12[b]) can be acquired in the same manner (described in Problem 3) as the position of separated air-bridges.

Visualization Our parameterized approach based on `point_parameter`, `arc_parameter` and `center_parameter` data structures enables the comprehensive visualization and logging of the entire process. `PointTypes`, `arcs`, `centers`, `channels` etc., can all be stored in different layers in a GDSII file (see Figure 4.17 [d]), allowing users to easily check, modify, or fold them, and enabling a better understanding of the program mechanisms and more efficient debugging.

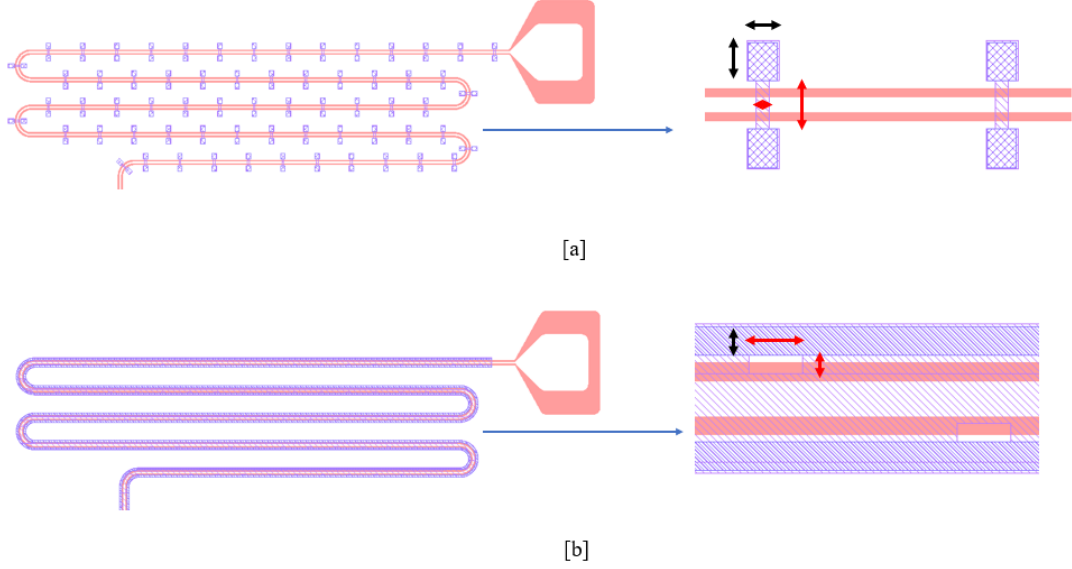


Figure 4.12: Two types of air-bridge: [a] Separated air-bridges and a close-up of pad size information (in black arrows) and central body size information (in red arrows); [b] Full package air-bridges and a close-up of single small window size information (in red arrows) and pad size information (in black arrows). (©2024 IEEE. Reprinted with permission from [31]).

Experiment

Dataset To test the effectiveness of our approach in automatically generating air-bridges in quantum chips, we composed a dataset consisting of open-source quantum chip layouts from KQCircuits [156] to evaluate our algorithms (see Table 4.2). This dataset contains different types of designs with basic quantum chip elements, such as CPWs, qubits, etc. Various structures and architectures are included.

Chip	Qubits	Polygons	Points	Avg. Points/Polygon
1	1	99	2652	26
2	0	94	3131	31
3	0	98	4687	47
4	3	109	6070	55
5	1	117	6491	55
6	4	114	7142	62
7	4	29	12321	424
8	6	117	24859	212

Table 4.2: Details of open-source quantum chips from KQCircuits [156]. Chips are sorted in order of the number of points on each chip. Chips 2 and 3 (with 0 qubits) contain only resonators and control lines. (©2024 IEEE. Reprinted with permission from [31]).

Metrics There appear to be few proposed methods in the literature for automating the construction of air-bridges on quantum chip layouts, and thus there are not yet standard benchmarks and metrics for evaluating air-bridge placement. To assess our approach, we propose two original and meaningful benchmarks: accuracy and overlap; and one reference value, coverage. The definitions are as follows:

$$\text{accuracy} = 1 - \frac{\sum_{i=1}^N e_i}{\sum_{i=1}^N l_i} \quad (4.10)$$

where e_i is the perpendicular distance from the central point of the i -th air-bridge to the corresponding channel, l_i denotes the length of the i -th air-bridge, and N refers to the total number of air-bridges (see Figure 4.13).

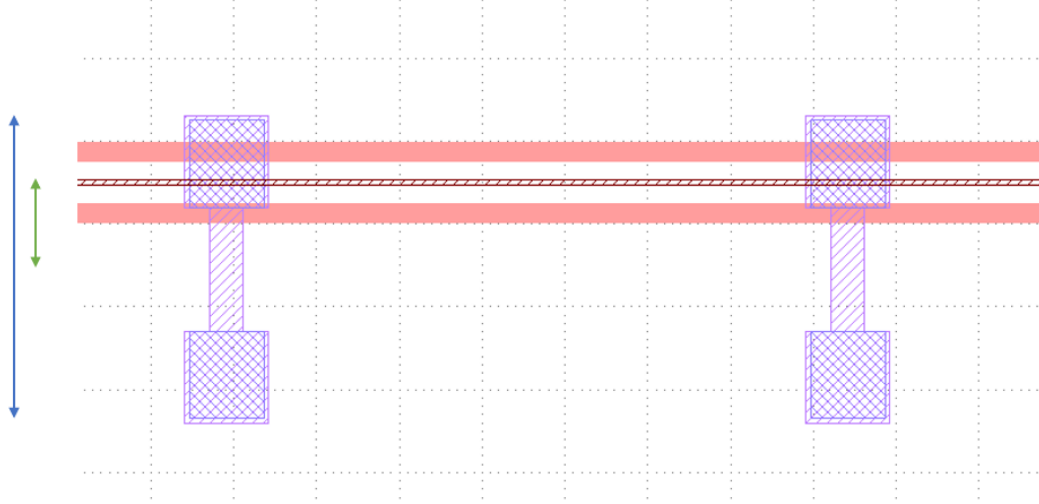


Figure 4.13: An exaggerated example of air-bridges with inaccurate locations. This phenomenon is commonly encountered when manually placed. The e_i and l_i in Eq. 4.10 refer to the length of the green arrow and the length of the blue arrow. (©2024 IEEE. Reprinted with permission from [31]).

$$\text{overlap} = \frac{\sum_{i=1}^N o_i}{\sum_{i=1}^N s_i} \quad (4.11)$$

where o_i is the area of the i -th air-bridge that erroneously overlaps with other components (see Figure 4.14), s_i is the area of the i -th air-bridge, and N is again the total number of air-bridges. Since the central part of bridge will always overlap with the corresponding channel, only the area of the bridge pad is considered in when calculating the overlap. Note that if two air-bridges overlap with each other, we do not double count the overlap area.

$$\text{coverage} = \frac{\sum_{i=1}^M \sum_{k=1}^{N_i} a_{ik}}{\sum_{i=1}^M c_i} \quad (4.12)$$

where c_i is the length of the i -th channel and a_{ik} is the width of the k -th air-bridge on the i -th channel, M is the number of channels, and N_i denotes the number of air-bridges on the i -th channel (see Figure 4.15).

Accuracy describes the deviation of the generated air-bridges from the channel. The higher the accuracy, the more precise the location of the air-bridge. Overlap describes the fraction of air-bridge area that erroneously overlaps with other components. From a manufacturing perspective, overlap between components in the layout will cause damage to their electrical properties and in practice, zero overlap is needed in the design process. Coverage is a reference value for designers to recognize how densely a chip is covered by air-bridges.

The above metrics can be determined through a combination of coordinate calculation and Boolean operations. Note that, in the actual development process of quantum chips, different electrical components may undergo distinct processing steps and diverse materials and manufacturing techniques may be required. Consequently, designers typically depict different components on separate layers based on

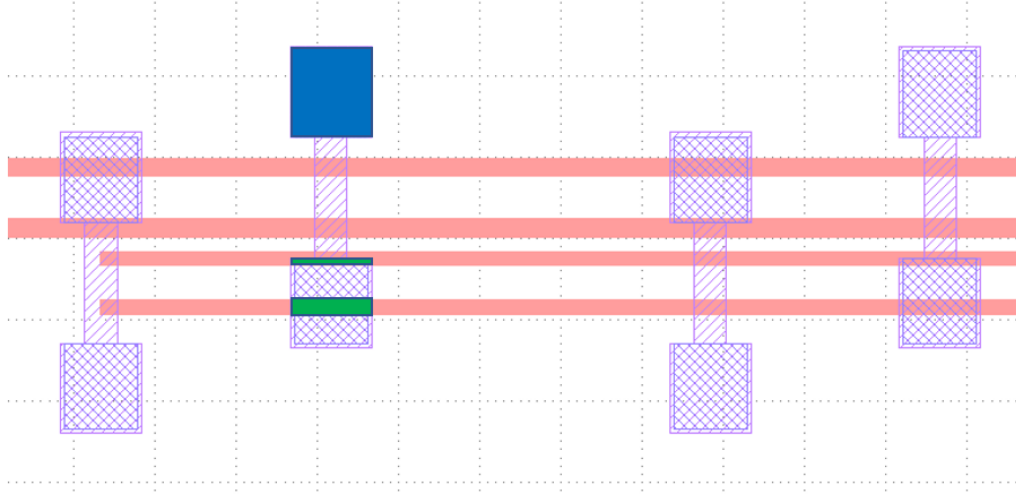


Figure 4.14: An example of overlap: air-bridges may erroneously overlap with components such as control lines, or other air-bridges. This phenomenon is commonly encountered at coupled CPWs. The o_i and $s_i/2$ in Eq. 4.11 refer to the green area and the blue area in the second air-bridge. (©2024 IEEE. Reprinted with permission from [31]).

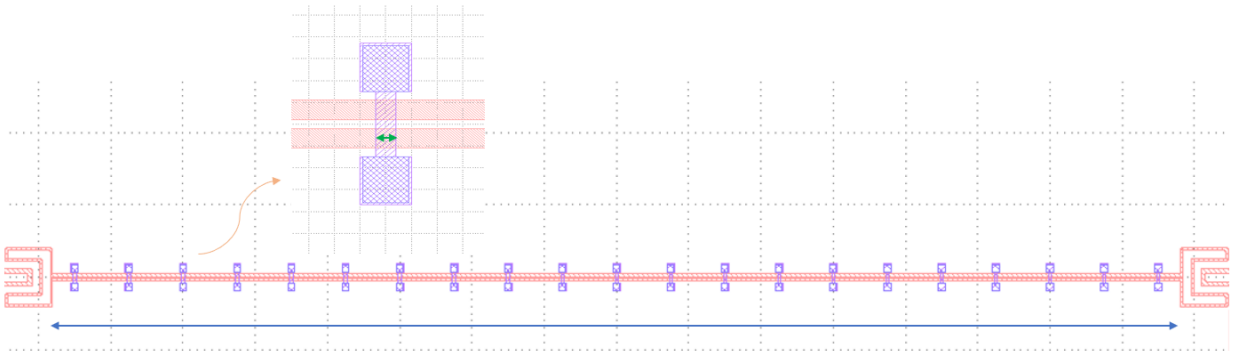


Figure 4.15: An example of air-bridges on control lines. The a_{ik} and c_i in Eq. 4.12 refer to the length of the green arrow and the length of the blue arrow. (©2024 IEEE. Reprinted with permission from [31]).

their processing needs and common conventions. For instance, CPW structures, such as superconducting resonators and control lines, are typically located in the same superconducting metal layer, e.g., layer 1, while the generated air-bridges are designated in other layers, e.g., layer 2 (air-bridge central body) and 3 (air-bridge pad). Thus, layer-based Boolean operations, including union (OR), intersection (AND), subtraction (NOT), and symmetric difference (XOR), can be utilized to obtain the aforementioned metrics. Then, the calculation of the **overlap** metric could be performed by an AND Boolean operation between layer 3 containing the air-bridge pads (similar to the blue shapes in Figure 4.14) and layer 1 containing the CPW, followed by saving the resulting shapes (similar to the green shapes in Figure 4.14) as a separate layer, e.g., layer 4. Subsequently, by summing the areas of the overlap regions in layer 4 and the air-bridge pads in layer 3, we can calculate the area ratio using Eq. 4.11 to obtain the overlap metric.

Quantitative Analysis Table 4.3 reports the results on the dataset described above by our method based on the metrics we propose. Generally, for well-designed quantum chip layouts, the number of points reflects the complexity of the corresponding designs, i.e., the more points a layout contains, the more complex its design is. Therefore, in terms of the complexity of all chip layouts in the dataset, we report the accuracy and overlap in Figure 4.16. Figure 4.16 shows that our algorithm can find accurate locations for air-bridges in both simple or complex chip designs. As the complexity increases, we find that the overlap tends to increase, while remaining at low level overall. This is expected, because the more complex the layout design, the less free area there is likely to be and therefore the more overlap will occur. For coverage value, it is a good reference value for designers to control the spacing between neighboring air-bridges. The results highlight the robustness and accuracy of our method in generating air-bridges automatically and efficiently.

Chip	Accuracy	Overlap	Coverage	n_p	n_c	Time(s)
1	1.000	0.0000	0.0527	2652	0	0.047
2	1.000	0.0039	0.0549	3131	4	0.078
3	1.000	0.0007	0.0624	4687	28	0.094
4	1.000	0.0063	0.0574	6070	15	0.109
5	1.000	0.0015	0.0699	6491	30	0.124
6	1.000	0.0019	0.0640	7142	24	0.124
7	1.000	0.0056	0.0665	12321	48	0.218
8	1.000	0.0695	0.0774	24859	161	0.515

Table 4.3: Performance metrics achieved by our approach for quantum chips in the dataset. n_p and n_c denote the number of points and number of centers in the corresponding layouts, respectively. (©2024 IEEE. Reprinted with permission from [31]).

Qualitative Analysis Although the accuracy and coverage metrics give a good indication of the effectiveness of our approach, for quantum chips requiring high precision, additional careful checks on the details of the air-bridges generated by our method are needed.

We demonstrate the qualitative analysis of our approach using an open-source 4-qubit chip layout provided by KQCCircuits [156] (see Figure 4.17[a]) and generating separated air-bridges (Figure 4.17[b]) and full package ones (Figure 4.17[c]). Compared to the several hours required for manual air-bridge design (particularly time-consuming in the full package case), our approach automatically generates the separated and full package air-bridges in 0.218s and 0.578s, respectively. Figure 4.12 depicts the details of air-bridges. Figure 4.17[b] and [c] shows the results of our approach at the macro level, demonstrating an accurate, complete, and efficient air-bridge arrangement.

In addition, our approach enables multiple layers of informative data visualization (see Figure 4.17[d]) for debugging errors including, but not limited to, incorrect placement of air-bridges. If our approach

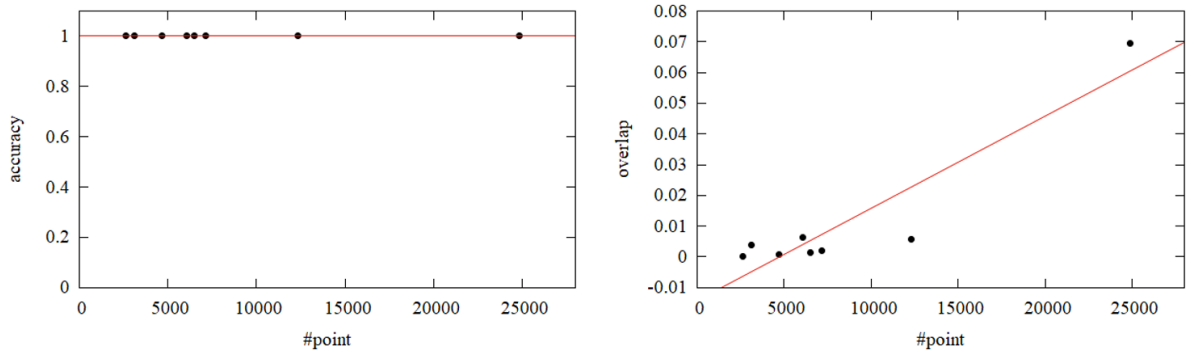


Figure 4.16: Accuracy and Overlap in terms of the number of points in different quantum chips. (©2024 IEEE. Reprinted with permission from [31]).

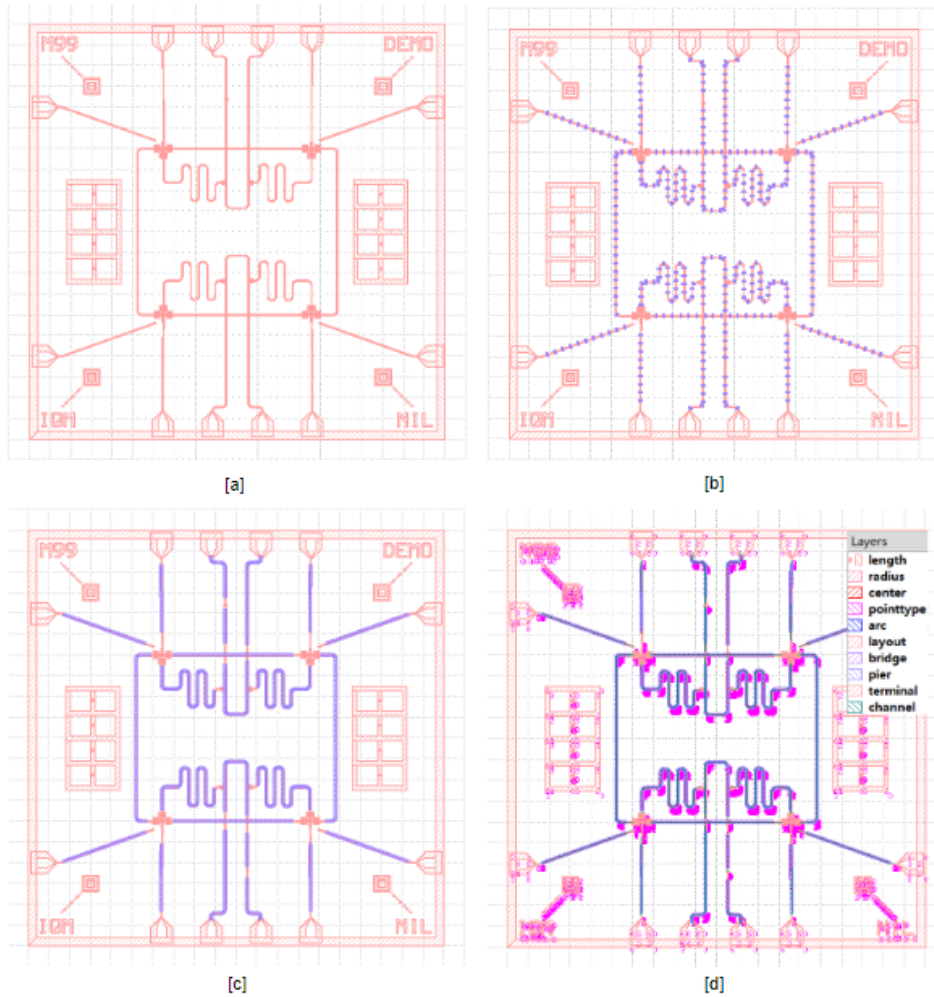


Figure 4.17: A 4-qubit layout provided by KQCircuits [156] with air-bridges designed by our parametric approach. [a] Original layout. [b] Separated air-bridges. [c] Full package air-bridges. [d] Foldable visualization layers enabled by our approach. (©2024 IEEE. Reprinted with permission from [31]).

produces inaccurate results, users can easily locate errors via the visual information provided. More details are shown in Figure 4.18.

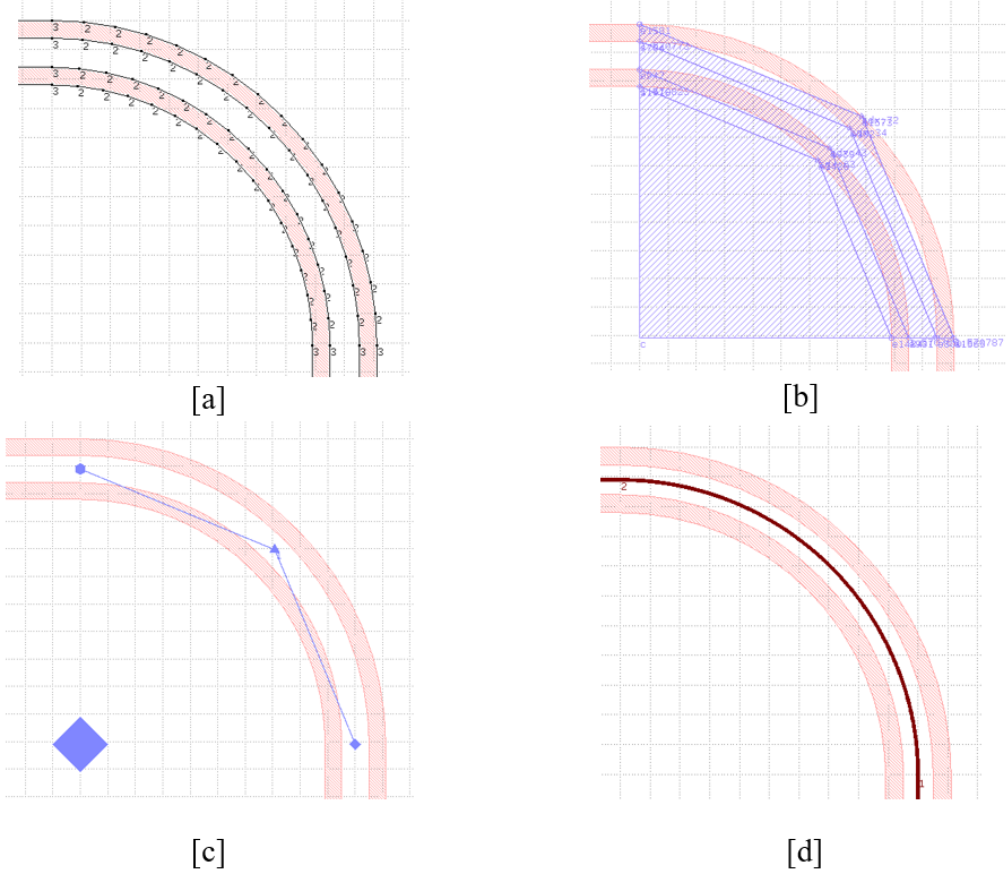


Figure 4.18: Details of debugging layers. [a] point type layer, [b] arc layer, [c] center layer, and [d] channel layer. (©2024 IEEE. Reprinted with permission from [31]).

We remark that, although we dealt with all CPW lines of the layout in Figure 4.17 [a] at once, the designer can actually process the layout in a more fine-grained manner. By separating the layout into different input, and then combining all the outputs, we can get the final layout with, e.g., separated air-bridges for readout resonators and transmission lines, and full package air-bridges for the control lines, as may be required in flip-chip architectures.

Complexity Analysis Let n_p , n_a , and n_c be the number of points (`point_parameter`), arcs (`arc_parameter`), and centers (`center_parameter`) corresponding to the the GDSII file input. Our algorithm running times are given in Table 4.4.

Algorithm	Complexity
Algorithm 1	$O(n_p)$
Algorithm 2	$O(n_a \cdot n_c) = O(n_c^2)$
Algorithm 3	$O(n_a) = O(n_c)$
Algorithm 4	$O(n_c)$
Total	$O(n_p + n_c^2)$

Table 4.4: Algorithm running time complexity. (©2024 IEEE. Reprinted with permission from [31]).

Note that $n_a = 4n_c$, and that n_c is upper bounded by n_p , although in practice is orders of magnitude smaller (see Table 4.3). The linear scaling in the number of points, and quadratic scaling in the number of centers (or arcs) means that our proposed method can efficiently deal with complex quantum chip layouts (see Time column in Table 4.3).

Feedback and Characterization from Real Chips To further verify the effectiveness of our method, we apply it to the development process of a real superconducting quantum chip. We use the GDSII file of the surface-13 quantum processor layout [146] and construct separated and full package air-bridges at the corresponding locations as required. This layout contains 13 qubits and 16 couplers, with a total of 56 polygons, 154 centers and 55,174 points. The entire air-bridge construction process takes 0.984 seconds for separated air-bridges and 1.045 seconds for full package air-bridges, with accuracy of 1.000, overlap of 0.0012, and coverage of 0.0471, respectively. After delivering the design layout to the manufacturing personnel for fabrication, we observe and check the overlaps between the air-bridges and other electrical elements under an optical microscope and Scanning Electron Microscope (SEM) (see Figure 4.19). When it comes to the accuracy and coverage metrics, the opaque nature of the air-bridges covering the CPWs after fabrication makes it impossible to directly obtain accurate values from the coordinates or dimension measurements. Nevertheless, we can observe the quality of the air-bridge placement by testing the fabricated chips in a dilution refrigerator, which provides an extremely low-temperature (approximately 10 mK) working environment for superconducting quantum chips. As air-bridges can affect the crosstalk of the qubit control signals and the simultaneous single qubit gate fidelity [143], measuring these properties gives an operational way to evaluate the quality of the air-bridge placement. Characterizing the signal crosstalk matrix [157] of the fabricated surface-13 processor, we find negligible flux crosstalk of 1.4×10^{-4} and microwave crosstalk at -45dB. We also benchmark the single qubit gate fidelity using both isolated (99.95%) and simultaneous (99.94%) Randomized Benchmarking (RB) ⁴. The tiny decrease (1.3×10^{-4}) between isolated and simultaneous RB indicates low airbridge-induced microwave crosstalk and great isolation.

Subsequent research proposed more empirical studies on the effect of tantalum airbridges employing this algorithm in [146]. The method was implemented in both multiqubit and few-qubit superconducting processors, demonstrating that fully-capped tantalum airbridges can serve not only as control-line jumpers and ground-plane crossovers, but also as reliable coupling elements without introducing observable degradation in qubit coherence. In a 13-qubit flip-chip tunable-coupling processor, parametric EDA-designed airbridges fabricated via a lift-off process enabled median qubit energy relaxation times exceeding 100 μ s for most qubits, while simultaneously suppressing microwave and flux crosstalk to -45 dB (median) and the 10^{-4} level, respectively. As a result, consistent single-qubit gate fidelities were achieved under both isolated and simultaneous randomized benchmarking, indicating effective suppression of residual crosstalk without the need for active compensation.

Furthermore, the feasibility of airbridge-mediated qubit-qubit coupling was systematically investigated using a dedicated two-qubit device, in which tantalum airbridges were introduced solely for coupling purposes. The extracted effective coupling strength was approximately 3.9 MHz, comparable to that of control devices employing conventional coupling schemes. Importantly, detailed coherence measurements revealed that both T_1 and T_2 times of the qubits remained statistically indistinguishable from those of the control group, confirming that the introduction of tantalum airbridges does not compromise qubit lifetimes or coherence properties. Building upon this coherence preservation, high-fidelity two-qubit operations were demonstrated via a diabatic CZ gate, yielding an interleaved randomized benchmarking fidelity of 99.2(2)%, together with a Bell-state preparation fidelity of 99.70% obtained from quantum state tomography.

Collectively, these experimental results provide compelling evidence that the proposed automated

⁴Randomized Benchmarking (RB) is a widely used experimental technique for characterizing the average fidelity of quantum gate operations by applying sequences of randomly chosen gates and measuring the resulting decay of state survival probability, which is largely insensitive to state preparation and measurement (SPAM) errors.

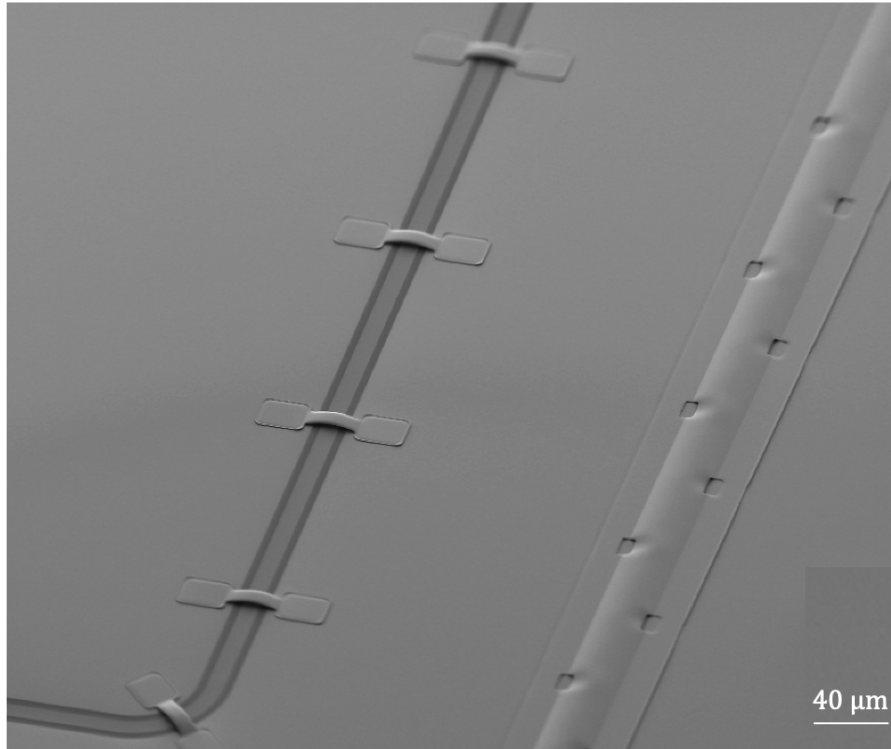


Figure 4.19: SEM images of the separated and full package air-bridges in the fabricated surface-13 quantum chips. (©2024 IEEE. Reprinted with permission from [31]).

airbridge generation and optimization methodology enables scalable interconnects and flexible coupling geometries, while preserving long qubit coherence times and high gate fidelities across different device scales. By elevating airbridge design from a device-specific implementation to an algorithmic, layout-aware, and reproducible process, this approach substantially lowers the engineering and fabrication barriers associated with constructing large-scale superconducting quantum processors. Such an enabling capability is particularly critical for emerging quantum architectures and quantum error correction schemes that rely on non-nearest-neighbor interactions, and it further facilitates the transition of quantum machine learning—especially quantum graph learning with complex connectivity patterns—from proof-of-concept demonstrations toward scalable, deployable, and industrially viable quantum systems.

Conclusion and Future Work

Designing a high performance superconducting quantum chip is a delicate and complex process. Increases in chip size are accompanied by increases in the number of signal lines, and thus more numerous and more effective air-bridges are required to reduce cross-talk and leakage. The significant time required to perform manual placement of air-bridges on a circuit layout, and the risks of errors and mistakes associated with doing so, necessitate the automation of air-bridge generation. This section proposes an effective parametric EDA method for air-bridge construction, suitable for use when the CPW channels are not pre-established. Our method can be applied to a variety of complex layout designs, and can speed up the process of air-bridge generation from hours, in the manual case, to seconds. In addition, our parametric approach provides all the information required for clear and informative visualization and logging, which facilitates analysis and debugging. Our approach can easily be integrated as a module in broader EDA packages for superconducting processor design. We also propose valuable metrics, accuracy and overlap, for evaluating the arrangement of air-bridges and report corresponding results on open-source quantum chip layouts, which can serve as a benchmark for future works. We validate our method by testing it

on a real fabricated 13-qubit chip for air-bridge construction and observe outstanding performance with minimal microwave and flux crosstalk. Future extensions could consider additional special structures for compatibility with an even richer set of layout designs, and the merging with other EDA modules to build an integrated, end-to-end superconducting quantum EDA suite.

4.5 Conclusion of the Chapter

This chapter explored the emerging intersection between graph learning and quantum computation, examining both the theoretical motivations for quantum graph neural networks (QGNNs) and the practical limitations imposed by current quantum hardware. While quantum machine learning offers the potential for enhanced expressive power—including richer entanglement-based representations and computational advantages for certain classes of graph problems—present-day superconducting devices remain constrained by limited qubit counts, high error rates, decoherence, and hardware noise phenomena such as parasitic coupling. As a result, the theoretical benefits of QGNNs have yet to be reliably realized in practical applications.

In addressing this gap, the chapter presented a parametric algorithm for automated airbridge placement on superconducting quantum chips, aimed at suppressing parasitic modes that degrade qubit coherence and signal integrity. This work illustrates how layout optimization can contribute to improving the physical reliability of quantum processors. Although the contribution is preliminary and does not directly resolve core challenges such as scalable error correction or full barren plateau mitigation, it demonstrates a hardware–software co-design perspective in which insights from graph and computer graphics can inform the engineering of more stable quantum systems.

In a nutshell, the findings of this chapter highlight the importance of considering both algorithmic and hardware factors in advancing quantum machine learning. Progress in QGNNs will likely require coordinated development across quantum architecture design, noise reduction techniques, model expressiveness, and task-specific optimization. As quantum devices continue to evolve, opportunities may arise for deeper integration between graph learning methodologies and quantum hardware capabilities, offering a promising direction toward practical quantum graph intelligence.

Chapter 5

Risks, Challenges and Ethical Issues

Artificial intelligence has undergone an unprecedented acceleration in recent years. Breakthroughs across machine learning, multimodal foundation models, protein design, and autonomous systems have reshaped both scientific research and everyday life. From the success of large language models to advances in graph-based learning and generative AI, the pace of innovation has outstripped that of any previous technological wave. These developments, reviewed in Section 5.1, illustrate not only AI’s scientific momentum but also its rapidly expanding societal footprint.

Yet the same capabilities that empower modern AI systems also amplify a range of risks and challenges. What begins as technical opacity in model architectures—such as graph neural networks in drug discovery or risk assessment—extends to broader societal concerns including safety, accountability, fairness, misinformation, intellectual property, and long-term governance. Section 5.2 examines this progression: starting from concerns specific to graph-based models and moving outward to the systemic risks that accompany advanced AI deployment. As AI becomes embedded in high-stakes decision-making, understanding these risks becomes indispensable.

In response, researchers and policymakers have turned to mechanisms such as eXplainable AI (XAI), auditing techniques, transparency standards, and regulatory interventions. Section 5.3 discusses these emerging strategies, highlighting both their potential and their limitations. International efforts—from the EU AI Act to the US NIST AI Risk Management Framework and UNESCO recommendations—reflect the growing consensus that AI systems must be understandable, accountable, and governable.

Despite these developments, research on AI safety, transparency, and ethics remains significantly underdeveloped compared with progress in AI theory and applications. The gap between model capability and our ability to understand or regulate such systems continues to widen. Section 5.4 presents our contribution toward narrowing this gap: a quantitative ethical study on AI transparency based on more than 200 survey responses. By examining user trust, model openness, and development dynamics, our study provides empirical insight into how transparency practices shape perceptions of AI reliability and reusability. This complements the technical discussions of earlier chapters and offers a unique perspective grounded in human-centered evaluation.

5.1 Rapid Advancement of Modern AI

As Geoffrey Hinton remarked in 2023, recent advances in neural networks represent “a completely new form of intelligence” whose capabilities are growing faster than many researchers anticipated [158]. Artificial intelligence has experienced a dramatic surge in capability and influence over the past few years. The combination of increased computational power, large-scale datasets, and novel architectures—particularly foundation models and graph-based learning—has accelerated AI progress at a pace previously unseen in the history of computing. This rapid advancement has not only generated scientific breakthroughs across physics, chemistry, and biology, but has also reshaped industrial practices and daily life. From

AlphaFold's revolution in protein structure prediction to multimodal models capable of reasoning across text, images, and molecular structures, AI is now operating at the frontier of scientific discovery and societal transformation.

One of the most widely recognized milestones in recent AI development is the success of AlphaFold [68], which achieved highly accurate prediction of protein three-dimensional structures from their amino acid sequences. Protein folding had remained an open grand challenge in biochemistry for over five decades, and AlphaFold's performance in the Critical Assessment of Structure Prediction (CASP14) competition was regarded as a transformative moment by the biological sciences community. Its impact has already extended into drug discovery, enzyme engineering, and molecular design, enabling researchers to explore protein function and interactions at a scale previously impossible. The breakthrough demonstrates how deep learning models, when trained with large biological datasets and carefully designed inductive biases, can surpass traditional computational methods and reshape scientific discovery workflows.

Another defining development in contemporary AI is the emergence of large language models (LLMs), particularly the GPT series [159, 160]. Models with hundreds of billions of parameters have demonstrated unprecedented abilities in natural language understanding, reasoning, coding, knowledge retrieval, and zero-shot generalization. GPT-4, for example, integrates multimodal inputs and exhibits capabilities that approach human-level performance in various standardized evaluations, including professional exams, logical reasoning tasks, and science problem solving. These models are not only reshaping the field of natural language processing, but also fundamentally altering education, creative industries, software development, and scientific communication. The rapid scaling of LLMs has become emblematic of AI's acceleration in both capability and economic impact.

The rise of generative AI further illustrates the breadth of recent advances. Models such as DALL·E 2 [161] and Stable Diffusion [162] have enabled high-quality image generation from natural language descriptions. These systems combine powerful latent diffusion models with large-scale vision-language pretraining, allowing users to produce photorealistic, artistic, or highly abstract images with simple textual prompts. Figure 5.1 demonstrates an example of applying generative AI to draw a singing cat. The societal impact of generative models is profound: they have accelerated innovation in digital art, entertainment, gaming, and advertising, while also raising important questions about copyright, authenticity, and creative labor. Their rapid adoption exemplifies the speed at which AI research transitions into real-world applications.

In 2024, during my six-month internship as an AI developer at **Araya Inc.**, a company at the forefront of AI and neurotechnology research and applications, I gained a profound firsthand understanding of the rapid advancement of AI technologies and the ways they are transforming everyday life. At Araya Inc., I worked on advanced AI projects spanning computer vision, natural language processing, and large language models. The project involves a cloud knowledge distillation framework based on different kinds of pre-trained models, allowing users to personalize their own machine learning models. The performance of models trained on small-scale personal datasets naturally cannot outperform that of large-scale models developed by corporations leveraging extensive datasets. To tackle this, knowledge distillation [163] provides a promising framework, enabling smaller datasets and models to leverage the knowledge embedded in pre-trained models derived from large datasets, while allowing for fine-tuning and specialization on the smaller datasets (see Figure 5.2).

Knowledge distillation has so far been mainly employed in Computer Vision (CV), Natural Language Processing (NLP), etc., solving the problems that large models are cumbersome, computationally costly, and difficult to deploy. In Araya Inc., we developed an AI platform where Graph Learning (GL) techniques as one of my major research topics is applied into it as well (see Figure 5.3) that allows people to participate in constructing datasets and training models, with the rest of process being completed on the cloud, eliminating the need for individuals to own expensive computing devices. This industrial experience contrasted sharply with my academic work and provided direct exposure to the practical challenges of building and deploying state-of-the-art AI systems.

Despite the company's technical sophistication, I observed firsthand that even high-performing AI



Figure 5.1: An example of drawing a singing cat generated by Generative AI.

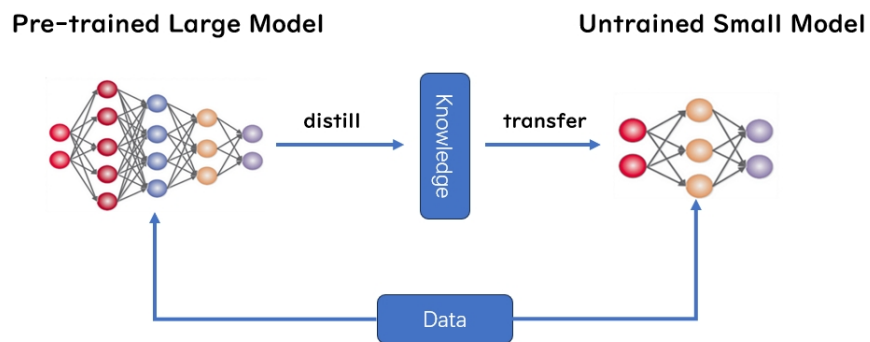


Figure 5.2: An illustration on knowledge distillation. (Work by the author)

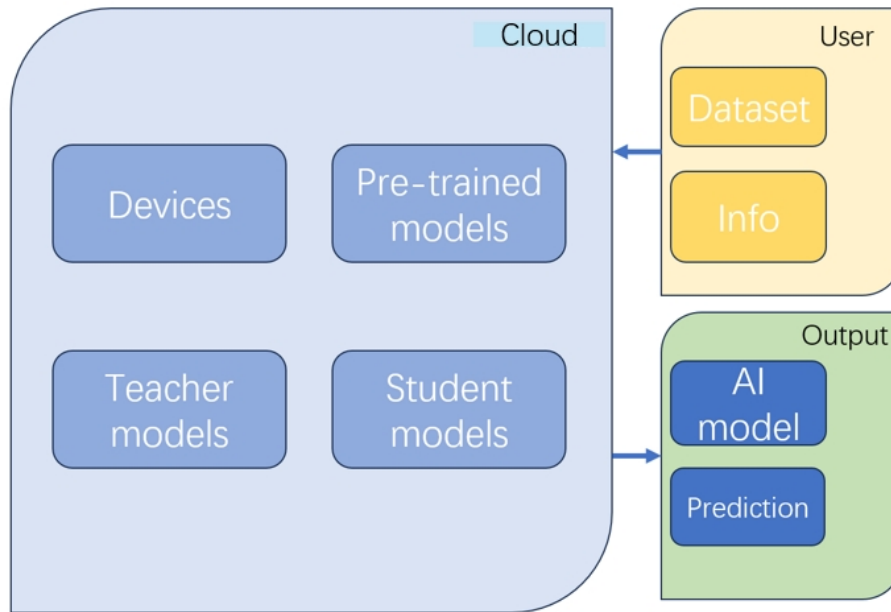


Figure 5.3: A schematic diagram of a user-friendly AI development platform based on distillation. (Work by the author)

applications—whether in vision, language, or other domains—encounter persistent challenges:

- **Trust and accountability**, when models operate as opaque black boxes.
- **Adaptability and personalization**, where large models excel at general performance but struggle to meet niche, user-specific needs.
- **Responsible deployment**, which requires not only accuracy but also clarity and compliance with emerging regulations.

These experiences made me realize that research on AI safety, ethics, and transparency is progressing far more slowly than research on AI capabilities and applications. Yet these areas are essential for ensuring that AI systems remain reliable, trustworthy, and aligned with long-term societal needs. This imbalance highlights the urgent need to confront the risks and challenges posed by rapidly developing AI technologies, which we discuss in the next section.

5.2 Risks and Challenges of Rapid-developing AI

This section starts by presenting the principal risks and challenges brought about by AI. Understanding those risks also gives crucial marching orders for researchers and makers, forcing a re-staring at AI and rethinking where it is heading for. And then we turn to some of the most important global governance regimes directed at these challenges.

AI has been increasingly everywhere in different verticals in our lives in recent years. Therefore, it is not only important but also highly desirable when a significant portion of the research community concentrates on addressing the risks and challenges of AI, whether ethical or practical. Jobin et al. systematized the most common ethical principles extracted from all available AI policy guidelines, pointing out the dominant, diversified risks and challenges concerned with AI [164] (see Figure 5.4). Jobin et al. categorized existential principles of AI that are common to international guidelines: *privacy and data protection*

(of the individual users), to ensure their privacy and protection from intrusive technology, *safety, security and robustness*, referring to the reliable and secure functioning of AI systems across different real-life conditions, *fairness and equity*, aiming to guide AI algorithms to make decisions which are unbiased and do not discriminate, to avoid from perpetuating any injustices, *responsibility and accountability*, in other words to put the value of ethical concerns in the forefront of the algorithms, meaning to make AI systems that carry-out functions ethically, *social and environmental impact*, which is assessing the broader impact of AI development, whether this has a positive or negative effect on society structures and ecosystems and last but not least, *transparency and explainability*, in order to clarify the machine-based decision-making to reduce or avoid any misunderstandings of the black-box models and procedures of AI.



Figure 5.4: Aspects of responsible AI given in [164].

Privacy and Data Protection. A fundamental problem since machine learning models are widely used in various practical scenarios is how to effectively protect the privacy and security of these learned models. To provide users with reliable AI applications, safety, privacy and robustness should be thoroughly thought of throughout their full lifecycle. Varshney defined safety here as lowering the likelihood of expected and unexpected harm [165].

Prior works demonstrate that ML models are susceptible to privacy and security attacks at each stage of the ML lifecycle [166, 167, 168, 169, 170]. These AI models need to be trained with massive data for achieving better performance, and then the training process of these AI models is decentralized into untrusted parties, which is typically an outsourced training process or trained on the basis of pre-trained models from untrusted sources, and therefore it leaves chances of unexpected attacks from inside implementation when trained from them. Furthermore, successful machine learning application is intuitively novel target for attacks in the test phase or application phase, as it inherently involves important infor-

mation such as training data, premiums and users information [171]. Privacy attacks violate which the privacy of the model in terms of its data or its internal components, while security attacks compromise the model quality and trustworthiness.

Safety, Security and Robustness. Attacks on machine learning models can be broken down into different types, depending on the architecture and the capabilities of the attackers. Concerning security issues, there could be the threat of attacks trying to access the model, and stealing the model's or even its users' information, making it dysfunctional, etc. As an example, an undue adversary can aim at a spam filtering AI model to disrupt its ability to classify spam. It is poisoned by corrupting its training data [172].

Pitropakis et al. classified AI model attacks as white-box, grey-box, and black-box attacks based on the information that the attackers have access to black-box attacks [173]). In white-box setting, attackers have complete knowledge of the model, including its architecture, parameters, training data, predictions vector, and all internal aspects. White-box (or clear-box) attacks are on the other end of the spectrum and are the strongest type of attacks, while in black-box attacks the attacker has no access to the model or training data. Gray-box (or partially white-box) attacks can be regarded as an intermediate level of access in which the attacker has some knowledge about the model architecture and the probability distribution of the training data. On the other hand, in the black-box attacks, the attackers don't know anything about the model structure or training data, and the only interaction they can have with the model is to query the model and observe its outputs. For instance, membership inference, whose attacks come to verify given a certain data sample whether it belongs to the original training set, is frequently used to interrogate the machine learning model with some black-box skills. Shokri et al. used those attacks to win strong accuracy in the membership inference against Google and Amazon's models [174]. Tramer et al. applied model extraction attack that generated a model that could be used to gain insight or to steal data from the original training data set [170].

By Varshney's definition of security for AI models [165], an AI model needs to be robust enough against different uncertainty, or, different attacks; in other words, in the event of any emergency, the model needs to deal with the emergency in a proper way and maintain normal behaviors and performance. To avoid AI models themselves from such security issues, one of the principal properties of AI is resilience to multiple shifts [175]. In addition, the designer of AI models should know the right objective function, and have an ability to evaluate it; moreover, there must exist the required resources—time, data and infrastructure—to train and evaluate the model. Once any of the above properties is not ensured, the model can be attacked by various manners more easily.

To effectively respond the attacks above and safeguard the machine learning models, researchers have proposed defense strategies against security and privacy attacks. Here we present a selection of the most typical methods briefly. Sweeney introduced k -anonymity to protect against privacy attacks, which has been applied to pseudonymizations in general [176]. This method guarantees that each data sample becomes extraordinarily close and/or indistinguishable when the other $k - 1$ data samples are properly provided. But when some side information is given, or these attacks accumulated more side information, this trick may not be powerful anymore. Dwork et al. proposed algorithmic differential privacy, an important defense methodology to training machine learning models, which offers an alternative structural way to introduce a quantification of the amount of risks on the data during training [177]. Besides the privacy attacks mentioned above, adversarial training was studied in Goodfellow et al. by training the model with some adversarial attacks so that the model can be more robust [178].

Careful attention needs to be paid when developing these defensive strategies with stakeholders. Improving model privacy and robustness naturally comes at the cost of a reduced level of accuracy. As a result, model owners, data owners and users should decide their acceptable level for accuracy degradation, along with the level of security and privacy they aim for.

Fairness and Equity. Equity means treating people or groups as equals, without bias or favoritism based on race, gender, religion or other inherent or socially constructed characteristics. One of the most common expectations for AI is fairness. But countless AI systems have not made the cut. AI regularly

replicate social inequalities because of discriminatory input data or flawed design. Facial recognition technologies, for instance, feature an error rate that is up to 34% higher for darker-skinned subjects, and hiring algorithms, such as Amazon's now canceled recruitment application systematically disadvantaged female applicants [179]; Angwin et al. examined a US software to determine the risk level of defendants, demonstrated that black defendants are twice as likely as white defendants to re-offend, without displaying second offense behaviors [180]; Jeffrey demonstrated that the hiring system of Amazon, which is based on AI for screening resumes of job-hunters, presented gender discrimination [181]. These biases exacerbate discrimination in important areas of life, including criminal justice, health care and employment, thereby motivating the interest in designing machine learning models that are fair and inclusive by design, so as to reduce the potential for bias.

Responsibility and Accountability. As earlier sections argued, AI often works like a complex technical "black box" which may produce effects that its designers did not intend, making it difficult to assign responsibility when something goes wrong. It is also important to understand the interaction among new technologies, the development of AI regulations, and ethical considerations. Examples such as Amazon's gender-biased hiring system and adversarial network attacks (as discussed earlier) paint a picture of the ways AI can get it wrong. Companies usually try to pin blame when mistakes happen. But AI systems themselves cannot be the only ones held accountable for unintended consequences. This in turn brings up a pivotal point when AI and other technologies do cause unintended consequences or harm: who is responsible?

Accountability has been a key concept in the disciplines of political science and law and provides a set of moral principles that are favorable for directing an individual or a group of individuals or an organization to do behaviors with ethics, because if they are accountable action accountable for what they do. It is in light of this that accountability is fundamental in engendering trust in groups, organizations, and in society in general. Within the realm of AI accountability plays a critical role to enable models of AI to be trustworthy the ethical guidelines are implemented and integrated throughout the entire lifecycle of such kind of models [182, 183, 184, 185].

Social and Environmental Impact. While AI models are not natural agents of harm for society and the environment, their deployment can have impacts that need to be carefully considered [186, 187]. The social effect: As the cases of information leakage, systematical bias, etc., show in the history, can be observed as such, And the running phase cost of machine learning models is very high energy-consuming, especially when running using GPUs for acceleration of computing.

Transparency and Explainability. As discussed in Chapter 3, explainable AI users need to learn how a model operates and makes decisions. Explainability, an essential aspect of machine learning, closes the distance between predictiveness and human intuition. It has become a fundamental instrument to gain an insight into the rationale of model decisions. Although prediction accuracy has long been the only driving force, the current trend of interpretability has induced academic attention to the black-box interpretation problem of neural networks [188]. In addition, easy to understand explanations enable machine learning models to be deployed while adhering to high ethical standards [189].

These six principles, although they deliver an ethical grounding for responsible AI development, necessitate a concrete regulatory framework for their operationalization. These normative benchmarks, many of them incorporated into global guidelines, have directly led to the development of legally binding institutional arrangements by governments. In the next section, we then consider how these principles are being transformed into the operational rigor of new (partial) AI regulation, and address the legislative approaches appropriate to the translation of abstract ethical ideals into practical compliance demands.

5.3 Possible Solutions and AI Regulations to Address Risks and Challenges

XAI is more than an experimental technique; it is also a critical, viable, and promising route to mitigate these systemic risks, to effect compliant deployment, and to build the trustworthy human-AI partnership that is essential if AI is to realize its potential morally and responsibly.

The need for XAI extends far beyond enhancing a tool’s utility or improving debugging efficiency. It touches upon fundamental pillars of responsible innovation:

- **Trust and Adoption:** Humans trust what they understand. XAI fosters trust among end-users (doctors, loan officers, engineers), enabling more confident adoption and reliance on AI assistance.
- **Accountability and Fairness:** When AI decisions impact lives (loans denied, diagnoses made, cars steered), determining responsibility is paramount. XAI helps identify errors, uncover biases, and ensure systems adhere to ethical and legal standards.
- **Scientific Discovery:** In domains like AlphaFold’s protein folding, understanding *how* the model arrives at its prediction can lead to novel scientific insights about fundamental biological processes, turning the AI from a prediction engine into a discovery tool.
- **Regulation and Governance:** Effective oversight of AI requires scrutability. Regulatory bodies globally are mandating explainability for algorithmic systems making high-impact decisions.
- **Human-AI Collaboration:** The most powerful outcomes often arise when humans and AI collaborate. Explainability is the bridge enabling meaningful dialogue between human intuition, expertise, and the computational power of AI.

AI alignment is another promising direction to deal with the above challenges. As AI systems become increasingly capable, especially with the emergence of large language models and autonomous agents, a central challenge is ensuring that their behavior remains consistent with human goals, ethical norms, and societal values. This problem, referred to as *alignment*, extends beyond merely improving accuracy or performance: it concerns whether an AI system will act in ways that humans deem acceptable, even in novel or ambiguous situations. Modern alignment research spans multiple levels. At the behavioral level, techniques such as Reinforcement Learning from Human Feedback (RLHF) and preference modeling are used to fine-tune models toward human-endorsed outputs. At the normative level, approaches such as Constitutional AI encode higher-level ethical principles that guide model behavior even in the absence of explicit human labels. At the systemic level, alignment focuses on preventing harmful emergent behavior, mitigating biases, ensuring value-sensitive decision making, and maintaining human oversight. Misalignment can lead to discriminatory decisions, unsafe autonomous actions, or unintended strategic behavior in multi-agent systems. Value safety therefore emphasizes embedding ethical, cultural, and domain-specific values into AI models so that they remain not only useful, but also trustworthy and socially compatible. As models grow more autonomous and general, alignment and value safety become indispensable pillars for ensuring that AI technologies remain beneficial in the long term.

The accelerating integration of AI technologies necessitates robust ethical frameworks and coordinated regulatory approaches. As evidenced by Stanford University’s 2023 AI Index Report [190], 127 countries enacted approximately 37 distinct AI-related legislative measures in 2022 alone, reflecting an increase in regulatory activity year-on-year. This tendency underscores a global consensus on the urgency of AI governance, particularly as ethical challenges—such as algorithmic bias and data privacy violations. Against this backdrop, initiatives promoting **Trusted AI** (emphasizing transparency, fairness, and accountability) have emerged as central pillars in global policy discourse.

National Strategic Initiatives

The United States' National AI Initiative Act of 2020 [191] is a recent effort to systematically link technological development with moral ideals. Key priorities are to raise \$1.7 billion of AI R&D - 209% more than what was announced in 2018 - in the effort to drive innovation in trustworthy AI across sectors [190]. Despite these attempts, the lack of comprehensive federal legislation has resulted in a patchwork of regulatory strategies at the state level, with some states having legislation and some not.

The EU has emerged as a global leader in regulation, with cutting-edge regulations on artificial intelligence and data. The General Data Protection Regulation (GDPR) [192] that took effect in May 2018, is a game-changer in terms of empowering EU citizens to control their personal data. Even their core principles (especially, Data Minimisation) have implications led upon training data acquisition and training for AI systems. On top of GDPR, the European Commission put forth the Artificial Intelligence Act in April 2021 [193], going so far as to create explicit AI-dedicated regulations. Recent legislative developments have taken this initiative into the trilogue negotiation stage, putting the EU on track set to adopt the first ever broad-scope AI regulatory framework by early 2024 [194].

Sector-Specific Governance

This was similarly the case with the UK with regards to regulation around Deepfake - with the Online Safety Bill introduced in the previous year, with the confirmation that sharing sexual deepfakes without consent is a crime, and the law for AI-generated material is still being worked out [195]. China has, in the meantime, released the Deep Synthesis Provisions from January 10, 2023 that govern deep synthesis technology with other removal, rights retention, transparency of model with additional obligations such as data retention, relevant content [196].

5.4 Ethical Study about the Transparency of AI

AI systems operate as intricate sociotechnical systems where transparency is a key enabling factor for responsible production. This work frames AI transparency in terms of two related but different dimensions: process transparency (in the form of model explainability) and output transparency (evidenced by open-source implementations). This dual-axis framework extends prior work by Arrieta et al. [73] on XAI and the analysis of how open-source machine learning softwares shape AI development by Langenkamp et al. [197]. Building on complex systems theory [198], we hypothesize that these transparency mechanisms are emergent properties which support higher levels of adaptiveness and resilience of AI systems in different application domains such as healthcare diagnostics, financial risk management, or social infrastructure.

While Arrieta et al. provide a valuable synthesis of XAI literature, and Langenkamp et al. offer key insights from developer/researcher interviews, neither systematically investigates the user perspective on how these transparency dimensions affect key factors like user reliability, transfer/reuse barriers, or AI's future development. Specifically, we need a clearer understanding of how different aspects of transparency produce measurable benefits for trustworthy AI deployment.

Building on this motivation, this section rethinks AI transparency in two dimensions:

- **Process transparency** (explainability of model reasoning)
- **Output transparency** (openness of model outputs and resources)

In this section, we propose a hypothesis model with four key impact areas—user reliability, AI development efficiency, model reusability, and model transferability—and develop a questionnaire to measure how these transparency dimensions influence those outcomes. By providing empirical evidence, this work aims to bridge the gap between the promise of XAI and its demonstrated value in responsible AI practice.

5.4.1 Hypothesis Framework

Here, we propose hypothesis framework H1 - H4 (see Figure 5.5):

H1: User reliability requires AI Explainability.

H2: AI development necessitates AI Explainability.

Sculley et al.'s (2015) technical debt framework [199] states that thinking of the boom of machine learning, especially lacking explainability as coming for free is dangerous. Based on this, we proposed H1: User reliability requires AI Explainability, indicating that AI model need to pay off the debt for its lacking explainability in order to earn user reliability. This hypothesis contends that process transparency (explainability of AI) creates compounding returns through enhanced model auditability, error correction efficiency, and stakeholder trust. Moreover, Miller [200] discusses ways that these can be infused with work on explainable artificial intelligence, triggering us to propose H2: AI development necessitates AI Explainability. H2 follows such process transparency can enable and trigger human-AI collaborations for future development.

H3: AI model's reusability requires AI model to be open-source.

H4: AI's transferability necessitates open-source AI models.

Von Hippel's (2005) [201] theory of "Open source software projects as user innovation networks" claims that open-source software are favorable to user innovations. Moreover, Lerner et al. [202] analyzes previous open-source software cases and identifies a number of attributes that make a project a good or poor candidate for open source development. But it lacks discussion for its future potential. To mitigate the gap, we propose the hypotheses that predict that transparent outcome structures (open source AI models) reduce duplication in research and development (R&D) (H3 - resuability) while enabling cross-domain knowledge transfer (H4 - transferability).

5.4.2 Questionnaire and Workshop Design

In order to validate the relationship between the transparency and the development of AI, we design a corresponding research based on mixed-methods.

Quantitative Phase: Structured Questionnaire

This research employs a sequentially embedded mixed-methods design beginning with psychometrically validated instruments. The questionnaire (details can be found in Appendix A) incorporates:

- Information gathering for professional context and technical proficiency.
- Balanced 5-point Likert scales with reverse-coded items to mitigate the acquiescence bias [203].
- Weighted priority allocation tasks using constant-sum scaling.
- Open questions for potential thoughts and opinions.

This questionnaire investigates user perceptions of AI from different angles. It comprises 20 core questions divided into four sections:

- **Respondent Background (Q1-Q7):** Q1-Q2 cover gender and age; Q3-Q4 inquire education and occupation; Q5-Q7 reveal AI exposure frequency, AI exposure genres, and self-assessed AI knowledge.

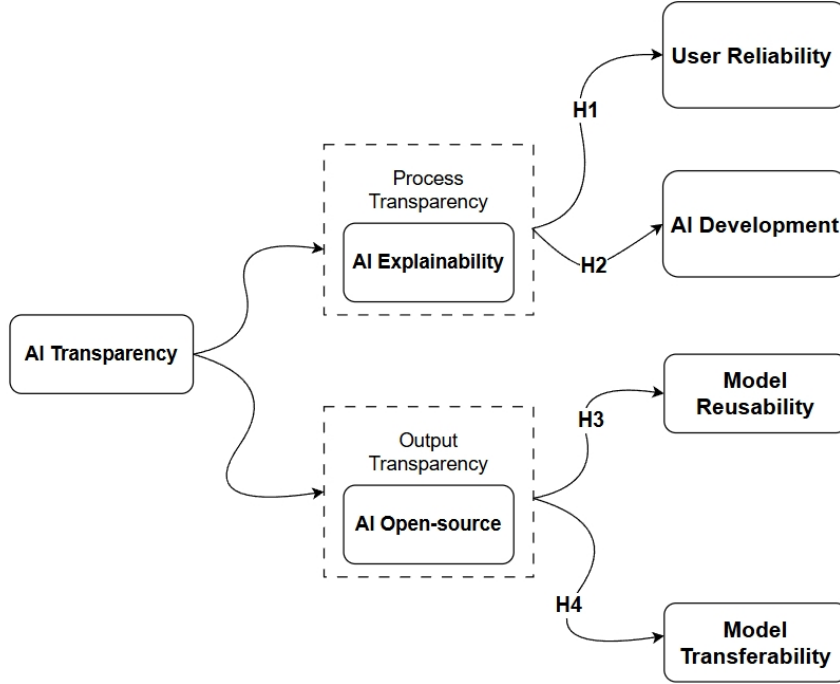


Figure 5.5: Framework of AI transparency hypotheses.

- **Explainability, User Reliability and Development (testing H1 & H2):** Q15 perceives importance of AI explainability; Q18 perceives pace of AI technology development; Q14 perceives AI technology development as **reverse-coded** using cost of implementation; Q12 and Q17 collectively measure **user reliability**.
- **Open-Source, Resuability and Transferability (testing H3 & H4):** Q9 evaluates value of AI open-sourcing; Q11 perceives model reusability; Q10 perceives barriers to resuability as a **reverse-coded** question; and Q8 perceives model transferability across domains.
- **Qualitative and Trade-off Analysis:** Q13, Q19 are open-ended questions for perspectives on AI explainability and open-source adoption; and Q20 ranks AI priorities (Open-source vs. Explainability vs. Performance).

5.4.3 Results and Discussion

To ensure a more robust and geographically diverse response, we employed a dual-pronged distribution strategy for our questionnaire:

- **Offline Workshop in Kyoto:** We organized targeted workshops in Kyoto, Japan, utilizing physical materials such as posters and flyers to directly engage participants and distribute the questionnaire link. More details can be found in Appendix B.
- **Online Social Media & Community Outreach:** The questionnaire was widely promoted through various online platforms and social communities, primarily leveraging popular regional applications like Line and WeChat.

The questionnaire itself was implemented as an anonymous online form (e.g. Google Forms), ensuring respondent privacy. We have collected 206 valid responses between May 2025 and July 2025. To

gauge geographical distribution, we recorded the originating IP addresses of submissions and aggregated the location data. The results are summarized below:

As illustrated in Table 5.1, our sample possesses significant geographical breadth. While the majority of respondents are located in East Asia (China: 140, Japan: 22), the survey successfully captured responses from Southeast Asia (Vietnam: 24, Philippines: 2), Oceania (Australia: 3), Europe (Germany: 6, United Kingdom: 1), and the Americas (United States: 8). This extensive geographical spread, reaching beyond localized networks, significantly reduces the risk of high spatial bias or local clustering in the sample.

Table 5.1: Geographical distribution of valid responses (by Country).

Country	Number of Responses
China	140
Japan	22
Vietnam	24
United States	8
Germany	6
Australia	3
Philippines	2
United Kingdom	1

Furthermore, the demographic composition of the 206 valid responses collected between May 2025 and July 2025 enhances the potential generalizability of our findings:

- **Gender Distribution:** 60.1% male, 35.4% female, and 4.5% preferring not to disclose (see Figure 5.6).
- **Age Distribution:** The largest age cohort comprises individuals aged 25-34 years (60.7%), followed by the 18-24 years group (18.4%). Crucially, respondents aged 18 to 44 years collectively represent 87.9% of the total sample (see Figure 5.7), indicating a strong focus on the prime working-age and young adult population.
- **Education Distribution:** 44.7% have bachelor's degree, consisting of the largest group; 30% have master's degree, consisting of the second largest group; and those education situation is equal to or higher than bachelor account for 82.5% (see Figure 5.8).

Combined with this broad geographic coverage and a significant representation across key demographic variables such as age and gender, the overall profile of the respondent pool supports the argument that the questionnaire results possess a reasonable degree of diversity and general applicability concerning the studied population characteristics. This multifaceted diversity strengthens the **robustness** of the analysis derived from this dataset.

Diversity in professional background was observed among the 206 respondents. A significant proportion (44.7%, n=92) work in non-AI-related industries yet utilize AI tools in their roles. Other represented occupational groups include freelancers (18.0%, n=37) and students (10.1%, n=21) (see Figure 5.9).

When queried about their frequency of AI use (Q5), engagement levels were notably high. The top three reported frequencies were: several times per week (27.2%, n=56), daily (25.2%, n=52), and multiple times per day (22.3%, n=46). As Figure 5.10 illustrates, only a small minority (17.0%) reported rare or no usage of AI tools. This pattern strongly indicates that AI has transitioned beyond a niche technology to become deeply integrated into the daily routines of a significant majority of respondents. This conclusion is further reinforced by Q7 results (Figure 5.11 & 5.12), which assessed self-reported AI knowledge: an overwhelming 91.0% of respondents possess at least basic familiarity with AI, with only 9.0% reporting no knowledge.

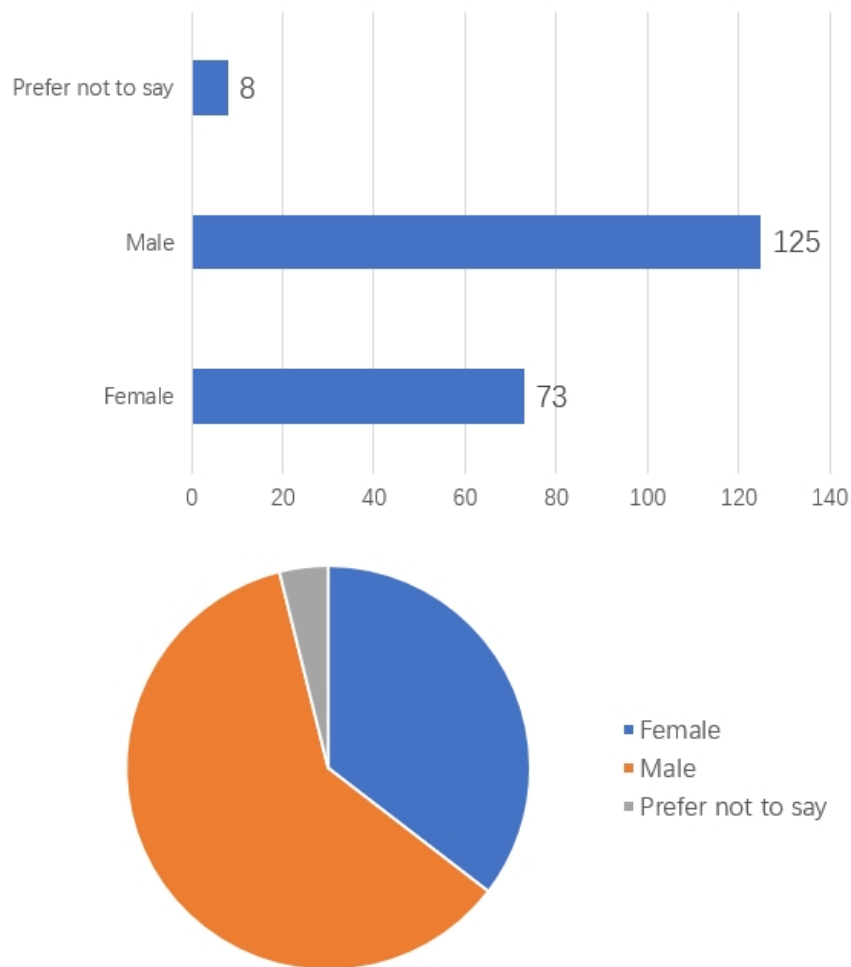


Figure 5.6: Gender distribution of survey respondents (N=206).

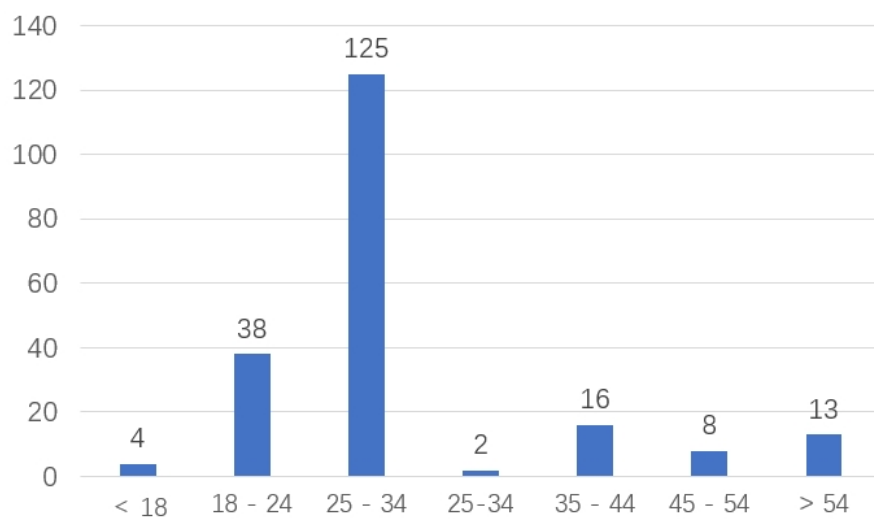


Figure 5.7: Age distribution of survey respondents (N=206).

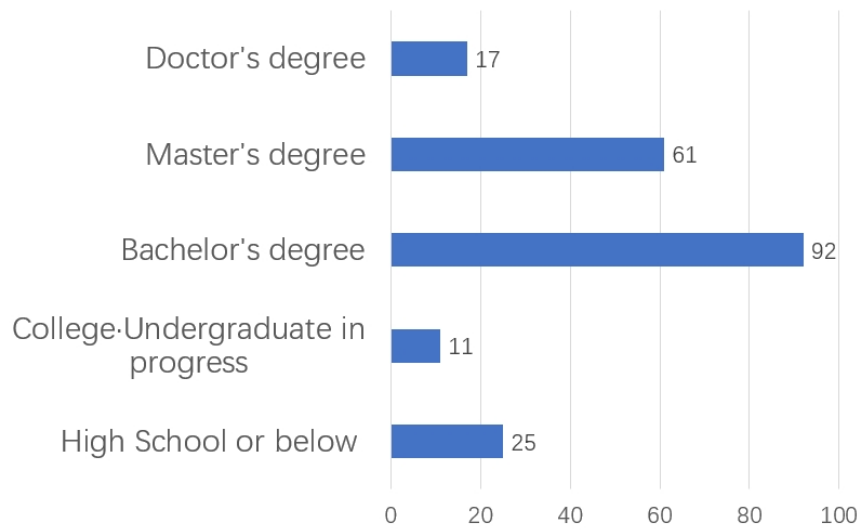


Figure 5.8: Education distribution of survey respondents (N=206).

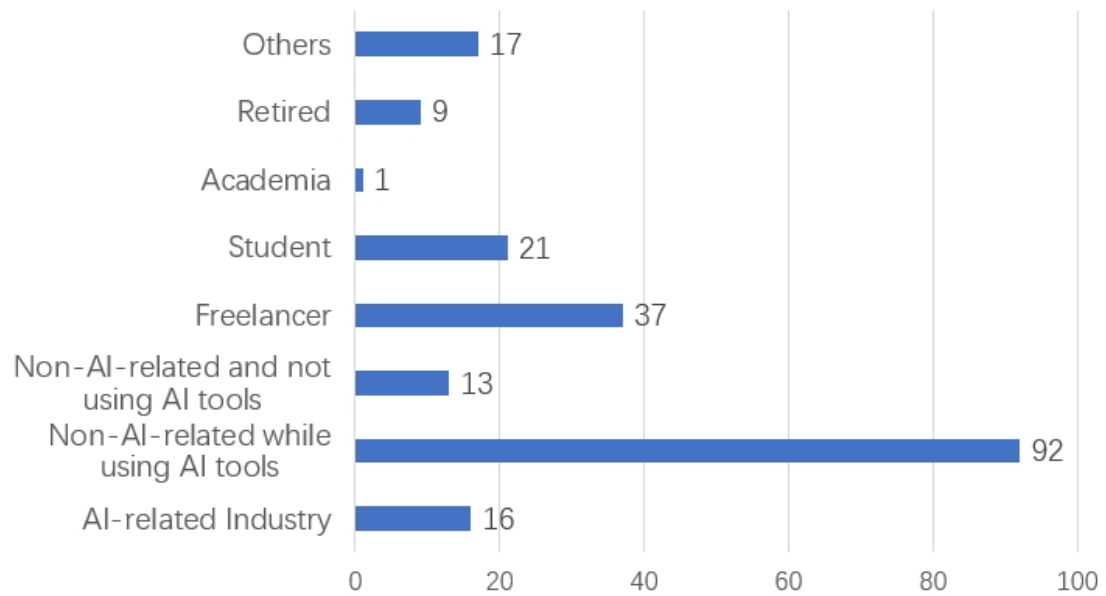


Figure 5.9: Occupation distribution of survey respondents (N=206).

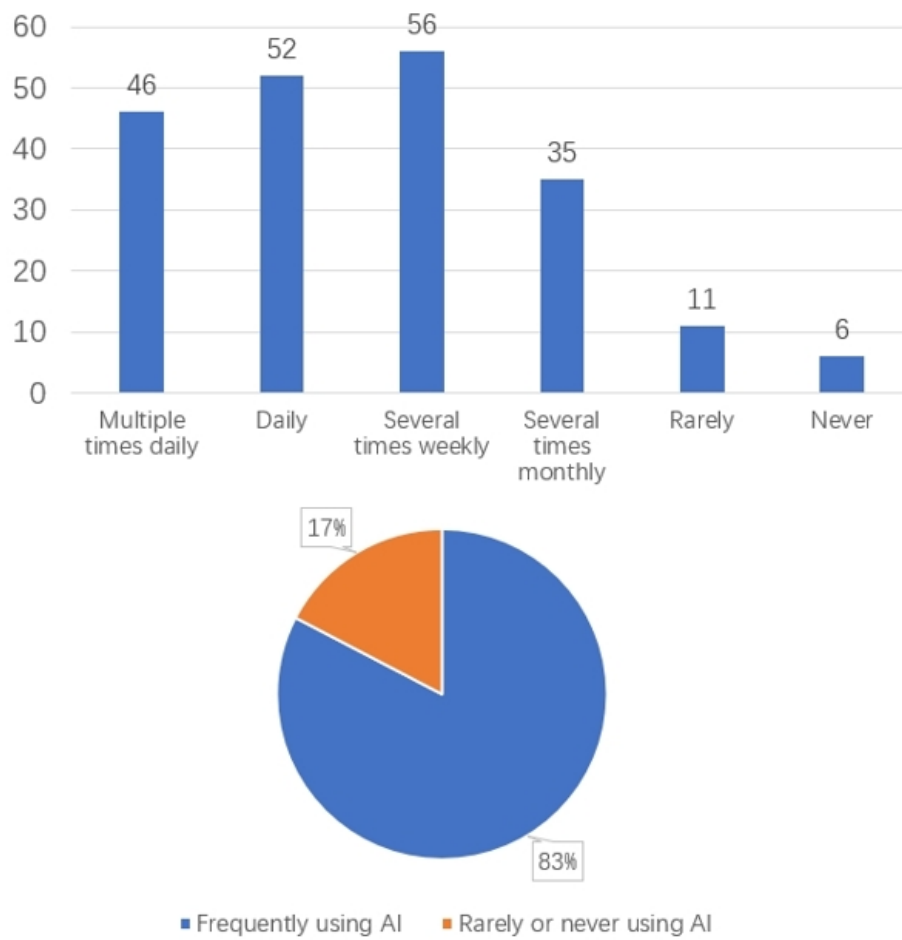


Figure 5.10: AI exposure frequency of respondents (N=206).

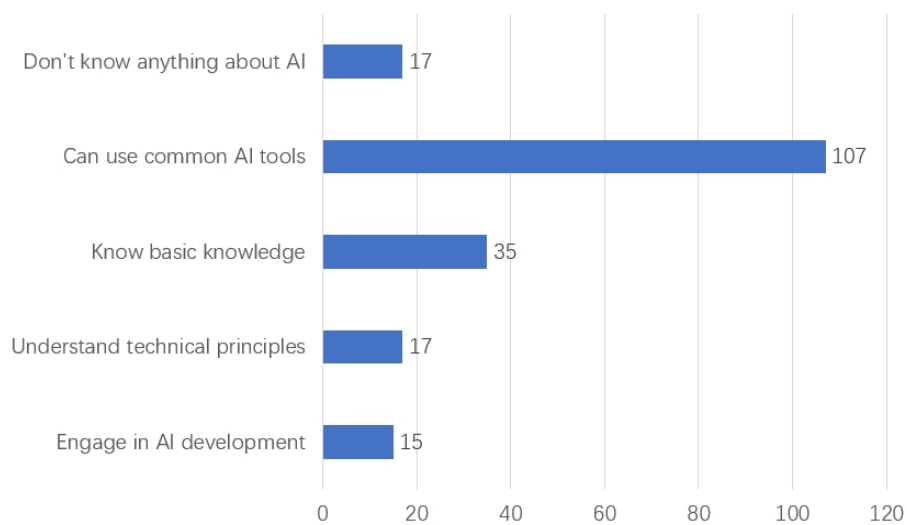


Figure 5.11: Self-assessed knowledge towards AI of respondents (N=206).

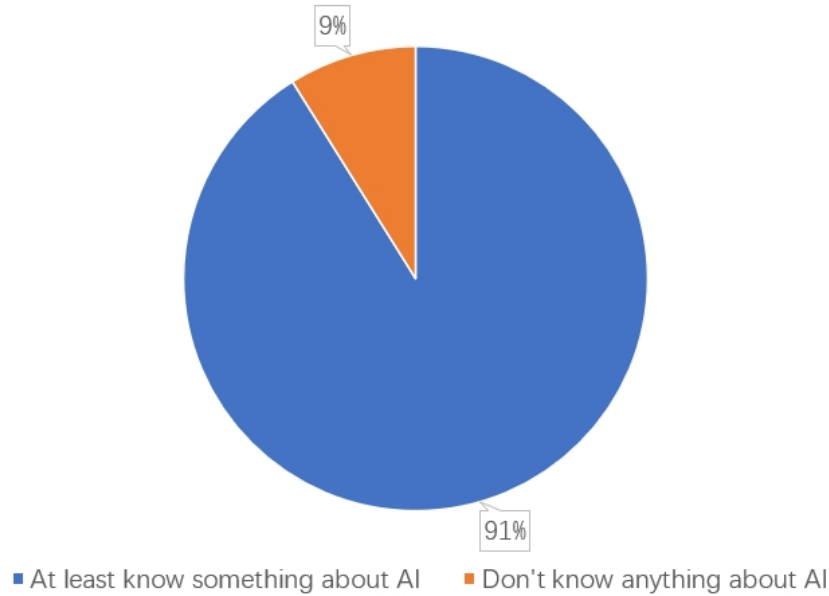


Figure 5.12: AI knowledge coverage in respondents.

Respondents identified (Q6, multiple selections allowed) the AI applications they commonly employ in daily life. As shown in Figure 5.13, the top three most prevalent applications are:

- **1. AI-generated Content (AIGC) tools:** (e.g., ChatGPT, Deepseek) - 78.2% (n=161)
- **2. Voice Assistants:** (e.g., Siri, Google Assistant) - 52.9% (n=109)
- **3. Recommendation Algorithms:** (e.g., as used by Amazon, TikTok, YouTube) - 49.0% (n=101)

This distribution aligns closely with current mainstream adoption trends. Conversely, specialized applications such as financial AI tools or medical AI diagnostics were reported less frequently, indicating their current use remains relatively niche compared to the broad accessibility and utility of AIGC, voice assistants, and recommendation systems.

After the respondent background analysis, we assessed the internal consistency of the multi-item scale (Q8–Q12 and Q14–Q18) using Cronbach's α [204]. All items were measured on a 5-point Likert scale (1 = strongly disagree, 5 = strongly agree). Cronbach's α is deeply embedded in scale development literature as a first-line diagnostic tool, providing a single intuitive coefficient (0-1) that's easily comparable across studies. Tavakol and Dennick confirmed the efficacy of Cronbach's α and proved if a test has a Cronbach's α of 0.80, there is 0.36 error variance (random error) in the scores [205]. The analysis yielded a coefficient value of **0.874**, which exceeds the threshold of 0.8, indicating good internal reliability for the scale.

Subsequently, we examined the correlations between the multi-item scales. Figure 5.14 visualizes the correlations for Q8-Q12 and Q14-Q18. As Q10 and Q16 exhibited weak correlations with other questions (most $|r| < 0.3$), we excluded them from the primary hypothesis framework. This decision aligns with standard psychometric procedures for refining scales to improve internal consistency and construct validity [206]. These items will be analyzed separately after confirming the framework.

After excluding Q10 and Q16, formal Pearson correlation tests were performed to assess significance. Figure 5.15 reveals all inter-question correlations were statistically significant (2-tailed $p < .001$).

Based on the results of Pearson correlation tests [207], we can confidently following the hypotheses transform the multi-scale questions into variables as Table 5.2.

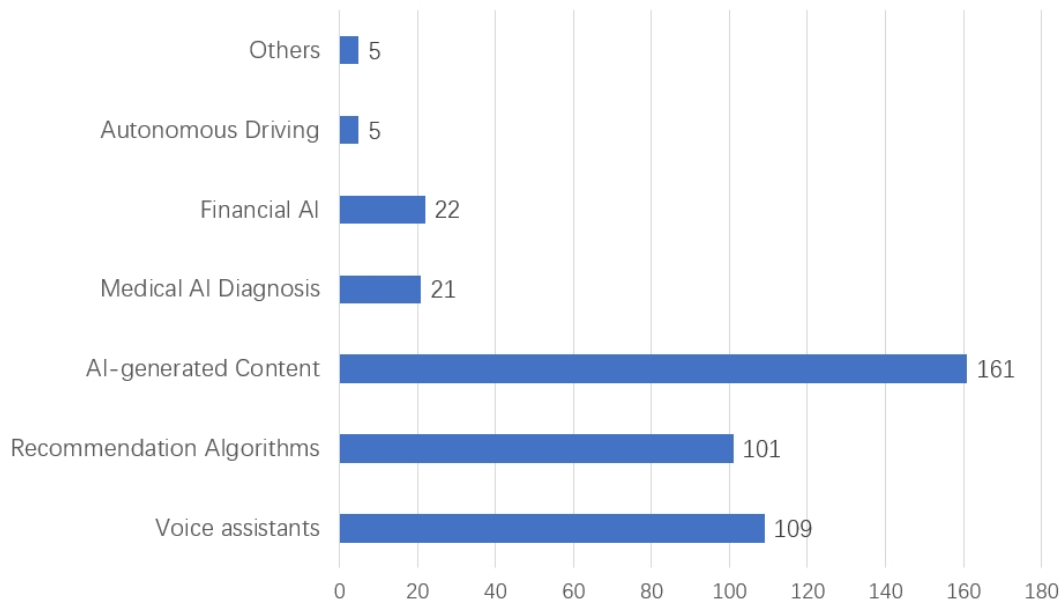


Figure 5.13: AI application exposure to respondents, where multiple choices are allowed.

	Q8	Q9	Q10	Q11	Q12	Q14	Q15	Q16	Q17	Q18
Q8	1									
Q9	0.6390432	1								
Q10	0.1408575	0.24235	1							
Q11	0.5885166	0.603434	0.265532	1						
Q12	0.5904092	0.531996	0.279249	0.609154	1					
Q14	0.4908334	0.433086	0.317767	0.463936	0.42351	1				
Q15	0.5042125	0.43977	0.234904	0.50302	0.45205	0.528036	1			
Q16	0.2450272	0.264676	0.389806	0.231754	0.23776	0.335053	0.220306	1		
Q17	0.6018684	0.539685	0.202865	0.543178	0.48673	0.54115	0.590548	0.237613	1	
Q18	0.5757359	0.57575	0.157668	0.622849	0.55753	0.518297	0.581076	0.175792	0.690351	1

Figure 5.14: Correlations between the multi-item scale questions.

Table 5.2: Mapping between questionnaire items and model variables.

Question Item	Variable
Q15	y_{explain}
Q9	y_{open}
Q12	x_{relia1}
Q17	x_{relia2}
Q18	x_{develop1}
Q14	x_{develop2}
Q11	x_{reuse}
Q8	x_{transfer}

		Correlations							
		Q8	Q9	Q11	Q12	Q14	Q15	Q17	Q18
Q8	Pearson Correlation	1	.639**	.589**	.590**	.491**	.504**	.602**	.576**
	Sig. (2-tailed)		<.001	<.001	<.001	<.001	<.001	<.001	<.001
	N	206	206	206	206	206	206	206	206
Q9	Pearson Correlation	.639**	1	.603**	.532**	.433**	.440**	.540**	.576**
	Sig. (2-tailed)	<.001		<.001	<.001	<.001	<.001	<.001	<.001
	N	206	206	206	206	206	206	206	206
Q11	Pearson Correlation	.589**	.603**	1	.609**	.464**	.503**	.543**	.623**
	Sig. (2-tailed)	<.001	<.001		<.001	<.001	<.001	<.001	<.001
	N	206	206	206	206	206	206	206	206
Q12	Pearson Correlation	.590**	.532**	.609**	1	.424**	.452**	.487**	.558**
	Sig. (2-tailed)	<.001	<.001	<.001		<.001	<.001	<.001	<.001
	N	206	206	206	206	206	206	206	206
Q14	Pearson Correlation	.491**	.433**	.464**	.424**	1	.528**	.541**	.518**
	Sig. (2-tailed)	<.001	<.001	<.001	<.001		<.001	<.001	<.001
	N	206	206	206	206	206	206	206	206
Q15	Pearson Correlation	.504**	.440**	.503**	.452**	.528**	1	.591**	.581**
	Sig. (2-tailed)	<.001	<.001	<.001	<.001	<.001		<.001	<.001
	N	206	206	206	206	206	206	206	206
Q17	Pearson Correlation	.602**	.540**	.543**	.487**	.541**	.591**	1	.690**
	Sig. (2-tailed)	<.001	<.001	<.001	<.001	<.001	<.001		<.001
	N	206	206	206	206	206	206	206	206
Q18	Pearson Correlation	.576**	.576**	.623**	.558**	.518**	.581**	.690**	1
	Sig. (2-tailed)	<.001	<.001	<.001	<.001	<.001	<.001	<.001	
	N	206	206	206	206	206	206	206	206

** . Correlation is significant at the 0.01 level (2-tailed).

Figure 5.15: Pearson correlations between questions (N=206).

Table 5.2 maps responses from Q15 to AI explainability (y_{explain}) and from Q9 to AI open-source level (y_{open}). Moreover, we use results from Q11 to denote model reusability as x_{reuse} and results from Q8 to denote model transferability as x_{transfer} . Figure 5.15 reveals a significant positive correlation between Q12 and Q17. Clark and Watson recommended item aggregation to capture broader aspects of constructs [208]. Hence, we can reasonably define **user reliability** as the composite variable $x_{\text{relia}} = \frac{x_{\text{relia1}} + x_{\text{relia2}}}{2}$ to represent the user reliability. Similarly, we use $x_{\text{develop}} = \frac{x_{\text{develop1}} + x_{\text{develop2}}}{2}$ to represent the AI development status. Note that Q14 was initially hypothesized as a reverse-coded measure of AI development. However, Figure 5.15 shows that Q14 exhibits a significant positive correlated with Q18 (another AI development indicator). This correlation pattern contradicts the hypothesis, suggesting that investments in AI explainability may not inherently hinder development. We therefore omit reverse coding for Q14 in subsequent analysis.

A multiple linear regression (MLR) analysis was conducted to investigate the relationship between AI explainability (y_{explain}) and two predictor variables: user reliability (x_{relia}) and AI development status (x_{develop}). The model demonstrated substantial explanatory power (Figure 5.16). With $R^2 = .444$, the predictors collectively accounted for 44.4% of variance in AI explainability. The adjusted $R^2 = .439 > 0.3$ (exceeding the .30 threshold for practical significance) further confirms the model's robustness against overfitting.

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.666 ^a	.444	.439	.76269

a. Predictors: (Constant), Development, Reliability

Figure 5.16: Multiple Linear Regression model summary for dependent variable y_{explain} and predictors x_{develop} , x_{relia} .

Model significance was formally assessed through ANOVA (Figure 5.17) [209, 210]. Results indicated the regression model was statistically significant (with $p < .001$), rejecting the null hypothesis. The final parameter estimates (Figure 5.18) yielded the following significant ($p < .001$) predictive equation, which supports the Hypothesis H1 and H2:

$$y_{\text{explain}} = 0.358 * x_{\text{relia}} + 0.512 * x_{\text{develop}} + 0.511. \quad (5.1)$$

Note that $y_{\text{explain}} = 0.511$ when $x_{\text{relia}} \rightarrow 0$ and $x_{\text{develop}} \rightarrow 0$. This signals that users inherently demand a baseline level of explainability from AI models. Since y_{explain} reflects the importance users place on explainability, this persistent minimum value confirms a key consensus: users interacting with an AI application typically engage in basic due diligence. At the very least, they expect transparency regarding the model's developer company or the methodology employed. In other words, a fully opaque "black-box" AI is generally unacceptable to typical users.

Similarly, we conducted a multiple linear regression (MLR) analysis to investigate the relationship between AI open-source status (y_{open}) and two predictor variables: model reusability (x_{reuse}) and model transferability (x_{transfer}). The model also showed substantial explanatory power (Figure 5.19). With $R^2 = .487$, the predictors collectively accounted for 48.7% of variance in AI open-source status. The adjusted $R^2 = .482 > 0.3$ (exceeding the .30 threshold for practical significance) further confirms the model's robustness.

Model significance was formally assessed through ANOVA (Figure 5.20). Results indicated the regression model was statistically significant (with $p < .001$), rejecting the null hypothesis. The final parameter estimates (Figure 5.21) yielded the following significant ($p < .001$) predictive equation:

$$y_{\text{open}} = 0.444 * x_{\text{transfer}} + 0.353 * x_{\text{reuse}} + 0.713. \quad (5.2)$$

ANOVA ^a						
Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	94.327	2	47.164	81.079	<.001 ^b
	Residual	118.085	203	.582		
	Total	212.413	205			

a. Dependent Variable: Explainability

b. Predictors: (Constant), Development, Reliability

Figure 5.17: ANOVA results of Multiple Linear Regression model for dependent variable y_{explain} and predictors x_{develop} , x_{relia} .

Coefficients ^a						
Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	.511	.276		1.850	.066
	Reliability	.358	.092	.297	3.898	<.001
	Development	.512	.093	.418	5.488	<.001

a. Dependent Variable: Explainability

Figure 5.18: Coefficients of Multiple Linear Regression model for dependent variable y_{explain} and predictors x_{develop} , x_{relia} .

Model Summary				
Model	R	R Square	Adjusted R Square	Std. Error of the Estimate
1	.698 ^a	.487	.482	.73234

a. Predictors: (Constant), Reusability, Transferability

Figure 5.19: Multiple Linear Regression model summary for dependent variable y_{open} and predictors x_{reuse} , x_{transfer} .

Similarly, $y_{\text{open}} = 0.713$ when $x_{\text{transfer}} \rightarrow 0$ and $x_{\text{reuse}} \rightarrow 0$. This indicates that respondents inherently demand a baseline level of open-source from AI models, i.e., at least the model needs to open-source the model's developer company or the methodology employed. Meanwhile, if the developers/researchers expect to enhance the transferability and reusability of the model, the open-source level also requires increasing. This equation and corresponding analysis confirmed H3 and H4.

ANOVA^a

Model		Sum of Squares	df	Mean Square	F	Sig.
1	Regression	103.540	2	51.770	96.529	<.001 ^b
	Residual	108.872	203	.536		
	Total	212.413	205			

a. Dependent Variable: Opensource
b. Predictors: (Constant), Reusability, Transferability

Figure 5.20: ANOVA results of Multiple Linear Regression model for dependent variable y_{open} and predictors x_{reuse} , x_{transfer} .

Additionally, although Q10 and Q16 were not integrated into the regression models for hypothesis testing, they still yield insightful conclusions. For instance, Pearson correlation analysis of Q10 shows no significant relationship between AI misuse and AI development, user reliability, model reusability, or model transferability. Furthermore, the mean response for Q16 (3.2) indicates that a majority of respondents do not perceive an inherent contradiction between AI explainability and model accuracy.

Beyond the five-point Likert scale items, Question 20 employed a clinical simulation requiring participants to allocate constrained resources. Respondents distributed 100 points (representing total development resources) across AI model performance, explainability, and open-source availability according to their perceived importance.

After excluding four invalid responses specifically for Q20 (due to non-participation or misunderstanding), 202 valid allocations remained. Table 5.3 summarizes the results, revealing that participants prioritized AI model performance most heavily, followed by explainability and model open-source availability respectively. Nevertheless, although not the primary determinants, model explainability and model open-source status exhibited non-negligible influence in these outcomes. These findings may serve as multidimensional optimization strategies for AI model developers in subsequent research.

In addition, we have two open-ended questions to gather deeper thoughts for AI explainability and AI open-source status. Q13 inquires the key bottlenecks for AI open-sourcing; and Q19 asks about the practical challenges for XAI implementation. In the following, we've summarized responses that are both profound and high-frequency.

Coefficients^a

Model		Unstandardized Coefficients		Standardized Coefficients	t	Sig.
		B	Std. Error	Beta		
1	(Constant)	.713	.238		2.993	.003
	Transferability	.444	.064	.434	6.989	<.001
	Reusability	.353	.063	.348	5.596	<.001

a. Dependent Variable: Opensource

Figure 5.21: Coefficients of Multiple Linear Regression model for dependent variable y_{open} and predictors x_{reuse} , x_{transfer} .

Table 5.3: Mean and standard deviation of results from Q20 (N=202).

	Model open-source	Model explainability	Model performance
Mean	21.3	30.9	47.8
Std. Dev.	0.1165	0.1019	0.1463

Top 5 answers to Q13:

- Conflicting interests of all parties.
- Worries about data and privacy security.
- Possibilities of AI abusing.
- Intellectual Property Rights Disputes and Protection Issues.
- Legal Liability Issues in Open-Source AI.

And top 5 answers to Q17:

- Performance-Explainability Trade-off.
- Pseudo-explanations / Mis-explanations.
- Divergent needs across stakeholder groups (e.g., users vs. developers) and problem types (e.g., medical vs. retail), hindering standardized benchmarks.
- Inherent complexity of deep learning architectures obstructing human-intelligible interpretations.
- Cost implications of developing supplementary explainability solutions.

Furthermore, to substantiate these findings and deepen the discussion surrounding AI’s possibilities and challenges, we organized a structured workshop titled “Harmonizing Innovation and Threats – Explainable AI,” held on **Saturday, June 7, 2025**, at Room 113, 1F, Higashi Ichijo Building, Kyoto University, in a hybrid format (30 in-person participants and 50 online via Zoom). The workshop brought together AI experts from government agencies, academia, AI start-ups, and AI-driven industries to examine the rapid development of AI technologies and the growing concerns associated with their deployment. Against the backdrop of AI’s unprecedented global recognition in 2024—including Nobel Prizes awarded in both Physics and Chemistry for contributions related to neural networks and computational protein design—the workshop addressed pressing issues such as data bias, intellectual property concerns, model opacity, and risks of misuse.

During the event, distinguished speakers provided insights that reinforced the key themes of this chapter. Hamada Taiyo from Araya Inc. examined the geometric structure of internal neural representations (“representational geometry”) and explained how understanding these latent spaces may help resolve alignment challenges and improve model explainability. Hayashi Yuseke from AI Alignment Network highlighted two central research directions essential to modern AI safety: (1) mechanistic explainability, which aims to make Transformer-based large language models intelligible to humans; and (2) AI alignment, which ensures that model outputs and autonomous behaviors conform to human intentions and values—while also acknowledging that alignment research may expose vulnerabilities or pathways to model failure. Watanabe Takuya from Ministry of Economy, Trade and Industry (METI) provided an overview of Japan’s governmental frameworks, institutions, and emerging regulatory measures governing AI development and deployment. Hashimoto Takashi from Bank of Japan (BOJ) offered practical perspectives on adopting AI technologies within financial institutions, emphasizing real-world constraints and risk considerations. Finally, Iida Masahito from Mitsubishi Research Institute, Inc. (MRI) discussed long-term challenges and emerging opportunities in the evolution of AI technologies.

Collectively, the workshop underscored the growing urgency of enhancing AI transparency, particularly through explainability, as a cornerstone of responsible and trustworthy AI development.

5.4.4 Conclusion

This section commenced with a review of the prevailing risks and challenges posed by artificial intelligence (AI), alongside an examination of current regulatory frameworks emerging globally. Building upon this foundation, we introduced eXplainable AI (XAI) as a potentially crucial solution to these challenges. To advance the understanding of the relationship between XAI principles and broader AI development, we adopted a higher-level, conceptual framework analyzing AI model transparency through two distinct, complementary lenses: **process transparency** (operationalized as AI explainability) and **output transparency** (operationalized as AI open-source level).

We further proposed a hypothesized framework relating these transparency dimensions to four critical aspects: user reliability, AI development status, model reusability, and model transferability. To empirically investigate these relationships, a **validated scale was developed and deployed** via an online questionnaire. Between May and July 2025, we successfully collected **206 valid responses**. The questionnaire demonstrated strong internal consistency, evidenced by a Cronbach's α coefficient of 0.874 across all constructs.

Our analysis yielded significant findings. Two key multiple linear regression models provided robust empirical support for our core hypotheses. The results clearly demonstrate that AI explainability (process transparency) exhibits a significant positive correlation with both user reliability and AI development status. Furthermore, the analysis conclusively showed that higher levels of open-source disclosure (output transparency) are fundamentally required to enable model reusability and transferability.

These quantitative insights were powerfully reinforced during our Kyoto workshop. Keynote presentations and subsequent discussions consistently highlighted the critical nature of transparency-related challenges and underscored the transformative potential of effective explainability mechanisms. Collectively, the empirical data and expert discourse converge to unequivocally affirm the foundational importance of process and output transparency in AI models. This synthesis not only validates our framework but also provides a clear direction towards achieving responsible and trustworthy AI development.

5.5 Conclusion of the Chapter

Despite the huge success achieved by from graph-based AI to general AI, this chapter explored risks, challenges, and ethical considerations relevant to the application in real-world environments. While graph learning methods continue to mature and demonstrate strong technical potential, the discussion and empirical observations in this chapter suggest that the development of explainability, safety, and governance mechanisms has not advanced at the same pace. The survey results provide indications that expectations regarding transparency and openness—particularly from users and non-expert stakeholders—may be higher than what current research and deployed systems typically provide. Although this study is not exhaustive, these findings highlight areas where further investigation and improvement could be valuable.

The chapter also suggests that ethical and societal considerations are closely connected to core technical questions rather than existing as separate concerns. Issues such as trust, interpretability, and accountability appear to play an important role in determining whether AI systems can be responsibly integrated into practice, especially in situations involving sensitive or high-impact decisions. These observations reinforce the perspective that technical performance alone may not be sufficient for real-world acceptance, and that complementary methodological and institutional developments may be needed.

Overall, this chapter does not claim to offer definitive solutions, but instead aims to contribute preliminary insights and empirical evidence supporting the future importance of research at the intersection of graph learning, explainability, and responsible AI. It is hoped that these initial findings may encourage further collaborative work across disciplines—combining expertise from machine learning, human-centered design, governance, and emerging computational technologies—to build AI systems that are not only more capable but also more transparent, reliable, and aligned with societal needs.

Chapter 6

Conclusion, Discussion, and Future Direction

6.1 Summary of Contributions

This thesis has investigated the learning for graphs, its explainability challenges, its potential in and intersection with quantum computation, and its broader ethical implications within the rapidly evolving landscape of artificial intelligence. Across the chapters, the thesis integrates mathematical foundations, methodological developments, theoretical analyses, empirical studies, and ethical reflections into a coherent narrative that studies both technical understanding and responsible AI practice.

Chapter 1 provided the overall motivation and contextualized the importance of learning and learning on graphs at a time when artificial intelligence is experiencing rapid technological acceleration and rising societal impact. It outlined the significance that “why learning on graphs is important” throughout the thesis and introduced the thematic structure connecting subsequent chapters.

Chapter 2 established the mathematical and methodological foundations necessary for studying modern graph learning. It reviewed the properties of graphs, the limitations of traditional machine learning approaches for graph-structured data, and the principles behind contemporary graph learning, especially GNNs. Through a comparative experiment between mol-infer and GraphSAGE, the chapter not only highlights the advantages of both structural learning over traditional machine-learning approaches that rely on handcrafted feature engineering and contemporary graph learning, but also illustrates the respective strengths and limitations of the two paradigms. By providing quantitative evidence and clear performance contrasts, it offers practical guidance for future developers and researchers on how to select an appropriate method when facing real-world graph-related problems. At the same time, the findings from this experiment establish an empirical foundation for the explainability issues discussed in the subsequent chapters.

Chapter 3 addressed the black-box nature of graph learning models by analyzing the landscape of graph explainability research. After reviewing existing explainability paradigms and their inductive reasoning foundations, the chapter identified the insufficient treatment of *connectedness* as a key gap in the literature. To remedy this, two novel contributions were proposed: *Connected-CF²*, which approximates connectedness through linear algebra representation, and *CRoCExplainer*, which introduces a principled continuous connectedness metric and incorporates it into explainer frameworks. Theoretical insights and empirical evaluations demonstrated that these approaches offer complementary and structurally faithful forms of explanation.

Chapter 4 explored the intersection between graph learning and quantum computation. After introducing the fundamentals of quantum computing relevant to machine learning, the chapter surveyed how QGNN is promising in its theoretical higher efficiency and stronger capacity of representation. However, it further analyzed current hardware bottlenecks that limit quantum machine learning and motivated

the need for structural optimization. A case study on quantum electronic design automation illustrated how graph-related, computer graphics-based methods can address practical challenges in quantum circuit design, highlighting an emerging yet promising direction at the interface of quantum technology.

Chapter 5 broadened the scope from technical methodologies to societal and ethical considerations. It reviewed the risks, challenges, and governance frameworks associated with modern AI systems, with particular emphasis on transparency and public trust. Building upon these discussions, the chapter presented a quantitative study based on over 200 survey responses to evaluate public perceptions of AI transparency. The empirical findings underscored the importance of explainability and openness in fostering responsible AI development and provided data-driven insights into an area where empirical evidence has been limited.

Together, the contributions of this thesis advance the understanding of graph learning from foundational theory to explainability, from quantum applications to ethical implications. The combination of methodological innovation and ethical reflection offers a multifaceted perspective that supports both the development of reliable graph-based AI systems and the broader pursuit of trustworthy artificial intelligence.

6.2 Discussion and Conclusion

The findings of this thesis contribute to several ongoing conversations within graph learning, explainable AI, quantum machine learning, and AI governance. At the technical level, the proposed explainability methods in Chapter 3 demonstrate that incorporating structural priors—particularly graph connectedness—can meaningfully enhance the fidelity and interpretability of explanations.

However, it is important to note that this analysis implicitly relies on an underlying assumption: **connectedness matters**, and therefore considering it in explainability is meaningful only when this property plays a significant role in the task. One explicit advantage of our proposed approach is that, because we quantify connectedness through an approximate metric, the corresponding term in the final explanation optimization can be flexibly adjusted—rather than being fixed to a particular setting—when connectedness is no longer a primary factor. This, in turn, reveals a general characteristic of graph-based models: many of their computations inherently depend on specific structural assumptions.

As discussed in Chapter 2, we introduced the distinction between homophilic and heterophilic graphs (see Figure 6.1). Most standard GNNs, due to their dependence on message passing, effectively operate under the assumption of graph homophily; that is, they presume that nodes with similar local structures share similar properties. When the target domain involves heterophilic graphs, specialized architectures are often required for the model to remain effective. Given the wide diversity of graph types and structural characteristics, a universal “one model/approach fits all” solution is essentially unattainable.

The same principle applies to explainability in graph learning. When focusing on a particular application, one must first understand the structural properties of the graphs involved and then select appropriate methods and metrics accordingly in order to achieve theoretically sound and practically meaningful results. Nonetheless, within this process, the methodological framework adopted in our publication remains valid and applicable.

The analysis in Chapter 4 highlights both the theoretical promise and the practical limitations of quantum machine learning. While quantum models offer intriguing advantages—such as the potential for exponential state representation, quantum parallelism, and new forms of graph-based computation—the current state of quantum hardware remains a significant bottleneck. Contemporary devices are constrained by limited qubit counts, high error rates, decoherence, and substantial operational costs, all of which prevent quantum models from achieving the stability and scalability required for real-world machine learning applications.

To explore this gap, Chapter 4 presented a parametric algorithm for automated airbridge placement on superconducting quantum chips, addressing a previously unsolved layout problem essential for suppressing parasitic modes in qubit arrays. Although this contribution demonstrates how graph-related reasoning

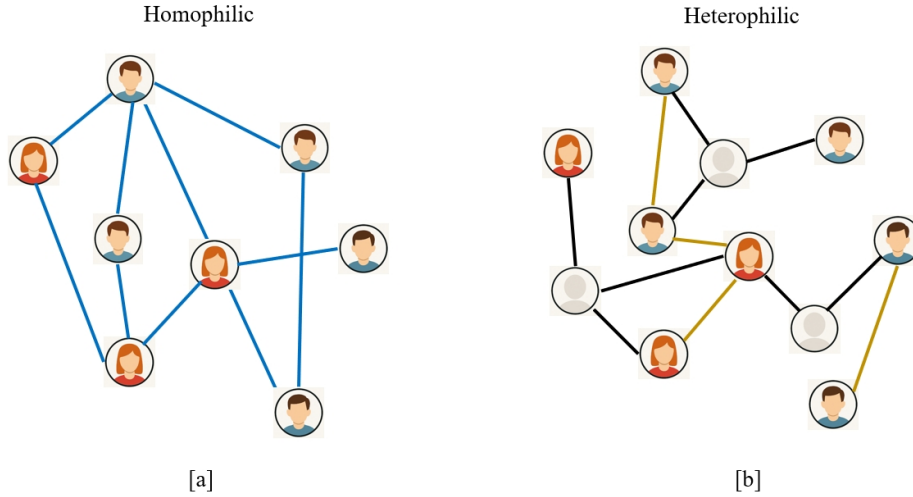


Figure 6.1: An illustration of homophilic and heterophilic graphs. [a] A homophilic social network, where blue edges represent friendship relations. In such networks, connected (adjacent) nodes tend to share similar attributes, reflecting the common assumption that “friends are alike.” [b] A heterophilic social network that encodes negative or conflicting relationships. Black edges denote strongly adversarial ties, while brown edges indicate past disputes. In contrast to panel [a], adjacent nodes in heterophilic graphs are more likely to be dissimilar. This highlights the fundamentally different structural assumptions underlying the two types of graphs.

can assist in quantum hardware design, it represents only a small step toward the broader goal of enabling practical quantum machine learning. Considerable advances in qubit fabrication, error correction, chip architecture, and design automation are still required before quantum devices can fully realize the computational advantages predicted by quantum learning theory. In this sense, the chapter underscores not only the potential of quantum-enhanced graph methods but also the substantial hardware-focused research still needed for that potential to be meaningfully realized.

Beyond methodological advances, Chapter 5 extends the thesis into broader societal and ethical considerations. As AI systems become embedded in high-stakes environments, issues such as transparency, accountability, fairness, and governance become inseparable from technical performance. The results of the transparency perception study indicate that public expectations for responsible AI are not yet matched by current research or industry practice. In particular, the strong influence of explainability on user trust and of openness on model reusability emphasizes that technical innovation must be accompanied by ethical and institutional development. These insights underscore a recurring theme throughout the thesis: AI technologies cannot be understood or evaluated solely through their predictive accuracy; they must be situated within the social, cultural, and regulatory contexts in which they operate.

Conclusion. Despite the rapid progress of artificial intelligence and its accelerated integration into high-stakes domains, research on explainability, safety, and ethical considerations remains substantially behind the pace of technological deployment. As demonstrated throughout this thesis, predictive performance alone is no longer sufficient for AI systems that influence scientific discovery, medical decision-making, financial risk analysis, public infrastructure, and many other critical societal functions. Without transparency, accountability, and trust, even the most advanced models risk becoming unusable or harmful in real-world settings.

This thesis has made modest but meaningful contributions toward addressing this gap. From the methodological advances in explainable graph learning and the development of connectedness-aware reasoning frameworks, to the exploration of quantum hardware design and the empirical study of public expectations regarding AI transparency, the work presented herein illustrates how technical innovation

and ethical responsibility can be mutually reinforcing. These insights affirm that the intersection of explainability, ethical governance, and emerging computation paradigms is not only an important research frontier but also a foundation for the long-term sustainability and trustworthiness of AI.

Looking forward, the author hopes that more researchers across disciplines—spanning machine learning, human-computer interaction, systems engineering, policy, and philosophy—will join in advancing responsible AI. Building AI that is understandable, secure, and aligned with human values is a collective undertaking: one that will require persistent effort, openness, and collaboration. It is only through such shared commitment that AI can evolve into a technology that empowers rather than displaces, supports rather than threatens, and truly contributes to a more reliable and just future.

6.3 Future Directions

The work presented in this thesis suggests multiple avenues for future investigation. First, the graph explainability methods developed in Chapter 3 may be extended by incorporating additional structural priors beyond connectedness. For instance, future explainers could integrate motif detection, causal substructures, or spectral properties to capture richer forms of relational reasoning. A promising direction is to develop explainers that operate jointly on graph topology and node feature geometry, enabling explanations that reflect both structural and semantic coherence.

Another possible direction is to promote the CRoC metric. CRoC is capable of evaluating both connectedness and symmetry, while this ability also brings side-effect that it cannot distinguish which part causes a minor difference. If more explorations to be made to set apart its capability in connectedness and symmetry, more delicate analysis can be made to graphs. Moreover, the computational complexity of connectedness-aware explanation remains an open challenge. While CRoC improves upon previous approximations, achieving sub-cubic computation or incremental evaluation for large dynamic graphs would significantly broaden the applicability of connectedness-based metrics. This invites research in numerical linear algebra, spectral sparsification, and randomized graph algorithms.

Furthermore, there is substantial opportunity to connect graph explainability with causal inference. Counterfactual explainability naturally intersects with causal reasoning, and integrating causal graph discovery or structural causal models could yield explanations that more faithfully reflect real-world mechanisms rather than statistical associations. Such integration would be particularly impactful in scientific domains such as chemistry, biology, and materials science.

The interplay between graph learning and quantum computation deserves deeper exploration. More expressive quantum graph models, quantum-enhanced spectral methods, or hybrid classical-quantum explainers could expand the capabilities of quantum machine learning while offering new insights into quantum devices themselves. As quantum hardware evolves, graph-based methods may play a pivotal role in quantum error mitigation, circuit optimization, and quantum algorithm design.

More broadly, graph learning as a paradigm is expected to continue evolving toward greater scalability, robustness, and alignment with real-world decision-making processes. Future research will likely focus on unifying learning, reasoning, and explanation within a single framework, enabling graph models that are not only predictive but also transparent and reliable under distribution shifts.

Fifth, the empirical study in Chapter 5 points toward a broader human-centered research agenda. Future studies could incorporate more diverse populations, cross-national comparisons, or domain-specific transparency requirements (e.g., healthcare, finance, chemistry). There is also strong potential for developing psychometric instruments that measure trust, interpretability, and user mental models in graph-based AI systems. These tools would support evidence-based governance and policy design.

Lastly, the rapid pace of AI development demands integrative frameworks that span technical, ethical, and regulatory dimensions. Building bridges between explainability research, AI safety, and governance could lead to standardized transparency benchmarks, certification mechanisms, and formal audit procedures. Such interdisciplinary work represents an essential pathway toward ensuring that advanced AI systems remain beneficial and aligned with human values.

Closing Remarks

Together, the contributions of this thesis highlight both the promise and the complexity of modern AI systems. From the foundations of graph learning to the challenges of explainability, from emerging quantum technologies to the social expectations surrounding AI transparency, the work underscores the multifaceted nature of responsible AI development. As AI continues to evolve at an unprecedented pace, its future will depend not only on technical innovation but also on our collective ability to guide these technologies with clarity, accountability, and human-centered values. It is my hope that the ideas presented in this thesis—particularly those emphasizing structural faithfulness and transparent decision-making—will contribute to this ongoing effort and inspire further research at the intersection of theory, practice, and ethics.

Acknowledgement

First, I would like to express my deepest gratitude to my supervisor, Liang Zhao. I still clearly remember that five years ago, during the lockdown period in my hometown due to COVID-19, I first came into contact with him and started learning together online. Throughout my five years at Kyoto University, he has provided invaluable guidance on the direction of my research and has dedicated tremendous effort, time, and patience in advising my thesis and dissertation work. Beyond academics, he has also taken thoughtful care of my daily life. I am a rather careless person, and his high standards and rigorous discipline have always inspired me to push toward the limits of my ability and continuously improve myself. His philosophical approach to research has profoundly influenced me and gradually shaped my own way of thinking—to aim high instead of getting trapped in details.

I would also like to express my sincere appreciation to the other wonderful and supportive professors I have met at GSAIS, Kyoto University. My vice supervisor, Kei Saito, has broadened my research horizon and encouraged me to think beyond the boundaries of my major field. Other professors, Yosuke Yamashiki, Takashi Sekiyama, Dimiter Savov Ialnazov, Yuichi Ikeda, Hiroaki Nagayama, Takashi Kanamura, etc., have also taught and helped me a lot. I am grateful as well to Ms. Yonehara from the office, who has provided generous help and support across many projects.

I am thankful to my laboratory colleagues for their collaboration and companionship: Wenlong Li and Wenruo Lü for sharing their valuable experience, as well as Rimsa Goperma, Zhenyu Zuo, and Rojan Basnet for the time we worked together and the moments we shared.

Moreover, I would like to express my sincere appreciation to the JST SPRING Program for their recognition and support. Their financial assistance relieved me from economic pressure and enabled me to focus fully on my research without distraction.

During my internship at Tencent, I am grateful to my mentor Xiong Xu for his guidance in technology, professional experience, and mindset. My sincere thanks also go to Sainan Huai for providing ideas and project support, and to my colleagues Shengming Ma, Tianyu Zhang, Yixin Feng, Kai Ye, Yue Chen, Zhao Huang, and Jianming Wang for their collaboration and encouragement. At Araya Inc., I am deeply appreciative of Mr. Kaoru Nozue for his recognition and guidance.

I am also grateful to my friends in Japan—Wenliang Yin, Jincheng Wang, Shiya Zhao, and Xiaonijun Chen—for their mutual care and companionship, and to my friends from my hometown—Jigao Xu, Jiahuan He, Yuxuan Qin, Guanyu Pei, and Jiangning Liu—whose long-distance support has always kept me mentally strong and emotionally grounded.

My heartfelt thanks go to my parents and family, who have provided unconditional love and support, allowing me to pursue my studies abroad without worry and always feel the warmth of home.

Finally, my deepest gratitude goes to my fiancée, my closest friend and soul mate, Jiayi Zhu. With every moment we share, my love for you grows stronger. Thank you for your unwavering support, understanding, and companionship over all the years.

In addition, thank the contribution of generative models, e.g., ChatGPT, DeepSeek in polishing this thesis.

Bibliography

- [1] The Open University. *What is learning*. Last modified: Friday, 20 September 2024, 9:42 AM. Accessed: 2025-11-22. OpenLearn Create. 2024. URL: <https://www.open.edu/openlearncreate/mod/page/view.php?id=146084>.
- [2] 趙 亮. “情報乱雑さで生きを考える——機械は賢くなれるか”. 日本語. In: 実践する総合生存学. Ed. by 池田 裕一. 京都大学学術出版会, 2021. ISBN: 9784814002962.
- [3] M. K. Smith. *Learning theory*. The encyclopedia of pedagogy and informal education. 1999–2020. URL: <https://infed.org/dir/learning-theory-models-product-and-process/>.
- [4] Wikipedia contributors. *Robert Gibbon Johnson*. Accessed: 2025-11-22. 2025. URL: https://en.wikipedia.org/wiki/Robert_Gibbon_Johnson.
- [5] Wikipedia contributors. *Pietro Andrea Mattioli*. Accessed: 2025-11-22. Wikimedia Foundation. 2025. URL: https://en.wikipedia.org/wiki/Pietro_Andrea_Mattioli#cite_note-2.
- [6] Global Alliance for Improved Nutrition (GAIN). *Tomatoes –the world’s most popular vegetable*. English. Accessed: 2025-11-22. Global Alliance for Improved Nutrition (GAIN). 2020. URL: <https://www.gainhealth.org/resources/reports-and-publications/tomatoes-worlds-most-popular-vegetable>.
- [7] Geoffrey West. *Scale: The Universal Laws of Growth, Innovation, Sustainability, and the Pace of Life in Organisms, Cities, Economies, and Companies*. Penguin Group, 2017. ISBN: 1594205582.
- [8] Sebastian Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016).
- [9] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.
- [10] Wikipedia contributors. *Seven Bridges of Königsberg*. Accessed: 2025-11-22. Wikimedia Foundation. 2025. URL: https://en.wikipedia.org/wiki/Seven_Bridges_of_K%C3%B6nigsberg.
- [11] Leonhard Euler. “Solutio problematis ad geometriam situs pertinentis”. In: *Commentarii academiae scientiarum Petropolitanae* (1741), pp. 128–140.
- [12] Richard M Karp. “Reducibility among combinatorial problems”. In: *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*. Springer, 2009, pp. 219–241.
- [13] Michael Held and Richard M Karp. “The traveling-salesman problem and minimum spanning trees”. In: *Operations research* 18.6 (1970), pp. 1138–1162.
- [14] Francesco Carrabs, Raffaele Cerulli, and Maria Grazia Speranza. “A branch-and-bound algorithm for the double travelling salesman problem with two stacks”. In: *Networks* 61.1 (2013), pp. 58–75.
- [15] Cor AJ Hurkens and Gerhard J Woeginger. “On the nearest neighbor rule for the traveling salesman problem”. In: *Operations Research Letters* 32.1 (2004), pp. 1–4.

- [16] Gözde Kizilateş and Fidan Nuriyeva. “On the nearest neighbor algorithms for the traveling salesman problem”. In: *Advances in Computational Science, Engineering and Information Technology: Proceedings of the Third International Conference on Computational Science, Engineering and Information Technology (CCSEIT-2013), KTO Karatay University, June 7-9, 2013, Konya, Turkey-Volume 1*. Springer. 2013, pp. 111–118.
- [17] Li-Ying Wang, Jie Zhang, and Hua Li. “An improved genetic algorithm for tsp”. In: *2007 International Conference on Machine Learning and Cybernetics*. Vol. 2. IEEE. 2007, pp. 925–928.
- [18] Noraini Mohd Razali, John Geraghty, et al. “Genetic algorithm performance with different selection strategies in solving TSP”. In: *Proceedings of the world congress on engineering*. Vol. 2. 1. International Association of Engineers Hong Kong, China. 2011, pp. 1–6.
- [19] Yong Deng, Yang Liu, and Deyun Zhou. “An improved genetic algorithm with initial population strategy for symmetric TSP”. In: *Mathematical problems in engineering* 2015.1 (2015), p. 212794.
- [20] Anna R Karlin, Nathan Klein, and Shayan Oveis Gharan. “A (slightly) improved approximation algorithm for metric TSP”. In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. 2021, pp. 32–45.
- [21] Roberto Battiti and Alan A. Bertossi. “Greedy, prohibition, and reactive heuristics for graph partitioning”. In: *IEEE transactions on computers* 48.4 (2002), pp. 361–385.
- [22] Sanjeev Arora, Satish Rao, and Umesh Vazirani. “Geometry, flows, and graph-partitioning algorithms”. In: *Communications of the ACM* 51.10 (2008), pp. 96–105.
- [23] Justin Gilmer et al. “Neural message passing for quantum chemistry”. In: *International conference on machine learning*. PMLR. 2017, pp. 1263–1272.
- [24] Amazon Freight. *Wide network, wider smiles: How Amazon Freight ships with care*. [Online; accessed 25-November-2024]. Nov. 25, 2024. URL: <https://freight.amazon.com/news/wide-network-wider-smiles-how-amazon-freight-ships-with-care>.
- [25] Greg Linden, Brent Smith, and Jeremy York. “Amazon. com recommendations: Item-to-item collaborative filtering”. In: *IEEE Internet computing* 7.1 (2003), pp. 76–80.
- [26] Amil Merchant et al. “Scaling deep learning for materials discovery”. In: *Nature* 624.7990 (2023), pp. 80–85.
- [27] Babak Sokouti. “A review on in silico virtual screening methods in COVID-19 using anticancer drugs and other natural/chemical inhibitors”. In: *Exploration of Targeted Anti-tumor Therapy* 4.5 (2023), p. 994.
- [28] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. doi: 10.1137/S0097539795293172.
- [29] Yanghepu Li. “Connected-CF2: Learning the Explainability of Graph Neural Network on Counterfactual and Factual Reasoning via Connected Component”. In: *International Conference on AI-generated Content*. Springer. 2023, pp. 83–94.
- [30] Yanghepu Li and Liang Zhao. “CRoCExplainer: A Tool to Explain Graph Learning with Improved Connectedness Representation”. In: *2025 International Joint Conference on Neural Networks (IJCNN)*. 2025, pp. 1–8. doi: 10.1109/IJCNN64981.2025.11228942.
- [31] Yanghepu Li et al. “A parametric EDA method for coplanar waveguide channel recognition and air-bridge construction in quantum chip design”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43.10 (2024), pp. 3004–3014.

- [32] Andrea Landherr, Bettina Friedl, and Julia Heidemann. “A critical review of centrality measures in social networks”. In: *Business & Information Systems Engineering* 2.6 (2010), pp. 371–385.
- [33] Joan Bruna and Stéphane Mallat. “Invariant scattering convolution networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1872–1886.
- [34] Stéphane Mallat. “Group invariant scattering”. In: *Communications on Pure and Applied Mathematics* 65.10 (2012), pp. 1331–1398.
- [35] Mark Tygert et al. “A mathematical motivation for complex-valued convolutional networks”. In: *Neural computation* 28.5 (2016), pp. 815–825.
- [36] David Lazer et al. “Computational social science”. In: *Science* 323.5915 (2009), pp. 721–723.
- [37] Eric H. Davidson et al. “A Genomic Regulatory Network for Development”. In: *Science* 295.5560 (2002), pp. 1669–1678.
- [38] Michael M Bronstein et al. “Geometric deep learning: going beyond euclidean data”. In: *IEEE Signal Processing Magazine* 34.4 (2017), pp. 18–42.
- [39] Weihua Hu et al. “Open Graph Benchmark: Datasets for Machine Learning on Graphs”. In: *arXiv preprint arXiv:2005.00687* (2020).
- [40] Weihua Hu et al. “OGB-LSC: A Large-Scale Challenge for Machine Learning on Graphs”. In: *arXiv preprint arXiv:2103.09430* (2021).
- [41] Lorenz C Blum and Jean-Louis Reymond. “970 million druglike small molecules for virtual screening in the chemical universe database GDB-13”. In: *Journal of the American Chemical Society* 131.25 (2009), pp. 8732–8733.
- [42] Christoph Schuhmann et al. “Laion-5b: An open large-scale dataset for training next generation image-text models”. In: *Advances in neural information processing systems* 35 (2022), pp. 25278–25294.
- [43] Maurice Weber et al. “RedPajama: an Open Dataset for Training Large Language Models”. In: *NeurIPS Datasets and Benchmarks Track* (2024).
- [44] Jure Leskovec. *CS224W: Machine Learning with Graphs —Lecture 01 “Introduction” (Slide 45)*. <https://web.stanford.edu/class/cs224w/slides/01-intro.pdf>. Slide 45, accessed 24 Nov 2025. 2025.
- [45] Peter V Cornish and Taekjip Ha. “A survey of single-molecule techniques in chemical biology”. In: *ACS chemical biology* 2.1 (2007), pp. 53–61.
- [46] Pierre Hohenberg and Walter Kohn. “Inhomogeneous electron gas”. In: *Physical review* 136.3B (1964), B864.
- [47] Robert G Parr. “Density functional theory of atoms and molecules”. In: *Horizons of quantum chemistry*. Springer, 1980, pp. 5–15.
- [48] Jianshen Zhu et al. “A method for molecular design based on linear regression and integer programming”. In: *Proceedings of the 2022 12th international conference on bioscience, biochemistry and bioinformatics*. 2022, pp. 21–28.
- [49] Xiaowen Dong et al. “Learning Laplacian matrix in smooth graph signal representations”. In: *IEEE Transactions on Signal Processing* 64.23 (2016), pp. 6160–6173.
- [50] Vassilis Kalofolias. “How to learn a graph from smooth signals”. In: *Artificial intelligence and statistics*. PMLR. 2016, pp. 920–929.
- [51] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. “Deepwalk: Online learning of social representations”. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2014, pp. 701–710.

- [52] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.
- [53] William L Hamilton. “Graph representation learning”. In: *Synthesis Lectures on Artificial Intelligence and Machine Learning* 14.3 (2020), pp. 1–159.
- [54] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).
- [55] William L Hamilton, Rex Ying, and Jure Leskovec. “Inductive representation learning on large graphs”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 2017, pp. 1025–1035.
- [56] Yue Kong et al. “Integrating concept of pharmacophore with graph neural networks for chemical property prediction and interpretation”. In: *Journal of Cheminformatics* 14.1 (2022), p. 52.
- [57] Ling Zhao et al. “T-GCN: A temporal graph convolutional network for traffic prediction”. In: *IEEE transactions on intelligent transportation systems* 21.9 (2019), pp. 3848–3858.
- [58] Wenqi Fan et al. “Graph neural networks for social recommendation”. In: *The world wide web conference*. 2019, pp. 417–426.
- [59] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. “A survey on oversmoothing in graph neural networks”. In: *arXiv preprint arXiv:2303.10993* (2023).
- [60] Keyulu Xu et al. “How Powerful are Graph Neural Networks?” In: *International Conference on Learning Representations*. 2019. URL: <https://openreview.net/forum?id=ryGs6iA5Km>.
- [61] Jiawei Zhang et al. “Graph-bert: Only attention is needed for learning graph representations”. In: *arXiv preprint arXiv:2001.05140* (2020).
- [62] Chengxuan Ying et al. “Do transformers really perform badly for graph representation?” In: *Advances in neural information processing systems* 34 (2021), pp. 28877–28888.
- [63] Raghunathan Ramakrishnan et al. “Quantum chemistry structures and properties of 134 kilo molecules”. In: *Scientific data* 1.1 (2014), pp. 1–7.
- [64] Lars Ruddigkeit et al. “Enumeration of 166 billion organic small molecules in the chemical universe database GDB-17”. In: *Journal of chemical information and modeling* 52.11 (2012), pp. 2864–2875.
- [65] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: <http://arxiv.org/abs/1412.6980>.
- [66] Nambury S Raju et al. “Methodology review: Estimation of population validity and cross-validity, and the use of equal weights in prediction”. In: *Applied Psychological Measurement* 21.4 (1997), pp. 291–305.
- [67] Scott Mayer McKinney et al. “International evaluation of an AI system for breast cancer screening”. In: *Nature* 577.7788 (2020), pp. 89–94.
- [68] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *Nature* 596.7873 (2021), pp. 583–589.
- [69] Nobel Prize Outreach. *The Nobel Prize in Chemistry 2024: Summary*. Accessed: 2025-11-22. Nobel Prize Outreach. 2025. URL: <https://www.nobelprize.org/prizes/chemistry/2024/summary/>.
- [70] Praveena Penmetsa et al. “Effects of the autonomous vehicle crashes on public perception of the technology”. In: *IATSS research* 45.4 (2021), pp. 485–492.

- [71] Neville A Stanton et al. “Models and methods for collision analysis: A comparison study based on the Uber collision with a pedestrian”. In: *Safety Science* 120 (2019), pp. 117–128.
- [72] Shanshan He. “Who is liable for the UBER self-driving crash? Analysis of the liability allocation and the regulatory model for autonomous vehicles”. In: *Autonomous vehicles: Business, technology and law*. Springer, 2020, pp. 93–111.
- [73] Alejandro Barredo Arrieta et al. “Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI”. In: *Information fusion* 58 (2020), pp. 82–115.
- [74] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ““ Why should i trust you?” Explaining the predictions of any classifier”. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016, pp. 1135–1144.
- [75] Finale Doshi-Velez and Been Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608* (2017).
- [76] Amina Adadi and Mohammed Berrada. “Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)”. In: *IEEE access* 6 (2018), pp. 52138–52160.
- [77] Chengliang Yang, Anand Rangarajan, and Sanjay Ranka. “Global model interpretation via recursive partitioning”. In: *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*. IEEE. 2018, pp. 1563–1570.
- [78] Rich Caruana et al. “Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission”. In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 1721–1730.
- [79] Benjamin Letham et al. “Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model”. In: *The Annals of Applied Statistics* 9.3 (Sept. 2015), pp. 1350–1371.
- [80] Andrew Zupon et al. “Lightly-supervised representation learning with global interpretability”. In: *Proceedings of the third Workshop on Structured Prediction for NLP*. 2019, pp. 18–28.
- [81] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. “Anchors: High-precision model-agnostic explanations”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.
- [82] Jing Lei et al. “Distribution-free predictive inference for regression”. In: *Journal of the American Statistical Association* 113.523 (2018), pp. 1094–1111.
- [83] David Baehrens et al. “How to explain individual classification decisions”. In: *The Journal of Machine Learning Research* 11 (2010), pp. 1803–1831.
- [84] Zhitao Ying et al. “Gnnexplainer: Generating explanations for graph neural networks”. In: *Advances in Neural Information Processing Systems*. 2019, pp. 9240–9251.
- [85] Ana Lucic et al. “Cf-gnnexplainer: Counterfactual explanations for graph neural networks”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2022, pp. 4499–4511.
- [86] John H Holland. *Induction: Processes of inference, learning, and discovery*. MIT press, 1986.
- [87] Evan Heit. “Properties of inductive reasoning”. In: *Psychonomic bulletin & review* 7.4 (2000), pp. 569–592.
- [88] Amos Tversky and Daniel Kahneman. “Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty.” In: *Science* 185.4157 (1974), pp. 1124–1131.
- [89] Alison Gopnik et al. “A theory of causal learning in children: causal maps and Bayes nets.” In: *Psychological review* 111.1 (2004), p. 3.

- [90] Steven Sloman and Steven A Sloman. *Causal models: How people think about the world and its alternatives*. Oxford University Press, 2009.
- [91] Neal J Roese. “Counterfactual thinking.” In: *Psychological bulletin* 121.1 (1997), p. 133.
- [92] Philip N Johnson-Laird. “Deductive reasoning”. In: *Annual review of psychology* 50.1 (1999), pp. 109–135.
- [93] Keith J Holyoak and Paul Thagard. *Mental leaps: Analogy in creative thought*. MIT press, 1996.
- [94] Hao Yuan et al. “On explainability of graph neural networks via subgraph explorations”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12241–12252.
- [95] Juntao Tan et al. “Learning and evaluating graph neural network explanations based on counterfactual and factual reasoning”. In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 1018–1027.
- [96] Eyal Winter. “The shapley value”. In: *Handbook of game theory with economic applications* 3 (2002), pp. 2025–2054.
- [97] William N Anderson Jr and Thomas D Morley. “Eigenvalues of the Laplacian of a graph”. In: *Linear and multilinear algebra* 18.2 (1985), pp. 141–145.
- [98] Dragoš M Cvetković, Peter Rowlinson, and Slobodan Simic. *Eigenspaces of graphs*. 66. Cambridge University Press, 1997.
- [99] Türker Bıyıkoglu, Josef Leydold, and Peter F Stadler. “Laplacian eigenvectors of graphs”. In: *Lecture notes in mathematics* 1915 (2007).
- [100] Gilbert Strang. *Linear Algebra and Its Applications*. 2nd. Orlando, FL: Academic Press, Inc., 1980, p. 222.
- [101] Paul Erdos, Alfréd Rényi, et al. “On the evolution of random graphs”. In: *Publ. math. inst. hung. acad. sci* 5.1 (1960), pp. 17–60.
- [102] Edgar N Gilbert. “Random graphs”. In: *The Annals of Mathematical Statistics* 30.4 (1959), pp. 1141–1144.
- [103] Mikhail Belkin and Partha Niyogi. “Laplacian eigenmaps for dimensionality reduction and data representation”. In: *Neural computation* 15.6 (2003), pp. 1373–1396.
- [104] Juntao Tan et al. “Learning and Evaluating Graph Neural Network Explanations based on Counterfactual and Factual Reasoning”. In: *Proceedings of the ACM Web Conference 2022*. WWW ’22. Virtual Event, Lyon, France: Association for Computing Machinery, 2022, pp. 1018–1027. URL: <https://doi.org/10.1145/3485447.3511948>.
- [105] Asim Kumar Debnath et al. “Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. correlation with molecular orbital energies and hydrophobicity”. In: *Journal of medicinal chemistry* 34.2 (1991), pp. 786–797.
- [106] Dongsheng Luo et al. “Parameterized explainer for graph neural network”. In: *Advances in neural information processing systems* 33 (2020), pp. 19620–19631.
- [107] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. ICLR ’17. Palais des Congrès Neptune, Toulon, France, 2017. URL: <https://openreview.net/forum?id=SJU4ayYgl>.
- [108] Alessandro Sebastianelli et al. “On circuit-based hybrid quantum neural networks for remote sensing imagery classification”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15 (2021), pp. 565–580.

- [109] Amira Abbas et al. “The power of quantum neural networks”. In: *Nature Computational Science* 1.6 (2021), pp. 403–409.
- [110] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. “A rigorous and robust quantum speed-up in supervised machine learning”. In: *Nature Physics* 17.9 (2021), pp. 1013–1017.
- [111] Matthias C Caro et al. “Generalization in quantum machine learning from few training data”. In: *Nature communications* 13.1 (2022), p. 4919.
- [112] Hsin-Yuan Huang et al. “Quantum advantage in learning from experiments”. In: *Science* 376.6598 (2022), pp. 1182–1186.
- [113] Francesco Tacchino et al. “An artificial neuron implemented on an actual quantum processor”. In: *npj Quantum Information* 5.1 (2019), pp. 1–8.
- [114] Jinkai Tian et al. “Recent advances for quantum neural networks in generative learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.10 (2023), pp. 12321–12340.
- [115] John Preskill. “Quantum computing in the NISQ era and beyond”. In: *Quantum* 2 (2018), p. 79.
- [116] Kosuke Mitarai et al. “Quantum circuit learning”. In: *Physical Review A* 98.3 (2018), p. 032309.
- [117] Francesco Tacchino et al. “Variational learning for quantum artificial neural networks”. In: *IEEE Transactions on Quantum Engineering* 2 (2021), pp. 1–10.
- [118] Stefano Mangini et al. “Quantum neural network autoencoder and classifier applied to an industrial case study”. In: *Quantum Machine Intelligence* 4.2 (2022), p. 13.
- [119] Zoë Holmes et al. “Nonlinear transformations in quantum computation”. In: *Physical Review Research* 5.1 (2023), p. 013105.
- [120] Marco Cerezo et al. “Cost function dependent barren plateaus in shallow parametrized quantum circuits”. In: *Nature communications* 12.1 (2021), p. 1791.
- [121] Guillaume Verdon et al. “Quantum graph neural networks”. In: *arXiv preprint arXiv:1909.12264* (2019).
- [122] H Jeff Kimble. “The quantum internet”. In: *Nature* 453.7198 (2008), pp. 1023–1030.
- [123] Kevin Qian et al. “Heisenberg-scaling measurement protocol for analytic functions with quantum sensor networks”. In: *Physical Review A* 100.4 (2019), p. 042304.
- [124] Nouhaila Innan et al. “Financial fraud detection using quantum graph neural networks”. In: *Quantum Machine Intelligence* 6.1 (2024), p. 7.
- [125] Hale F Trotter. “On the product of semi-groups of operators”. In: *Proceedings of the American Mathematical Society* 10.4 (1959), pp. 545–551.
- [126] Masuo Suzuki. “Generalized Trotter’s formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems”. In: *Communications in Mathematical Physics* 51.2 (1976), pp. 183–190.
- [127] Péter Mernyei, Konstantinos Meichanetzidis, and Ismail Ilkan Ceylan. “Equivariant quantum graph circuits”. In: *International conference on machine learning*. PMLR. 2022, pp. 15401–15420.
- [128] Kishor Bharti et al. “Noisy intermediate-scale quantum algorithms”. In: *Reviews of Modern Physics* 94.1 (2022), p. 015004.
- [129] Thaddeus D Ladd et al. “Quantum computers”. In: *Nature* 464.7285 (2010), pp. 45–53.
- [130] Earl T Campbell, Barbara M Terhal, and Christophe Vuillot. “Roads towards fault-tolerant universal quantum computation”. In: *Nature* 549.7671 (2017), pp. 172–179.

- [131] Abhinav Kandala et al. “Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets”. In: *Nature* 549.7671 (2017), pp. 242–246.
- [132] Jarrod R McClean et al. “Barren plateaus in quantum neural network training landscapes”. In: *Nature communications* 9.1 (2018), p. 4812.
- [133] Morten Kjaergaard et al. “Superconducting qubits: Current state of play”. In: *Annual Review of Condensed Matter Physics* 11.1 (2020), pp. 369–395.
- [134] Sebastian Krinner et al. “Engineering cryogenic setups for 100-qubit scale superconducting circuit systems”. In: *EPJ Quantum Technology* 6.1 (2019), p. 2.
- [135] Prakash Murali et al. “Formal constraint-based compilation for noisy intermediate-scale quantum systems”. In: *Microprocessors and Microsystems* 66 (2019), pp. 102–112.
- [136] Morten Kjaergaard et al. “Superconducting Qubits: Current State of Play”. In: *Annual Review of Condensed Matter Physics* 11.1 (2020), pp. 369–395. DOI: 10.1146/annurev-conmatphys-031119-050605.
- [137] David M Pozar. *Microwave engineering*. John Wiley & sons, 2011.
- [138] George E Ponchak, John Papapolymerou, and Manos M Tentzeris. “Excitation of coupled slot-line mode in finite-ground CPW with unequal ground-plane widths”. In: *IEEE transactions on microwave theory and techniques* 53.2 (2005), pp. 713–717.
- [139] AA Houck et al. “Controlling the spontaneous emission of a superconducting transmon qubit”. In: *Physical review letters* 101.8 (2008), p. 080502.
- [140] Chung-Yi Lee, Yaozhong Liu, and Tatsuo Itoh. “The effects of the coupled slotline mode and air-bridges on CPW and NLC waveguide discontinuities”. In: *IEEE Transactions on microwave theory and techniques* 43.12 (1995), pp. 2759–2765.
- [141] Jongjoo Lee et al. “Suppression of coupled-slotline mode on CPW using air-bridges measured by picosecond photoconductive sampling”. In: *IEEE microwave and guided wave letters* 9.7 (1999), pp. 265–267.
- [142] E. Girgis, J. Liu, and M. L. Benkheadar. “Fabrication of metallic air bridges using multiple-dose electron beam lithography”. In: *Applied Physics Letters* 88.20 (May 2006), p. 202103. URL: <https://doi.org/10.1063/1.2204833>.
- [143] Zijun Chen et al. “Fabrication and characterization of aluminum airbridges for superconducting microwave circuits”. In: *Applied Physics Letters* 104.5 (2014), p. 052602.
- [144] A Dunsworth et al. “A method for building low loss multi-layer wiring for superconducting microwave devices”. In: *Applied Physics Letters* 112.6 (2018), p. 063502.
- [145] Yuting Sun et al. “Fabrication of airbridges with gradient exposure”. In: *Applied Physics Letters* 121.7 (Aug. 2022), p. 074001. URL: <https://doi.org/10.1063/5.0102555>.
- [146] Kunliang Bu et al. “Tantalum airbridges for scalable superconducting quantum processors”. In: *npj Quantum Information* 11.1 (2025), p. 17.
- [147] D Rosenberg et al. “3D integrated superconducting qubits”. In: *npj quantum information* 3.1 (2017), pp. 1–5.
- [148] Frank Arute et al. “Quantum supremacy using a programmable superconducting processor”. In: *Nature* 574.7779 (2019), pp. 505–510.
- [149] Yulin Wu et al. “Strong quantum computational advantage using a superconducting quantum processor”. In: *Physical review letters* 127.18 (2021), p. 180501.
- [150] Pengfei Zhang et al. “Many-body Hilbert space scarring on a superconducting processor”. In: *arXiv preprint arXiv:2201.03438v2* (2022).

- [151] Donna-Ruth W Yost et al. “Solid-state qubits integrated with superconducting through-silicon vias”. In: *npj Quantum Information* 6.1 (2020), pp. 1–7.
- [152] Justin L Mallek et al. “Fabrication of superconducting through-silicon vias”. In: *arXiv preprint arXiv:2103.08536* (2021).
- [153] Zlatko K Minev et al. *Qiskit Metal: An Open-Source Framework for Quantum Device Design & Analysis*. 2021. URL: <https://doi.org/10.5281/zenodo.4618153>.
- [154] Tal Dayan. *Rubber-band based topological router*. University of California, Santa Cruz, 1997.
- [155] Peter Hart, Nils Nilsson, and Bertram Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. URL: <https://doi.org/10.1109/tssc.1968.300136>.
- [156] Johannes Heinsoo et al. *KQCircuits*. June 2021. DOI: 10.5281/zenodo.4944796. URL: <https://github.com/iqm-finland/KQCircuits>.
- [157] Julian Kelly et al. “State preservation by repetitive error detection in a superconducting quantum circuit”. In: *Nature* 519.7541 (2015), pp. 66–69.
- [158] Geoffrey Hinton. “Godfather of Artificial Intelligence” *Geoffrey Hinton on the promise, risks of advanced AI*. Updated on: June 16 2024. Accessed: 2025-11-22. CBS News. 2023. URL: <https://www.cbsnews.com/news/geoffrey-hinton-ai-dangers-60-minutes-transcript/>.
- [159] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.
- [160] Josh Achiam et al. “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (2023).
- [161] Aditya Ramesh et al. “Hierarchical text-conditional image generation with clip latents”. In: *arXiv preprint arXiv:2204.06125* 1.2 (2022), p. 3.
- [162] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.
- [163] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).
- [164] Anna Jobin, Marcello Ienca, and Effy Vayena. “The global landscape of AI ethics guidelines”. In: *Nature machine intelligence* 1.9 (2019), pp. 389–399.
- [165] Kush R Varshney. “Trustworthy machine learning and artificial intelligence”. In: *XRDS: Crossroads, The ACM Magazine for Students* 25.3 (2019), pp. 26–29.
- [166] Anirban Chakraborty et al. “Adversarial attacks and defences: A survey”. In: *arXiv preprint arXiv:1810.00069* (2018).
- [167] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures”. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1322–1333.
- [168] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. “Deep models under the GAN: information leakage from collaborative deep learning”. In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017, pp. 603–618.
- [169] Nicolas Papernot et al. “Practical black-box attacks against machine learning”. In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 2017, pp. 506–519.
- [170] Florian Tramèr et al. “Stealing machine learning models via prediction {APIs}”. In: *25th USENIX security symposium (USENIX Security 16)*. 2016, pp. 601–618.

- [171] Mingfu Xue et al. “Machine learning security: Threats, countermeasures, and evaluations”. In: *IEEE Access* 8 (2020), pp. 74720–74742.
- [172] Marco Barreno et al. “The security of machine learning”. In: *Machine learning* 81 (2010), pp. 121–148.
- [173] Nikolaos Pitropakis et al. “A taxonomy and survey of attacks against machine learning”. In: *Computer Science Review* 34 (2019), p. 100199.
- [174] Reza Shokri et al. “Membership inference attacks against machine learning models”. In: *2017 IEEE symposium on security and privacy (SP)*. IEEE. 2017, pp. 3–18.
- [175] Dario Amodei et al. “Concrete problems in AI safety”. In: *arXiv preprint arXiv:1606.06565* (2016).
- [176] Latanya Sweeney. “Achieving k-anonymity privacy protection using generalization and suppression”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10.05 (2002), pp. 571–588.
- [177] Cynthia Dwork and Aaron Roth. “The Algorithmic Foundations of Differential Privacy”. In: *Foundations and Trends in Theoretical Computer Science* 9.3-4 (Aug. 2014), pp. 211–487. URL: <https://doi.org/10.1561/04000000042>.
- [178] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).
- [179] Thilo Hagendorff. “The ethics of AI ethics: An evaluation of guidelines”. In: *Minds and machines* 30.1 (2020), pp. 99–120.
- [180] Julia Angwin et al. “Machine bias”. In: *Ethics of data and analytics*. Auerbach Publications, 2022, pp. 254–264.
- [181] Jeffrey Dastin. “Amazon scraps secret AI recruiting tool that showed bias against women”. In: *Ethics of data and analytics*. Auerbach Publications, 2022, pp. 296–299.
- [182] Beena Ammanath. *Trustworthy AI: a business guide for navigating trust and ethics in AI*. John Wiley & Sons, 2022.
- [183] Finale Doshi-Velez et al. “Accountability of AI under the law: The role of explanation”. In: *arXiv preprint arXiv:1711.01134* (2017).
- [184] Buomsoo Kim, Jinsoo Park, and Jihae Suh. “Transparency and accountability in AI decision support: Explaining and visualizing convolutional neural networks for text information”. In: *Decision Support Systems* 134 (2020), p. 113302.
- [185] Beatriz San Miguel, Aisha Naseer, and Hiroya Inakoshi. “Putting accountability of AI systems into practice”. In: *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 2021, pp. 5276–5278.
- [186] David Patterson et al. “Carbon emissions and large neural network training”. In: *arXiv preprint arXiv:2104.10350* (2021).
- [187] Roy Schwartz et al. “Green ai”. In: *Communications of the ACM* 63.12 (2020), pp. 54–63.
- [188] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.
- [189] John A McDermid et al. “Artificial intelligence explainability: the technical and ethical dimensions”. In: *Philosophical Transactions of the Royal Society A* 379.2207 (2021), p. 20200363.
- [190] Shana Lynch. *2023 State of AI in 14 Charts*. Stanford Institute for Human-Centered Artificial Intelligence (HAI). 2023. URL: <https://hai.stanford.edu/news/2023-state-ai-14-charts> (visited on 10/06/2023).

- [191] US Congress. *National Artificial Intelligence Initiative Act of 2020*. Public Law. Division E of Public Law 116-283, current as of April 15, 2023. 2020.
- [192] European Union. *General Data Protection Regulation (GDPR)*. Official Journal of the European Union. Regulation (EU) 2016/679. 2018.
- [193] European Commission. *Proposal for a Regulation laying down harmonised rules on artificial intelligence (Artificial Intelligence Act)*. European Commission Proposal. 2021. URL: <https://artificialintelligenceact.eu/>.
- [194] Chandler Morse. “What the GDPR can teach us about AI regulation”. In: *World Economic Forum* (2023). Vice President Corporate Affairs, Workday. URL: <https://www.weforum.org/agenda/2023/06/gdpr-artificial-intelligence-regulation-europe-us/>.
- [195] Shiona McCallum. “Revenge and deepfake porn laws to be toughened”. In: *BBC News* (2023). Online; accessed June 10, 2023. URL: <https://www.bbc.com/news/technology-66021643> (visited on 06/10/2023).
- [196] Giulia Interesse. “China to Regulate Deep Synthesis (Deepfake) Technology from 2023”. In: *China Briefing* (2023). Online; accessed June 10, 2023. URL: <https://www.china-briefing.com/news/china-to-regulate-deep-synthesis-deep-fake-technology-starting-january-2023/> (visited on 06/10/2023).
- [197] Max Langenkamp and Daniel N Yue. “How open source machine learning software shapes ai”. In: *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*. 2022, pp. 385–395.
- [198] Yaneer Bar-Yam. “General features of complex systems”. In: *Knowledge management, organizational intelligence and learning, and complexity 1* (2002), pp. 43–95.
- [199] D. Sculley et al. “Hidden technical debt in Machine learning systems”. In: *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*. NIPS’15. Montreal, Canada: MIT Press, 2015, pp. 2503–2511.
- [200] Tim Miller. “Explanation in artificial intelligence: Insights from the social sciences”. In: *Artificial intelligence* 267 (2019), pp. 1–38.
- [201] Eric Von Hippel. “Open source software projects as user innovation networks”. In: *Perspectives on free and open source software* (2005), pp. 267–278.
- [202] Josh Lerner and Jean Tirole. “Some simple economics of open source”. In: *The journal of industrial economics* 50.2 (2002), pp. 197–234.
- [203] Rensis Likert. “A technique for the measurement of attitudes”. In: *Archives of psychology* 140 (1932), pp. 5–55.
- [204] Lee J Cronbach. “Coefficient alpha and the internal structure of tests”. In: *psychometrika* 16.3 (1951), pp. 297–334.
- [205] Mohsen Tavakol and Reg Dennick. “Making sense of Cronbach’s alpha”. In: *International journal of medical education* 2 (2011), p. 53.
- [206] Robert F. DeVellis and Carolyn T. Thorpe. *Scale Development: Theory and Applications*. 4th ed. SAGE Publications, 2021.
- [207] Karl Pearson. “VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia”. In: *Philosophical Transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character* 187 (1896), pp. 253–318.
- [208] Lee A. Clark and David Watson. “Constructing Validity: Basic Issues in Objective Scale Development”. In: *Psychological Assessment* 7.3 (1995), pp. 309–319.

- [209] Ronald Aylmer Fisher. “Theory of statistical estimation”. In: *Mathematical proceedings of the Cambridge philosophical society*. Vol. 22. 5. Cambridge University Press. 1925, pp. 700–725.
- [210] Ellen R. Girden. *ANOVA: Repeated Measures*. Quantitative Applications in the Social Sciences 84. SAGE Publications, 1992.

Appendices

A Questionnaire on Transparency of AI

1. **Gender*** (Select one answer only)

- ☐ Male
- ☐ Female
- ☐ Prefer not to say

2. **Age Group*** (Select one answer only)

- ☐ Under 18
- ☐ 18-24
- ☐ 25-34
- ☐ 35-44
- ☐ 45-54
- ☐ 55+

3. **Highest Education Level*** (Select one answer only)

- ☐ High School or below
- ☐ College/Undergraduate in progress
- ☐ Bachelor's degree
- ☐ Master's degree
- ☐ Doctor's degree
- ☐ Other: _____

4. **Occupational Category*** (Select one answer only)

- ☐ Tech Industry Professional (incl. AI-related)
- ☐ Non-tech Professional (using AI tools)
- ☐ Non-tech Professional (not using AI tools)
- ☐ Academia
- ☐ Student
- ☐ Freelancer
- ☐ Retired
- ☐ Other: _____

5. **AI Tool Usage Frequency*** (Select one answer only)

- ☐ Multiple times daily
- ☐ Daily
- ☐ Several times weekly
- ☐ Several times monthly
- ☐ Rarely
- ☐ Never
- ☐ Other: _____

6. AI Application Exposure* (Select all that apply)

- ☐ Voice assistants (e.g., Siri)
- ☐ Recommendation Algorithms (e.g., TikTok/Amazon)
- ☐ Autonomous Driving
- ☐ AI-Generated Content (e.g., ChatGPT/Midjourney)
- ☐ Medical AI Diagnosis
- ☐ Financial Risk Management Systems
- ☐ Other: _____

7. Technical Proficiency Self-Assessment* (Select one answer only)

- ☐ No understanding
- ☐ Basic Conceptual Knowledge
- ☐ Can operate common AI tools
- ☐ Understand technical principles
- ☐ Engage in AI development

8. Open-source accelerates cross-domain innovation in generative AI (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

9. Models trained on public data should be mandatorily open-sourced (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

10. Closed-source better prevents AI misuse (e.g., deepfakes) (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

11. Technology monopolies hinder healthy AI development (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral

- ☐ Disagree
- ☐ Strongly Disagree

12. **Enterprises should establish ethics boards for open-source AI models** (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

13. **Key bottlenecks for AI open-sourcing?**

14. **AI Explainability enhancement significantly increases development costs**

(AI explainability generally refers to the output given by the AI model being explainable in human-understandable language or knowledge)

(Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

15. **XAI (Explainable AI) prioritization in healthcare surpasses recommender systems** (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

16. **Irreconcilable conflict exists between AI accuracy and AI explainability** (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

17. **User trust positively correlates with AI explanation depth** (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral

- ☐ Disagree
- ☐ Strongly Disagree

18. **Enhanced AI explainability facilitates long-term AI model optimization** (Select one answer only)

- ☐ Strongly Agree
- ☐ Agree
- ☐ Neutral
- ☐ Disagree
- ☐ Strongly Disagree

19. **What are the practical challenges for XAI implementation?**

20. **Medical AI Weight Allocation (total 100%)**

Please allocate weights for the following three aspects (total 100%)

Code openness + Diagnostic explainability + Prediction accuracy = 100

- Code openness: _____
- Diagnostic explainability: _____
- Prediction accuracy: _____

B Workshop on Explainable AI

English Version

Notice: The Japanese version follows after the English content.

Event Overview

Harmonizing Innovation and Threats - Explainable AI

Date: Saturday, June 7, 2025

Time: 13:00 - 17:00 (Venue opens at 12:15)

Venue: Room 113, 1F, Higashi Ichijo Building, Kyoto University

Format: Hybrid (In-person & Online via Zoom)

Language: Japanese

Capacity: 30 in-person, 70 online

Background

2024 witnessed AI's unprecedented recognition in global science, with AI researchers receiving Nobel Prizes in both Physics and Chemistry. Applications like ChatGPT, DeepSeek, and image-generation AI are rapidly transforming daily life and work processes. Yet significant concerns persist: complex architectures hinder human understanding, crucial applications in fields like drug development are impeded, data dependencies raise bias and intellectual property issues, and potential misuse poses societal risks.

This workshop explores AI's possibilities and challenges, focusing on explainable AI development, AI governance, and related policies through industry-government-academia collaboration.

Featured Speakers

- **Takuya Watanabe:** Director, IT Infrastructure Industry Office, METI
- **Masahito Iida:** Researcher, Generative AI Lab, Mitsubishi Research Institute
- **Taiyo Hamada:** Research Team Lead, ARAYA Inc.
- **Yusuke Hayashi:** Director, AI Alignment Network
- **Takashi Hashimoto:** Planning Officer, Financial Systems and Services Bureau, Bank of Japan
- **Liang Zhao:** Professor, Graduate School of Advanced Integrated Studies in Human Survivability, Kyoto University

Schedule Highlights

Session 1: Presentations (13:00-15:00) Chair: Yanghepu Li

Expert discussions on:

- METI's AI initiatives and Japan's strategy
- AI applications in financial services and challenges
- Geometric approaches to AI alignment

- Collective predictive coding theory for human-AI coordination
- Generative AI governance frameworks

Panel Discussion (15:15-16:15) Moderator: Taiyo Hamada

Debates on:

- Open-source AI: Balancing transparency vs. risk management
- Explainability paradigms in the generative AI era

Closing Session (16:15-17:00) Open discussion for in-person participants

Survey

We welcome feedback on explainable AI technologies and governance frameworks. All responses are anonymized.

Japanese Version

UTF8min

技術革新と脅威の調和—説明可能 AI を考える。

時間： 2025 年 6 月 7 日（土）

13:00 - 17:00（12:15 開場）

会場： 京都大学東一条館 1F 113 号室

言語： 日本語

定員： 会場 30 名、オンライン 70 名

趣旨

2024 年、人工知能（AI）は世界的な科学の舞台で前例のない評価を獲得した。ノーベル物理学賞も化学賞も AI の研究開発者たちに授与されたのである。さらに、ChatGPT や DeepSeek、画像生成 AI などのアプリケーションは日常生活に急速に浸透し、人間の経験や作業プロセスを再構築しようとしている。

一方、AI に関して重大な懸念もある。複雑で精巧なアーキテクチャは、その内部構造を人間が理解することを困難にし、製薬開発などの重要な分野での応用を妨げる。また、特定のデータに依存する AI 生成モデルの学習は、偏見や知的財産権に関する懸念を引き起こす可能性がある。さらに、誤った利用より社会が深刻な危険にさらされることもある。

このワークショップでは、AI がもたらす可能性と課題、ならびにそのために必要な説明可能 AI の開発、AI ガバナンス、AI に関連する政策を探求することを目的とし、産官学連携で情報共有を図る。

登壇者（順不同、敬称略）

- 渡辺琢也：経産省商務情報政策局情報処理基盤産業室長
- 飯田正仁：三菱総合研究所生成 AI ラボ研究員
紹介：東京大学工学部計数工学科数理コース卒業、京都大学大学院情報学研究科数理工学専攻修了。人工知能学会会員（汎用人工知能研究会専門委員：2025 年～）、日本オペレーションズ・リサーチ学会会員（研究普及委員：2010～2016 年）。2002 年入社後、AI・XR・量子など新しい技術の最新動向や社会に与える影響の調査研究に従事。現在は、生成 AI ラボにて、生成 AI のテクノロジー/トレンド/リスク/規制などを研究。共著に、『13 番目の人類「人間が AI を超えたら」』など。
- 濱田太陽：株式会社アラヤ Research Team Lead
紹介：2019 年沖縄科学技術大学院大学 (OIST) 科学技術研究科博士課程修了。2022 年より、Moonshot R&D プログラム (目標 9)「逆境の中でも前向きに生きられる社会の実現」(山田 PM グループ) の Principal Investigator として前向き状態に関するモデル化に従事。2024 年より現職。研究テーマは好奇心の神経計算メカニズムの解明や LLM によるデジタルツイン研究に従事。AI とニューロサイエンスが生み出す未来のために社会実装に日々明け暮れている。

- 林祐輔：AI Alignment Network 理事
紹介：データ活用支援・AI 研究開発に企業研究者として携わりながら、プライベートにおいても独立研究者として活動。
- 橋本崇：日本銀行金融機構局企画役
紹介：新日銀ネット構築プロジェクトの PMO、決済企画、FinTech センターを経て、2017 年 6 月から 2023 年 6 月まで国際標準化機構金融サービス専門委員会（ISO/TC 68）の国内委員会事務局にて事務局長を担当。現在は、金融機構局でサイバーセキュリティを担当。
- 趙亮：京都大学大学院教育支援機構大学院総合生存学館教授
紹介：中国清華大学応用数学学部卒、京都大学工学研究科修士・情報学研究科博士。宇都宮大学工学部情報工学科助教、京都大学情報学研究科講師、ドイツ・カールスルーエ工科大学（KIT）客員教授、京都大学総合生存学館准教授を経て現職。研究は、機械学習（特にグラフ学習）、AI 倫理、グラフ・ネットワークアルゴリズム、経済社会指標設計、国会・議会規模と議席配分など。情報の視点から生命や智慧、人間、社会を考える情報智慧論を提唱。IEEE、ACM、IEICE、ORSJ 会員。

スケジュール

司会：李楊和璞（京都大学大学院総合生存学館博士課程）

13:00 - 15:00 講演（含質疑応答時間）

- **13:00 - 13:20**：趙亮開会挨拶
趣旨説明や未来智慧研究会の研究紹介など
- **13:20 - 13:40**：渡辺琢也「AI に関わる経済産業省の取組」
- **13:40 - 14:00**：橋本崇「金融機関における AI の活用例と課題」
AI 技術は金融サービス業界に革新をもたらしているが、その活用にはさまざまな課題も伴う。プレゼンテーションでは、金融業界における AI の活用事例を紹介するとともに、主な課題とそれに対する取り組み内容について探求する。
- **14:00 - 14:20**：濱田太陽「表現幾何学による AI アライメントの可能性と課題」
人工知能（AI）の進展によりその影響力の高さから行動や判断を人間の意図や価値観に一致させる AI アライメントの実現が現代 AI 研究における重要な課題となっている。本講演では、AI が持つ内部表現の幾何学的な性質に着目し、この「表現幾何学」が AI アライメント問題の理解と解決にどのように寄与できるかを議論したい。まず AI アライメントの現状と、AI が学習した内部表現の幾何学的性質（潜在空間の構造、類似性の概念、概念間距離）について説明する。次に、表現幾何学を用いて AI の行動や判断を人間の価値観と照合する方法を議論する。最後に、表現幾何学的アプローチによって実現される社会について議論し、課題と今後の研究方向性を提示する。
- **14:20 - 14:40**：林祐輔「AI アライメントの新たな地平：集合的予測符号化（CPC）理論から展望する人間と AI の協調的未来」
人工知能（AI）の急速な社会実装は、誤情報・偽情報の拡散や情報の真偽評価といった課題を顕在化させている。本講演では「機械論的解釈可能性」および「AI アライメント」研究の最近の動向を概観するとともに、谷口忠大教授が提唱した「集合的予測符号化（CPC）理論」が提供する新たな理論的視座を提示する。CPC 理論は個々のエージェントが相互作用を通じて集合的な信念体系を形成するプロセス

を記述するもので、AI と人間が共存する社会システムへ拡張することで、AI が人間の認知モデルをいかに学習・予測し行動を調整しうるかを考察する。

- **14:40 - 15:00**：飯田正仁「生成 AI の光と影：5 つの潮流から読み解くガバナンスの未来」

生成 AI の急速な進化は社会やビジネスに多大な影響を与えている。本講演では生成 AI の技術革新とそれに伴うリスクを整理し、5 つの主要な潮流を通じて AI ガバナンスの現状と未来について考察する。特に日本が直面する課題と果たすべき役割に焦点を当て、国際的な協調の中での日本の立ち位置を探る。

15:00 - 15:15 休憩

15:15 - 16:15 パネルディスカッション

ファシリテーター：濱田太陽（株式会社アラヤ）

- **15:15 - 15:45**：生成 AI はオープンソース化すべきか？透明性とリスク管理の両立を考える

Wikipedia 等の公共財を活用している点から広く開放すべきとの主張と、ディープフェイク作成等の悪用リスクから慎重な運用を求める声の両論を多角的に検討し、技術革新と社会受容性のバランスを議論する。

- **15:45 - 16:15**：生成 AI 以後の説明可能性とどう向き合うべきか？ブラックボックス化のトレードオフ再考

高リスク領域での説明可能性（XAI）は精度や開発コストとのトレードオフが課題だったが、大規模生成 AI は AI 自らが挙動を説明する新たな可能性を開いた。本セッションでは生成 AI 時代における説明性要求のパラダイムシフトと実装戦略を再考する。

16:15 - 17:00 フリーディスカッション

会場の参加者による自由懇談が予定されています。オンラインのセッションはここで終了しますが、ぜひアンケート調査にご協力ください。

アンケート調査協力をお願い

ワークショップ参加後、説明可能 AI（XAI）の技術基盤や倫理の実装フレームワーク、産学官連携モデルなどに関するご意見をお聞かせください。回答データは完全に匿名化され、個人を特定する情報を収集しません。集計結果の分析レポートをご希望の方は、アンケート最終項目でメールアドレスを登録ください。ワークショップ終了後 90 日以内に研究成果を共有いたします。

京都大学大学院総合生存学館
未来智慧研究会

ワークショップ

技術革新と 脅威の調和

ー説明可能AIを考える

2024 年、人工知能 (AI) は世界的な科学の舞台で前例のない評価を獲得した。ノーベル物理学賞も化学賞も AI の研究開発者たちに授与されたのである。さらに、ChatGPT や DeepSeek、画像生成 AI などのアプリケーションは日常生活に急速に浸透し、人間の経験や作業プロセスを再構築しようとしている。

一方、AI に関して重大な懸念もある。複雑で精巧なアーキテクチャは、その内部構造を人間が理解することを困難にし、製薬開発などの重要な分野での応用を妨げる。また、特定のデータに依存する AI 生成モデルの学習は、偏見や知的財産権に関する懸念を引き起こす可能性がある。さらに、誤った利用より社会が深刻な危険にさらされることもある。

このワークショップでは、AI がもたらす可能性と課題、ならびにそのために必要な説明可能 AI の開発、AI ガバナンス、AI に関連する政策を探究することを目的とし、産官学連携で情報共有を図る。

登壇者(順不同)

渡辺 琢也(経産省)、飯田 正仁(三菱総研)、濱田 太陽(株式会社アラヤ)

林 祐輔(AI Alignment Network)、橋本 崇(日本銀行)、趙 亮(京都大学)

2025 年 6 月 7 日[土]

京都大学東一条館1F 112/113号室

第1部	講演&パネルディスカッション	第2部	フリーディスカッション	企画&講演司会	李楊和璞(京都大学)
	13:00 - 16:15		16:15 - 17:00	ファシリテーター	濱田太陽(株式会社アラヤ)
形式	会場	配信	(Zoom にてコメント受付)	言語	日本語
		参加費	無料	参加対象者	どなたでも歓迎します!

ウェブ <https://u.kyoto-u.jp/31h7o>

参加登録 <https://forms.gle/VdL5U22pbHACpwH29>

参加申込期限 2025 年 6 月 6 日[金] (申し込み多数の場合は早めに会場参加を締め切る可能性があります。)

主催 京都大学大学院総合生存学館未来智慧研究会

共催 京都大学大学院教育支援機構

fwi-contact@gsais.kyoto-u.ac.jp

ウェブ

参加登録

