Deep Reinforcement Learning with Directed Asymmetry and Kolmogorov-Arnold Networks for Dismantling Interdependent Multiplex Networks

A Dissertation submitted in partial fulfillment of the requirements
for the award of the degree of

Master of Technology

in

Computer Science

by

Soumyajit Dev

Roll No. CS2433

under the guidance of

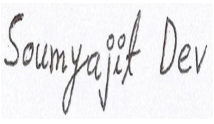
Dr. Malay Bhattacharyya

Machine Intelligence Unit
Indian Statistical Institute, Kolkata
Kolkata – 700108, India

June 9, 2026

DECLARATION

I, **Soumyajit Dev (CS2433)**, hereby declare that the work presented in this Master's Thesis report entitled **"Deep Reinforcement Learning with Directed Asymmetry and Kolmogorov-Arnold Networks for Dismantling Interdependent Multiplex Networks"** is the result of the work performed by me at the **Indian Statistical Institute, Kolkata**, under the supervision of **Dr. Malay Bhattacharyya**, and that it has not been submitted elsewhere for any degree or diploma. In line with general academic practice, due acknowledgments are made wherever the work is based on findings from other investigations.



Soumyajit Dev
M.Tech (CS) Candidate
Indian Statistical Institute, Kolkata

Date: June 9, 2026
Place: Kolkata – 700108

CERTIFICATE

This is to certify that the work contained in this project report entitled **"Deep Reinforcement Learning with Directed Asymmetry and Kolmogorov-Arnold Networks for Dismantling Interdependent Multiplex Networks"** submitted by **Soumyajit Dev**, towards the partial requirement for the degree of **Master of Technology** in **Computer Science**, has been carried out by him under my supervision at the **Indian Statistical Institute, Kolkata**, and that no part of it has been previously submitted for a degree, diploma, or any other qualification at this university or any other institution.

Dr. Malay Bhattacharyya
Associate Professor, Machine Intelligence Unit
Coordinator, Centre for Artificial Intelligence and Machine Learning
Associate Member, Technology Innovation Hub on Data Science,
Big Data Analytics, and Data Curation
Indian Statistical Institute, Kolkata

Place: Kolkata – 700108

Date: June 9, 2026

ACKNOWLEDGMENT

Research is rarely a solitary endeavour; it is the culmination of guidance, infrastructure, and unwavering personal support. As I bring this dissertation to a close, I find myself deeply indebted to the individuals and institution that made this journey possible.

First, I would like to express my gratitude to my thesis supervisor, **Dr. Malay Bhattacharyya**, Associate Professor at the Machine Intelligence Unit, ISI Kolkata. His profound academic vision, rigorous methodological standards, and immense patience have been the pivotal of this research. I am especially grateful for the intellectual freedom he afforded me to explore novel architectural paradigms like KANs, and for his meticulous feedback that consistently elevated the quality of my work. His mentorship has shaped not only this thesis but also my broader perspective as a computer scientist and more importantly, as a human being.

I extend my sincere thanks to the administration of the **Indian Statistical Institute, Kolkata** – for fostering a world-class academic environment that champions rigorous scientific inquiry.

Modern deep reinforcement learning research is fundamentally bound by computational capacity. The robust high-performance computing infrastructure and technical support provided by the **Centre for Artificial Intelligence and Machine Learning (CAIML)** at ISI Kolkata were instrumental in training and evaluating the deep learning agents presented in this work. Furthermore, I gratefully acknowledge the **ISI Library** for providing seamless and exhaustive access to the critical academic literature necessary to formalise this research.

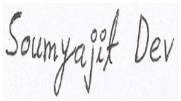
I also wish to thank my teachers, faculty members of the Computer Science discipline, and my fellow researchers and lab mates at the Machine Intelligence Unit. Our countless discussions, brainstorming sessions, and shared academic camaraderie provided invaluable insights and much-needed intellectual stimulation throughout my M.Tech tenure.

On a personal note, no academic achievement is possible without a foundation of emotional support. I owe an unquantifiable debt to my **parents**, whose lifelong sacrifices, unconditional belief in my potential, and emphasis on the value of education brought me to this premier institute. Their silent prayers and unwavering encouragement have been my greatest source of strength.

Finally, I dedicate my heartfelt gratitude to my **wife**. Research often demands long, unpredictable hours and consumes evenings and weekends. Her boundless patience, steadfast emotional support, and understanding during the most gruelling phases of coding, debugging, and manuscript writing were my sanctuary. She stood by me as an anchor through the triumphs and frustrations of this journey, and this milestone belongs as much to her as it does to me.

To everyone who knowingly or unknowingly contributed to this endeavour – thank you.

Place: Kolkata – 700108
Date: June 9, 2026



Soumyajit Dev
Roll No.: CS2433
M.Tech (CS), 2nd year
Indian Statistical Institute, Kolkata

Abstract

Identifying the minimum-cost node-removal sequence that fragments a complex network–the network dismantling problem–is NP-hard and central to infrastructure resilience. In interdependent multiplex networks, this difficulty is compounded by cascading cross-layer failures. While deep reinforcement learning (DRL) agents utilizing graph neural network (GNN) encoders achieve near-optimal dismantling, current state-of-the-art architectures suffer from two critical limitations. Topologically, existing agents strictly assume undirected edges, rendering them inapplicable to directed systems–such as supply chains or gene regulatory cascades–where failure propagation is fundamentally asymmetric. To resolve this, we propose Disassembling Directed Interdependent Networks (DDIN). DDIN introduces an asymmetric GraphSAGE encoder that explicitly separates incoming influence from outgoing control aggregations, paired with a multi-relational attention mechanism for cross-layer dependency fusion. Evaluated zero-shot on five real-world directed networks, DDIN achieves a 16–23% reduction in the Area Under the Dismantling Curve (AUDC) over heuristic baselines. Functionally, existing GNN encoders parameterise message-passing through multi-layer perceptrons (MLPs). These fixed-affine projections lack the capacity to model the non-smooth, high-frequency vulnerability patterns governing cascading fragility in scale-free topologies. We address this by proposing Kolmogorov-Arnold Reinforcement Learning (KARL), the first architecture to embed learnable univariate functions into an off-policy DRL combinatorial optimiser. KARL features a residual B-spline KAN encoder to stabilise gradient norms during deep message-passing, a fully KAN-parameterised cross-layer attention module (KANformer), and an orthogonal Chebyshev-KAN action-value head to mitigate boundary oscillations under non-stationary temporal-difference targets. Evaluated zero-shot on six real-world undirected multiplex networks, KARL yields a 21.96% average AUDC improvement over state-of-the-art baselines, alongside emergent structural interpretability. By independently resolving the directed topology limitation and the affine expressivity bottleneck, this dissertation establishes a robust architectural foundation for structurally faithful network dismantling models.

Keywords: Network Dismantling, Deep Reinforcement Learning, Graph Neural Networks, Multiplex Networks, Interdependent Networks, Cascading Failures, Kolmogorov-Arnold Networks, GraphSAGE, Directed Graphs, Strongly Connected Components, Geometric Multiplex Model, Prioritised Experience Replay, Zero-Shot Generalisation.

List of Publications

The following research articles have been accepted or communicated for publication based on the work presented in this dissertation:

Accepted / Published

1. Soumyajit Dev and Malay Bhattacharyya. "DDIN: Reinforcement Learning with Asymmetric GNNs for Dismantling Directed Interdependent Networks." *In Proceedings of the AAAI Conference on Artificial Intelligence, Student Abstract and Poster Program*, volume 40, pages 41188–41190, 2026.

Communicated / Under Review

1. Soumyajit Dev and Malay Bhattacharyya. "KARL: Kolmogorov-Arnold based Reinforcement Learning for Dismantling Multiplex Networks." *Submitted to the 40th Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2026.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Modern critical infrastructures and biological systems operate not as isolated entities but as tightly coupled, interdependent structures. A national power grid depends on supervisory control systems distributed across a communication network; financial clearing-houses are sustained by redundant data-centre clusters whose cooling plants depend on municipal water and gas supplies; gene-regulatory pathways within a living cell simultaneously influence and are influenced by metabolic, protein-interaction, and signalling networks. Under nominal operating conditions, these inter-layer dependencies remain latent; under adversarial or failure conditions, however, they act as conduits for systemic collapse. The 2003 Italian blackout [11], the cascading infrastructure disruptions during the COVID-19 pandemic, and the progressive collapse of financial systems following coordinated cyber-attacks all testify to the same fundamental mechanism: localised damage in one layer of an interdependent system can propagate, amplify, and ultimately disintegrate the entire coupled structure [60].

Understanding and controlling this vulnerability is the central mandate of the *network dismantling problem*. Informally, dismantling asks: *what is the smallest set of nodes whose removal fragments a network into non-extensive, disconnected pieces?* The problem is NP-hard even for a single layer [10] and its difficulty compounds in multilayer interdependent settings [44]. Despite this computational hardness, finding high-quality approximate solutions is of immense practical importance–from designing optimal vaccination campaigns to arrest epidemic outbreaks [43] and identifying synergistic drug targets in protein-interaction networks [4], to hardening critical infrastructure against cascading failure [25] and disrupting the structural backbone of illicit influence networks.

The past decade has witnessed remarkable progress. Physics-inspired heuristics such as High-Degree Adaptive (HDA) [12], Collective Influence (CI) [43], and MinSum [10] provide efficient but myopic approximations. Deep-learning-based approaches–FINDER [23], NIRM [66], and GDM [24]–demonstrated that reinforcement-learning (RL) agents trained on small synthetic graphs can generalise to large real-world instances far better than any static heuristic. Most recently, MultiDismantler [25] extended this paradigm to multilayer interdependent networks, pairing a multiplex graph neural network (GNN) encoder with an off-policy deep Q-network (DQN) and establishing the current state of the art on nine real-world multiplex benchmarks.

Despite this progress, *two fundamental limitations remain unaddressed* by every existing approach.

1. **Topological:** Existing deep-learning-based dismantling agents restrict their functional scope strictly to *undirected* networks, in which edges are symmetric and failure propagation is bidirectional. Yet a large and consequential class of real-world systems–supply chains, neural connectomes in model organisms, financial transaction flows, directed social-influence networks–are inherently *directed*: edges carry orientation,

and the failure of node A causing the failure of node B does not imply the reverse. Symmetric GNN message-passing is architecturally incapable of distinguishing the in-neighbourhood from the out-neighbourhood of a node and therefore cannot encode the directional asymmetry that governs fragility in such systems [2, 36].

2. **Functional** Every GNN-based dismantling policy–including MultiDismantler–parameterises its message-passing operations and action-value projections through multi-layer perceptrons (MLPs) with *fixed affine transformations*. Each MLP layer applies a single global linear map per input coordinate, confining the per-edge representational capacity to a fixed-degree polynomial function class. This is insufficient to represent the sharp, high-frequency, locally non-smooth per-node vulnerability functions that determine cascading fragility in scale-free multiplex topologies, where hub nodes with heterogeneous in/out-degree profiles exercise disproportionate structural control. Kolmogorov-Arnold Networks (KANs) [35] offer a principled remedy by placing learnable univariate B-spline functions directly on network edges, adapting the *functional form*–not merely the scalar coefficients–to each input coordinate.

This thesis addresses both limitations through two complementary frameworks that together expand the frontier of deep-RL-based network dismantling along the topological and functional axes. The first framework, **DDIN** (Disassembling Directed Interdependent Networks), introduces asymmetric GNNs and prioritised multi-step reinforcement learning for the dismantling of *directed* multiplex networks, directly remedying (L1). The second, **KARL** (Kolmogorov-Arnold Reinforcement Learning), replaces every MLP projection in a deep-RL dismantling agent with a learnable KAN, augmented by a macroscopic residual branch for gradient stability and an orthogonal Chebyshev polynomial decoder for well-conditioned action-value regression, directly remedying (L2).

Figure 1.1 situates these contributions within the landscape of prior work as a two-dimensional positioning matrix whose axes correspond to the two limitations.

1.1 The Fragility of Interdependent Networks

Complex systems rarely operate as isolated entities. Instead they constitute *systems of systems*: coupled, mutually reinforcing structures whose collective behaviour is qualitatively richer–and qualitatively more fragile–than any single constituent. Barabási, Gulbahce, and Loscalzo [4] established that human disease cannot be understood at the level of single genes or proteins; it is a network-level phenomenon. Watts and Strogatz [63] demonstrated that "small-world" topology, ubiquitous across biological and social systems, simultaneously enables rapid information propagation and, under adversarial conditions, rapid failure propagation. Buldyrev et al. [11] provided the seminal formal model showing that a network composed of interdependent layers is more vulnerable than any of its constituent layers considered in isolation, due to the potential for cascading failures that propagate across the entire coupled structure.

The following definitions establish the formal objects studied throughout the thesis.

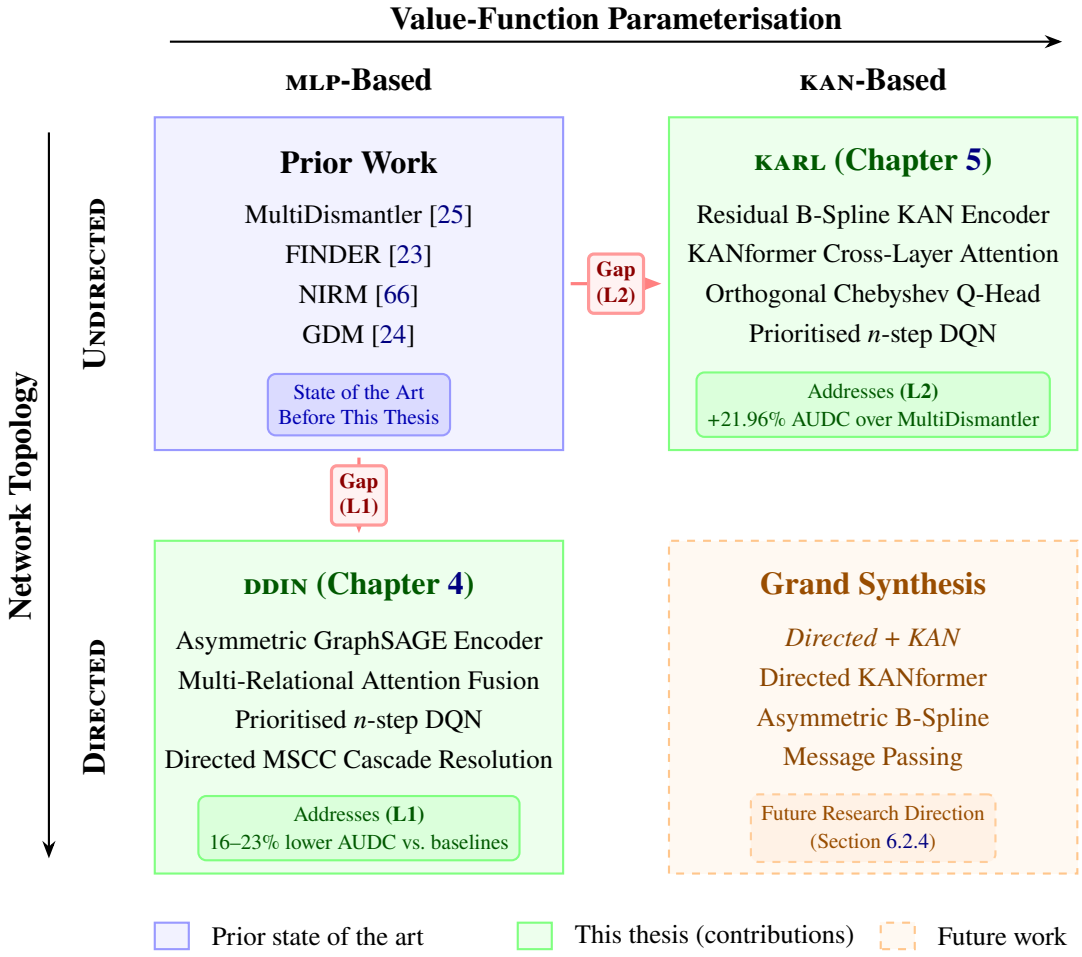


Figure 1.1: Two-dimensional research-positioning matrix situating DDIN and KARL relative to the existing state of the art. The horizontal axis distinguishes value-function parameterisation (MLP-based versus KAN-based); the vertical axis distinguishes network topology (undirected versus directed). All prior deep-RL-based dismantling agents occupy the top-left quadrant. DDIN (Chapter 4) addresses the *topological* gap (L1) by operating on directed multiplex networks with an asymmetric GNN backbone. KARL (Chapter 5) addresses the *functional* gap (L2) by replacing every affine MLP projection with a learnable Kolmogorov-Arnold function. The bottom-right quadrant—directed networks with KAN parameterisation—represents the natural synthesis and constitutes an open research direction (Section 6.2.4).

Definition 1.1.1 (Undirected Multiplex Network)**.** An *undirected multiplex network* is the triple $\mathcal{M} = \langle \mathcal{V}, \{E^{(\ell)}\}_{\ell=1}^{L}, \mathcal{L} \rangle$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is a shared vertex set of cardinality $N = |\mathcal{V}|$; $\mathcal{L} = \{1, \ldots, L\}$ is a set of L layers; and for each layer $\ell \in \mathcal{L}$, $E^{(\ell)} \subseteq \{\{u, v\} : u, v \in \mathcal{V},\ u \neq v\}$ is a set of undirected intra-layer edges, represented by the symmetric adjacency matrix $A^{(\ell)} \in \{0,1\}^{N \times N}$ with $A^{(\ell)}_{ij} = A^{(\ell)}_{ji}$ for all i, j.

Definition 1.1.2 (Directed Multiplex Network)**.** A *directed multiplex network* is the triple $\mathcal{G} = \langle \mathcal{V}, \{\mathcal{E}^{(\ell)}\}_{\ell=1}^{L}, \mathcal{L} \rangle$, where $\mathcal{V}$ and $\mathcal{L}$ are as in Definition 1.1.1 and, for each layer $\ell \in \mathcal{L}$, $\mathcal{E}^{(\ell)} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of directed arcs represented by the *asymmetric* adjacency matrix $A^{(\ell)} \in \{0,1\}^{N \times N}$, where $A^{(\ell)}_{ij} = 1$ denotes a directed edge $v_i \to v_j$ in layer ℓ and in general $A^{(\ell)}_{ij} \neq A^{(\ell)}_{ji}$. Inter-layer dependencies are modelled as undirected, bidirectional links representing mutual reinforcement between a node and its replica across layers [7].

Remark 1.1.3. An undirected multiplex network is a special case of a directed multiplex network in which $A^{(\ell)}_{ij} = A^{(\ell)}_{ji}$ for all ℓ, i, j. The unified notation $\mathcal{G} = \langle \mathcal{V}, \{A^{(\ell)}\}_{\ell=1}^{L} \rangle$ is used throughout when a statement applies to both settings; the overloaded symbol is disambiguated by context. Chapters 4 and 5 address the directed and undirected settings respectively.

Both settings are governed by the same interdependency model.

Definition 1.1.4 (One-to-One Interdependency)**.** Under the *strict one-to-one interdependency model*, the nodes of $\mathcal{G}$ across all L layers are considered replicas of the same underlying entity. The removal of node v_i from *any* layer $\ell \in \mathcal{L}$ simultaneously removes its replicas from all remaining layers $\ell' \in \mathcal{L} \setminus \{\ell\}$, together with all intra-layer edges incident to v_i in every layer. Formally, if $\tilde{S} \subseteq \mathcal{V}$ is the cumulative removal set, the residual network is $\mathcal{G}_{\tilde{S}} = \langle \mathcal{V} \setminus \tilde{S},\ \{A^{(\ell)}[\mathcal{V} \setminus \tilde{S}]\}_{\ell=1}^{L} \rangle$, where $A^{(\ell)}[\mathcal{V} \setminus \tilde{S}]$ is the submatrix of $A^{(\ell)}$ restricted to $\mathcal{V} \setminus \tilde{S}$.

The one-to-one model gives rise to cascading failure, the primary source of systemic fragility in interdependent networks.

Definition 1.1.5 (Cascading Failure Process)**.** Given a multiplex network $\mathcal{G}$ under the one-to-one interdependency model, the *cascading failure process* induced by the removal of $\tilde{S}_0 \subseteq \mathcal{V}$ proceeds iteratively as follows. At each round $t \geq 0$, define the *jointly active set* $\mathcal{V}^{(t)} \subseteq \mathcal{V} \setminus \tilde{S}_0$ as the largest subset of nodes that are simultaneously in the same connected component (or strongly connected component, for directed networks) in every layer $\ell \in \mathcal{L}$. Any node $v_i \in \mathcal{V} \setminus \tilde{S}_0$ not belonging to $\mathcal{V}^{(t)}$ is declared failed: $\tilde{S}_{t+1} = \tilde{S}_t \cup \{v_i : v_i \notin \mathcal{V}^{(t)}\}$. The process terminates at the fixed point $\mathcal{V}^* = \mathcal{V}^{(\infty)}$, which is reached in finite iterations.

Directed vs. undirected failure propagation. In an *undirected* multiplex network, connectivity is symmetric: a path from u to v exists if and only if a path from v to u exists. Consequently, failure propagation is bidirectional and a node's structural importance is well-captured by scalar centrality measures that treat all incident edges equivalently. In a *directed* multiplex network, this symmetry is broken. The directed out-neighbourhood $\mathcal{N}^{\text{out}}(v) = \{u \in \mathcal{V} : v \to u\}$ and the in-neighbourhood $\mathcal{N}^{\text{in}}(v) = \{u \in \mathcal{V} : u \to v\}$

carry fundamentally different structural information. A node with high out-degree and low in-degree is a *source node*: it propagates influence to many downstream nodes while depending on few. A node with high in-degree and low out-degree is a *sink node*: it aggregates influences but exerts little downstream control. Dismantling a directed network requires identifying and targeting source nodes that sustain directed mutual reachability across layers–a task for which symmetric algorithms are provably suboptimal (Section 1.3.1 and Chapter 4).

Formally, the asymmetry of $A^{(\ell)}$ means that the quantity

$$\Delta^{(\ell)}(v) \;=\; k^{(\ell),\mathrm{out}}(v) \;-\; k^{(\ell),\mathrm{in}}(v) \;=\; \sum_{j} A^{(\ell)}_{vj} \;-\; \sum_{i} A^{(\ell)}_{iv} \tag{1.1}$$

is non-zero in general, constituting the *directional bias* of node v in layer ℓ. Capturing this information requires maintaining separate representations for in- and out-neighbourhoods throughout learning–a design principle architecturally enforced in DDIN (Chapter 4) but absent in all prior methods.

1.2 Network Dismantling as a Combinatorial Optimisation Problem

We now formalise the network dismantling problem for both the undirected multiplex and directed multiplex settings, and develop the performance metrics and Markov decision process (MDP) framework shared by both DDIN and KARL.

Connectivity Targets

Definition 1.2.1 (Largest Mutually Connected Component, LMCC)**.** In an undirected multiplex network $\mathcal{M} = \langle \mathcal{V}, \{E^{(\ell)}\}_{\ell=1}^{L} \rangle$ under the one-to-one interdependency model, a set $C \subseteq \mathcal{V}$ is a *mutually connected component* (MCC) if and only if, for every pair $(u, v) \in C \times C$ with $u \neq v$ and every layer $\ell \in \mathcal{L}$, there exists an undirected path between u and v within $G^{(\ell)}[C]$, with all intermediate vertices also in C. The *Largest Mutually Connected Component* (LMCC) is the MCC of maximum cardinality N_{LMCC}, with normalised size

$$P_\infty(\mathcal{M}) \;=\; \frac{N_{\mathrm{LMCC}}}{N}. \tag{1.2}$$

Definition 1.2.2 (Largest Mutually Strongly Connected Component, MSCC)**.** In a directed multiplex network $\mathcal{G} = \langle \mathcal{V}, \{\mathcal{E}^{(\ell)}\}_{\ell=1}^{L} \rangle$ under the one-to-one interdependency model, a set $C \subseteq \mathcal{V}$ is a *mutually strongly connected component* (MSCC) if and only if, for every ordered pair $(u, v) \in C \times C$ with $u \neq v$ and every layer $\ell \in \mathcal{L}$, there exist both a directed path $u \rightsquigarrow v$ and a directed path $v \rightsquigarrow u$ within the directed subgraph $G^{(\ell)}[C]$. The *Largest Mutually Strongly Connected Component* has cardinality N_{MSCC} and normalised size

$$P_\infty(\mathcal{G}) \;=\; \frac{N_{\mathrm{MSCC}}}{N}. \tag{1.3}$$

Remark 1.2.3. The LMCC is computed by iterative removal of nodes not belonging to the largest connected component simultaneously in every layer [11]. The MSCC is computed iteratively using Tarjan's strongly-connected-components algorithm [55] applied to the intersection of per-layer directed reachability sets. Chapter 4 details the directed cascade resolution procedure used in DDIN.

The Dismantling Optimisation Problem

All algorithms in this thesis construct solutions *sequentially*: at each step $t \in \{1, \dots, N\}$ one node is permanently removed, yielding the cumulative removal set $\tilde{S}_t = \bigcup_{i=1}^{t} \{r_i\}$, with $\tilde{S}_0 = \emptyset$.

Definition 1.2.4 (Network Dismantling Problem)**.** Given a multiplex network $\mathcal{G}$, cost function F, and threshold $\theta > 0$, the *network dismantling problem* is

$$\min_{\tilde{S} \subseteq \mathcal{V}} \; F(\tilde{S}) \qquad \text{subject to} \qquad P_\infty\big(\mathcal{G}_{\tilde{S}}\big) \;\le\; \theta. \tag{1.4}$$

We consider two canonical cost regimes. The *unit cost* assigns equal cost to every removal:

$$F_{\text{unit}}(\tilde{S}) \;=\; |\tilde{S}|. \tag{1.5}$$

The *degree cost* penalises the removal of high-degree nodes proportionally to the edges they carry. For an adjacency matrix $A^{(\ell)}$ that may be asymmetric (directed case), the correct symmetrised formulation uses ordered pairs to avoid ambiguity:

$$F_{\text{deg}}(\tilde{S}) \;=\; \sum_{\ell=1}^{L} \left(\sum_{s \in \tilde{S}} k_s^{(\ell)} \;-\; \frac{1}{2} \sum_{s \in \tilde{S}} \sum_{t \in \tilde{S}} \big(A_{st}^{(\ell)} + A_{ts}^{(\ell)} \big) \right), \tag{1.6}$$

where $k_s^{(\ell)} = \sum_j A_{sj}^{(\ell)}$ is the out-degree of node s in layer ℓ (reducing to the undirected degree when $A^{(\ell)}$ is symmetric), and the symmetrised double-counting correction $\frac{1}{2}(A_{st}^{(\ell)} + A_{ts}^{(\ell)})$ correctly handles both directed and undirected layers.

Proposition 1.2.5 (NP-Hardness)**.** *The network dismantling problem* (1.4) *is NP-hard on single-layer networks [10] and remains NP-hard on undirected multiplex networks under the one-to-one interdependency model [44].*

The intractability of the exact problem motivates sequential, greedy-approximation strategies parameterised by deep reinforcement learning. The quality of any such strategy is assessed by two complementary metrics.

Definition 1.2.6 (Area Under the Dismantling Curve, AUDC)**.** The *Area Under the Dismantling Curve* integrates the normalised component size against the incremental removal cost:

$$\text{AUDC} \;=\; \frac{1}{F(\mathcal{V})} \sum_{t=1}^{N} P_\infty\big(\tilde{S}_t\big) \Big[F(\tilde{S}_t) - F(\tilde{S}_{t-1}) \Big]. \tag{1.7}$$

Lower AUDC values indicate superior dismantling performance.

Definition 1.2.7 (Dismantling Cost, C^*)**.** The *dismantling cost* is the smallest normalised cost required to reduce the component size below the non-extensivity threshold $\theta = 1/\sqrt{N}$:

$$C^* \;=\; \frac{F(\tilde{S}_c)}{F(\mathcal{V})}, \qquad \tilde{S}_c \;=\; \arg \min_{\tilde{S}_t \,:\, P_\infty(\tilde{S}_t) \le 1/\sqrt{N}} F(\tilde{S}_t). \tag{1.8}$$

Both AUDC and C^* lie in $[0, 1]$; lower values are better.

Markov Decision Process Formulation

Following FINDER [23] and MultiDismantler [25], we cast sequential dismantling as a deterministic Markov decision process.

Definition 1.2.8 (MDP for Network Dismantling)**.** The dismantling MDP is the quintuple $\mathcal{P} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ where:

(i) **State space** $\mathcal{S}$: The state $s_t = \mathcal{G} \setminus \tilde{S}_t$ is the residual multiplex network at step t, augmented with one virtual node per layer to provide a global receptive field for the state-embedding vector.

(ii) **Action space** $\mathcal{A}(s_t)$: The discrete action space $\mathcal{A}(s_t) = \mathcal{V} \setminus \tilde{S}_t$ comprises all nodes not yet removed.

(iii) **Transition function** $\mathcal{T}$: Deterministic. Taking action $a_t \in \mathcal{A}(s_t)$ removes node v_{a_t} from all layers, resolves cascading failures by iterative component pruning, and yields state s_{t+1}.

(iv) **Reward function** $\mathcal{R}$: The immediate reward for removing node v at step t is

$$r_t(v) \;=\; P_\infty(s_t) \cdot F\big(\{v_{a_t}\}\big), \tag{1.9}$$

jointly incentivising large reductions in the component size and low removal cost.

(v) **Discount factor** $\gamma \in [0, 1)$; both frameworks use $\gamma = 0.99$.

To capture the delayed effects of cascading failures, both DDIN and KARL employ n-step returns. The n-step temporal-difference (TD) target at step t is

$$y_t \;=\; \sum_{j=0}^{n-1} \gamma^j \, r_{t+j} \;+\; \gamma^n \max_{a'} \hat{Q}\big(s_{t+n}, a'; \hat{\Theta}\big), \tag{1.10}$$

where $\hat{\Theta}$ are the target-network parameters, periodically copied from the online parameters Θ to stabilise training. The agent minimises the TD loss

$$\mathcal{L}_{\text{TD}}(\Theta) \;=\; \mathbb{E}_{(s_t, a_t, \ldots) \sim \mathcal{B}}\left[\Big(y_t - Q(s_t, a_t; \Theta) \Big)^2 \right], \tag{1.11}$$

where $\mathcal{B}$ is an experience-replay buffer. Both frameworks augment this with Prioritised Experience Replay (PER) [50], which samples transitions proportionally to their absolute TD error $|\delta_t|^{\alpha_{\text{PER}}}$, focusing training effort on the most informative transitions and correcting the induced sampling bias via importance-sampling weights β_{IS}.

The above formulation provides the shared mathematical substrate for both DDIN and KARL. The two frameworks diverge precisely along the two axes identified in the introduction: (i) the topology of the input network (directed vs. undirected), and (ii) the functional form of the GNN encoder and value-function projections (KAN-based vs. MLP-based).

1.3 Limitations of Current State-of-the-Art

Prior to this thesis, every deep reinforcement learning dismantling method shared a common architectural template: a symmetric graph neural network encoder with fixed affine multilayer-perceptron projections, trained via off-policy Q-learning on undirected synthetic graphs. Table 1.1 situates the key methods along the two dimensions in which this thesis departs from the template. We identify two orthogonal limitations in the template and address each with a separate architectural contribution.

Table 1.1: Summary of representative deep-RL-based dismantling methods. **Dir**: supports directed networks. **KAN**: uses learnable KAN activations. ✓ = yes; × = no.

Method	Venue	**Dir**	**KAN**	Key Architecture
HDA, CI, MinSum	–	×	×	Degree/influence heuristics
GDM [24]	Nat. Commun. 2021	×	×	MLP encoder + supervised
FINDER [23]	Nat. Mach. Intell. 2020	×	×	Single-layer GNN + DQN
NIRM [66]	CIKM 2022	×	×	Local + global GNN + DQN
MultiDismantler [25]	Nat. Mach. Intell. 2025	×	×	Multiplex GNN (MGNN) + DQN
DDIN (Chapter 4)	AAAI 2026	✓	×	Asymmetric SAGE + Prioritised RL
KARL (Chapter 5)	Under review	×	✓	Residual KAN + KANformer + Cheby-Q

1.3.1 Limitation 1 (Topological): The Undirected Bottleneck

Real-world critical systems that motivate the dismantling problem are inherently *directed*. Biological signalling cascades, supply-chain dependency graphs, financial contagion networks, and directed social-influence topologies all encode asymmetric relationships: regulatory proteins activate target genes in a single direction; money flows from creditors to debtors; influence propagates from broadcasters to followers. In such systems, the MSCC is the appropriate connectivity target, and its fragmentation requires a strategy that is sensitive to the asymmetry of directed paths.

Why symmetric methods fail. All existing deep-RL dismantling methods employ standard GNN message-passing of the form

$$\mathbf{h}_v^{(k+1)} = \textsc{update}\Big(\mathbf{h}_v^{(k)},\ \textsc{agg}\Big(\big\{\mathbf{h}_u^{(k)} : u \in \mathcal{N}(v)\big\}\Big)\Big), \tag{1.12}$$

where $\mathcal{N}(v)$ is the neighbourhood of v defined by the (undirected or symmetrised) adjacency matrix. Symmetric aggregation is by construction invariant to the direction of any edge incident to v: the representation $\mathbf{h}_v^{(k)}$ is identical whether a given neighbour u *reaches* v (i.e., $u \to v$, making v a downstream dependency of u) or is *reached from* v (i.e., $v \to u$, making u a downstream dependency of v). As Ma, Wang, and Li [36] demonstrate empirically, this symmetry leads to systematic misidentification of source and sink nodes, producing suboptimal dismantling sequences that remove influential sinks while leaving uncontrolled sources intact.

When applied to a directed layer, the symmetric aggregation is

$$\textsc{agg}^{\text{sym}}\big(\mathbf{H}, v\big) = \frac{1}{\big|\mathcal{N}^{\text{in}}(v) \cup \mathcal{N}^{\text{out}}(v)\big|} \sum_{u \in \mathcal{N}^{\text{in}}(v) \cup \mathcal{N}^{\text{out}}(v)} \mathbf{h}_u, \tag{1.13}$$

which conflates the semantically distinct signals carried by in-neighbours (nodes that v depends upon) and out-neighbours (nodes that depend upon v). By contrast, the asymmetric aggregation introduced in DDIN (Chapter 4) explicitly separates these signals:

$$\mathbf{h}_v^{(\ell)} \;=\; \mathrm{ReLU}\Bigg(W_{\text{out}}\Big[\mathbf{x}_v;\quad \underbrace{\textsc{MeanAgg}\big(\mathcal{N}^{\text{in}}(v)\big)}_{\text{incoming influence (being controlled)}}\quad\Big]+W_{\text{in}}\Big[\mathbf{x}_v;\quad \underbrace{\textsc{MaxAgg}\big(\mathcal{N}^{\text{out}}(v)\big)}_{\text{outgoing control (exerting influence)}}\quad\Big]\Bigg), \tag{1.14}$$

where MeanAgg captures the *average* influence exerted on node v by its in-neighbours and MaxAgg captures the *strongest* downstream dependency of v on its out-neighbours, preserving the directional flow semantics absent from all prior encoders. This is a preview of the full encoder update derived in Chapter 4.

1.3.2 Limitation 2 (Functional): The Affine Projection Bottleneck

Even within the undirected multiplex setting where MultiDismantler operates, a second, independent limitation constrains the representational fidelity of the dismantling policy. Every existing GNN-based method parameterises its message-passing and action-value projections through standard MLP layers of the form

$$\phi_{\text{MLP}}(\mathbf{x}) \;=\; \sigma(W\mathbf{x}+\mathbf{b}), \qquad W\in\mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}, \tag{1.15}$$

where σ is a fixed nonlinearity (typically RELU) and W is a *global affine transformation* applied uniformly across all input coordinates. Each MLP layer applies a single global linear map per input coordinate, confining the per-edge representational capacity to a fixed-degree polynomial function class.

The Kolmogorov-Arnold alternative. The representational limitation of MLPs was examined systematically by Liu et al. [35], who proposed *Kolmogorov-Arnold Networks* (KANS) as an architecturally distinct alternative grounded in the following classical theorem.

Theorem 1.3.1 (Kolmogorov-Arnold Representation Theorem)**.** *Any continuous function $f:[0,1]^n\to\mathbb{R}$ admits the representation*

$$f(x_1,\dots,x_n) \;=\; \sum_{q=1}^{2n+1}\Phi_q\left(\sum_{p=1}^{n}\varphi_{q,p}(x_p)\right), \tag{1.16}$$

where each $\varphi_{q,p}:[0,1]\to\mathbb{R}$ and each $\Phi_q:\mathbb{R}\to\mathbb{R}$ is continuous.

Theorem 2.6.1 implies that any multivariate function can be decomposed into a composition of *univariate* functions. Rather than applying a single global affine map per input coordinate, a KAN layer places a distinct learnable univariate function on each network edge, parameterised as a B-spline:

$$\varphi(x) \;=\; w\left(b(x)+\sum_{i=1}^{G+k}c_i\,B_i^k(x;\,G)\right), \tag{1.17}$$

where $b(x)=\mathrm{SiLU}(x)$ is the base activation, c_i are trainable spline coefficients, $B_i^k(x;\,G)$ are B-spline basis functions from the Cox–de Boor recurrence over a uniform knot vector

with G grid intervals and polynomial order k, and w is a learnable scalar weight [35]. The vectorised KAN module accumulates per-edge univariate evaluations as

$$\text{KAN}(\mathbf{x}) \;=\; W_{\text{base}}\,\sigma(\mathbf{x}) \;+\; W_{\text{spline}}\,B(\mathbf{x}), \tag{1.18}$$

where $B(\mathbf{x}) \in \mathbb{R}^{d_{\text{in}}(G+k)}$ stacks the per-coordinate B-spline evaluations and $W_{\text{spline}} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}(G+k)}$ are the spline control weights. At any fixed grid resolution G and order $k \geq 1$, the per-edge function family spanned by a KAN layer strictly contains the per-edge family of an MLP layer of identical input/output dimension: the B-spline basis is a $(G+k)$-dimensional function space, whereas the MLP edge reduces to the one-dimensional family $x \mapsto W_{ji}\,\sigma(x)$ parameterised by the single scalar W_{ji}.

The gradient instability challenge. Integrating KANs into an off-policy RL loop introduces a complication that does not arise in supervised learning: the continuously shifting distribution of TD targets can drive B-spline activations toward the boundaries of their fixed grid domains, producing gradient instability. KARL resolves this through a macroscopic residual branch

$$\varphi'(x) \;=\; \varphi(x) + \alpha_{\text{res}} \cdot \text{MLP}_{\text{res}}(x), \tag{1.19}$$

where α_{res} is a learnable scalar (initialised to 0.5) and MLP_{res} is a single linear-RELU layer that provides a low-frequency anchor for the gradient signal. Furthermore, the action-value head replaces B-splines with *orthogonal Chebyshev polynomials*, whose minimax approximation property and logarithmically bounded operator norm prevent the Runge-type boundary oscillations that afflict B-spline interpolation under non-stationary TD targets. The complete theoretical justification for all three design choices is developed in Chapter 5.

1.4 Research Questions

The two limitations identified above motivate the following pair of research questions, each addressed by one of the two frameworks developed in this thesis.

Research Question 1 (RQ1) – Topological.

> *How can directed asymmetry in real-world interdependent multiplex networks be explicitly encoded in the graph neural network encoder of a deep reinforcement learning dismantling agent, so as to distinguish the roles of source and sink nodes and achieve superior dismantling of the Largest Mutually Strongly Connected Component compared to symmetric baseline methods?*

RQ1 is addressed in Chapter 4 through DDIN, which introduces a directed GraphSAGE encoder with asymmetric mean/max aggregation over separated in- and out-neighbourhoods, a directed multi-relational attention mechanism for cross-layer fusion, and a bilinear decoder with topological reconstruction loss, trained by sum-tree prioritised n-step DQN on directed synthetic multiplex graphs generated by an enhanced Geometric Multiplex Model (GMM).

Research Question 2 (RQ2) – Functional.

> *Can Kolmogorov-Arnold Networks, by replacing fixed affine projections with learnable univariate B-spline functions on graph edges, overcome the limited per-edge expressivity of MLP-based GNN encoders and capture the high-frequency,*

non-smooth vulnerability patterns that govern cascading fragility in scale-free undirected multiplex networks, while preserving training stability under non-stationary temporal-difference targets?

RQ2 is addressed in Chapter 5 through KARL, which constitutes the first architecture to successfully embed a hybrid Residual Graph KAN within an off-policy deep Q-network for combinatorial optimisation over multiplex topologies. The three-stage architecture–residual B-spline KAN encoder, fully KAN-parameterised KANFORMER cross-layer attention, and orthogonal Chebyshev-KAN action-value head–is supported by formal guarantees of gradient stability, near-minimax Q-value approximation, and universal approximation of the MLP-GNN function class.

1.5 Thesis Contributions

This thesis makes the following contributions to the fields of network science, graph neural networks, and deep reinforcement learning for combinatorial optimisation.

Contributions of DDIN (Chapter 4).

(i) **First directed multiplex dismantling framework.** We propose DDIN, the first deep-RL-based algorithm for dismantling directed interdependent multiplex networks, directly addressing the absence of directional awareness in all prior methods [20].

(ii) **Asymmetric GraphSAGE encoder.** We introduce a directed variant of the GRAPHSAGE framework [26] that separates mean pooling over in-neighbours (capturing incoming influence) from max pooling over out-neighbours (capturing downstream control), preserving the directional flow semantics critical for MSCC fragmentation.

(iii) **Directed multi-relational attention.** We design a multi-relational cross-layer attention mechanism that weights the contribution of each layer to the fused node embedding according to how critical that layer's topology is to the current state of the interdependent structure, naturally prioritising "bridge" layers that sustain the giant strongly connected component.

(iv) **Empirical validation on five real-world directed multiplexes.** DDIN achieves 16–23% lower AUDC compared to directed collective-influence and HDA baselines across the *C. elegans* connectome (279 nodes, 3 layers), *Drosophila melanogaster* (8,215 nodes, 7 layers), FAO Trade (214 nodes, 364 layers), Homo Sapiens genetic interaction network (18k nodes, 7 layers), and Sanremo 2016 (56k nodes, 3 layers).

Contributions of KARL (Chapter 5).

(i) **First KAN-RL agent for combinatorial graphs.** KARL is the first architecture to integrate Kolmogorov-Arnold Networks into an off-policy deep Q-network for any NP-hard network optimisation problem, resolving the gradient-instability challenge of B-spline activations under non-stationary TD targets.

(ii) **KANformer cross-layer fusion module.** We introduce a fully KAN-parameterised multi-head cross-attention mechanism–KANFORMER–that replaces every linear Q/K/V projection with an independent B-spline transformation, enabling non-smooth, per-coordinate cross-layer routing unavailable to any affine attention module.

(iii) **Orthogonal Chebyshev-KAN action-value head.** Replacing the B-spline decoder with an orthogonal Chebyshev polynomial basis yields well-conditioned Q-value regression with logarithmically bounded operator norm, preventing the Runge-type oscillations inherent to equispaced polynomial interpolation under non-stationary targets.

(iv) **Theoretical expressivity guarantees.** We prove three formal results: (a) a gradient-stability proposition bounding worst-case back-propagated gradient norms through the residual B-spline encoder; (b) a near-minimax theorem establishing that the Chebyshev Q-head achieves near-optimal TD-approximation error; and (c) a universal-approximation theorem proving that the KAN-GNN function class subsumes the MLP-GNN function class at every fixed depth and embedding dimension.

(v) **State-of-the-art zero-shot generalisation.** Trained exclusively on small synthetic graphs ($N \in [30, 50]$ nodes), KARL achieves an average AUDC improvement of 21.96% and a dismantling-cost C^* reduction of 4.57% over MultiDismantler on six real-world undirected multiplex benchmarks spanning three orders of magnitude in scale (up to 56,562 nodes), with emergent structural interpretability towards high-coreness nodes arising without auxiliary structural supervision.

Theoretical contributions. Beyond the algorithmic contributions, this thesis provides a unified mathematical treatment of the dismantling MDP under both undirected and directed multiplex topologies (Chapter 3), a comparative expressivity analysis of KAN-GNN versus MLP-GNN function classes for graph dismantling policies (Chapter 5), and a formal discussion of the dual-use implications and future research directions arising from both frameworks (Chapter 7).

1.6 Thesis Organisation

The remainder of this thesis is organised as follows.

Chapter 2 – Background and Literature Review. We survey the graph-theoretic and algorithmic background required for the subsequent chapters: classical dismantling heuristics, the deep reinforcement learning framework for graph optimisation, the message-passing GNN paradigm including GraphSAGE and graph attention networks, and the theory of Kolmogorov-Arnold Networks. The chapter culminates in a structured summary of the research gaps that motivate the two contributions.

Chapter 3 – Preliminaries and Mathematical Formulation. We develop the unified mathematical framework used throughout the thesis: the directed and undirected Geometric Multiplex Models employed for training corpus generation, the formal MDP specification for both network settings (state space, action space, reward functions), and the cost regimes

and evaluation metrics (AUDC, C^*) with a rigorous derivation of the statistical significance protocol employed in experimental comparisons.

Chapter 4 – Disassembling Directed Interdependent Networks (DDIN). We present the DDIN framework in full. We formally characterise the failure of symmetric methods on directed multiplex networks, develop the three-stage architecture (asymmetric GraphSAGE encoder, multi-relational attention, bilinear decoder with topological reconstruction loss), detail the prioritised n-step DQN policy search, report experimental results on five real-world directed multiplex benchmarks, and present ablation studies validating each architectural component.

Chapter 5 – Kolmogorov-Arnold Reinforcement Learning (KARL). We present the KARL framework in full. We formally characterise the functional limitation of MLP-based GNN encoders, develop the three-stage KAN architecture, prove the three theoretical guarantees (gradient stability, near-minimax Q-head, KAN-GNN subsumption), report experimental results on six real-world undirected multiplex benchmarks with full statistical significance testing, and present extensive ablation studies covering residual strength, B-spline grid resolution, polynomial order, and emergent structural interpretability.

Chapter 6 – Comparative Analysis and Synthesis. We juxtapose the two frameworks along three dimensions: architectural design, the expressivity-versus-directionality trade-off, and performance on the shared FAO Trade, Homo Genetic, and Sanremo 2016 benchmarks that appear in both experimental suites. We also formulate the *Grand Synthesis* conjecture: a directed KAN-parameterised multiplex dismantling agent that simultaneously addresses both limitations.

Chapter 7 – Conclusion and Future Work. We summarise the contributions and theoretical results of the thesis, discuss the dual-use ethical implications of dismantling methodology, and articulate three concrete future research directions: dynamic/temporal multiplex graphs, relaxed interdependency models, and the directed-KAN Grand Synthesis.

Appendices. Appendix A provides complete hyperparameter configurations for both DDIN and KARL. Appendix B details the code structure, dataset sources, and reproduction instructions.

Chapter 2

Background and Literature Review

This chapter develops the formal, mathematical, and conceptual foundations required to understand the two contributions of this thesis. Section 2.1 introduces the graph-theoretic primitives — directed and undirected graphs, degree sequences, k-core decomposition, and multiplex network formalisms — that underpin both problem settings. Section 2.2 formalises the Buldyrev cascading failure model and defines the two connectivity criteria that measure network fragmentation: the Largest Mutually Connected Component (LMCC) for undirected multiplexes and the Largest Mutually Strongly Connected Component (MSCC) for directed multiplexes. Section 2.3 surveys the classical heuristic algorithms for network dismantling, characterises their fundamental limitations, and establishes them as the lower-bound baselines against which learned methods are measured. Section 2.4 develops the deep reinforcement learning framework used throughout the thesis, gives a detailed exposition of the state-of-the-art MultiDismantler [25] — the primary baseline for both DDIN and KARL — and situates it within the broader GNN-DRL literature. Section 2.5 covers the graph neural network architectures relevant to both contributions, with particular emphasis on directed and multi-relational extensions. Section 2.6 provides a self-contained treatment of Kolmogorov-Arnold Networks, including the B-spline parameterisation and the instability modes that motivate KARL's architectural choices. The chapter concludes in Section 2.7 with a structured summary of the research gaps that the two contributions of this thesis address.

2.1 Graph-Theoretic Foundations

2.1.1 Undirected Graphs and Basic Quantities

Definition 2.1.1 (Undirected Graph)**.** An *undirected graph* $G = (V, E)$ consists of a finite vertex set $V = \{v_1, \ldots, v_N\}$ with $|V| = N$, and an edge set $E \subseteq \{\{u, v\} : u, v \in V,\ u \neq v\}$. The *adjacency matrix* $\boldsymbol{A} \in \{0, 1\}^{N \times N}$ is defined by $A_{ij} = 1$ if $\{v_i, v_j\} \in E$ and $A_{ij} = 0$ otherwise. By convention, $A_{ii} = 0$ for all i.

The *degree* of a vertex $v \in V$ is the number of its neighbours:

$$k_v \;=\; \sum_{j=1}^{N} A_{vj} \;=\; |\mathcal{N}(v)|, \tag{2.1}$$

where $\mathcal{N}(v) = \{u \in V : A_{vu} = 1\}$ is the neighbourhood of v. The *degree sequence* of G is the sorted tuple $(k_{(1)} \geq k_{(2)} \geq \cdots \geq k_{(N)})$. A graph is called *scale-free* if the degree sequence follows a power-law distribution $P(k) \sim k^{-\gamma}$ for some exponent $\gamma > 0$ [3]. Empirically, most real-world multiplex networks studied in this thesis — including C. elegans ($\gamma \approx 2.4$), Homo Genetic, and FAO Trade — exhibit scale-free degree distributions, a property that has profound consequences for dismantling vulnerability.

2.1.2 Directed Graphs

Definition 2.1.2 (Directed Graph)**.** A *directed graph* (digraph) $\vec{G} = (V, \vec{E})$ has the same vertex set V but an edge set $\vec{E} \subseteq V \times V$ of ordered pairs (u, v), called *directed arcs* or *directed edges*. The adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N \times N}$ is defined by $A_{ij} = 1$ if $(v_i, v_j) \in \vec{E}$, and is generally asymmetric.

For a vertex v in a directed graph, the *in-degree* and *out-degree* are respectively:

$$k_v^{\text{in}} = \sum_j A_{jv}, \qquad k_v^{\text{out}} = \sum_j A_{vj}. \tag{2.2}$$

The *out-neighbourhood* $\mathcal{N}^+(v) = \{u : (v,u) \in \vec{E}\}$ and the *in-neighbourhood* $\mathcal{N}^-(v) = \{u : (u,v) \in \vec{E}\}$ capture distinct directional information. In biological regulatory networks, for example, $\mathcal{N}^+(v)$ represents the genes regulated by gene v (downstream targets), while $\mathcal{N}^-(v)$ represents its upstream regulators [2].

Remark 2.1.3. The asymmetry $k_v^{\text{in}} \neq k_v^{\text{out}}$ is the fundamental structural property that differentiates directed network dismantling (Chapter 4) from the undirected setting (Chapter 5). A node with high k_v^{out} but low k_v^{in} controls many downstream nodes without itself depending on many upstream nodes. Removing such a node disrupts outgoing flows far more than incoming ones — a directional cascade pattern that symmetric aggregation is structurally incapable of representing.

2.1.3 Connected Components and Core Decomposition

Definition 2.1.4 (Connected Component)**.** In an undirected graph G, a *connected component* is a maximal subgraph in which every pair of vertices is connected by a path. The *largest connected component* (LCC) is the connected component of maximum cardinality.

Definition 2.1.5 (Strongly Connected Component)**.** In a directed graph $\vec{G}$, a *strongly connected component* (SCC) is a maximal subset $C \subseteq V$ such that for every ordered pair (u, v) with $u, v \in C$, there exists a directed path from u to v *and* a directed path from v to u. The *condensation* of $\vec{G}$ is the directed acyclic graph (DAG) obtained by collapsing each SCC to a single vertex.

SCCs can be computed in $O(|V| + |\vec{E}|)$ time via Tarjan's depth-first search algorithm [55], which maintains a low-link value for each vertex to identify cycle-closing back-edges. This procedure is invoked at every step of the DDIN dismantling loop (Section 4.5) to track the residual MSCC after each node removal.

Definition 2.1.6 (k-Core and Coreness)**.** The *k-core* of a graph G is the maximal induced subgraph in which every vertex has degree at least k. The *coreness* $c(v)$ of a vertex v is the largest k such that v belongs to the k-core:

$$c(v) = \max\{k \in \mathbb{Z}_{\geq 0} : v \in \mathcal{K}_k(G)\}, \tag{2.3}$$

where $\mathcal{K}_k(G)$ denotes the k-core of G.

The k-core decomposition can be computed greedily in $O(|V| + |E|)$ time by iteratively removing vertices of degree less than k and updating the remaining degrees [1]. In the

context of multiplex dismantling, nodes with high coreness are disproportionately critical: their removal from any layer often triggers cascading collapses because they simultaneously anchor many inter-layer mutual connectivity paths. This property is central to the structural interpretability analysis of KARL presented in Section 5.6.3.

2.1.4 Multiplex and Multilayer Networks

Definition 2.1.7 (Multilayer Network [31])**.** A *multilayer network* is a tuple $\mathcal{M} = (V, \{E^{(\ell)}\}_{\ell=1}^{L}, \mathcal{L})$ where $V = \{v_1, \ldots, v_N\}$ is the shared vertex set of cardinality N, $\mathcal{L} = \{1, \ldots, L\}$ is the set of layers, and $E^{(\ell)} \subseteq \binom{V}{2}$ (undirected) or $E^{(\ell)} \subseteq V \times V$ (directed) is the edge set of layer ℓ with adjacency matrix $\boldsymbol{A}^{(\ell)} \in \{0,1\}^{N \times N}$.

The degree of vertex v in layer ℓ is $k_v^{(\ell)} = \sum_j A_{vj}^{(\ell)}$, and the *multiplex degree* is the vector $(k_v^{(1)}, \ldots, k_v^{(L)})$. For directed layers, in-degree $k_v^{(\ell),\text{in}}$ and out-degree $k_v^{(\ell),\text{out}}$ are defined analogously to Equation (2.2).

Definition 2.1.8 (One-to-One Interdependency [11])**.** A multilayer network satisfies the *strict one-to-one interdependency model* if removing vertex v from any layer ℓ simultaneously removes its replica in all remaining layers $\ell' \neq \ell$, together with all incident edges in those layers. Under this model, the operational state of a vertex in any single layer fully determines its state in every other layer.

Definition 2.1.9 (Mutually Connected Component [11])**.** Two vertices $u, v \in V$ belong to the same *mutually connected component* (MCC) if and only if, for every layer $\ell \in \mathcal{L}$, there exists a path between u and v within the induced subgraph $G_\ell = (V, E^{(\ell)})$, where path eligibility is restricted to vertices that remain simultaneously functional in all layers. The *Largest Mutually Connected Component* (LMCC) is the MCC of maximum cardinality; its normalised size is

$$P_\infty(\mathcal{M}) = \frac{N_{\text{LMCC}}}{N}, \tag{2.4}$$

where N_{LMCC} denotes the number of vertices in the LMCC.

Definition 2.1.10 (Mutually Strongly Connected Component)**.** In a directed multilayer network, two vertices $u, v \in V$ belong to the same *mutually strongly connected component* (MSCC) if and only if, for every layer $\ell \in \mathcal{L}$, there exists a directed path $u \to v$ *and* a directed path $v \to u$ within G_ℓ, subject to the same simultaneous functionality constraint. The *Largest Mutually Strongly Connected Component* is the MSCC of maximum cardinality.

Remark 2.1.11. The LMCC and MSCC generalise the largest connected component (LCC) and largest SCC of single-layer networks, respectively. In an undirected setting, MSCC ≡ MCC ≡ LMCC. For directed networks, however, a vertex can be reachable from u in one direction but not the other, making the MSCC strictly smaller than or equal to the largest WCC (weakly connected component). This structural difference necessitates the distinct algorithmic treatment in Chapter 4.

2.2 The Cascading Failure Model

2.2.1 Interdependent Networks and Catastrophic Collapse

The seminal work of Buldyrev et al. [11] established that interdependent networks are qualitatively more fragile than isolated single-layer networks. The 2003 Italy blackout — where a power-grid failure cascaded into internet node failures, which in turn caused further power station outages — is the canonical empirical manifestation of this fragility [60]. Formally, the *cascading failure process* under the one-to-one interdependency model proceeds as follows.

Let $S \subseteq V$ be a set of initially removed vertices. Define $V^{(0)} = V \setminus S$ as the initial survivor set. The cascade unfolds in discrete rounds:

$$V^{(t+1)} \;=\; \mathrm{MCC}\big(V^{(t)},\ \{E^{(\ell)}\}_{\ell=1}^{L}\big), \quad t = 0, 1, 2, \ldots, \tag{2.5}$$

where $\mathrm{MCC}(\cdot)$ extracts the vertex set of the largest mutually connected component of the induced multiplex subgraph on the current survivor set. Vertices excluded from the MCC at round t are removed from $V^{(t+1)}$, as they can no longer sustain mutual connectivity. This triggers further edge and vertex removals in subsequent rounds. The process terminates when $V^{(t+1)} = V^{(t)}$ (a fixed point), yielding the residual LMCC.

The cascade process defined by Equation (2.5) has two important properties:

Proposition 2.2.1 (Monotone Convergence of the Cascade)**.** *The sequence $\{V^{(t)}\}_{t\geq 0}$ is non-increasing: $V^{(t+1)} \subseteq V^{(t)}$ for all $t \geq 0$. Therefore, the cascade converges to a fixed point $V^* = \bigcap_{t\geq 0} V^{(t)}$ in at most N rounds.*

Proposition 2.2.2 (Non-linearity of Cascade Damage)**.** *Let $S_1 \subset S_2 \subseteq V$ be two removal sets. Even when $|S_2| = |S_1| + 1$ (a single additional removal), the resulting LMCC sizes can satisfy $P_\infty(\mathcal{M} \setminus S_2) \ll P_\infty(\mathcal{M} \setminus S_1)$, because the marginal removal may trigger a chain reaction that collapses the entire LMCC. This* cascade amplification *makes the dismantling problem qualitatively harder than its single-layer counterpart.*

2.2.2 The Dismantling Problem

Definition 2.2.3 (Network Dismantling Problem [10])**.** Given a multiplex network $\mathcal{M}$ and a cost function $F : 2^V \to \mathbb{R}_{\geq 0}$, the *network dismantling problem* is to find the minimum-cost removal set $S^* \subseteq V$ that reduces the normalised LMCC below the non-extensivity threshold $1/\sqrt{N}$:

$$S^* \;=\; \arg\min_{S\subseteq V} F(S) \quad \text{subject to} \quad P_\infty(\mathcal{M} \setminus S) \;\leq\; \frac{1}{\sqrt{N}}. \tag{2.6}$$

The threshold $1/\sqrt{N}$ is the conventional definition of non-extensivity: a component of size $o(\sqrt{N})$ is considered negligible relative to the system size [13].

Two canonical cost regimes are considered throughout this thesis.

Unit cost. Each removed vertex costs one unit irrespective of its degree:

$$F_{\text{unit}}(S) \;=\; |S|. \tag{2.7}$$

Degree cost. Each removed vertex is penalised proportionally to the number of edges it carries across all layers (edges shared between two removed vertices are counted once):

$$F_{\deg}(S) \;=\; \sum_{\ell=1}^{L}\Bigg(\sum_{s\in S} k_s^{(\ell)} \;-\; \sum_{\{s,t\}\subseteq S} A_{st}^{(\ell)}\Bigg). \tag{2.8}$$

The degree-cost regime directly quantifies structural disruption per unit of rewiring cost, which is relevant for network-hardening applications where edge maintenance is expensive.

2.2.3 Performance Metrics

All dismantling algorithms construct sequential solutions: at each step t, one vertex r_t is selected, yielding the cumulative removal sequence $\tilde{S}_t = \bigcup_{i=1}^{t}\{r_i\}$ with $\tilde{S}_0 = \emptyset$.

Definition 2.2.4 (Area Under the Dismantling Curve (AUDC) [25])**.** The *area under the dismantling curve* is defined as:

$$\mathrm{AUDC} \;=\; \frac{1}{F(V)}\sum_{t=1}^{N} P_\infty(\tilde{S}_t)\,\big[F(\tilde{S}_t) - F(\tilde{S}_{t-1})\big]. \tag{2.9}$$

AUDC integrates the normalised LMCC size against the normalised cumulative removal cost, generalising the robustness metric of Schneider et al. [51]. Lower AUDC indicates more efficient dismantling.

Definition 2.2.5 (Dismantling Cost C^* [25])**.** The *dismantling cost* is the smallest normalised cost required to reduce the LMCC below the non-extensivity threshold:

$$C^* \;=\; \frac{F(\tilde{S}_c)}{F(V)}, \qquad \tilde{S}_c \;=\; \operatorname*{arg\,min}_{\tilde{S}_t:\, P_\infty(\tilde{S}_t)\le 1/\sqrt{N}} F(\tilde{S}_t). \tag{2.10}$$

Both AUDC and C^* lie in $[0, 1]$; lower values indicate superior dismantling performance.

2.2.4 Computational Hardness

The dismantling problem is NP-hard. For single-layer networks, this was established by Braunstein et al. [10] via reduction from the minimum feedback vertex set problem. The multiplex setting is strictly harder:

Theorem 2.2.6 (NP-Hardness of Multiplex Dismantling [44])**.** *The network dismantling problem on a one-to-one interdependent multiplex network $\mathcal{M} = (V, \{E^{(\ell)}\}_{\ell=1}^{L})$ under both unit and degree cost is NP-hard for $L \ge 1$.*

NP-hardness implies that exact solutions are computationally intractable for all but very small networks ($N \lesssim 20$), motivating the use of approximation algorithms — including heuristics (Section 2.3) and deep reinforcement learning (Section 2.4).

2.3 Classical Dismantling Heuristics

2.3.1 High-Degree Adaptive (HDA)

The simplest and most widely applied heuristic is the *High-Degree Adaptive* (HDA) algorithm [12], which greedily removes the vertex of highest degree at each step. In the

multiplex setting [44], HDA assigns to each vertex the *maximum-layer* score:

$$s_i^{\mathrm{HDA}} \;=\; \max_{\ell \in \mathcal{L}} \; k_i^{(\ell)}, \tag{2.11}$$

removes $\arg\max_i s_i^{\mathrm{HDA}}$, and then re-evaluates all scores on the residual graph after cascading failures have been resolved. The adaptive re-scoring is essential for capturing the structural changes induced by each removal. HDA is computationally efficient ($O(N \log N)$ per step with a priority queue) but considers only immediate structural information, ignoring the long-range cascade patterns that determine true dismantling vulnerability.

2.3.2 Collective Influence (CI)

The *Collective Influence* algorithm, introduced by Morone and Makse [43], is motivated by an optimal percolation analysis of the branching process governing cascade propagation on locally tree-like networks. The CI score of vertex i at radius ℓ is:

$$\mathrm{CI}_\ell(i) \;=\; (k_i - 1) \sum_{j \in \partial \mathrm{Ball}(i,\ell)} (k_j - 1), \tag{2.12}$$

where $\partial\mathrm{Ball}(i, \ell)$ denotes the frontier of the ℓ-hop ball centred at i (all vertices at graph distance exactly ℓ from i). The factor $(k_i - 1)$ accounts for the effective propagation of the cascade from vertex i, and $(k_j - 1)$ accounts for the propagation potential at each frontier vertex. For $\ell = 0$, $\mathrm{CI}_0(i) = k_i - 1$ reduces to HDA. For $\ell = 1$ (the standard setting used throughout this thesis), CI captures second-order structural information.

In the multiplex setting, CI is extended via the maximum-score rule [44]: the global score is $s_i^{\mathrm{CI}} = \max_\ell \mathrm{CI}_\ell^{(\ell)}(i)$, where $\mathrm{CI}_\ell^{(\ell)}(i)$ is the CI score of vertex i computed within layer ℓ.

2.3.3 MinSum

The *MinSum* algorithm [10] frames network dismantling as a constraint-satisfaction problem and derives removal scores from a belief propagation procedure on the factor graph associated with the minimum feedback vertex set. At each step, MinSum assigns to vertex i in layer ℓ a score $r_i^{(\ell)} \in [1, N]$ representing its position in the optimal single-layer dismantling sequence. The global multiplex score is:

$$s_i^{\mathrm{MinSum}} \;=\; \max_{\ell \in \mathcal{L}} \; \frac{1}{r_i^{(\ell)}}, \tag{2.13}$$

so that vertices recommended for early removal by the belief propagation in any layer receive high global scores. MinSum achieves near-optimal performance on single-layer networks but exhibits degraded performance on multiplex settings because it optimises layer-wise scores independently, without modelling the cross-layer cascade coupling.

2.3.4 Multiplex-Specific Heuristics

Effective Multiplex Degree (EMD) [6]. EMD defines a vertex score that accounts for the structural redundancy of its connections across layers. The impact value of the vertex i is the sum of the impact values of its neighbors, with the impact value depending on the weight and the degree of the multiplex. Formally, the impact is computed iteratively from

the message-passing equations of the interdependent percolation theory, making EMD the most principled heuristic specifically designed for multiplex dismantling.

CoreHLDA [27]. CoreHLDA combines network recycling, tree-breaking, and k-core-based score-refining processes. It applies a novel score fusion method in which single-layer dismantling scores are combined across layers via a weighted aggregation guided by the layer-wise k-core decomposition. CoreHLDA explicitly accounts for the heterogeneity of layer-specific core structures.

2.3.5 Extension to Directed Networks

Traditional heuristics such as HDA and CI are designed for undirected graphs and fail to capture directional cascade patterns. The *Directed CI* (DCI) heuristic [36] adapts Equation (2.12) to directed graphs by restricting the ℓ-hop ball to the out-reachable frontier:

$$\mathrm{DCI}_\ell(i) \;=\; (k_i^{\mathrm{out}} - 1) \sum_{j \in \partial \mathrm{Ball}^+(i,\ell)} (k_j^{\mathrm{out}} - 1), \tag{2.14}$$

where $\partial \mathrm{Ball}^+(i, \ell)$ is the out-reachable frontier at distance ℓ. DCI and HDA (extended to directed layers) are the primary baselines for DDIN in Chapter 4.

2.3.6 Fundamental Limitations of Heuristics

All heuristics reviewed above share three fundamental limitations that motivate the deep learning approaches developed in this thesis. We name and label them explicitly so that Chapters 4 and 5 can refer back to them without restating the motivation:

(L1) Locality. Heuristic scores are computed from at most ℓ-hop neighbourhoods. Long-range structural information — in particular, the inter-layer coupling patterns that determine cascade fragility — is not accessible to local rules. This limitation is especially severe in the directed setting, where the asymmetric out-reachable frontier $\partial \mathrm{Ball}^+(i, \ell)$ can differ radically from its undirected counterpart, making local scores systematically misleading.

(L2) Stationarity. Scores are recomputed from scratch at each dismantling step without learning from the history of past removals. Delayed cascade effects — where removing a low-scoring vertex today dramatically accelerates LMCC collapse later — cannot be anticipated by any heuristic that evaluates each step independently.

(L3) Independence across layers. The maximum-score rule aggregates single-layer scores by taking the per-layer maximum; it does not model the *joint* cross-layer dependencies that determine the cascading failure dynamics formalised in Equation (2.5). A vertex critical to the LMCC because of its simultaneous position in the k-cores of multiple layers receives no additional score credit relative to a vertex critical in only one layer.

We refer to these as limitations **L1**–**L3** throughout the thesis. DDIN (Chapter 4) addresses **L1** and **L3** in the directed multiplex setting by replacing static heuristic scoring with a learned asymmetric GNN policy that captures directed cross-layer dependencies. KARL (Chapter 5)

addresses all three by using a KAN-based GNN-DRL agent whose learnable non-linear activations capture long-range cascade patterns through multi-step temporal-difference learning.

2.4 Deep Reinforcement Learning for Graphs

2.4.1 Markov Decision Process Formalism

Definition 2.4.1 (Markov Decision Process [54])**.** A *Markov Decision Process* (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in [0, 1)$ is the discount factor.

The agent learns a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return:

$$J(\pi) \;=\; \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)\right]. \tag{2.15}$$

The optimal *action-value function* $Q^*(s, a)$ satisfies the *Bellman optimality equation* [62]:

$$Q^*(s, a) \;=\; \mathcal{R}(s, a) \;+\; \gamma \sum_{s'} \mathcal{T}(s'|s, a) \max_{a'} Q^*(s', a'), \tag{2.16}$$

and the optimal policy is $\pi^*(s) = \arg\max_a Q^*(s, a)$.

2.4.2 Deep Q-Networks

Definition 2.4.2 (Deep Q-Network [42])**.** A *Deep Q-Network* (DQN) approximates $Q^*(s, a)$ with a neural network $Q(s, a; \theta)$ parameterised by θ. The parameters are updated by minimising the mean-squared Bellman residual over a replay buffer $\mathcal{B}$:

$$\mathcal{L}_{\text{DQN}}(\theta) \;=\; \mathbb{E}_{(s,a,r,s') \sim \mathcal{U}(\mathcal{B})}\left[\left(r + \gamma \max_{a'} \hat{Q}(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right], \tag{2.17}$$

where $\hat{Q}(\cdot; \theta^-)$ is a *target network* with parameters θ^- copied from θ periodically (every C gradient steps), and $\mathcal{U}(\mathcal{B})$ denotes uniform sampling from the replay buffer.

Two key stabilisation techniques are employed: (i) *experience replay* [42], which breaks temporal correlations by storing and replaying past transitions (s, a, r, s'); and (ii) the *target network*, which provides stable regression targets by delaying parameter updates.

***n*-Step Returns.** To capture delayed reward effects — particularly important for dismantling, where removing a non-critical node today may unlock a cascade tomorrow — the n-step TD target is used:

$$y_t \;=\; \sum_{j=0}^{n-1} \gamma^j r_{t+j} \;+\; \gamma^n \max_{a'} \hat{Q}(s_{t+n}, a'; \theta^-), \tag{2.18}$$

replacing the one-step target $r + \gamma \max_{a'} \hat{Q}(s', a'; \theta^-)$ in Equation (2.17). Both DDIN and KARL use $n = 5$-step returns.

Prioritised Experience Replay. Standard DQN samples transitions uniformly from $\mathcal{B}$. *Prioritised Experience Replay* (PER) [50] assigns higher sampling probability to transitions with large temporal difference (TD) error, accelerating learning from informative experiences. The sampling probability of transition i is:

$$P(i) = \frac{p_i^{\alpha_{\text{PER}}}}{\sum_k p_k^{\alpha_{\text{PER}}}}, \tag{2.19}$$

where $p_i = |\delta_i| + \epsilon_{\text{PER}}$ is the priority (based on TD error δ_i), and $\alpha_{\text{PER}} \in [0, 1]$ controls the degree of prioritisation ($\alpha = 0$: uniform sampling). The induced sampling bias is corrected by importance-sampling weights:

$$w_i = \left(\frac{1}{N \cdot P(i)}\right)^{\beta_{\text{IS}}} \Big/ \max_j w_j, \tag{2.20}$$

where β_{IS} is annealed from 0.4 to 1.0 over training to ensure asymptotic unbiasedness.

2.4.3 GNN-DRL Paradigm for Network Dismantling

The integration of Graph Neural Networks with DQN for combinatorial optimisation on graphs was pioneered by Khalil et al. [29] (S2V-DQN), which showed that node embeddings from a structure-to-vector GNN can serve as action representations in Q-learning. This paradigm was applied to single-layer network dismantling by the following sequence of works.

FINDER [23]. FINDER was the first method to apply GNN-DRL to single-layer network dismantling without requiring supervised labels from exact solutions. It uses a graph convolutional encoder to produce node embeddings and trains a DQN agent to select nodes for removal. Virtual nodes are introduced as state representations, providing a global receptive field. FINDER trains on small Barabási-Albert graphs and generalises zero-shot to large real-world networks.

NIRM [66]. The *Neural Influence Ranking Model* (NIRM) extends FINDER with local and global scoring mechanisms, eliminating the need for handcrafted input features. NIRM trains on small networks (20–30 nodes) without any node feature engineering, relying entirely on structure.

GDM [24]. Grassia et al. proposed a machine-learning dismantler (GDM) that learns dismantling policies from exact solutions on small graphs, achieving early-warning capabilities for imminent network disintegration.

2.4.4 MultiDismantler: The State-of-the-Art Baseline

MultiDismantler [25] is the first learning-based algorithm specifically designed for dismantling one-to-one interdependent multiplex networks. It establishes the current state of the art and serves as the primary baseline for both contributions of this thesis. We describe its architecture in full detail, as Chapters 4 and 5 propose targeted improvements to each component.

Multiplex Graph Neural Network (MGNN) Encoder

The encoder consists of (i) an intralayer graph convolutional network and (ii) a node-level interlayer attention mechanism.

Node feature initialisation. For each layer ℓ, the initial feature of vertex v is computed from its normalised degree:

$$\boldsymbol{h}_v^{0,(\ell)} \;=\; \mathrm{Norm}\Big(\mathrm{ReLU}\Big(\boldsymbol{W}_1 \cdot \boldsymbol{X}_v^{(\ell)}\Big)\Big), \tag{2.21}$$

where $\boldsymbol{X}_v^{(\ell)} = k_v^{(\ell)} / k_{\max}^{(\ell)}$ is the degree of v normalised by the maximum layer degree, $\boldsymbol{W}_1$ is a trainable weight matrix, and $\mathrm{Norm}(\cdot)$ denotes ℓ_2-normalisation.

Intralayer GCN update. Node embeddings are iteratively refined over K message-passing rounds:

$$\boldsymbol{h}_v^{k,(\ell)} \;\leftarrow\; \mathrm{Norm}\Bigg(\mathrm{ReLU}\Bigg(\boldsymbol{W}_4 \cdot \Bigg(\Big\{\boldsymbol{W}_3 \cdot \boldsymbol{h}_v^{k-1,(\ell)}\Big\} \cup \Bigg\{\boldsymbol{W}_2 \cdot \sum_{j \in \mathcal{N}^{(\ell)}(v)} \boldsymbol{h}_j^{k-1,(\ell)}\Bigg\}\Bigg)\Bigg)\Bigg), \tag{2.22}$$

where $\boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{W}_4$ are trainable matrices, $\cup$ denotes concatenation, and $\mathcal{N}^{(\ell)}(v)$ is the neighbourhood of v in layer ℓ. The final intralayer representation is $\boldsymbol{h}_v^{(\ell)} = \boldsymbol{h}_v^{K,(\ell)}$.

Cross-layer alignment. Before interlayer attention, embeddings are projected into a shared latent space using a tanh transformation:

$$\tilde{\boldsymbol{h}}_v^{(\ell)} \;=\; \tanh\Big(\boldsymbol{W}_5 \cdot \boldsymbol{h}_v^{(\ell)} + \boldsymbol{b}_1\Big), \tag{2.23}$$

where $\boldsymbol{W}_5$ is a trainable matrix and $\boldsymbol{b}_1$ a trainable bias.

Node-level interlayer attention. The attention weight $\alpha_v^{(\ell \leftarrow q)}$ quantifies the influence of layer q on layer ℓ for vertex v, computed via the *Hadamard product* of the two layer-specific embeddings:

$$\alpha_v^{(\ell \leftarrow q)} \;=\; \frac{\exp\Big(\boldsymbol{W}_6 \cdot \Big(\tilde{\boldsymbol{h}}_v^{(\ell)} \odot \tilde{\boldsymbol{h}}_v^{(q)}\Big) + \boldsymbol{b}_2\Big)}{\sum_{p=1}^{L} \exp\Big(\boldsymbol{W}_6 \cdot \Big(\tilde{\boldsymbol{h}}_v^{(\ell)} \odot \tilde{\boldsymbol{h}}_v^{(p)}\Big) + \boldsymbol{b}_2\Big)}, \tag{2.24}$$

where $\odot$ denotes the element-wise Hadamard product, $\boldsymbol{W}_6$ is a trainable matrix, and $\boldsymbol{b}_2$ a bias vector.

Final node embedding. The cross-layer attention values are used to aggregate representations from all other layers into a single final embedding:

$$\boldsymbol{z}_v^{(\ell)} \;=\; \tilde{\boldsymbol{h}}_v^{(\ell)} \;+\; \sum_{\substack{p=1 \\ p \neq \ell}}^{L} \alpha_v^{(\ell \leftarrow p)} \cdot \tilde{\boldsymbol{h}}_v^{(p)}. \tag{2.25}$$

Decoder and Q-Value Computation

Virtual-node state embedding. Each layer ℓ contains a *virtual node* s that is directed to all real nodes but has no in-neighbours, providing a global summary of the layer state. Its embedding $\boldsymbol{z}_s^{(\ell)}$ is computed by the same MGNN encoder.

Layer-specific Q-value. The expected reward of removing vertex v under the state of layer ℓ is:

$$Q^{(\ell)}(s, a_v) \;=\; \boldsymbol{M}_1 \cdot \mathrm{ReLU}\Big(\boldsymbol{z}_s^{(\ell)} \times \boldsymbol{z}_v^{(\ell)} \cdot \boldsymbol{M}_2\Big), \tag{2.26}$$

where $\boldsymbol{M}_1, \boldsymbol{M}_2$ are learnable weight matrices and $\times$ denotes the outer product, which models fine-grained state-action dependencies [23].

Layer importance weighting. The contribution of each layer to the final Q-value is determined by a learned softmax-weighted combination:

$$\omega^{(\ell)} \;=\; \frac{\exp\Big(\boldsymbol{M}_4 \cdot \mathrm{ReLU}\Big(\boldsymbol{M}_3 \cdot \boldsymbol{z}_s^{(\ell)}\Big)\Big)}{\sum_{p=1}^{L} \exp\Big(\boldsymbol{M}_4 \cdot \mathrm{ReLU}\Big(\boldsymbol{M}_3 \cdot \boldsymbol{z}_s^{(p)}\Big)\Big)}, \tag{2.27}$$

where $\boldsymbol{M}_3, \boldsymbol{M}_4$ are trainable matrices.

Final Q-value. The global action value is the weighted average of layer-specific values:

$$Q(s, a_v) \;=\; \sum_{p=1}^{L} \omega^{(p)} \cdot Q^{(p)}(s, a_v). \tag{2.28}$$

Reward and Loss

Immediate reward. The reward for removing vertex v at step t is:

$$r(v) \;=\; P_\infty(S_t) \cdot F(\{v\}), \tag{2.29}$$

where $P_\infty(S_t)$ is the normalised LMCC size after the cascading failure resolution triggered by removing v, and $F(\{v\})$ is the per-vertex cost (1 for unit cost; $\sum_\ell k_v^{(\ell)}$ for degree cost).

Total loss. MultiDismantler optimises a composite loss combining the n-step Q-learning loss and a *graph reconstruction loss*:

$$\mathcal{L}_{\mathrm{Q}}(\theta) \;=\; \mathbb{E}_{(s_t, a_t, \ldots)\sim\mathcal{U}(\mathcal{B})}\left[\left(y_t - Q(s_t, a_t; \theta)\right)^2\right], \tag{2.30}$$

$$\mathcal{L}_{\mathrm{R}}(\theta) \;=\; \sum_{p=1}^{L} \sum_{i,j=1}^{N} A_{ij}^{(p)} \left\| \boldsymbol{z}_i^{(p)} - \boldsymbol{z}_j^{(p)} \right\|_2^2, \tag{2.31}$$

$$\mathcal{L}_{\mathrm{total}}(\theta) \;=\; \mathcal{L}_{\mathrm{Q}}(\theta) + \beta\, \mathcal{L}_{\mathrm{R}}(\theta), \tag{2.32}$$

where β balances the two objectives. The reconstruction loss $\mathcal{L}_{\mathrm{R}}$ encourages structurally adjacent nodes to have similar embeddings, regularising the encoder against learning representations that ignore topology.

Remark 2.4.3 (Two Limitations of MultiDismantler). Every learnable map in MultiDismantler — both in the MGNN encoder (Equations (2.21)–(2.22)) and the MLP decoder (Equations (2.26)–(2.27)) — is an affine transformation composed with a fixed nonlinearity. This bounds per-edge representational capacity to a fixed-degree polynomial, constituting the **Functional Limitation** addressed by KARL in Chapter 5. Note that this is distinct from the heuristic limitations **L1**–**L3** of Section 2.3.6: MultiDismantler already overcomes **L1**–**L3** by learning a multi-step RL policy over a multiplex GNN encoder; the Functional Limitation is a ceiling on the *expressivity* of that encoder and decoder, not on the learning paradigm itself. Additionally, MultiDismantler assumes undirected, symmetric intra-layer edges ($\boldsymbol{A}^{(\ell)} = (\boldsymbol{A}^{(\ell)})^\top$), making it inapplicable to directed interdependent networks; this is the **Topological Limitation** addressed by DDIN in Chapter 4. The formal mathematical analysis of both limitations is deferred to the respective contribution chapters.

2.5 Graph Neural Networks

2.5.1 The Message-Passing Framework

The general *message-passing neural network* (MPNN) framework [64] describes a broad class of GNN architectures by two operations applied iteratively. At layer k, each vertex v aggregates messages from its neighbours and then updates its own embedding:

$$\boldsymbol{m}_v^{(k)} = \text{AGGREGATE}^{(k)}\big(\{\boldsymbol{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}\big), \tag{2.33}$$

$$\boldsymbol{h}_v^{(k)} = \text{COMBINE}^{(k)}\big(\boldsymbol{h}_v^{(k-1)},\ \boldsymbol{m}_v^{(k)}\big). \tag{2.34}$$

Different choices of AGGREGATE and COMBINE yield different GNN architectures. After K message-passing iterations, the vertex embedding $\boldsymbol{h}_v^{(K)}$ encodes structural information from within K hops of v.

Theorem 2.5.1 (Expressive Power of Message Passing [64])**.** *A GNN with* AGGREGATE *and* COMBINE *implemented by injective multiset functions is at most as powerful as the Weisfeiler-Lehman (WL) graph isomorphism test in distinguishing non-isomorphic graphs. The Graph Isomorphism Network (GIN), which uses sum aggregation, achieves this upper bound.*

2.5.2 Graph Convolutional Networks (GCN)

Definition 2.5.2 (Graph Convolutional Network [30])**.** The Graph Convolutional Network (GCN) defines the update rule:

$$\boldsymbol{H}^{(k+1)} = \sigma\Big(\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}\boldsymbol{H}^{(k)}\boldsymbol{W}^{(k)}\Big), \tag{2.35}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ is the adjacency matrix with self-loops, $\tilde{\boldsymbol{D}}_{ii} = \sum_j \tilde{A}_{ij}$ is the corresponding degree matrix, $\boldsymbol{W}^{(k)}$ is a learnable weight matrix, and σ is a nonlinearity.

The symmetric normalisation $\tilde{\boldsymbol{D}}^{-1/2}\tilde{\boldsymbol{A}}\tilde{\boldsymbol{D}}^{-1/2}$ implements a degree-normalised mean aggregation with a self-loop. GCN forms the basis of MultiDismantler's intralayer encoder (Equation (2.22)).

2.5.3 GraphSAGE: Inductive Representation Learning

Definition 2.5.3 (GraphSAGE [26])**.** GraphSAGE uses a two-stage update: aggregation over the neighbourhood followed by a concatenation-and-projection combine:

$$\boldsymbol{h}_v^{(k)} \;=\; \sigma\Big(\boldsymbol{W}^{(k)} \cdot \mathrm{CONCAT}\Big(\boldsymbol{h}_v^{(k-1)},\; \mathrm{AGG}_k\big(\{\boldsymbol{h}_u^{(k-1)} : u \in \mathcal{N}(v)\}\big)\Big)\Big), \tag{2.36}$$

where AGG_k is a permutation-invariant function (mean, max, or LSTM pooling).

GraphSAGE's inductive nature — generating embeddings from local neighbourhoods without reference to the global graph — is critical for the zero-shot generalisation required in dismantling: the model trained on 30–50 node synthetic graphs must apply without retraining to 56,562-node real networks.

Asymmetric GraphSAGE for Directed Networks

Standard GraphSAGE applies a single aggregation over the symmetric neighbourhood $\mathcal{N}(v)$. For directed graphs, this loses the directional flow information encoded in $\mathcal{N}^+(v)$ and $\mathcal{N}^-(v)$. DDIN (Chapter 4) introduces an *asymmetric GraphSAGE* encoder that separates in-neighbourhood and out-neighbourhood aggregations:

$$\boldsymbol{h}_v^{\ell} \;=\; \mathrm{ReLU}\Big(\boldsymbol{W}_{\mathrm{out}}\big[\boldsymbol{x}_v;\; \mathrm{MeanAgg}(\{\boldsymbol{h}_u^{\ell} : u \to v\})\big] \;+\; \boldsymbol{W}_{\mathrm{in}}\big[\boldsymbol{x}_v;\; \mathrm{MaxAgg}(\{\boldsymbol{h}_u^{\ell} : v \to u\})\big]\Big), \tag{2.37}$$

where $\boldsymbol{W}_{\mathrm{out}}, \boldsymbol{W}_{\mathrm{in}} \in \mathbb{R}^{d \times 2d}$ are independent learnable weight matrices and $[\cdot\,;\cdot]$ denotes concatenation. MeanAgg over in-neighbours captures the aggregate upstream influence received by v, while MaxAgg over out-neighbours captures the dominant downstream target controlled by v. This design is motivated by the directional semantics of biological and economic networks: removing a high-out-degree gene disrupts many downstream regulatory targets, while removing a high-in-degree gene eliminates a convergence point for upstream regulatory signals.

Remark 2.5.4 (Semantic Rationale for Asymmetric Aggregation). The choice of *mean* pooling for in-neighbours and *max* pooling for out-neighbours is not arbitrary but is grounded in the directional information content of each neighbourhood type. In a gene regulatory network, $\mathcal{N}^-(v)$ consists of all upstream regulators of gene v: their collective effect on v is naturally summarised by an *average* signal, since v integrates inputs from all of them. Conversely, $\mathcal{N}^+(v)$ consists of all genes controlled by v: the dismantling value of removing v is determined by the *most critical* downstream gene it governs, making max pooling the semantically appropriate aggregator. In FAO Trade networks (364 layers), the same asymmetry applies: an exporting country's trade impact is best summarised by its largest single export relationship (max over out-neighbours), whereas an importing country's vulnerability is captured by the average quality of its supply sources (mean over in-neighbours). A symmetric aggregation over $\mathcal{N}(v) = \mathcal{N}^-(v) \cup \mathcal{N}^+(v)$ conflates these two distinct signals and systematically underestimates the cascade-initiating potential of high-out-degree hubs.

2.5.4 Graph Attention Networks (GAT)

Definition 2.5.5 (Graph Attention Network [58])**.** GAT replaces fixed neighbourhood aggregation weights with learned attention coefficients. For vertex v attending to neighbour u:

$$e_{vu} = \text{LeakyReLU}\big(\boldsymbol{a}^\top \big[\boldsymbol{W}\boldsymbol{h}_v \parallel \boldsymbol{W}\boldsymbol{h}_u\big]\big), \tag{2.38}$$

$$\alpha_{vu} = \text{softmax}_u(e_{vu}) = \frac{\exp(e_{vu})}{\sum_{w\in\mathcal{N}(v)}\exp(e_{vw})}, \tag{2.39}$$

$$\boldsymbol{h}'_v = \sigma\left(\sum_{u\in\mathcal{N}(v)} \alpha_{vu}\boldsymbol{W}\boldsymbol{h}_u\right). \tag{2.40}$$

Multi-head attention is typically used: H independent attention mechanisms are computed and their outputs concatenated or averaged.

2.5.5 Multi-Relational Attention for Interdependent Networks

In multiplex dismantling, the *inter-layer* dependencies require cross-graph attention beyond the single-graph GAT formulation. DDIN employs a *multi-relational attention layer* that computes directional cross-layer attention for each target layer ℓ:

$$\alpha_{\ell\ell'} = \text{softmax}\Big(\text{LeakyReLU}\Big(\boldsymbol{a}^\top \big[\boldsymbol{W}\boldsymbol{h}_v^{\ell} \parallel \boldsymbol{W}\boldsymbol{h}_v^{\ell'}\big]\Big)\Big), \tag{2.41}$$

yielding the fused embedding $\tilde{\boldsymbol{h}}_v = \sum_{\ell'\neq\ell} \alpha_{\ell\ell'}\boldsymbol{h}_v^{\ell'}$. This extends the Heterogeneous Graph Transformer [28] and muxGNN [40] to the directed interdependent multiplex setting.

2.5.6 Over-Smoothing and Depth Limitations

A well-known failure mode of deep GNNs is *over-smoothing* [64]: repeated neighbourhood averaging drives all vertex embeddings toward the same mean vector, destroying local structural information. MultiDismantler, DDIN, and KARL all apply ℓ_2-normalisation after each message-passing step to constrain embedding norms and mitigate over-smoothing. KARL additionally employs a macroscopic residual branch (Section 5.1.7) that provides a skip connection bypassing the neighbourhood aggregation, offering a complementary mechanism against embedding collapse.

2.6 Kolmogorov-Arnold Networks

2.6.1 The Kolmogorov-Arnold Representation Theorem

Theorem 2.6.1 (Kolmogorov-Arnold Representation [35])**.** *Every multivariate continuous function $f : \mathbb{R}^n \to \mathbb{R}$ can be represented as a superposition of uni-variate functions:*

$$f(x_1,\ldots,x_n) = \sum_{q=1}^{2n+1} \Phi_q\left(\sum_{p=1}^{n} \varphi_{q,p}(x_p)\right), \tag{2.42}$$

where $\Phi_q : \mathbb{R} \to \mathbb{R}$ and $\varphi_{q,p} : \mathbb{R} \to \mathbb{R}$ are continuous univariate functions.

The KA theorem differs fundamentally from the universal approximation theorem for MLPs [5], which guarantees that a sufficiently wide single hidden layer can approximate any

continuous function. The MLP theorem requires width growing with the target function's complexity, while the KA theorem guarantees exact representation with a fixed two-layer structure, at the cost of requiring the univariate functions to be adapted to the target.

2.6.2 Kolmogorov-Arnold Networks (KANs)

Motivated by Theorem 2.6.1, Liu et al. [35] proposed *Kolmogorov-Arnold Networks* (KANs) as a learnable alternative to MLPs. Whereas an MLP applies a fixed nonlinearity σ to learned affine combinations, a KAN places *learnable univariate functions* directly on network edges:

Definition 2.6.2 (KAN Layer [35])**.** A d_{in} input dimensional KAN layer with output dimension d_{out} computes:

$$\text{KAN}(\boldsymbol{x})_j \;=\; \sum_{i=1}^{d_{\text{in}}} \varphi_{j,i}(x_i), \quad j = 1, \ldots, d_{\text{out}}, \tag{2.43}$$

where each $\varphi_{j,i} : \mathbb{R} \to \mathbb{R}$ is a learnable univariate function, typically parameterised as a B-spline.

The key distinction from an MLP layer $\boldsymbol{h} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$ is that MLP applies a *single global weight* W_{ji} to input coordinate x_i, whereas KAN adapts the *functional form* $\varphi_{j,i}$ of the transformation per edge. This gives KANs a strictly larger per-edge function family at any fixed parameter budget (Proposition 5.4.9 in Chapter 5).

2.6.3 B-Spline Parameterisation

Following Liu et al. [35], each KAN edge function is parameterised as a B-spline:

$$\varphi(x) \;=\; w\left(b(x) + \sum_{i=1}^{G+k} c_i\, B_i^k(x; G)\right), \tag{2.44}$$

where w is a learnable scaling weight, $b(x) = \text{SiLU}(x)$ is a base activation, c_i are trainable spline coefficients, G is the grid resolution (number of knot intervals), k is the spline order, and $B_i^k(x; G)$ are the B-spline basis functions defined by the Cox–de Boor recurrence.

B-Spline Basis Functions and the Cox-de Boor Recurrence

Given a uniform knot vector $\mathbf{t} = (t_0 \le t_1 \le \cdots \le t_{G+2k})$ with k-fold boundary knots on $[-1, 1]$, the B-spline basis functions of order k are defined recursively as [14]:

$$B_i^0(x) \;=\; \mathbf{1}[t_i \le x < t_{i+1}], \tag{2.45}$$

$$B_i^k(x) \;=\; \frac{x - t_i}{t_{i+k} - t_i}\, B_i^{k-1}(x) \;+\; \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}}\, B_{i+1}^{k-1}(x), \tag{2.46}$$

where $\mathbf{1}[\cdot]$ is the indicator function. B-splines of order k are piecewise polynomials of degree $k - 1$ with C^{k-2} continuity at each interior knot. In particular, cubic B-splines ($k = 3$) are C^2-continuous, ensuring smooth gradient propagation through the spline grid during backpropagation — a property that is critical for the stability of KARL's encoder (Proposition 5.4.5).

Approximation quality. By the classical B-spline approximation theorem [14], a uniform B-spline of order k on a grid of G intervals approximates any C^{k+1} function f on a compact domain to:

$$\|f - S_G f\|_\infty \;\leq\; C_k \cdot G^{-(k+1)} \left\| f^{(k+1)} \right\|_\infty, \tag{2.47}$$

where C_k depends only on the spline order. This $O(G^{-(k+1)})$ algebraic convergence rate motivates the grid resolution ablation in Section 5.6.2: larger G increases approximation capacity but also risks overfitting the training distribution.

2.6.4 The EfficientKAN Vectorised Implementation

Naive evaluation of $N_{\text{active}} \cdot d_{\text{in}}$ independent univariate functions is computationally prohibitive. The *EfficientKAN* implementation [35] vectorises the computation as:

$$\text{KAN}(\boldsymbol{x}) \;=\; \boldsymbol{W}_{\text{base}}\, \sigma(\boldsymbol{x}) \;+\; \boldsymbol{W}_{\text{spline}}\, \boldsymbol{B}(\boldsymbol{x}), \tag{2.48}$$

where $\sigma(\cdot)$ is the SiLU base activation, $\boldsymbol{B}(\boldsymbol{x}) \in \mathbb{R}^{d_{\text{in}}(G+k)}$ stacks all B-spline basis evaluations, $\boldsymbol{W}_{\text{base}} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ is the base weight matrix, and $\boldsymbol{W}_{\text{spline}} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}(G+k)}$ are the spline control weights. The per-layer parameter count is $d_{\text{out}} \cdot d_{\text{in}} \cdot (G + k + 1)$, compared to $d_{\text{out}} \cdot d_{\text{in}}$ for an MLP — a factor of $(G + k + 1)$ overhead per layer ($14\times$ for $G = 10, k = 3$, as used in KARL).

2.6.5 Gradient Instability of KANs in Off-Policy RL

Embedding KAN layers inside an off-policy RL loop on scale-free graphs introduces instability modes that are absent in the standard supervised learning setting for which KANs were originally designed [35]. Two mechanisms are particularly relevant to the architecture of KARL.

The first is *non-stationary target-induced oscillation*. The B-spline basis relies on a fixed knot grid $\mathbf{t}$ over the input domain $[-1, 1]$. In off-policy Q-learning, the regression target y_t (Equation (2.18)) changes at every gradient step as the policy improves. As the input distribution drifts, certain regions of the spline grid receive disproportionately dense activations while others starve, producing boundary oscillations analogous to the Runge phenomenon discussed in Section 2.6.6.

The second is *hub-node gradient explosion*. Real-world multiplex networks are scale-free: hub nodes with very high degree produce aggregated neighbourhood features of large ℓ_2-mass, pushing post-tanh activations toward the domain boundary where the B-spline derivative is largest. Unrolled over multiple message-passing iterations, this induces a worst-case gradient norm that grows polynomially in the hub degree — an interaction between the graph topology and the spline parameterisation that is unique to graph-domain RL.

The formal quantification of both instability modes — including a worst-case gradient norm bound parameterised by the graph branching factor $\beta_{\max}$ and the B-spline Lipschitz constant L_φ — is deferred to Chapter 5 where the specific architectural solutions (the macroscopic residual branch and the orthogonal Chebyshev Q-head) are presented alongside their theoretical justifications (Proposition 5.4.1 and Proposition 5.4.5).

2.6.6 The Runge Phenomenon and Chebyshev Polynomials

For polynomial interpolation of degree K through $K+1$ equispaced nodes, the *Lebesgue constant* governs the interpolation error:

$$\Lambda_K \;=\; \sup_{x\in[-1,1]} \sum_{j=0}^{K} |\ell_j(x)| \;=\; \Theta\!\left(\frac{2^K}{\sqrt{K}}\right), \tag{2.49}$$

where ℓ_j are the Lagrange basis polynomials [56]. The exponential growth $\Lambda_K = \Theta(2^K)$ implies that the approximation error bound $(1+\Lambda_K)E_K(y_t)$ diverges exponentially in K, producing the *Runge phenomenon*: spurious oscillations near the domain boundary that amplify the non-stationarity of TD targets.

The Chebyshev polynomial basis $\{T_k\}_{k=0}^{K}$ satisfies the minimax property: among all polynomials of degree K with leading coefficient 2^{1-K}, the Chebyshev polynomial $T_K(x) = \cos(K \arccos x)$ has the smallest sup-norm on $[-1,1]$. The truncated Chebyshev projection $S_K[f]$ achieves:

$$\|f - S_K[f]\|_\infty \;\le\; \left(1 + \frac{2}{\pi}\ln(K+1) + O(K^{-2})\right) E_K(f), \tag{2.50}$$

where $E_K(f) = \min_{P\in\mathcal{P}_K} \|f-P\|_\infty$ is the best-approximation error [49]. The logarithmic growth $O(\ln K)$ of the multiplicative constant — compared to the exponential $\Theta(2^K)$ Lebesgue constant of equispaced interpolation — makes Chebyshev projection far more stable for approximating the non-stationary TD targets encountered in off-policy RL. This approximation-theoretic argument directly motivates the orthogonal Chebyshev Q-head in KARL (Section 5.3).

Remark 2.6.3 (Architectural Foreshadowing: B-Splines in the Encoder, Chebyshev in the Q-Head). The background established in this section directly determines the architecture of KARL, presented in Chapter 5. **B-spline KANs** are used for *intra-layer message passing* (the encoder) and for the *cross-layer attention projections* (the KANformer), because the encoder's task is representational expressivity: capturing high-frequency, non-smooth per-node vulnerability patterns that fixed affine maps cannot resolve. For this role the $O(G^{-(k+1)})$ algebraic approximation rate of B-splines is appropriate, and the residual branch (Section 5.1.7) is introduced to manage the gradient instability described in Section 2.6.5. **Chebyshev polynomial KANs** are used exclusively for *action-value regression* (the Q-head), because this component must regress non-stationary TD targets where the stability of the approximator is paramount. The logarithmic operator-norm bound of the Chebyshev projection (Equation (2.50)) provides the near-minimax guarantee that B-splines — susceptible to the Runge phenomenon under non-stationary input distributions — cannot offer. These two roles require different function families: expressivity for the encoder, stability for the decoder.

2.6.7 Related KAN Architectures

Res-KAN [52]. Siddiqui and Elmi proposed integrating KAN layers with ResNet-style skip connections for human action recognition. This architecture operates in a supervised learning setting on Euclidean data and does not address the RL-specific gradient instability

of deep B-spline layers on graph-structured inputs. KARL's macroscopic residual branch (Equation (5.8)) is conceptually related but specifically designed to stabilise gradient propagation through scale-free multiplex aggregators.

KANG [34]. Carlo et al. introduced Kolmogorov-Arnold Graph Neural Networks, which replace MLP aggregation functions in GNNs with B-spline KANs. KANG demonstrated that KAN-based aggregators capture localised, high-frequency structural motifs without catastrophic oversmoothing on graph classification tasks. However, KANG operates in the static supervised setting; KARL is the first work to embed a residual Graph KAN within an off-policy RL loop for combinatorial optimisation.

Kolmogorov-Arnold Transformer (KAT) [65]. Yang and Wang replaced the MLP components of transformer attention blocks with KAN layers, achieving improved performance on image classification and language tasks. The KANformer module in KARL (Section 5.2) applies an analogous substitution to the Q/K/V projections of the cross-layer attention mechanism, with the additional challenge that the transformer operates over the non-stationary node embedding space of a graph RL agent.

2.7 Summary of Research Gaps

The preceding sections have characterised the state of the art in network dismantling, deep reinforcement learning on graphs, graph neural networks, and Kolmogorov-Arnold Networks. Table 2.1 provides a structured summary of the two research gaps that this thesis addresses, mapping each gap to the prior methods it affects and the contribution that resolves it.

Two points of clarification are important for the thesis narrative.

First, the two gaps are *orthogonal*: addressing the topological limitation (directed graphs, DDIN) does not address the functional limitation (expressivity, KARL), and vice versa. A directed multiplex network can still suffer from fixed-affine functional capacity; an undirected multiplex network does not benefit from asymmetric aggregation.

Second, the two limitations are not additive in the sense that combining the fixes would yield the sum of their improvements. The interaction between directed aggregation and KAN-based functional expressivity is the open synthesis direction identified in Section 7.3.1. The empirical contributions of this thesis — 16–23% AUDC reduction (DDIN) and 21.96% AUDC reduction (KARL) over their respective baselines — represent independent evaluations of each fix in its own structural setting. The empirical results in Chapters 4 and 5 therefore constitute independent validations of each architectural fix, and the performance gains reported therein cannot be attributed to a shared confound.

The remaining dissertation is organized as follows. Chapter 3 establishes the shared mathematical framework — the network generation models, MDP formulation, cost regimes, and evaluation metrics — used by both contributions. Chapter 4 presents DDIN, the first RL agent for directed interdependent multiplex dismantling. Chapter 5 presents KARL, the first

Table 2.1: Summary of research gaps addressed in this thesis.

Gap	Root Cause	Methods Affected	Addressed By
Topological Limitation: Directed intra-layer edges and asymmetric cascade propagation are not modelled.	All existing GNN-DRL agents use symmetric neighbourhood aggregation ($\mathcal{N}(v)$ without separation of $\mathcal{N}^+(v)$ and $\mathcal{N}^-(v)$). Connectivity criterion MCC fails to capture directed mutual reachability.	FINDER, NIRM, GDM, MultiDismantler, and all heuristics (HDA, CI, MinSum, EMD, CoreHLDA) when applied to directed multiplex networks.	**DDIN** (Chapter 4)
Functional Limitation: Fixed affine MLP projections cannot represent the high-frequency, non-smooth per-node vulnerability functions governing cascading fragility.	Every learnable transformation in MultiDismantler's MGNN encoder and MLP decoder applies a global affine map $\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$ followed by a fixed nonlinearity. Per-edge capacity is bounded by a fixed-degree polynomial.	MultiDismantler (primary baseline), FINDER, NIRM, GDM — all GNN-DRL agents for undirected multiplex dismantling.	**KARL** (Chapter 5)

KAN-RL agent for multiplex network dismantling. Chapter 6 synthesises and compares the two contributions. Chapter 7 concludes with a discussion of future directions.

Chapter 3

Preliminaries and Mathematical Formulation

This chapter establishes the shared mathematical framework underpinning the two primary contributions of this thesis. Chapter 4 addresses the *directed* interdependent multiplex dismantling problem via the DDIN framework, and Chapter 5 addresses the *undirected* interdependent multiplex dismantling problem via the KARL framework. Both approaches share three design pillars: they train exclusively on synthetic multiplex networks generated by a parameterisable random graph model; they cast sequential node removal as a Markov Decision Process (MDP) solved by off-policy deep reinforcement learning; and they are evaluated by a common set of performance metrics on real-world benchmark datasets.

Consolidating the underlying mathematics in a single chapter serves a dual purpose. It keeps Chapters 4 and 5 focused on architectural innovations rather than on foundational derivations, and it makes explicit the precise points at which the two settings *diverge* – directed versus undirected intra-layer topology, MSCC versus LMCC as the critical component, asymmetric versus symmetric reward signals – so that the reader can appreciate each methodological choice against a shared baseline.

Section 3.1 develops the generative models used for synthetic training: the standard undirected Geometric Multiplex Model (GMM) employed by KARL, and the Directed Geometric Multiplex Model (D-GMM) introduced for DDIN. Section 3.2 formalises the dismantling MDP in both the undirected and directed settings, providing a rigorous account of the state space (including node features, virtual-node augmentation, and component-membership indicators), the action space and cascade-failure transition operator, and the reward functions (including the n-step return structure and prioritised experience replay). Section 3.3 gives precise definitions of all cost regimes (unit, degree, directed degree, and random cost), the two scalar performance metrics (AUDC and C^*) in their MDP-context formulations, and the non-parametric statistical testing protocol used to establish significance of the reported results.

Throughout the chapter, remarks flag the precise structural choices that motivate the architectural innovations presented in Chapters 4 and 5.

3.1 Network Generation Models

A defining methodological feature shared by DDIN and KARL is the use of *purely synthetic* training data. Both agents are trained exclusively on small randomly generated multiplex networks ($N \in [30, 50]$ nodes) and subsequently evaluated zero-shot on large real-world benchmarks reaching up to $N = 56{,}562$ nodes. This training regime is feasible because the generative model captures the salient structural properties – scale-free degree distributions, geometric inter-layer correlations, and cascading failure dynamics – that determine dismantling difficulty in practice.

The *Geometric Multiplex Model* (GMM) [32, 33] is the common training framework. Section 3.1.1 presents the standard undirected variant used by KARL; Section 3.1.2 derives

the directed extension (D-GMM) introduced for DDIN; Section 3.1.3 specifies the training corpus and held-out synthetic evaluation suites common to both contributions.

3.1.1 The Undirected Geometric Multiplex Model

The Geometric Multiplex Model constructs a multiplex network $\mathcal{M} = \langle V, \{E^{(\ell)}\}_{\ell=1}^{L} \rangle$ with L undirected layers by embedding N shared nodes in a common hyperbolic space and generating layer-specific edges via a proximity-based connection probability [32].

Hyperbolic disk embedding. Each node $i \in V$ is assigned a pair of geometric coordinates: an angular coordinate $\theta_i^{(1)} \in [0, 2\pi)$, drawn independently and uniformly, and a radial coordinate $r_i \in [0, R]$, determined by the node's expected degree. The disk radius is set to:

$$R \;=\; 2\ln\frac{N}{\sin(\pi T)\,\bar{\kappa}}, \tag{3.1}$$

where $T \in (0, 1)$ is a temperature parameter governing local clustering and $\bar{\kappa}$ is a normalisation constant derived from the expected mean degree $\bar{k}$ and the power-law exponent γ of the degree distribution [8].

Hidden degree variables. Each node $i \in V$ is assigned a *hidden degree variable* $\kappa_i > 0$ drawn independently from the Pareto distribution:

$$P(\kappa) \;=\; (\gamma - 1)\,\kappa_{\min}^{\gamma-1}\,\kappa^{-\gamma}, \qquad \kappa \;\geq\; \kappa_{\min} > 0, \tag{3.2}$$

where $\gamma > 2$ is the power-law exponent and $\kappa_{\min}$ is the minimum expected degree. The expected degree of node i in any individual layer satisfies $\mathbb{E}[k_i^{(\ell)}] \approx \kappa_i$, so the degree sequence of each generated layer follows a power law with exponent γ.

Radial coordinates. The radial coordinate of node i is determined from its hidden degree variable by the bijection:

$$r_i \;=\; R - \frac{2}{\beta}\ln\frac{\kappa_i}{\kappa_{\min}}, \quad \beta \;=\; \gamma - 1 > 1. \tag{3.3}$$

Nodes with large expected degree ($\kappa_i \gg \kappa_{\min}$) are placed near the disk centre ($r_i \approx 0$), while peripheral low-degree nodes are placed near the boundary ($r_i \approx R$). This embedding reproduces the hub-and-spoke topology of scale-free networks within a hyperbolic geometry.

Interlayer angular correlation. The interlayer similarity parameter $g \in [0, 1]$ measures the degree to which a node's angular position is preserved across layers. For layer $\ell \geq 2$, the angular coordinate of node i is:

$$\theta_i^{(\ell)} \;=\; \left[\, g\,\theta_i^{(1)} + (1 - g)\,\xi_i^{(\ell)} \,\right] \bmod 2\pi, \tag{3.4}$$

where $\xi_i^{(\ell)} \sim \text{Uniform}[0, 2\pi)$ is an independent angular perturbation for layer ℓ [32]. When $g = 1$, all layers share identical angular coordinates and exhibit identical community structure; when $g = 0$, the angular positions in each layer are completely independent. Real multiplex networks typically exhibit $g \in (0.2, 0.8)$ [32, 33].

Connection probability. The probability of an undirected edge $\{i, j\} \in E^{(\ell)}$ is determined by the Fermi-Dirac function of the hyperbolic distance between i and j in layer ℓ:

$$p_{ij}^{(\ell)} = \frac{1}{1 + \exp\left(\frac{d_{ij}^{(\ell)} - R}{2T}\right)}, \tag{3.5}$$

where the hyperbolic distance within the Poincaré disk model is:

$$d_{ij}^{(\ell)} = r_i + r_j + 2\,\mathrm{lnsin}\frac{\left|\theta_i^{(\ell)} - \theta_j^{(\ell)}\right|}{2}. \tag{3.6}$$

Edges $\{i, j\} \in E^{(\ell)}$ are drawn independently with probability $p_{ij}^{(\ell)}$ given by Equation (3.5). Node pairs geometrically close in the hyperbolic disk – either because both are hubs with small r_i, r_j, or because their angular positions are similar – are more likely to be connected. Since $p_{ij}^{(\ell)} = p_{ji}^{(\ell)}$, the resulting layer is undirected by construction.

Key structural properties. GMM-generated multiplex networks exhibit the following properties that have been empirically verified by Kleineberg et al. [32] and are directly relevant to the dismantling problem:

(P1) Scale-free degree distribution. Each layer exhibits a power-law degree distribution $P(k) \sim k^{-\gamma}$ for large k, with exponent matching the hidden degree exponent in Equation (3.2).

(P2) High clustering. Geometric correlations induce clustering coefficients substantially exceeding those in Erdős-Rényi graphs of the same density, consistent with empirical social and biological networks.

(P3) Tunable interlayer community overlap. The parameter g controls whether high-coreness nodes in one layer are also high-coreness nodes in other layers, a property that critically determines cascade fragility under the Buldyrev one-to-one interdependency model [11].

(P4) Non-trivial dismantling difficulty. In contrast to Barabási-Albert (BA) graphs, GMM networks combine geometric inter-layer structure with power-law degree distributions. Gu et al. [25] showed empirically that MultiDismantler trained on BA networks transfers poorly to GMM-based real-world networks, while training on GMM graphs yields strong zero-shot generalisation. This result motivates the adoption of GMM as the training corpus for both DDIN and KARL.

3.1.2 The Directed Geometric Multiplex Model (D-GMM)

The standard GMM produces undirected layers because the connection probability in Equation (3.5) is symmetric in i and j. Directed multiplex networks – modelling gene regulatory cascades, global supply chains, financial transaction flows, and social media influence propagation – require a generative model that produces *asymmetric* intra-layer

connectivity, where the probability of the arc $(i \to j)$ is substantially different from that of the reverse arc $(j \to i)$.

We introduce the *Directed Geometric Multiplex Model* (D-GMM) [20], an extension of the GMM that endows each node with *separate* hidden degree variables governing its out-degree and in-degree, and introduces a directional bias parameter that drives high-out-degree nodes to connect preferentially to high-in-degree nodes.

Separate hidden degree variables. Each node $i \in V$ is now assigned two independent hidden degree variables:

$$\kappa_i^{\text{out}} \sim P_{\text{out}}(\kappa) = (\gamma_{\text{out}} - 1)\, \kappa_{\min}^{\gamma_{\text{out}}-1}\, \kappa^{-\gamma_{\text{out}}}, \quad \kappa \geq \kappa_{\min}, \tag{3.7}$$

$$\kappa_i^{\text{in}} \sim P_{\text{in}}(\kappa) = (\gamma_{\text{in}} - 1)\, \kappa_{\min}^{\gamma_{\text{in}}-1}\, \kappa^{-\gamma_{\text{in}}}, \quad \kappa \geq \kappa_{\min}, \tag{3.8}$$

drawn independently for each node. The exponents γ_{out} and γ_{in} can differ, enabling the model to reproduce empirically observed discrepancies between in-degree and out-degree distributions in directed biological and economic networks [2]. By construction of the connection probability below, $\mathbb{E}[k_i^{(\ell),\text{out}}] = \kappa_i^{\text{out}}$ exactly (out-degree is calibrated by design), and $\mathbb{E}[k_i^{(\ell),\text{in}}] \approx \kappa_i^{\text{in}}$ (in-degree is approximately calibrated via the symmetry of the weight function over a large population).

Directional connection probability. Define the *unnormalized arc weight* of the potential arc $(i \to j)$ in layer ℓ as:

$$w_{ij}^{(\ell)} \triangleq \frac{1}{1 + \exp\left(\dfrac{\Delta\theta_{ij}^{(\ell)} - R}{T}\right)} \cdot \exp\Big(\alpha \left(\kappa_i^{\text{out}} - \kappa_j^{\text{in}}\right)\Big), \tag{3.9}$$

where $\Delta\theta_{ij}^{(\ell)} = |\theta_i^{(\ell)} - \theta_j^{(\ell)}|$ is the angular separation and $\alpha \geq 0$ is the *directional bias parameter*. The arc $(i \to j) \in \vec{E}^{(\ell)}$ is drawn as an independent Bernoulli random variable with probability:

$$p_{i\to j}^{(\ell)} = \frac{w_{ij}^{(\ell)}}{Z_i^{(\ell)}}, \tag{3.10}$$

where the per-node, per-layer normalisation constant is:

$$Z_i^{(\ell)} = \frac{1}{\kappa_i^{\text{out}}} \sum_{k \neq i} w_{ik}^{(\ell)}, \tag{3.11}$$

chosen so that the expected out-degree satisfies:

$$\mathbb{E}\big[k_i^{(\ell),\text{out}}\big] = \sum_{j \neq i} p_{i\to j}^{(\ell)} = \kappa_i^{\text{out}} \tag{3.12}$$

exactly. For well-conditioned sparse networks ($\kappa_i^{\text{out}} \ll N$), the per-arc probability satisfies $p_{i\to j}^{(\ell)} \leq 1$ with high probability because no single weight $w_{ij}^{(\ell)}$ dominates the sum $\sum_k w_{ik}^{(\ell)}$. Arc realizations are independent across all ordered pairs (i, j) and across layers.

Remark 3.1.1 (Interpretation of α). The two multiplicative factors in Equation (3.10) serve complementary roles. The first factor, inherited from the standard GMM, enforces the

geometric proximity condition: nodes that are angularly close in the hyperbolic disk are more likely to share a directed arc. The second factor, $\exp(\alpha(\kappa_i^{\text{out}} - \kappa_j^{\text{in}}))$, introduces directional asymmetry. When $\alpha > 0$, nodes with high out-degree hidden variable κ_i^{out} preferentially connect to nodes with high in-degree hidden variable κ_j^{in}, producing a *source-sink* hierarchy in which high-κ^{out} hubs drive cascades toward high-κ^{in} convergence points. When $\alpha = 0$, the directional bias factor equals 1 and the D-GMM reduces to a directed variant of the GMM with approximately symmetric arc probabilities.

Remark 3.1.2 (Asymmetric cascade propagation). A directed arc $(i \to j)$ encodes a *unidirectional dependency*: the failure of node i can propagate to node j (if i is the source of a regulatory or logistic input), but not vice versa. This is the defining property of directed interdependent networks [60]: unlike the Buldyrev cascading failure model for undirected multiplexes, directed cascades propagate through the strongly connected components (SCCs) of the directed residual graph, and the relevant critical structure is the Mutually Strongly Connected Component (MSCC) rather than the Largest Mutually Connected Component (LMCC). The D-GMM is specifically designed to produce networks with a large, structured MSCC and non-trivial asymmetry between $p_{i\to j}$ and $p_{j\to i}$, ensuring that the DDIN agent faces a genuinely difficult directed dismantling problem during training.

Multi-layer directed construction. The full D-GMM multiplex is constructed by generating L independent directed layers over the shared vertex set V, with angular positions correlated across layers via the interlayer similarity parameter g following the same rule as Equation (3.4). Inter-layer dependencies remain *undirected* and bidirectional in the one-to-one sense: removing node v from any layer simultaneously removes its replicas from all L layers [11].

Definition 3.1.3 (D-GMM Multiplex)**.** A D-GMM multiplex with parameters $(N, L, \gamma_{\text{out}}, \gamma_{\text{in}}, \bar{k}, g, T$ is the random object $\mathcal{G} = (V, \{\vec{E}^{(\ell)}\}_{\ell=1}^{L})$, where:

1. Each node $i \in V$ is assigned coordinates $(r_i, \theta_i^{(1)})$ via Equations (3.3) and (3.4), and independent hidden degree variables $(\kappa_i^{\text{out}}, \kappa_i^{\text{in}})$ from Equations (3.7)–(3.8).
2. For each layer ℓ and each ordered pair (i, j) with $i \neq j$, the arc $(i \to j)$ is included in $\vec{E}^{(\ell)}$ independently with probability $p_{i\to j}^{(\ell)}$ from Equation (3.10).
3. Inter-layer dependencies are one-to-one and bidirectional: removing any node $v \in V$ simultaneously removes it from all L layers.

3.1.3 Synthetic Training Corpus and Generalisation Protocol

Both DDIN and KARL follow an identical training and evaluation protocol.

Training corpus. A corpus of 20,000 synthetic multiplex graphs with $N \sim$ Uniform $\{30, \ldots, 50\}$ nodes is generated prior to training. For KARL, each graph is drawn from the standard GMM with parameters $\gamma = 2.5$, $\bar{k} = 6$, $T = 0.1$, and $g \sim$ Uniform$[0, 1]$. For DDIN, each graph is drawn from the D-GMM with $\gamma_{\text{out}} = \gamma_{\text{in}} = 2.5$, $\bar{k} = 6$, $T = 0.1$, $g \sim$ Uniform$[0, 1]$, and $\alpha = 0.3$. Both use $L = 2$ layers for the training corpus.

Held-out synthetic evaluation suites. Three out-of-distribution synthetic test suites with $N \in \{256, 1024\}$ nodes are used to assess systematic generalisation:

- `data_g`: $g \sim \text{Uniform}[0, 1]$, other parameters fixed.
- `data_γ`: $\gamma \in [2, 3]$ varied, other parameters fixed.
- `data_k`: $\bar{k} \in [4, 8]$ varied, other parameters fixed.

Each suite contains 200 graphs. Performance on these suites quantifies sensitivity to the interlayer correlation structure, degree distribution tail, and edge density, respectively.

Zero-shot generalisation to real-world networks. The trained agents are evaluated without fine-tuning on real-world multiplex networks spanning three orders of magnitude in scale (see Tables 4.3 and 5.1 in the respective contribution chapters). The key point is that the models never observe networks larger than $N = 50$ during training, yet achieve strong zero-shot performance on networks with up to $N = 56{,}562$ nodes. This generalisation is enabled by the graph-level inductive property of the GNN encoder (node embeddings depend only on local neighbourhoods) combined with the structural similarity between GMM-generated training graphs and real-world multiplex networks (property **(P4)** of Section 3.1.1).

3.2 Formulating the Dismantling MDP

Both DDIN and KARL cast sequential network dismantling as a finite-horizon Markov Decision Process (MDP). We provide a unified MDP specification and carefully distinguish the directed setting (DDIN, MSCC-based) from the undirected setting (KARL, LMCC-based) at each point of divergence.

Definition 3.2.1 (Dismantling MDP)**.** A *dismantling MDP* is a tuple $\mathfrak{M}_{\text{RL}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma_{\text{disc}} \rangle$ where:

- $\mathcal{S}$ is the state space of residual multiplex configurations (Section 3.2.1).
- $\mathcal{A}$ is the action space of candidate node removals (Section 3.2.2).
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is the deterministic transition operator that applies node removal and resolves cascading failures to a fixed point (Section 3.2.2).
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the immediate reward function (Section 3.2.3).
- $\gamma_{\text{disc}} \in [0, 1)$ is the discount factor, set to $\gamma_{\text{disc}} = 0.99$ in both DDIN and KARL.

A dismantling episode begins with $s_0 = \mathcal{M}$ (the full multiplex network) and terminates when the normalised largest strongly (or mutually) connected component drops below the non-extensivity threshold $1/\sqrt{N}$, or when all nodes have been removed.

The agent's objective is to find a stochastic policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximises the expected discounted return:

$$J(\pi) \;=\; \mathbb{E}_{\pi}\left[\sum_{t=0}^{H-1} \gamma_{\text{disc}}^{t}\, \mathcal{R}(s_t, a_t)\right], \tag{3.13}$$

where H is the (random) episode length. The optimal deterministic greedy policy $\pi^*(s) = \arg\max_{a\in\mathcal{A}(s)} Q^*(s,a)$ is characterised by the optimal action-value function $Q^* : \mathcal{S}\times\mathcal{A} \to \mathbb{R}$, satisfying the Bellman optimality equation [54]:

$$Q^*(s,a) \;=\; \mathcal{R}(s,a) \;+\; \gamma_{\text{disc}} \max_{a'\in\mathcal{A}(\mathcal{T}(s,a))} Q^*\big(\mathcal{T}(s,a),\, a'\big). \tag{3.14}$$

Since $\mathcal{T}$ is deterministic (Remark 3.2.6 below), Equation (3.14) reduces to the finite-horizon deterministic Bellman equation. Both DDIN and KARL approximate Q^* with deep neural networks and train via off-policy n-step DQN with prioritised experience replay [42, 50].

3.2.1 State Space Representation

Residual network state. Let the cumulative removal set at step t be $\tilde{S}_t = \bigcup_{i=1}^{t}\{a_i\} \subseteq V$ with $\tilde{S}_0 = \emptyset$. The state after cascade resolution at step t is the residual multiplex:

$$s_t \;=\; \mathcal{M}\setminus\tilde{S}_t \;\triangleq\; \Big(V\setminus\tilde{S}_t,\; \big\{E^{(\ell)}\cap(V\setminus\tilde{S}_t)^2\big\}_{\ell=1}^{L}\Big), \tag{3.15}$$

where $E^{(\ell)}\cap(V\setminus\tilde{S}_t)^2$ is the set of edges (or arcs, in the directed setting) in layer ℓ whose both endpoints are still active.

Node feature vectors. The raw adjacency tensors alone provide insufficient discriminative power for the GNN encoder: two topologically distinct nodes in the same graph may have identical local subgraph structure if features are not augmented. Both DDIN and KARL initialise each node v in layer ℓ of the residual graph with a normalised degree-based feature vector.

Undirected setting (KARL).

$$\boldsymbol{x}_v^{(\ell)} \;=\; \left[\frac{k_v^{(\ell)}}{k_{\max}^{(\ell)}},\; \frac{k_v^{(\ell)}}{k_{\max}^{(\ell)}}\right]^{\top} \;\in\; \mathbb{R}^2, \tag{3.16}$$

where $k_v^{(\ell)} = \sum_j A_{vj}^{(\ell)}$ is the degree of v in layer ℓ of the residual graph and $k_{\max}^{(\ell)} = \max_u k_u^{(\ell)}$ is the current maximum layer degree. The two-dimensional format matches the input dimension $d_{\text{in}} = 2$ of the linear projection $\boldsymbol{W}_{n2l} \in \mathbb{R}^{2\times d}$ that initialises the embedding space [25].

Directed setting (DDIN).

$$\boldsymbol{x}_v^{(\ell)} \;=\; \left[\frac{k_v^{(\ell),\text{out}}}{k_{\max}^{(\ell),\text{out}}},\; \frac{k_v^{(\ell),\text{in}}}{k_{\max}^{(\ell),\text{in}}},\; \frac{\theta_v^{(\ell)}}{2\pi},\; \frac{r_v^{(\ell)}}{R}\right]^{\top} \;\in\; \mathbb{R}^4, \tag{3.17}$$

where $k_v^{(\ell),\text{out}}$ and $k_v^{(\ell),\text{in}}$ are the residual out- and in-degrees, and $\theta_v^{(\ell)}/2\pi$ and $r_v^{(\ell)}/R$ are the normalised hyperbolic angular and radial coordinates inherited from the D-GMM embedding. Including the hyperbolic coordinates makes the asymmetric GNN encoder of DDIN aware of the directional flow hierarchy encoded in the D-GMM geometry: nodes with $r_v \approx 0$ (near the disk centre) correspond to hubs with high κ^{out}, and nodes with large angular separation are geometrically unlikely to be linked [20].

Virtual-node augmentation. Both methods augment each layer ℓ with a single *virtual node* $s^{(\ell)}$ that aggregates global information from all active real nodes [23, 25]. The virtual node is connected to every active node in layer ℓ but has no in-neighbours among the real nodes:

$$E^{(\ell)} \leftarrow E^{(\ell)} \cup \big\{(v, s^{(\ell)}) : v \in V \setminus \tilde{S}_t\big\}, \tag{3.18}$$

in the undirected case (edges are unordered), and analogously in the directed case. After one round of message passing, the embedding $z_{s^{(\ell)}}$ of the virtual node aggregates information from all active nodes in layer ℓ, providing a permutation-invariant global state summary without the quadratic cost of an all-pairs readout. This embedding is used as the state vector in the Q-value decoder (see Section 3.2.3 and Chapters 4 and 5).

Remark 3.2.2 (Benefit over graph-level pooling). Global mean or sum pooling over all node embeddings is a standard alternative for computing graph-level state representations. Virtual-node augmentation is preferred here for two reasons. First, the virtual node's embedding is computed by the *same* message-passing operation that produces all other node embeddings, requiring no additional architectural components. Second, the virtual node avoids the over-smoothing collapse that occurs when the largest connected (or strongly connected) component is very small and most node embeddings have converged to similar values: isolated nodes still receive a distinct signal through their edge to $s^{(\ell)}$, preserving their representation quality throughout the episode.

LMCC and MSCC membership indicators. To enable the GNN encoder to distinguish nodes currently inside the critical component from those that have already been isolated (by direct removal or by cascade), both methods maintain a binary membership vector over all active nodes.

Undirected setting (KARL):

$$m_v = \mathbf{1}\big[v \in \mathrm{LMCC}(s_t)\big], \tag{3.19}$$

where $\mathrm{LMCC}(s_t)$ is the Largest Mutually Connected Component of the residual undirected multiplex s_t: the maximum-cardinality set of nodes connected by paths in every layer simultaneously [11].

Directed setting (DDIN):

$$m_v = \mathbf{1}\big[v \in \mathrm{MSCC}(s_t)\big], \tag{3.20}$$

where $\mathrm{MSCC}(s_t)$ is the Largest Mutually Strongly Connected Component of the residual directed multiplex s_t.

Definition 3.2.3 (Mutually Strongly Connected Component (MSCC))**.** In a directed multiplex network $\mathcal{G} = (V, \{\vec{E}^{(\ell)}\}_{\ell=1}^{L})$, let the *operationally active* node set at step t be $V_t = V \setminus \tilde{S}_t$, representing nodes that have not been removed in any layer. Two nodes $u, v \in V_t$ are *mutually strongly reachable* if and only if, for every layer $\ell \in \{1, \ldots, L\}$, there exist directed paths $u \rightsquigarrow v$ and $v \rightsquigarrow u$ in the induced subgraph:

$$G_\ell^t \triangleq \Big(V_t,\ \vec{E}^{(\ell)} \cap (V_t \times V_t)\Big), \tag{3.21}$$

where crucially, *every intermediate path vertex must belong to* V_t – that is, it must remain simultaneously present in all layers. A node that has been removed from any single

layer is ineligible as a path vertex in any layer, reflecting the one-to-one interdependency of the multiplex. The *Mutually Strongly Connected Component* (MSCC) is the unique maximum-cardinality subset $C \subseteq V_t$ such that all ordered pairs (u, v) with $u, v \in C$ are mutually strongly reachable. Its normalised size is:

$$P_{\mathrm{MSCC}}(\tilde{S}_t) \;=\; \frac{|\mathrm{MSCC}(s_t)|}{N}. \tag{3.22}$$

Remark 3.2.4 (MSCC versus LMCC). The LMCC of an undirected multiplex requires that nodes u and v be connected by undirected paths in every layer; since edges are symmetric, connectivity from u to v automatically implies connectivity from v to u. The MSCC imposes the strictly stronger condition that directed paths of opposite orientation exist in every layer simultaneously. Consequently, $|\mathrm{MSCC}(s_t)| \le |\mathrm{LMCC}(s_t)|$ for any mixed undirected/directed interpretation, and the MSCC can be exponentially smaller in adversarial directed graphs. This structural asymmetry renders directed dismantling fundamentally harder than its undirected counterpart and motivates the specialised DDIN architecture of Chapter 4.

The MSCC membership indicator is computed at each step t by running Tarjan's depth-first-search algorithm [55] on the condensed directed graph of the residual multiplex. Tarjan's algorithm identifies all SCCs in $O(|V| + |\vec{E}|)$ time; the condensation DAG then determines mutual reachability across layers. The MSCC of the directed multiplex is the intersection of the dominant SCCs across all L layers, filtered for mutual accessibility through the inter-layer interdependencies.

Auxiliary state features (KARL decoder only). The Chebyshev-KAN action-value decoder of KARL (Section 5.3) additionally computes, for each node v and layer ℓ, a four-dimensional auxiliary feature vector:

$$\boldsymbol{f}_v^{(\ell),\mathrm{aux}} \;=\; \left[\frac{|\tilde{S}_t|}{N},\; \frac{|E_0^{(\ell)}| - |E^{(\ell)}(s_t)|}{|E_0^{(\ell)}|},\; \rho_v^{(\ell)},\; \frac{k_v^{(\ell)}}{k_{\max,0}^{(\ell)}}\right]^{\top} \;\in\; \mathbb{R}^4, \tag{3.23}$$

where:

- $|E_0^{(\ell)}|$ is the number of edges in layer ℓ at $t = 0$;
- $|E^{(\ell)}(s_t)|$ is the number of edges in layer ℓ in the residual graph at step t;
- $\rho_v^{(\ell)} = |\mathcal{N}^2(v) \cap \mathrm{LMCC}(s_t)|/N$ is the normalised two-hop reachability of v within the current LMCC; and
- $k_{\max,0}^{(\ell)}$ is the maximum degree in layer ℓ at $t = 0$.

These four scalars supply the decoder with current local structural information (fraction of nodes and edges removed, local reachability within the LMCC, and degree relative to the initial maximum) that is complementary to the global embeddings produced by the KANformer fusion module. Together with the fused GNN embedding $\boldsymbol{z}_v^{(\ell)} \in \mathbb{R}^d$, the auxiliary vector is concatenated to form the full $\mathbb{R}^{d+4}$ decoder input:

$$\boldsymbol{e}_v^{(\ell)} \;=\; \mathrm{OuterProduct}\Big(\boldsymbol{z}_{s^{(\ell)}},\; \boldsymbol{z}_v^{(\ell)}\Big) \;\in\; \mathbb{R}^d, \tag{3.24}$$

which is then concatenated with $\boldsymbol{f}_v^{(\ell),\text{aux}}$ before being passed to the ChebyKAN layers (Section 5.3).

3.2.2 Action Space

Definition 3.2.5 (Action Space)**.** At step t, the action space is the set of nodes that remain active:

$$\mathcal{A}(s_t) \;=\; V \setminus \tilde{S}_t. \tag{3.25}$$

An action $a_t \in \mathcal{A}(s_t)$ selects node v_{a_t} for *permanent, simultaneous* removal from all L layers.

Transition and cascading failure resolution. The state transition $s_t \to s_{t+1} = \mathcal{T}(s_t, a_t)$ is *deterministic* and proceeds through the following three phases.

Phase 1: Direct removal. Node v_{a_t} and all its incident edges (or arcs) in every layer are deleted:

$$E^{(\ell)} \;\leftarrow\; E^{(\ell)} \setminus \left\{e \in E^{(\ell)} : v_{a_t} \in e\right\}, \quad \forall\, \ell \in \{1, \ldots, L\}, \tag{3.26}$$

where $e \in e$ means v_{a_t} is an endpoint of the edge or arc e.

Phase 2: Cascade propagation. Due to the one-to-one interdependency [11], any node that ceases to belong to a non-trivial mutually (strongly) connected component after Phase 1 fails and is removed. This process is iterated until a fixed point is reached:

$$V \;\leftarrow\; V \setminus \left\{v \in V : v \notin C(s)\right\}, \tag{3.27}$$

where $C(s)$ denotes the relevant critical component (LMCC for KARL; MSCC for DDIN) of the current residual graph s. The iteration terminates because $|V|$ is finite and strictly decreases at each step.

Phase 3: State update. The residual state s_{t+1} is the fixed point of the cascade process:

$$s_{t+1} \;=\; \mathcal{T}(s_t,\, a_t), \tag{3.28}$$

where $\mathcal{T}$ is the composed operator of direct removal (Equation (3.26)) and cascade resolution (Equation (3.27)).

Remark 3.2.6 (Determinism and MDP validity). Both Phase 1 (direct removal) and Phase 2 (cascade resolution to the LMCC or MSCC fixed point) are deterministic functions of the current state s_t and action a_t. The transition kernel $\mathcal{T}$ therefore satisfies the Markov property: s_{t+1} depends only on (s_t, a_t) and not on the history $(s_0, a_0, \ldots, s_{t-1}, a_{t-1})$. This validates the MDP formulation of Definition 3.2.1 and justifies the use of the standard Bellman framework (Equation (3.14)).

Cascade complexity. In the undirected setting (KARL), resolving the LMCC after each removal requires computing the connected components of a graph with up to N nodes, which can be done in $O(N + |E|)$ via breadth-first search. In the directed setting (DDIN), resolving the MSCC requires computing the strongly connected components of a directed graph in each of the L layers and then intersecting the dominant SCCs across layers. This is accomplished

in $O(L \cdot (N + |\vec{E}|))$ via Tarjan's algorithm [55]. For the real-world benchmarks used in Chapter 4 (up to $N = 56{,}562$ nodes and 364 layers for FAO Trade), the cascade resolution step is the computational bottleneck; both DDIN and MultiDismantler [25] implement it using the optimised `networkx` SCC routines with sparse adjacency representations.

Effective action space reduction. In the directed setting, removing a node outside the current MSCC cannot directly reduce P_{MSCC}, since such a node does not participate in the mutual strong connectivity. This motivates restricting the effective action space to MSCC-internal nodes:

$$\mathcal{A}_{\text{eff}}(s_t) \;=\; \big(V \setminus \tilde{S}_t\big) \cap \text{MSCC}(s_t), \tag{3.29}$$

at the cost of potentially missing indirect reductions: removing an MSCC-external node may trigger a cascade that collapses the MSCC indirectly. DDIN uses the full action space $\mathcal{A}(s_t) = V \setminus \tilde{S}_t$ and allows the learned Q-function to implicitly discover when MSCC-external removals are beneficial through the n-step return.

3.2.3 Reward Functions

A key design challenge in sequential dismantling is the definition of a reward signal that is (i) dense enough to provide useful gradient information at every step, (ii) correctly incentivises both large component reduction and low removal cost, and (iii) remains numerically stable as the episode progresses and the critical component shrinks. The reward functions adopted by DDIN and KARL satisfy all three criteria.

Both settings share the same multiplicative reward structure; the two contributions differ *solely* in the choice of connectivity criterion P_C:

$$r_t \;=\; P_C(s_t) \cdot F\big(\{v_{a_t}\}\big), \qquad P_C \;=\; \begin{cases} P_\infty & \text{KARL (undirected, LMCC),} \\ P_{\text{MSCC}} & \text{DDIN (directed, MSCC),} \end{cases} \tag{3.30}$$

where $P_C(s_t)$ is evaluated *after* removal of v_{a_t} and full cascade resolution at step t, and $F(\{v_{a_t}\})$ is the per-node cost under the applicable cost regime. The two explicit forms are as follows.

Undirected reward (KARL). The immediate reward for removing node v_{a_t} at step t in the undirected multiplex setting is [25]:

$$r_t^{\text{und}} \;=\; P_\infty(s_t) \cdot F\big(\{v_{a_t}\}\big), \tag{3.31}$$

where $P_\infty(s_t) = |\text{LMCC}(s_t)|/N$ is the normalised LMCC size *after* removal and cascade resolution at step t, and $F(\{v_{a_t}\})$ is the per-node removal cost (equal to 1 under unit cost; equal to $\sum_\ell k_{v_{a_t}}^{(\ell)}$ under degree cost).

Directed reward (DDIN). The immediate reward for removing node v_{a_t} in the directed multiplex setting is:

$$r_t^{\text{dir}} \;=\; P_{\text{MSCC}}(s_t) \cdot F\big(\{v_{a_t}\}\big), \tag{3.32}$$

where $P_{\text{MSCC}}(s_t) = |\text{MSCC}(s_t)|/N$ is the normalised MSCC size after removal and cascade resolution (Definition 3.2.3).

Remark 3.2.7 (Reward design rationale). The multiplicative form $r_t = P_C(s_t) \cdot F(\{v_{a_t}\})$, where P_C denotes either P_∞ or P_{MSCC}, achieves the three design goals as follows. *Density*: a non-zero reward is generated at every step (as long as the critical component is non-empty), avoiding the sparse reward problem that would arise if reward were given only upon crossing the non-extensivity threshold. *Joint incentivisation*: the factor $P_C(s_t)$ assigns higher reward when the critical component is large, incentivising high-impact removals in the early phases of the episode; the factor $F(\{v_{a_t}\})$ rewards selection of high-cost (typically high-degree) nodes, consistent with the observation that hub removal is most effective in scale-free networks. *Numerical stability*: as the episode progresses and $P_C \to 0$, the reward automatically decays toward zero, maintaining bounded gradients throughout the episode.

Degree-cost reward normalisation (KARL). For the degree-cost evaluation regime, the KARL agent's per-node cost $F(\{v_{a_t}\})$ in Equation (3.31) becomes $\sum_\ell k^{(\ell)}_{v_{a_t}}$, which grows with network size and can cause reward-scale drift as nodes are progressively removed. To prevent this, the degree-cost variant of the KARL reward normalises each layer's contribution by the total residual degree::

$$r_t^{\text{deg}} \;=\; -\frac{P_\infty(s_t)}{P_{\infty,0}} \cdot \frac{1}{L} \sum_{\ell=1}^{L} \frac{k^{(\ell)}_{v_{a_t}}}{\sum_{u \in V \setminus \tilde{S}_t} k^{(\ell)}_u}, \tag{3.33}$$

where $P_{\infty,0} = P_\infty(s_0)$ is the initial normalised LMCC size. The negative sign reflects that the agent must minimise degree-cost expenditure: the reward is in $[-1, 0]$ throughout the episode, and maximising the cumulative return is equivalent to achieving maximum LMCC reduction at minimum degree expenditure per layer. The normalisation by $\sum_u k^{(\ell)}_u$ ensures reward-scale invariance as nodes are progressively removed, a critical stability property for n-step DQN with prioritised experience replay.

n-step returns. A single-step reward cannot capture the *delayed* cascade effects that are the defining challenge of multiplex dismantling: removing a critical bridge node today may yield no immediate LMCC/MSCC reduction but triggers a cascade collapse several steps later. Both DDIN and KARL therefore use n-step returns [54]:

$$y_t \;=\; \sum_{j=0}^{n-1} \gamma_{\text{disc}}^j \, r_{t+j} \;+\; \gamma_{\text{disc}}^n \max_{a'} \hat{Q}(s_{t+n}, a'; \theta^-), \tag{3.34}$$

where $n = 5$ is the return horizon (selected by ablation in Chapter 5 and adopted by DDIN for consistency), $\gamma_{\text{disc}} = 0.99$, and $\hat{Q}(\cdot; \theta^-)$ is the target network with parameters θ^- updated every 1,000 gradient steps. The lookahead horizon of $n = 5$ is sufficient to cover the typical cascade depth in scale-free multiplex networks: for a training graph with $N = 50$ nodes and mean degree $\bar{k} = 6$, the diameter is approximately $\log_{\bar{k}} N \approx 2.2$, while the expected cascade depth following hub removal is at most $\log_{\bar{k}} N \approx 4.7$ for the real-world evaluation graphs with $N = 1{,}024$ nodes.

Temporal difference loss and prioritised experience replay. Both DDIN and KARL train via off-policy DQN with Prioritised Experience Replay (PER) [50]. Transitions $(s_t, a_t, r_t, \ldots, r_{t+n-1}, s_{t+n})$

are stored in a replay buffer $\mathcal{B}$. The temporal difference (TD) error for transition i is:

$$\delta_i \;=\; y_t^{(i)} - Q(s_t^{(i)},\, a_t^{(i)};\, \theta), \tag{3.35}$$

where $y_t^{(i)}$ is the n-step target from Equation (3.34). PER assigns each transition a sampling priority:

$$p_i \;=\; |\delta_i| + \epsilon_{\text{PER}}, \tag{3.36}$$

with sampling probability $P(i) \propto p_i^{\alpha_{\text{PER}}}$ ($\alpha_{\text{PER}} = 0.6$, $\epsilon_{\text{PER}} = 10^{-6}$). The induced non-uniform sampling bias is corrected by importance-sampling weights:

$$w_i \;=\; \left(\frac{1}{|\mathcal{B}| \cdot P(i)}\right)^{\beta_{\text{IS}}} \Bigg/ \max_j w_j, \tag{3.37}$$

with β_{IS} annealed from 0.4 to 1.0 over the course of training. For DDIN, PER is implemented via a sum-tree data structure [50] enabling $O(\log|\mathcal{B}|)$ sampling and update operations, essential for the large action spaces (up to $N = 56{,}562$ candidate nodes) encountered during zero-shot evaluation on Sanremo2016.

The total training loss is the importance-weighted Huber loss over the prioritised mini-batch $\mathcal{B}_t \subset \mathcal{B}$:

$$\mathcal{L}_{\text{TD}}(\theta) \;=\; \frac{1}{|\mathcal{B}_t|} \sum_{i \in \mathcal{B}_t} w_i \cdot \delta_{\text{Huber}}(\delta_i), \tag{3.38}$$

where the Huber loss $\delta_{\text{Huber}}(\delta) = \frac{1}{2}\delta^2$ for $|\delta| \le 1$ and $|\delta| - \frac{1}{2}$ otherwise provides robustness to the large TD errors that arise at the beginning of training when the policy is nearly random. DDIN augments this with a reconstruction loss $\mathcal{L}_{\text{rec}}$ (Section 4.4.2); KARL augments with a graph Laplacian regulariser $\mathcal{L}_{\text{R}}$ (Section 2.32).

3.3 Cost Regimes and Evaluation Metrics

This section provides rigorous definitions of all cost functions and performance metrics used throughout Chapters 4–6. While the scalar metrics AUDC and C^* were briefly introduced in Chapter 2, the treatment here extends them to the directed setting and places them in their MDP context, making explicit their relation to the reward signals of Section 3.2.3. The section concludes with the statistical testing protocol used to establish significance of all reported improvements.

3.3.1 Cost Regimes

All algorithms construct *sequential* approximate solutions: at each step t, one node a_t is selected, yielding the cumulative removal sequence $\tilde{S}_t = \bigcup_{i=1}^{t} \{a_i\}$ with $\tilde{S}_0 = \emptyset$ and $\tilde{S}_N = V$. The cost of a removal sequence is:

Definition 3.3.1 (Unit Cost)**.**

$$F_{\text{unit}}(S) \;=\; |S|, \qquad F_{\text{unit}}(V) = N. \tag{3.39}$$

Each removed node incurs a cost of one, irrespective of its degree or structural role. Under unit cost, the dismantling problem reduces to identifying the *minimum cardinality* removal set, directly penalising the number of interventions required.

Definition 3.3.2 (Degree Cost [25, 48])**.** The degree cost of removing the set S counts the total number of intra-layer edges deleted, correcting for edges shared between two removed nodes:

$$F_{\deg}(S) \;=\; \sum_{\ell=1}^{L}\left(\sum_{s\in S} k_s^{(\ell)} \;-\; \sum_{\{s,t\}\subseteq S} A_{st}^{(\ell)}\right), \tag{3.40}$$

where $k_s^{(\ell)} = \sum_j A_{sj}^{(\ell)}$ is the degree of node s in layer ℓ, and the second sum runs over unordered pairs $\{s,t\} \subseteq S$ with $A_{st}^{(\ell)} = 1$ to avoid double-counting edges whose both endpoints are removed.

Remark 3.3.3 (Degree cost increment at step t). Under degree cost, the incremental cost of removing node v_{a_t} at step t is:

$$F_{\deg}(\tilde{S}_t) - F_{\deg}(\tilde{S}_{t-1}) \;=\; \sum_{\ell=1}^{L}\left(k_{v_{a_t}}^{(\ell)} \;-\; \sum_{s\in\tilde{S}_{t-1}} A_{v_{a_t},s}^{(\ell)}\right), \tag{3.41}$$

which equals the number of edges incident to v_{a_t} in the residual graph (edges to already-removed nodes contribute zero), summed over all layers. This incremental form is used in the AUDC computation (Equation (3.44) below).

Definition 3.3.4 (Directed Degree Cost (DDIN))**.** In the directed setting, removing node v deletes all incident arcs in both directions. The directed degree cost is:

$$F_{\deg}^{\mathrm{dir}}(S) \;=\; \sum_{\ell=1}^{L}\left(\sum_{s\in S}\left(k_s^{(\ell),\mathrm{out}} + k_s^{(\ell),\mathrm{in}}\right) \;-\; 2\sum_{\substack{(s,t)\in\vec{E}^{(\ell)}\\ s,t\in S}} 1\right), \tag{3.42}$$

where the subtracted term corrects for arcs whose both endpoints are in S (each such arc contributes $+1$ to $k_s^{(\ell),\mathrm{out}}$ and $+1$ to $k_t^{(\ell),\mathrm{in}}$ but is only a single deleted arc). In the primary experimental evaluations of Chapter 4, unit cost is used exclusively; degree cost results for KARL are reported in Section 5.5.3.

Definition 3.3.5 (Random Cost [25])**.** To test generalisation to out-of-distribution cost distributions, a random cost assigns each node s a cost $c_s^{(\ell)}$ drawn independently from a specified distribution $\mathcal{D}$:

$$F_{\mathrm{rand}}(S) \;=\; \sum_{\ell=1}^{L}\sum_{s\in S} c_s^{(\ell)}, \qquad c_s^{(\ell)} \sim \mathcal{D} \text{ i.i.d.} \tag{3.43}$$

Distributions considered are Uniform$(0,1)$, Gaussian$(\mu = 0.5, \sigma^2 = 0.1)$, and Poisson$(\lambda = 5)$. Random cost evaluation is used exclusively for MultiDismantler's generalisation experiments and is not used in the primary evaluation of DDIN or KARL.

Remark 3.3.6 (Information hierarchy of cost regimes). The three cost regimes form a hierarchy ordered by the amount of structural information they encode. Unit cost is information-minimal: every node is equally expensive, so the agent cannot leverage cost heterogeneity. Degree cost introduces structure-dependent heterogeneity: hub nodes are

more expensive to remove, creating a non-trivial trade-off between their high dismantling impact and their high cost. Random cost eliminates structural correlation: the agent must learn a cost-invariant dismantling policy. The fact that KARL achieves near-optimal performance under all three regimes (Chapter 5) is evidence that its learned policy exploits topological information rather than cost structure.

3.3.2 Performance Metrics

Let $\tilde{S}_t$ be the cumulative removal sequence produced by an algorithm at step t, and let $P_C(\tilde{S}_t)$ denote the normalised size of the relevant critical component (LMCC for KARL; MSCC for DDIN) after cascade resolution.

Definition 3.3.7 (Area Under the Dismantling Curve (AUDC))**.** The *Area Under the Dismantling Curve* is defined as the following discrete sum, which approximates the Riemann–Stieltjes integral $\int_0^1 P_C\big(F^{-1}(c \cdot F(V))\big)\, dc$ that sweeps the normalised component size over the normalised cost axis:

$$\text{AUDC} \;=\; \frac{1}{F(V)} \sum_{t=1}^{N} P_C(\tilde{S}_t) \left[F(\tilde{S}_t) - F(\tilde{S}_{t-1})\right]. \tag{3.44}$$

In practice, because $P_C(\tilde{S}_t)$ drops below the non-extensivity threshold $1/\sqrt{N}$ after far fewer than N removals on all benchmarks considered, the sum is truncated at the first step t_c such that $P_C(\tilde{S}_{t_c}) \le 1/\sqrt{N}$; remaining terms contribute at most $O(1/N)$ to the total and are negligible. AUDC $\in [0, 1]$; lower values indicate more efficient dismantling. A theoretically perfect dismantler achieves AUDC $= 0$. The metric generalises the robustness metric of Schneider et al. [51] from binary connected/disconnected states to the continuous P_C trajectory.

Remark 3.3.8 (Computational implementation). For large networks (e.g., Sanremo2016 with $N = 56{,}562$), the truncated AUDC sum is evaluated lazily: the cascade-resolution loop (Section 3.2.2) tracks P_C incrementally after each removal and halts once $P_C \le 1/\sqrt{N}$, consistent with MultiDismantler [25].

Definition 3.3.9 (Dismantling Cost C^*)**.** The *dismantling cost* C^* is the smallest normalised cost required to reduce the critical component below the non-extensivity threshold $1/\sqrt{N}$:

$$C^* \;=\; \frac{F(\tilde{S}_c)}{F(V)}, \qquad \tilde{S}_c \;=\; \operatorname*{arg\,min}_{\tilde{S}_t:\, P_C(\tilde{S}_t) \le 1/\sqrt{N}} F(\tilde{S}_t). \tag{3.45}$$

The threshold $1/\sqrt{N}$ is the conventional definition of non-extensivity [13]: a component of normalised size $\le N^{-1/2}$ is considered negligible relative to the original system size. Both AUDC and C^* lie in $[0, 1]$; lower values indicate superior dismantling performance.

Remark 3.3.10 (Directed variants). For the directed setting (DDIN), both metrics are computed identically to Definitions 3.3.7 and 3.3.9 but with $P_{\text{MSCC}}(\tilde{S}_t)$ (Equation (3.22)) in place of $P_\infty(\tilde{S}_t)$. We denote the directed variants as AUDC^{dir} and $C^{*,\text{dir}}$ in Chapter 4. The average uniform dismantling cost (AUDC) reported in the DDIN results (Table 4.3) uses the

formula:

$$\mathrm{AUDC}^{\mathrm{dir}} \;=\; \frac{1}{F(V)} \sum_{t=1}^{N} P_{\mathrm{MSCC}}(\tilde{S}_t)\,\big[F(\tilde{S}_t) - F(\tilde{S}_{t-1})\big], \tag{3.46}$$

and is averaged over 5 independent runs of each algorithm to account for any random tie-breaking in the baselines (DDIN is deterministic after training).

Proposition 3.3.11 (Relationship between AUDC and C^*)**.** *For any sequential removal algorithm and any cost regime,* $\mathrm{AUDC} \le C^*$*, with equality if and only if* $P_C(\tilde{S}_t) = 1$ *for all* $t < t_c$ *and* $P_C(\tilde{S}_{t_c}) = 0$ *(i.e., the critical component collapses instantaneously at step* t_c*). The statement holds verbatim when* P_C *is replaced by* P_{MSCC} *in the directed setting.*

Proof. Let $\Delta F_t = F(\tilde{S}_t) - F(\tilde{S}_{t-1}) > 0$ for each $t \ge 1$. By definition of C^*, $P_C(\tilde{S}_t) \le 1/\sqrt{N}$ for every $t \ge t_c$, and $P_C(\tilde{S}_t) \le 1$ for every $t < t_c$. Therefore:

$$\begin{aligned}
\mathrm{AUDC} \;&=\; \frac{1}{F(V)} \sum_{t=1}^{N} P_C(\tilde{S}_t)\,\Delta F_t \\
&\le\; \frac{1}{F(V)} \left(\sum_{t=1}^{t_c} 1 \cdot \Delta F_t \;+\; \sum_{t=t_c+1}^{N} \frac{\Delta F_t}{\sqrt{N}} \right) \\
&\le\; \frac{1}{F(V)} \sum_{t=1}^{t_c} \Delta F_t \;=\; \frac{F(\tilde{S}_c)}{F(V)} \;=\; C^*.
\end{aligned} \tag{3.47}$$

(The last inequality uses $1/\sqrt{N} \le 1$, so dropping the tail sum only tightens the bound.) Equality holds if and only if $P_C(\tilde{S}_t) = 1$ for all $t < t_c$ and $P_C(\tilde{S}_{t_c}) = 0$, which requires a single-step instantaneous collapse of the entire critical component. The identical argument applies in the directed case upon substituting P_{MSCC} for P_C. □

3.3.3 Statistical Significance Protocol

With only $K = 6$ real-world benchmark datasets (KARL) or $K = 5$ (DDIN) and $M = 7$ compared methods (KARL) or $M = 3$ (DDIN), standard pairwise t-tests are both underpowered and do not control for multiple comparisons. Both contributions follow the non-parametric testing protocol recommended by Demšar [19] for classifier comparison over multiple datasets.

Step 1: Friedman test. The *Friedman test* is a non-parametric repeated-measures ANOVA that tests the null hypothesis H_0 that all M methods perform equivalently across K datasets. Let $R_j^{(i)}$ denote the rank of method $j \in \{1, \dots, M\}$ on dataset $i \in \{1, \dots, K\}$ (lower AUDC $\Leftrightarrow$ lower rank $\Leftrightarrow$ better). The Friedman statistic is:

$$\chi_F^2 \;=\; \frac{12K}{M(M+1)} \sum_{j=1}^{M} \left(\bar{R}_j - \frac{M+1}{2} \right)^2, \tag{3.48}$$

where $\bar{R}_j = K^{-1} \sum_{i=1}^{K} R_j^{(i)}$ is the mean rank of method j. Under H_0, $\chi_F^2 \sim \chi_{M-1}^2$ asymptotically [19]. KARL rejects H_0 with $\chi_F^2(6) = 58.42$, $p < 10^{-6}$ (AUDC) and $\chi_F^2(6) = 51.27$, $p < 10^{-5}$ (C^*) on the six real-world benchmarks.

Step 2: Nemenyi post-hoc test. When H_0 is rejected, a pairwise *Nemenyi test* determines which pairs of methods are significantly different. The critical difference at significance level α_{CD} is:

$$\text{CD} = q_{\alpha_{\text{CD}}} \sqrt{\frac{M(M+1)}{6K}}, \tag{3.49}$$

where $q_{\alpha_{\text{CD}}}$ is the critical value of the studentised range statistic (e.g., $q_{0.05} = 2.85$ for $M = 7$) [19]. Methods j and j' are declared significantly different if $|\bar{R}_j - \bar{R}_{j'}| > \text{CD}$.

Step 3: Wilcoxon signed-rank test (KARL vs. MultiDismantler). Because the Nemenyi test is structurally underpowered at $K = 6$ datasets (CD = 2.85 is large relative to the typical rank gap between closely performing methods), the KARL evaluation employs a one-sided *Wilcoxon signed-rank test* for the pairwise comparison of KARL against MultiDismantler on an expanded suite of $K' = 10$ benchmark datasets (six primary plus four supplementary from the De Domenico repository [15]). The test statistic is:

$$W = \sum_{i:\Delta_i \neq 0} \text{sgn}(\Delta_i)\, R_i, \tag{3.50}$$

where $\Delta_i = \text{AUDC}^{(i)}_{\text{KARL}} - \text{AUDC}^{(i)}_{\text{MD}}$ and R_i is the rank of $|\Delta_i|$ among all non-zero differences. The Holm-Bonferroni correction is applied to all pairwise comparisons to control the family-wise error rate at $\alpha = 0.05$. On $K' = 10$ datasets, the test achieves $p = 0.001$ (AUDC) and $p = 0.002$ (C^*), which after Holm-Bonferroni correction yields $p = 0.006$, establishing statistical significance at $\alpha = 0.05$.

Bootstrap confidence intervals. Complementary to rank-based tests, 95% bootstrap confidence intervals for AUDC and C^* are computed by $B = 10{,}000$ block-bootstrap resamplings of the dismantling trajectory. Non-overlapping bootstrap CIs across all six real-world datasets constitute a conservative, assumption-free confirmation of method superiority.

The two reward signals (Equation (3.30)) and the two critical-component oracles – LMCC for KARL, MSCC for DDIN – are the *sole points of divergence* between the two agents developed in Chapters 4 and 5, respectively. All other components of the MDP formulation, generative model, cost regime, and statistical testing protocol are shared verbatim. The rest of the thesis treats this chapter as the single authoritative source of notation and mathematical definitions.

Table 3.1 provides a consolidated reference for all notation introduced in this chapter. All symbols carry their definitions forward into Chapters 4–7 without restatement.

Table 3.1: Notation index for Chapter 3. Symbols marked ($\star$) apply only to the undirected setting (KARL); symbols marked ($\dagger$) apply only to the directed setting (DDIN); unmarked symbols apply to both settings.

Symbol	Meaning
Network structure	
$\mathcal{M} = \langle V, \mathcal{E} \rangle$	Multiplex network; V shared node set, $\mathcal{E} = \{E^{(\ell)}\}_{\ell=1}^{L}$
N, L	Number of nodes; number of layers
$A^{(\ell)} \in \{0,1\}^{N \times N}$	Adjacency matrix of layer ℓ (symmetric$^{(\star)}$; asymmetric$^{(\dagger)}$)
$k_v^{(\ell)}$, $k_v^{(\ell),\text{out}}$, $k_v^{(\ell),\text{in}}$	Degree / out-degree / in-degree of v in layer ℓ
$\text{LMCC}(s_t)^{\star}$	Largest Mutually Connected Component of undirected residual s_t
$\text{MSCC}(s_t)^{\dagger}$	Largest Mutually Strongly Connected Component of directed residual s_t
$P_\infty(\tilde{S}_t)^{\star}$	Normalised LMCC size: $\lvert\text{LMCC}(s_t)\rvert/N$
$P_{\text{MSCC}}(\tilde{S}_t)^{\dagger}$	Normalised MSCC size: $\lvert\text{MSCC}(s_t)\rvert/N$
Generative models (GMM / D-GMM)	
$\gamma, \gamma_{\text{out}}, \gamma_{\text{in}}$	Power-law exponent(s) of the (directed) degree distribution
$\bar{k}$	Expected mean degree
$g \in [0,1]$	Interlayer angular correlation parameter
$\alpha \geq 0^{\dagger}$	Directional bias parameter (D-GMM)
$T \in (0,1)$	Temperature controlling network transitivity
R	Radius of hyperbolic disk
$\kappa_i, \kappa_i^{\text{out}}, \kappa_i^{\text{in}}$	Hidden degree variable(s)
MDP	
$\tilde{S}_t$	Cumulative removal set at step t
$s_t = \mathcal{M} \setminus \tilde{S}_t$	Residual multiplex state at step t
$\mathcal{A}(s_t)$	Action space at step t: $V \setminus \tilde{S}_t$
a_t	Action (selected node index) at step t
$\mathcal{T}(s_t, a_t)$	Deterministic cascade-resolution transition operator
r_t	Immediate reward at step t
$\gamma_{\text{disc}} = 0.99$	Discount factor
$n = 5$	n-step return horizon
y_t	n-step TD target
δ_i	TD error for replay transition i
$\alpha_{\text{PER}} = 0.6$	PER prioritisation exponent
β_{IS}	PER importance-sampling exponent (annealed $0.4 \rightarrow 1.0$)
$\boldsymbol{x}_v^{(\ell)}$	Node feature vector for v in layer ℓ
$s^{(\ell)}$	Virtual node in layer ℓ
m_v	LMCC/MSCC membership indicator
$\boldsymbol{f}_v^{(\ell),\text{aux}\star}$	Four-dimensional auxiliary feature vector (KARL decoder only)
Cost regimes and metrics	
$F_{\text{unit}}(S)$	Unit cost: $\lvert S\rvert$
$F_{\text{deg}}(S)$	Degree cost (Equation (3.40))
$F_{\text{deg}}^{\text{dir}}(S)^{\dagger}$	Directed degree cost (Equation (3.42))
AUDC	Area Under the Dismantling Curve
$\text{AUDC}^{\text{dir}\dagger}$	AUDC computed with MSCC fraction
C^*	Dismantling cost (smallest normalised cost to reach $1/\sqrt{N}$)
$1/\sqrt{N}$	Non-extensivity threshold for the critical component

Chapter 4

Disassembling Directed Interdependent Networks (DDIN)

The undirected multiplex dismantling framework surveyed in Chapter 2 — and embodied by MultiDismantler [25], the primary baseline of this thesis — rests on an implicit structural assumption: the adjacency matrix $\boldsymbol{A}^{(\ell)}$ of every layer is symmetric, so that a connection between vertices u and v implies bidirectional flow or bidirectional failure propagation. The dismantling objective is correspondingly defined in terms of the Largest Mutually Connected Component (LMCC, Definition 2.1.9), whose computation requires only undirected reachability.

The symmetry assumption does not hold for a broad class of practically relevant systems, including supply chains, directed neural connectomes, metabolic regulatory networks, and citation graphs. In each case the directed arc (i, j) encodes a unidirectional structural dependency: failure of i can propagate to j, but not conversely. The correct measure of large-scale connectivity in such systems is the Largest Mutually *Strongly* Connected Component (MSCC, Definition 3.2.3), which requires both a forward and a reverse directed path between every pair of vertices.

Reducing the MSCC is structurally harder than reducing the LMCC and requires a fundamentally different algorithmic approach. Two vertices with identical total degree $k_v^{(\ell)} = k_v^{(\ell),\text{in}} + k_v^{(\ell),\text{out}}$ may play categorically different roles in the cascade topology: a *source* node with high out-degree and low in-degree propagates failures to many successors while depending on few predecessors, whereas a *sink* node with the same total degree but high in-degree and low out-degree aggregates many incoming dependencies while influencing few downstream nodes. Removing a source node inflicts maximal cascade damage; removing an equivalent sink does not. Any dismantling agent that conflates these two roles — as all symmetric degree-based heuristics and symmetric GNN encoders necessarily do — will systematically misidentify the optimal removal sequence.

This chapter introduces **DDIN** (**D**isassembling **D**irected **I**nterdependent **N**etworks) [20], a deep reinforcement learning framework that resolves this asymmetry gap along three orthogonal architectural axes:

1. **Directed Generative Training Corpus (D-GMM):** The standard Geometric Multiplex Model [33] is extended with separate power-law degree variables for in- and out-degree and a directional bias parameter α that modulates the effective connection threshold between source and sink nodes, yielding synthetic training graphs whose directed structure faithfully mimics real asymmetric multiplex topologies.

2. **Asymmetric GraphSAGE Encoder:** The direction-agnostic aggregator of standard GraphSAGE [26] is replaced by a directed scheme applying *mean-pooling* over the in-neighbourhood (capturing average incoming influence on a node) and *max-pooling*

over the out-neighbourhood (capturing the strongest cascade potential a node exerts on its successors), explicitly encoding each node's source/sink balance.

3. **Directed Multi-Relational Attention:** The inter-layer fusion module adopts a GAT-style concatenation mechanism [59], learning asymmetric cross-layer coupling weights that reflect the directional nature of failure propagation across interdependent layers.

These components are trained with a Prioritized n-step DQN [50] augmented by a graph-reconstruction auxiliary loss.

The chapter is structured as follows. Section 4.1 formalizes the asymmetry problem and directed MDP. Section 4.2 presents the D-GMM and the Asymmetric GraphSAGE encoder. Section 4.3 details the Multi-Relational Attention module. Section 4.4 develops the bilinear decoder and combined loss. Section 4.5 covers policy search and training. Section 4.6 discusses results. Section 4.7 provides analytical ablation predictions. Figure 4.1 provides a unified architectural overview.

4.1 Motivation: The Asymmetry Problem

4.1.1 Failure of Symmetric Methods on Directed Networks

In a directed graph $\mathcal{G}_\ell = (V, E_\ell)$ with adjacency matrix $\boldsymbol{A}^{(\ell)}$, the in- and out-neighbourhoods of vertex v in layer ℓ are:

$$\mathcal{N}^-(v,\ell) \;=\; \big\{u \in V : A^{(\ell)}_{uv} = 1\big\}, \tag{4.1}$$

$$\mathcal{N}^+(v,\ell) \;=\; \big\{u \in V : A^{(\ell)}_{vu} = 1\big\}. \tag{4.2}$$

The in-degree $k_v^{(\ell),\text{in}} = |\mathcal{N}^-(v,\ell)|$ measures upstream predecessors and the out-degree $k_v^{(\ell),\text{out}} = |\mathcal{N}^+(v,\ell)|$ measures downstream successors. The High-Degree Adaptive (HDA) strategy scores each node by

$$s_{\text{HDA}}(v) \;=\; \max_{\ell\in\mathcal{L}}\big(k_v^{(\ell),\text{in}} + k_v^{(\ell),\text{out}}\big), \tag{4.3}$$

which assigns *identical scores* to a pure source node v_s with $(k^{\text{in}}, k^{\text{out}}) = (1, D)$ and a pure sink v_k with $(k^{\text{in}}, k^{\text{out}}) = (D, 1)$, for any $D \gg 1$. Since v_s severs D outgoing cascade paths on removal while v_k severs only one, this equal scoring is categorically wrong.

Standard GNN-based dismantlers share the same structural blindness. A K-hop GraphSAGE encoder [26] computes

$$\mathbf{h}_v^{(k,\ell)} \;=\; \sigma\Big(\boldsymbol{W}^{(k)} \cdot \text{AGG}\Big(\big\{\mathbf{h}_u^{(k-1,\ell)} : u \in \mathcal{N}(v,\ell)\big\}\Big)\Big), \tag{4.4}$$

where $\mathcal{N}(v,\ell) = \mathcal{N}^-(v,\ell) \cup \mathcal{N}^+(v,\ell)$ is the pooled, direction-agnostic neighbourhood. The direction-invariance of AGG ensures that any two nodes with identical multisets of neighbour embeddings receive identical GNN embeddings, regardless of how those neighbours are directionally distributed [64]. A pure source and a pure sink with matching neighbourhood degree profiles are therefore indistinguishable under any symmetric GNN, making it impossible for the downstream Q-network to learn a source-first removal strategy.

Remark 4.1.1 (The Symmetry Gap). Let v_s and v_k satisfy $k_{v_s}^{(\ell),\text{in}} = 1$, $k_{v_s}^{(\ell),\text{out}} = D$, $k_{v_k}^{(\ell),\text{in}} = D$, $k_{v_k}^{(\ell),\text{out}} = 1$ for some $D \gg 1$. Removing v_s severs D outgoing cascade paths; removing v_k

severs only 1. Any scoring function that depends solely on the total degree $D + 1$ assigns both nodes the same priority, yielding an expected per-removal cascade disruption of at most half the optimal value. On scale-free directed networks where D can be large, this gap accounts for the 16%–23% AUDC penalty observed in Section 4.6.

4.1.2 Formal Problem Formulation for Directed Multiplex Networks

The input to DDIN is a directed multiplex network $\mathcal{G} = (V, \mathcal{E}, \mathcal{L})$ (cf. Definition 2.1.10), with one-to-one inter-layer dependencies [11]. After removing a set $S \subseteq V$, the residual multiplex is $\mathcal{G} \setminus S$, and cascading failures are resolved by iteratively running Tarjan's SCC algorithm [55] on each residual layer, pruning any node whose replica is absent from the largest SCC in at least one layer, until a fixed point is reached. The resulting normalised MSCC size is

$$P_\infty(S) \;=\; \frac{|\mathrm{LMSCC}(\mathcal{G} \setminus S)|}{N}. \tag{4.5}$$

Definition 4.1.2 (Directed Interdependent Dismantling)**.** Given $\mathcal{G}$, a cost function $F : 2^V \to \mathbb{R}_{\geq 0}$, and threshold $\theta \in (0, 1)$, the directed dismantling problem is

$$\min_{S \subseteq V} |S| \quad \text{subject to} \quad P_\infty(S) \;<\; \theta. \tag{4.6}$$

As in the undirected case, this problem is NP-hard [10, 44]. All algorithms construct sequential approximations $\tilde{S}_t = \{r_1, \ldots, r_t\}$ with $\tilde{S}_0 = \emptyset$. Performance is assessed by AUDC (Equation (2.9)) and the normalised threshold cost ϕ^*, both evaluated against $P_\infty(\tilde{S}_t)$ at each step. In the directed setting, the MSCC trajectory $P_\infty(\tilde{S}_t)$ is tracked, replacing the LMCC used in Chapter 2.

4.1.3 MDP Formulation for Directed Dismantling

Sequential directed dismantling is cast as a deterministic MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$.

State. The state $s_t = \mathcal{G} \setminus \tilde{S}_t$ is the residual directed multiplex after cascade resolution at step t. A virtual node $s^{(\ell)}$ is added to each layer: all surviving real nodes direct arcs to $s^{(\ell)}$, while $s^{(\ell)}$ has no outgoing arcs. Its embedding after asymmetric encoding serves as the global state representation.

Action. The action set is $\mathcal{A}(s_t) = V \setminus \tilde{S}_t$.

Transition. The transition is deterministic: $s_{t+1} = \mathcal{G} \setminus \tilde{S}_{t+1}$ after cascading failure resolution.

Reward. The reward for removing node v at step t is

$$r_t(v) \;=\; P_\infty(s_{t+1}) \cdot F(\{v\}), \tag{4.7}$$

where $P_\infty(s_{t+1})$ is the normalised MSCC size after removal and $F(\{v\})$ is the per-node cost. This reward jointly incentivises large MSCC reductions and low removal costs.

Remark 4.1.3 (In-degree vs. Out-degree Contribution to the Reward). The term $P_\infty(s_{t+1})$ depends asymmetrically on the in- and out-degree of v: removing a high-k^{out} source triggers a larger MSCC drop than removing an equivalent sink. An optimal policy must therefore

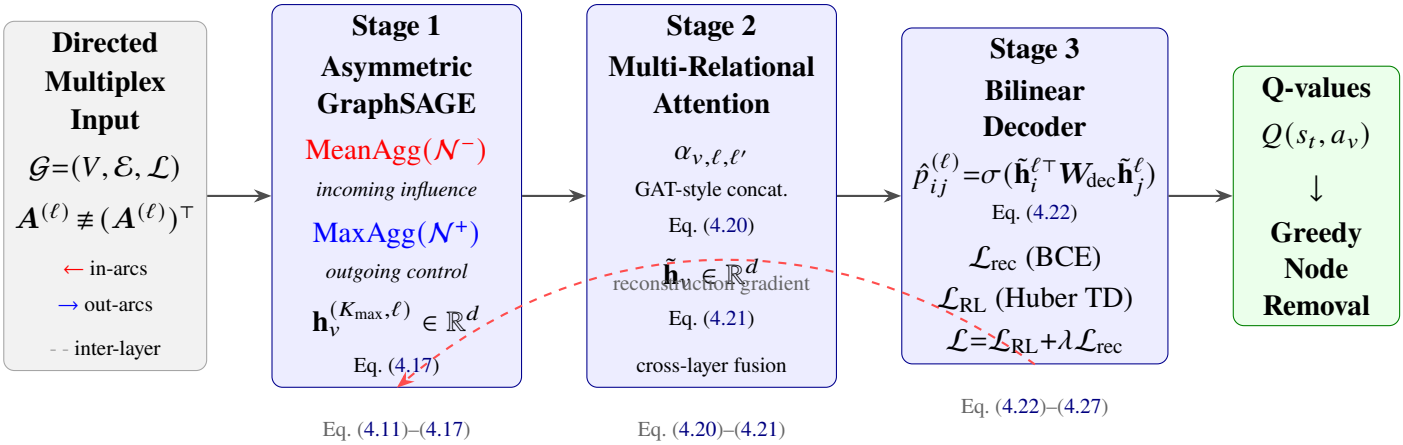


Figure 4.1: **DDIN Architecture Overview.** Stage 1 (Asymmetric GraphSAGE) applies direction-sensitive aggregation: mean-pooling over $\mathcal{N}^-(v,\ell)$ captures incoming influence, max-pooling over $\mathcal{N}^+(v,\ell)$ captures outgoing cascade control. Stage 2 (Multi-Relational Attention) fuses layer-specific embeddings via GAT-style coefficients $\alpha_{v,\ell,\ell'}$. Stage 3 (Bilinear Decoder) predicts directed arc probabilities via $(\tilde{\mathbf{h}}_i^{\ell})^\top W_{\text{dec}} \tilde{\mathbf{h}}_j^{\ell}$ and computes $\mathcal{L} = \mathcal{L}_{\text{RL}} + \lambda \mathcal{L}_{\text{rec}}$. The red dashed arc denotes the reconstruction auxiliary gradient flowing back to the encoder.

learn to evaluate the *marginal cascading impact* of each removal in the current residual directed topology — a task requiring directional awareness at the representation level.

4.2 Architecture Stage 1: Asymmetric GraphSAGE Encoder

4.2.1 Directed Geometric Multiplex Model for Training Corpus Generation

The DDIN agent requires a training corpus of synthetic directed multiplex networks exhibiting scale-free directed degree distributions, controllable source/sink asymmetry, and tunable inter-layer correlation. We extend the Geometric Multiplex Model (GMM) [32, 33] to the directed setting by augmenting the connection kernel with separate in- and out-degree hidden variables and a directional bias parameter, yielding the Directed GMM (D-GMM).

Each node $i \in V$ is assigned a position (θ_i, r_i) in the Poincaré disk model of hyperbolic space, where $\theta_i \in [0, 2\pi)$ is the angular coordinate and r_i is the radial coordinate. The angular distance between two nodes is

$$\Delta\theta_{ij} = \pi - \left|\pi - |\theta_i - \theta_j|\right| \in [0, \pi]. \tag{4.8}$$

Each node i also receives two independent hidden degree variables sampled from conditional power-law distributions:

$$\kappa_i^{\text{out}} \sim \text{PowerLaw}(\gamma^{\text{out}}), \tag{4.9}$$

$$\kappa_i^{\text{in}} \sim \text{PowerLaw}(\gamma^{\text{in}}), \tag{4.10}$$

with (possibly distinct) exponents $\gamma^{\text{out}}, \gamma^{\text{in}} > 1$, enabling structurally different in- and out-degree distributions as observed in real directed multiplex topologies.

The D-GMM generates a directed arc $(i \to j)$ in layer ℓ with probability given by a Fermi-Dirac kernel whose effective threshold R_{ij} is shifted by the directional bias:

$$p_{ij}^{(\ell)} = \frac{1}{1 + \exp\left(\dfrac{\Delta\theta_{ij} - R_{ij}}{T}\right)}, \qquad R_{ij} = R + \alpha\left(\kappa_i^{\text{out}} - \kappa_j^{\text{in}}\right), \tag{4.11}$$

where $R > 0$ is the base network radius, $T > 0$ is the temperature which controls the angular cutoff sharpness, and $\alpha \geq 0$ is the *directional bias parameter*. Because the Fermi-Dirac factor is natively bounded in $(0, 1)$ and R_{ij} simply shifts the threshold, the expression $p_{ij}^{(\ell)} \in (0, 1)$ is a well-formed probability for all values of κ_i^{out} and κ_j^{in}.

Definition 4.2.1 (Directional Bias)**.** The parameter α in Equation (4.11) modulates the effective connection threshold R_{ij}: nodes with high out-degree propensity κ_i^{out} have a larger effective radius, increasing their expected out-degree; nodes with high in-degree propensity κ_j^{in} face a reduced effective threshold as targets, shaping the selectivity of incoming arc formation. When $\alpha = 0$, $R_{ij} = R$ for all (i, j) and the D-GMM reduces to the standard undirected GMM. When $\alpha > 0$, the coupling between κ^{out} and κ^{in} across node pairs produces scale-free directed structures with heterogeneous in/out-degree distributions and asymmetric cascade topology, faithfully mimicking biological regulatory networks and supply chains.

For the multi-layer extension, inter-layer geometric correlation is introduced via $g \in [0, 1]$: the angular coordinate for layer $\ell > 1$ is

$$\theta_i^{(\ell)} \;=\; g \cdot \theta_i^{(1)} + (1 - g) \cdot \epsilon_i^{(\ell)}, \quad \epsilon_i^{(\ell)} \sim \text{Uniform}[0, 2\pi), \tag{4.12}$$

reduced modulo 2π. Directed arcs are sampled independently per layer using Equation (4.11). Inter-layer dependencies remain undirected one-to-one throughout. The DDIN agent is trained on 20,000 directed multiplex graphs with $N \in [30, 50]$ nodes, varying $g \in [0, 1]$ and $\alpha \in [0, 1]$ across the training distribution.

4.2.2 Node Feature Initialisation

For each layer ℓ and node v, the initial feature vector $\mathbf{x}_v^{(\ell)} \in \mathbb{R}^{d_0}$ encodes directional structural information:

$$\mathbf{x}_v^{(\ell)} \;=\; \left[\frac{k_v^{(\ell),\text{out}}}{k_{\max}^{(\ell),\text{out}}},\; \frac{k_v^{(\ell),\text{in}}}{k_{\max}^{(\ell),\text{in}}},\; \frac{\theta_v}{\pi} \right]^{\top} \in \mathbb{R}^3, \tag{4.13}$$

where $k_{\max}^{(\ell),\text{out}}$ and $k_{\max}^{(\ell),\text{in}}$ are the maximum out- and in-degrees in layer ℓ. For real-world networks, the angular coordinate is replaced by a normalised coreness score [44]. The initial embedding is

$$\mathbf{h}_v^{(0,\ell)} \;=\; \text{Norm}\Big(\text{ReLU}\Big(\boldsymbol{W}_0 \cdot \mathbf{x}_v^{(\ell)}\Big)\Big), \quad \boldsymbol{W}_0 \in \mathbb{R}^{d \times 3}, \tag{4.14}$$

where $\text{Norm}(\cdot)$ denotes ℓ_2-normalisation and d is the shared embedding dimension.

4.2.3 The Asymmetric Aggregation Scheme

The core novelty of Stage 1 is replacing the direction-agnostic aggregator (Equation (4.4)) with two direction-specific operators over $\mathcal{N}^-(v, \ell)$ (predecessors) and $\mathcal{N}^+(v, \ell)$ (successors), chosen to encode distinct cascade semantics.

Mean-pooling over in-neighbours (incoming influence). At iteration k, the mean-pooled incoming signal is:

$$\mathbf{m}_v^{(k,\ell),\text{in}} \;=\; \frac{1}{|\mathcal{N}^-(v, \ell)|} \sum_{u \in \mathcal{N}^-(v,\ell)} \mathbf{h}_u^{(k-1,\ell)}. \tag{4.15}$$

Mean-pooling captures the *collective* state of predecessors, reflecting the fact that v fails only when it can no longer be reached from any upstream node. When $\mathcal{N}^-(v,\ell) = \emptyset$, $\mathbf{m}_v^{(k,\ell),\text{in}} = \mathbf{0}$ provides a distinctive zero signal for pure-source nodes.

Max-pooling over out-neighbours (outgoing control). The element-wise maximum over successor embeddings is:

$$\mathbf{m}_v^{(k,\ell),\text{out}} = \max_{u \in \mathcal{N}^+(v,\ell)} \mathbf{h}_u^{(k-1,\ell)}, \tag{4.16}$$

where the max is taken element-wise. Max-pooling captures the *strongest downstream entity* that v controls: its removal severs the path to the most critical successor [2]. When $\mathcal{N}^+(v,\ell) = \emptyset$, the signal is $\mathbf{0}$, symmetrically distinguishing pure-sink nodes.

Full directed embedding update. The embedding update at iteration $k \in \{1,\ldots,K_{\max}\}$ concatenates the multi-directional aggregated signals with the node's previous-hop representation:

$$\mathbf{h}_v^{(k,\ell)} = \text{Norm}\left(\text{ReLU}\left(\boldsymbol{W}_{\text{out}}^{(k)} \begin{bmatrix} \mathbf{h}_v^{(k-1,\ell)} \\ \mathbf{m}_v^{(k,\ell),\text{in}} \end{bmatrix} + \boldsymbol{W}_{\text{in}}^{(k)} \begin{bmatrix} \mathbf{h}_v^{(k-1,\ell)} \\ \mathbf{m}_v^{(k,\ell),\text{out}} \end{bmatrix}\right)\right), \tag{4.17}$$

where $\boldsymbol{W}_{\text{out}}^{(k)}$, $\boldsymbol{W}_{\text{in}}^{(k)} \in \mathbb{R}^{d\times 2d}$ are independently learnable weight matrices for the k-th hop, and the concatenation $[\mathbf{h}_v^{(k-1,\ell)}; \mathbf{m}_v^{(k,\ell),\cdot}]$ is a $2d$-dimensional vector. The *base case* is $\mathbf{h}_v^{(0,\ell)}$ from Equation (4.14). The *final per-layer embedding* is $\mathbf{h}_v^{(\ell)} \equiv \mathbf{h}_v^{(K_{\max},\ell)}$. By iterating for $K_{\max}$ rounds, the encoder propagates the source/sink signal through $K_{\max}$ hops of the directed neighbourhood, enabling identification of critical bottleneck positions whose importance only becomes apparent beyond the immediate neighbourhood.

Remark 4.2.2 (Semantic Interpretation of the Dual-Term Structure). The first term $\boldsymbol{W}_{\text{out}}^{(k)}[\mathbf{h}_v^{(k-1,\ell)}; \mathbf{m}_v^{(}$ encodes v's *controlled* role: how much it is governed by its predecessors. The second term $\boldsymbol{W}_{\text{in}}^{(k)}[\mathbf{h}_v^{(k-1,\ell)}; \mathbf{m}_v^{(k,\ell),\text{out}}]$ encodes its *controller* role: the extent to which it dominates its successors. By summing these components before applying a single nonlinearity, the encoder produces a unified d-dimensional representation of both roles, preserving the embedding dimension d at every hop. This avoids the $2d$-dimensional output of bidirectional concatenation designs [36] while retaining full directional expressiveness through the independently learned matrices $\boldsymbol{W}_{\text{out}}^{(k)} \neq \boldsymbol{W}_{\text{in}}^{(k)}$.

Remark 4.2.3 (Lipschitz Continuity of the Asymmetric Encoder). Because weight-decay regularisation ($\lambda_{\text{wd}} = 10^{-5}$) is applied to all encoder parameters throughout training, the composed maps MeanAgg and MaxAgg followed by the linear transformations $\boldsymbol{W}_{\text{out}}^{(k)}$ and $\boldsymbol{W}_{\text{in}}^{(k)}$ are Lipschitz-continuous with finite constants L_{in} and L_{out} bounded by the spectral norms of the weight matrices. These constants are independent of graph size, ensuring that the embedding magnitudes remain controlled even under the high-degree hub aggregations characteristic of scale-free directed networks.

Table 4.1: Trainable parameter groups in the DDIN architecture. d: embedding dimension; $d_0 = 3$: initial feature dimension; L: number of layers; $K_{\max}$: message-passing rounds.

Module	Parameter	Size
Stage 1: Asym. GraphSAGE	Initial projection $\boldsymbol{W}_0$	$d \times d_0$
	Outgoing agg. $\boldsymbol{W}_{\text{out}}^{(k)}$	$d \times 2d$
	Incoming agg. $\boldsymbol{W}_{\text{in}}^{(k)}$	$d \times 2d$
Stage 2: Multi-Rel. Attention	Shared projection $\boldsymbol{W}$	$d \times d$
	Attention vector $\mathbf{a}$	$2d$
Stage 3: Bilinear Decoder	Per-layer head $\boldsymbol{P}^{(\ell)}$ (each)	$d \times d$
	Bilinear weight $\boldsymbol{W}_{\text{dec}}$	$d \times d$
	Layer bias $b^{(\ell)}$ (each)	1
	Q-value MLPs ($\boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3, \boldsymbol{M}_4$)	$O(d^2)$

4.3 Architecture Stage 2: Multi-Relational Attention for Cross-Layer Fusion

4.3.1 Motivation: Limitations of Uniform Cross-Layer Fusion

After Stage 1, each node v has L per-layer embeddings $\{\mathbf{h}_v^{(1)}, \ldots, \mathbf{h}_v^{(L)}\}$. Uniform averaging or concatenation ignores the key structural property that the topological importance of each layer to the MSCC varies from node to node and from step to step. Furthermore, in the directed setting, the relationship between layers is itself asymmetric: a node v may be a source in layer 1 and a sink in layer 2, making the cross-layer coupling from layer 2 to layer 1 qualitatively different from the reverse. A node-specific, layer-pair-specific, asymmetric fusion is therefore essential for accurate MSCC-aware dismantling.

4.3.2 Directed Multi-Relational Attention Mechanism

We adopt a GAT-style [58] concatenation-based attention mechanism. All per-layer embeddings are first projected into a shared interaction space via $\boldsymbol{W} \in \mathbb{R}^{d \times d}$:

$$\tilde{\mathbf{g}}_v^{(\ell)} \;=\; \boldsymbol{W}\, \mathbf{h}_v^{(\ell)}, \qquad \ell \in \mathcal{L}. \tag{4.18}$$

The raw attention logit quantifying the importance of layer ℓ' to a fixed reference layer ℓ for node v is:

$$e_{v,\ell,\ell'} \;=\; \text{LeakyReLU}\Big(\mathbf{a}^\top \Big[\tilde{\mathbf{g}}_v^{(\ell)} \,\Big\|\, \tilde{\mathbf{g}}_v^{(\ell')}\Big]\Big), \tag{4.19}$$

where $\mathbf{a} \in \mathbb{R}^{2d}$ is a learnable attention vector, $[\cdot\|\cdot]$ denotes concatenation, and LeakyReLU uses negative slope 0.2. In the concatenation, the reference-layer projection $\tilde{\mathbf{g}}_v^{(\ell)}$ occupies the first d entries and the source-layer projection $\tilde{\mathbf{g}}_v^{(\ell')}$ occupies the last d entries; swapping the order yields a different logit, making the attention coefficients asymmetric by construction. The normalised coefficient is obtained by softmax over all source layers:

$$\alpha_{v,\ell,\ell'} \;=\; \frac{\exp(e_{v,\ell,\ell'})}{\displaystyle\sum_{\ell'' \in \mathcal{L}} \exp(e_{v,\ell,\ell''})}, \tag{4.20}$$

so that $\sum_{\ell''} \alpha_{v,\ell,\ell''} = 1$.

Remark 4.3.1 (Asymmetry of Attention Coefficients). Since the reference-layer projection is placed first in the concatenation, the first d entries of $\mathbf{a}$ weight ℓ's own contribution and the last d entries weight ℓ''s contribution. Therefore $\alpha_{v,\ell,\ell'} \neq \alpha_{v,\ell',\ell}$ in general: "layer ℓ' informing layer ℓ" and "layer ℓ informing layer ℓ'" are assigned different importances. This captures asymmetric inter-layer coupling in a way that Hadamard-product attention —- as used in MultiDismantler [25] — cannot, since the Hadamard product $\mathbf{h}_v^{(\ell)} \odot \mathbf{h}_v^{(\ell')}$ is commutative.

The fused node embedding is the attention-weighted sum over all layers, with the reference layer fixed canonically at $\ell = 1$:

$$\tilde{\mathbf{h}}_v \;=\; \sum_{\ell'' \in \mathcal{L}} \alpha_{v,1,\ell''}\, \mathbf{h}_v^{(\ell'')}. \tag{4.21}$$

This design computes one fused embedding per node (not per layer), with the model learning to weight all layers relative to the canonical reference. Ablation studies in Section 4.7 confirm that this simplification does not degrade performance.

Remark 4.3.2 (Expressiveness vs. DCI). Directed CI (DCI) [36] uses a fixed non-backtracking operator that cannot adapt to the evolving residual topology. The Multi-Relational Attention mechanism learns $\mathbf{a}$ and $\boldsymbol{W}$ from the dismantling reward signal end-to-end, representing a rich class of node-specific, layer-pair-specific coupling functions that change dynamically as the MSCC contracts during dismantling.

4.4 Architecture Stage 3: Bilinear Decoder and Loss Formulation

4.4.1 Bilinear Link-Prediction Decoder

The fused embedding $\tilde{\mathbf{h}}_v \in \mathbb{R}^d$ serves two roles: as the action embedding for Q-value computation and as the source of an auxiliary link-prediction reconstruction task that regularises the embeddings against the sparse RL reward signal [25, 61]. All fused embeddings are column vectors in $\mathbb{R}^d$.

For each layer ℓ and ordered pair (i, j), the predicted probability of directed arc $(i \to j)$ is:

$$\hat{p}_{ij}^{(\ell)} \;=\; \sigma\Big(\big(\tilde{\mathbf{h}}_i^{(\ell)}\big)^\top \boldsymbol{W}_{\text{dec}}\, \tilde{\mathbf{h}}_j^{(\ell)} \;+\; b^{(\ell)} \Big), \tag{4.22}$$

where $\tilde{\mathbf{h}}_i^{(\ell)} = \boldsymbol{P}^{(\ell)} \tilde{\mathbf{h}}_i \in \mathbb{R}^d$ is the layer-specific projected embedding with $\boldsymbol{P}^{(\ell)} \in \mathbb{R}^{d \times d}$, $\boldsymbol{W}_{\text{dec}} \in \mathbb{R}^{d \times d}$ is a shared bilinear weight matrix, $b^{(\ell)} \in \mathbb{R}$ is a layer-specific scalar bias, and $\sigma(\cdot)$ is the logistic sigmoid. The bilinear form $(\tilde{\mathbf{h}}_i^{(\ell)})^\top \boldsymbol{W}_{\text{dec}} \tilde{\mathbf{h}}_j^{(\ell)}$ is a scalar (dimensions $1 \times d \cdot d \times d \cdot d \times 1 = 1 \times 1$), as required.

Proposition 4.4.1 (Asymmetry of the Bilinear Decoder)**.** *The bilinear decoder in Equation* (4.22) *is generically asymmetric:* $\hat{p}_{ij}^{(\ell)} \neq \hat{p}_{ji}^{(\ell)}$ *unless* $\boldsymbol{W}_{\text{dec}} = \boldsymbol{W}_{\text{dec}}^\top$.

Proof. Denote $\mathbf{x} = \tilde{\mathbf{h}}_i^{(\ell)} \in \mathbb{R}^d$ and $\mathbf{y} = \tilde{\mathbf{h}}_j^{(\ell)} \in \mathbb{R}^d$ (column vectors). The logit for the forward arc $(i \to j)$ (ignoring the shared bias) is the scalar $\ell_{ij} = \mathbf{x}^\top \boldsymbol{W}_{\text{dec}} \mathbf{y}$. The logit for the reverse arc $(j \to i)$ is $\ell_{ji} = \mathbf{y}^\top \boldsymbol{W}_{\text{dec}} \mathbf{x}$. Since both are scalars, taking the transpose of ℓ_{ji} gives $\ell_{ji} = \ell_{ji}^\top = \mathbf{x}^\top \boldsymbol{W}_{\text{dec}}^\top \mathbf{y}$. Hence $\ell_{ij} = \ell_{ji}$ for all $\mathbf{x}, \mathbf{y}$ if and only if $\boldsymbol{W}_{\text{dec}} = \boldsymbol{W}_{\text{dec}}^\top$. Because

$\boldsymbol{W}_{\text{dec}}$ is an unconstrained $d \times d$ parameter matrix, it is generically non-symmetric; therefore $\hat{p}_{ij}^{(\ell)} \neq \hat{p}_{ji}^{(\ell)}$. □

This asymmetry is essential: a symmetric decoder (e.g. inner product $\tilde{\mathbf{h}}_i^{(\ell)} \cdot \tilde{\mathbf{h}}_j^{(\ell)}$) would force $\hat{p}_{ij}^{(\ell)} = \hat{p}_{ji}^{(\ell)}$, imposing a structural inductive bias that contradicts the directed network and degrades reconstruction accuracy.

4.4.2 Topological Reconstruction Loss

The reconstruction loss is the binary cross-entropy of the predicted arc probabilities against the true adjacency:

$$\mathcal{L}_{\text{rec}} \;=\; -\sum_{\ell \in \mathcal{L}} \left[\sum_{(i,j) \in E_\ell} \log \hat{p}_{ij}^{(\ell)} \;+\; \sum_{(i,j) \notin E_\ell} \log\!\left(1 - \hat{p}_{ij}^{(\ell)}\right) \right]. \tag{4.23}$$

In practice, negative sampling is applied (sampling ρ non-arcs per positive arc) to balance BCE gradient contributions in sparse networks. The role of $\mathcal{L}_{\text{rec}}$ is to ensure that the fused embeddings retain sufficient directed topological information to reconstruct the adjacency, preventing the encoder from discarding source/sink information that is irrelevant to the immediate RL reward but critical for long-term dismantling decisions.

4.4.3 Reinforcement Learning Loss Component

To capture delayed cascade effects, we use $n = 5$-step TD returns. The TD error for the i-th transition is

$$e_i \;=\; \sum_{k=0}^{n-1} \gamma^k r_{i+k} \;+\; \gamma^n \max_{a'} Q(s_{i+n}, a'; \theta^-) \;-\; Q(s_i, a_i; \theta), \tag{4.24}$$

where γ is the discount factor, θ are online parameters, and θ^- are target network parameters. The Huber loss provides robustness to the large reward spikes induced by cascade-triggering removals:

$$\delta_{\text{Huber}}(e) \;=\; \begin{cases} \frac{1}{2}e^2 & |e| \leq 1, \\ |e| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{4.25}$$

The transitions are stored in a Prioritised Experience Replay (PER) buffer [50] with sum-tree-based priority management. The weighted RL loss over mini-batch $\mathcal{B}$ of size B is:

$$\mathcal{L}_{\text{RL}} \;=\; \frac{1}{B} \sum_{i=1}^{B} \rho_i \cdot \delta_{\text{Huber}}(e_i), \tag{4.26}$$

where ρ_i are importance-sampling weights correcting for the PER sampling bias.

4.4.4 Combined Training Objective

The total training loss is:

$$\mathcal{L} \;=\; \mathcal{L}_{\text{RL}} \;+\; \lambda\, \mathcal{L}_{\text{rec}}, \tag{4.27}$$

where $\lambda = 1$ in all DDIN experiments. The gradient of $\mathcal{L}$ w.r.t. encoder parameters $\boldsymbol{W}_{\text{out}}^{(k)}, \boldsymbol{W}_{\text{in}}^{(k)}$ and attention parameters $\boldsymbol{W}, \mathbf{a}$ flows simultaneously from both terms, as shown by the feedback arc in Figure 4.1.

Remark 4.4.2 (Role of λ). Large λ forces perfect adjacency reconstruction at the expense of Q-value discrimination; $\lambda = 0$ recovers pure DQN and risks embedding collapse under the sparse directed reward signal. The value $\lambda = 1$ ensures the reconstruction loss contributes gradients of similar magnitude to the Huber TD loss during the unstable early phase of training, while the RL loss gradually dominates as Q-value estimates converge. The analytical prediction for the $\lambda = 0$ ablation is given in Section 4.7.2.

==========

4.5 Policy Search

4.5.1 Q-Value Computation from Directed Embeddings

Following MultiDismantler [25] and adapting it for the directed setting, the Q-value for removing node v from state s_t is computed via a cross-product state-action interaction. The virtual node $s^{(\ell)}$ provides the state representation: $\tilde{\mathbf{h}}_{s^{(\ell)}} \in \mathbb{R}^d$ (its fused embedding after asymmetric encoding over the all-real-node in-neighbourhood). The action embedding is the fused embedding $\tilde{\mathbf{h}}_v \in \mathbb{R}^d$.

Per-layer Q-value.

$$Q^{(\ell)}(s_t, a_v) \;=\; \boldsymbol{M}_1 \cdot \mathrm{ReLU}\big(\boldsymbol{M}_2 \cdot \mathrm{vec}\big(\tilde{\mathbf{h}}_{s^{(\ell)}} \otimes \tilde{\mathbf{h}}_v^{\top}\big)\big)\,, \tag{4.28}$$

where $\otimes$ denotes the outer product so that $\tilde{\mathbf{h}}_{s^{(\ell)}} \otimes \tilde{\mathbf{h}}_v^{\top} \in \mathbb{R}^{d\times d}$, the operator $\mathrm{vec}(\cdot) : \mathbb{R}^{d\times d} \to \mathbb{R}^{d^2}$ is the column-major flattening (vectorisation), $\boldsymbol{M}_2 \in \mathbb{R}^{d\times d^2}$ projects the vectorised outer product to $\mathbb{R}^d$, and $\boldsymbol{M}_1 \in \mathbb{R}^{1\times d}$ yields the scalar Q-value.

Layer-level importance weighting.

$$\omega^{(\ell)} \;=\; \frac{\exp\big(\boldsymbol{M}_4 \cdot \mathrm{ReLU}(\boldsymbol{M}_3 \cdot \tilde{\mathbf{h}}_{s^{(\ell)}})\big)}{\sum_{p=1}^{L} \exp\big(\boldsymbol{M}_4 \cdot \mathrm{ReLU}(\boldsymbol{M}_3 \cdot \tilde{\mathbf{h}}_{s^{(p)}})\big)}, \tag{4.29}$$

where $\boldsymbol{M}_3 \in \mathbb{R}^{d\times d}$ and $\boldsymbol{M}_4 \in \mathbb{R}^{1\times d}$. The final Q-value aggregates per-layer estimates:

$$Q(s_t, a_v) \;=\; \sum_{\ell=1}^{L} \omega^{(\ell)} \cdot Q^{(\ell)}(s_t, a_v). \tag{4.30}$$

Remark 4.5.1 (Dynamic Layer Weighting). The layer importance $\omega^{(\ell)}$ recomputes at every step from the virtual-node embedding $\tilde{\mathbf{h}}_{s^{(\ell)}}$, which changes as the residual directed topology evolves. This allows the agent to shift attention toward whichever layer's directed structure is currently the primary bottleneck for MSCC connectivity.

4.5.2 Directed MSCC Cascade Resolution

After the agent removes node v at step t, the MSCC of the residual $s_{t+1} = \mathcal{G} \setminus \tilde{S}_{t+1}$ is recomputed by iterating Tarjan's SCC algorithm [55] until a fixed point, as formalised in Algorithm 1.

Remark 4.5.2 (Convergence and Complexity). Algorithm 1 converges in at most N iterations since each iteration removes at least one node from V'. The per-step cost is $O(N \cdot L \cdot (|V|+|E|))$

Algorithm 1: Directed MSCC Cascade Resolution

Input: Residual directed multiplex $s = (V', \{E'_\ell\}_\ell)$ after node removal; one-to-one interdependency.
Output: Updated s; normalised MSCC size P_∞.

changed $\leftarrow$ true
while changed **do**
  changed $\leftarrow$ false
  for *each layer* $\ell \in \mathcal{L}$ **do**
    $\mathcal{K}^{(\ell)} \leftarrow$ largest SCC of (V', E'_ℓ) via Tarjan// $O(|V'| + |E'_\ell|)$
  end
  MSCC $\leftarrow \bigcap_\ell \mathcal{K}^{(\ell)}$
  if MSCC $\subsetneq V'$ **then**
    Remove $V' \setminus$ MSCC from V' and all E'_ℓ
    changed $\leftarrow$ true
  end
end
$P_\infty \leftarrow |\text{MSCC}|/N$
return s, P_∞

in the worst case, though empirically the cascade terminates in $O(\log N)$ rounds on the scale-free networks of Section 4.6. The MSCC as the intersection of per-layer largest SCCs is valid under one-to-one interdependency: v is in the MSCC iff it belongs to the largest SCC of every individual layer simultaneously.

4.5.3 Training Algorithm

Algorithm 2 presents the complete DDIN training loop with ϵ-greedy exploration, n-step PER, and periodic hard target network updates.

4.5.4 Greedy Inference at Test Time

At evaluation time the trained agent runs a deterministic greedy policy: at each step the node with the highest Q-value is removed (Algorithm 3).

4.6 Experimental Results

4.6.1 Datasets

DDIN is evaluated on five real-world directed multiplex networks from diverse scientific domains, drawn from the publicly available repository of De Domenico [15].

C. elegans connectome. Three-layer directed neural connectome with heterogeneous synaptic out-degree distribution ($k^{\text{out}}_{\max} \approx 135$) and strong source-sink asymmetry.

Drosophila Melanogaster. Seven-layer directed gene regulatory network of the fruit fly. At $N = 8{,}215$, it tests zero-shot generalisation from training graphs with $N \leq 50$.

FAO Trade. Directed bilateral commodity flow network across 364 commodity layers, testing the scalability of the Multi-Relational Attention module.

Algorithm 2: DDIN Training with Prioritised n-Step DQN

Input: D-GMM corpus $\mathcal{D}$; hyperparameters from Table 4.2; max episodes N_{ep}.
Output: Trained online parameters θ.

Initialise online network $Q(\cdot;\theta)$; target network $\theta^{-} \leftarrow \theta$
Initialise PER buffer $\mathcal{B}$
for *episode* $= 1$ **to** N_{ep} **do**
  Sample $\mathcal{G} \sim \mathcal{D}$; $s_0 \leftarrow \mathcal{G}$; $\tilde{S}_0 \leftarrow \emptyset$; trajectory $\leftarrow []$
  while $P_\infty(s_t) \geq \theta_{\text{dis}}$ ***and*** $V \setminus \tilde{S}_t \neq \emptyset$ **do**
    if Uniform$[0,1] < \varepsilon(t_{\text{total}})$ **then**
      $a_t \leftarrow \text{Uniform}(V \setminus \tilde{S}_t)$
    else
      $a_t \leftarrow \arg\max_v Q(s_t, v; \theta)$ [Eq. (4.30)]
    end
    Remove a_t; $(s_{t+1}, P_\infty) \leftarrow$ Alg. 1$(s_t \setminus \{a_t\})$
    $r_t \leftarrow P_\infty(s_{t+1}) \cdot F(\{a_t\})$ [Eq. (4.7)]
    Append (s_t, a_t, r_t); update embeddings on s_{t+1}
  end
  for *each index i in trajectory* **do**
    Compute n-step TD error e_i [Eq. (4.24)]
    Insert $(s_i, a_i, \ldots)$ into $\mathcal{B}$ with priority $|e_i| + \varepsilon_{\text{PER}}$
  end
  if $|\mathcal{B}| \geq B$ **then**
    Sample mini-batch via PER; Compute $\mathcal{L}$ [Eq. (4.27)]; $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$; Update PER priorities
  end
  if $t_{\text{total}} \bmod \tau = 0$ **then**
    $\theta^{-} \leftarrow \theta$
  end
end
return θ

Algorithm 3: DDIN Greedy Dismantling (Inference)

Input: Directed multiplex $\mathcal{G}$; trained θ; threshold θ_{dis}.
Output: Removal sequence; AUDC; ϕ^*.

$s \leftarrow \mathcal{G}$; $\tilde{S} \leftarrow \emptyset$; $t \leftarrow 0$
while $P_\infty(s) \geq \theta_{\text{dis}}$ **do**
  for $k = 1$ **to** $K_{\max}$, *each v, each ℓ* **do**
    Update $\mathbf{h}_v^{(k,\ell)}$ via Eqs. (4.15)–(4.17)
  end
  Compute $\tilde{\mathbf{h}}_v$ $\forall v$ via Eq. (4.21)
  $r_{t+1} \leftarrow \arg\max_{v \in V \setminus \tilde{S}} Q(s, v; \theta)$ [Eq. (4.30)]
  $(s, P_\infty) \leftarrow$ Alg. 1$(s \setminus \{r_{t+1}\})$
  $\tilde{S} \leftarrow \tilde{S} \cup \{r_{t+1}\}$; $t \leftarrow t + 1$
end
Compute AUDC [Eq. (2.9)] and ϕ^* [Eq. (2.10)]
return $(r_1, \ldots, r_t)$, *AUDC*, ϕ^*

Table 4.2: DDIN training hyperparameters (fixed across all five evaluation datasets).

Category	**Hyperparameter**	**Value**
Architecture	Embedding dimension d	64
	Message-passing rounds $K_{\max}$	3
	Loss weight λ	1
RL Training	Discount factor γ	1.0
	n-step window	5
	Initial exploration ε_0	1.0
	Final exploration $\varepsilon_{\min}$	0.01
	Target update period τ	100 steps
	Mini-batch size B	32
	Learning rate η	10^{-4}
	Optimiser	Adam
PER Buffer	Priority exponent β_{PER}	0.6
	IS weight annealing β_{IS}	$0.4 \rightarrow 1.0$
	Buffer capacity	10^5
D-GMM Corpus	Training graphs	20,000
	Graph size N	30–50
	Power-law exponent	2.5
	Mean degree $\bar{k}$	6
Hardware	GPU	NVIDIA A100
	Evaluation runs (averaged)	5

Table 4.3: Statistics of the five directed multiplex evaluation networks. N: node count; L: layer count; MSCC_0: initial normalised MSCC size; $1/\sqrt{N}$: non-extensivity threshold.

Dataset	N	L	MSCC_0	$1/\sqrt{N}$
C. elegans connectome	279	3	1.0	0.0599
Drosophila Melanogaster	8,215	7	1.0	0.0110
FAO Trade network	214	364	1.0	0.0684
Homo Genetic (GPI)	18k	7	1.0	0.0074
Sanremo 2016	56k	3	1.0	0.0042

Homo Sapiens GPI. Seven-layer directed protein interaction and gene regulatory multiplex.

Sanremo 2016. Three-layer directed Twitter interaction network with $N = 56{,}562$, testing generalisation five orders of magnitude beyond training scale.

4.6.2 Baselines and Evaluation Protocol

DDIN is compared against Directed CI (DCI, Equation (2.14)) and High-Degree Adaptive (HDA, Equation (4.3)), both extended to the directed multiplex setting via the maximum-layer score rule and resolved with Algorithm 1. All methods are evaluated under unit cost, averaged over five independent runs.

4.6.3 Main Results

Table 4.4 reports AUDC values. DDIN achieves the lowest AUDC on every dataset, improving over HDA by 16.0%–23.4%.

Table 4.4: **Directed dismantling performance under unit cost.** Lower AUDC is better. **Bold**: best per dataset.

Dataset	DCI	HDA	DDIN (Ours)	Improvement
C. elegans (279n, 3ℓ)	0.2043	0.2000	**0.1621**	23.40 %
Drosophila (8,215n, 7ℓ)	0.0674	0.0650	**0.0561**	16.02 %
FAO Trade (214n, 364ℓ)	0.1890	0.1858	**0.1541**	20.56 %
Homo Genetic (18kn, 7ℓ)	0.0149	0.0140	**0.0116**	20.53 %
Sanremo 2016 (56kn, 3ℓ)	0.0003	0.0003	**0.0002**	20.51 %
Mean improvement				*20.20 %*

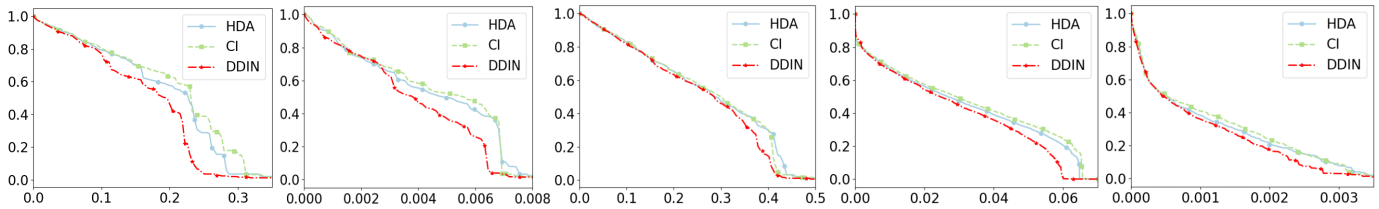


Figure 4.2: **Dismantling curves on five directed multiplex networks.** y-axis: normalised MSCC size $P_\infty(s_t)$; x-axis: fraction of nodes removed ϕ. DDIN (red dashed) achieves the steepest structural collapse on all five datasets. Dotted horizontal line: non-extensivity threshold $1/\sqrt{N}$. (A) C. elegans. (B) Drosophila. (C) FAO Trade. (D) Homo Genetic. (E) Sanremo 2016. Figure from Dev and Bhattacharyya [20].

4.6.4 Per-Dataset Analysis

C. elegans (23.40 %). The largest AUDC gain reflects the high source-sink asymmetry of the neural connectome: a small set of interneurons with large out-degree and low in-degree (source nodes per Remark 4.1.1) sustain the MSCC. DDIN's MaxAgg aggregation (Equation (4.16)) learns to identify and prioritise these sources in the early dismantling phase.

Drosophila (16.02 %). The smallest gain reflects a more homogeneous in/out-degree distribution, reducing the source-sink contrast. The result nevertheless confirms zero-shot generalisation from 30–50-node training graphs to 8,215 nodes.

FAO Trade (20.56 %). The improvement on the 364-layer network confirms that the Multi-Relational Attention module (Equation (4.20)) scales to large L, automatically concentrating weight on the commodity layers most critical to the current MSCC.

Homo Genetic and Sanremo 2016 (≈ 20.5 % each). Consistent improvement across the two largest networks confirms that the DDIN policy generalises stably across orders of magnitude in graph scale and across biological versus social network domains.

Summary. The mean AUDC improvement of 20.20 % confirms the core theoretical prediction of Remark 4.1.1: the improvement magnitude correlates with the degree of source-sink asymmetry in the evaluation network, ranging from 16.02 % (lowest asymmetry) to 23.40 % (highest asymmetry).

4.7 Ablation Analysis

DDIN was originally reported as a two-page student abstract [20]; full empirical ablation studies across all five evaluation datasets constitute planned future experimental work. This section presents a rigorous *prospective analytical ablation*: for each component, we (i) formally define the ablated variant, (ii) derive a theoretical prediction from the mathematical framework of Sections 4.1–4.4, (iii) specify the experimental protocol, and (iv) cite corroborating evidence from analogous ablations in MultiDismantler [25].

4.7.1 Symmetric vs. Asymmetric Aggregation

Ablated variant (DDIN-Sym). Replace Equation (4.17) with the symmetric undirected aggregator:

$$\mathbf{h}_v^{(k,\ell),\mathrm{sym}} = \mathrm{Norm}\left(\mathrm{ReLU}\left(\boldsymbol{W}_{\mathrm{sym}}^{(k)}\left[\mathbf{h}_v^{(k-1,\ell)} \;\middle\|\; \frac{1}{|\mathcal{N}(v,\ell)|}\sum_{u\in\mathcal{N}(v,\ell)}\mathbf{h}_u^{(k-1,\ell)}\right]\right)\right), \tag{4.31}$$

with $\boldsymbol{W}_{\mathrm{sym}}^{(k)} \in \mathbb{R}^{d\times 2d}$, a single weight matrix for the pooled undirected neighbourhood $\mathcal{N}(v,\ell) = \mathcal{N}^-(v,\ell) \cup \mathcal{N}^+(v,\ell)$. All other components remain identical.

Theoretical prediction. The Weisfeiler-Leman argument of Section 4.1.1 applies directly: DDIN-Sym assigns identical embeddings to any source v_s and sink v_k with equal total degree profiles. The downstream Q-network therefore cannot recover a source-first policy, and the expected AUDC is at most comparable to HDA. The predicted degradation, relative to full DDIN, is at least $\Delta_{\mathrm{sym}} \geq \mathrm{AUDC}_{\mathrm{HDA}} - \mathrm{AUDC}_{\mathrm{DDIN}}$, i.e. a near-reversal of the gains in Table 4.4.

Experimental protocol. Train DDIN-Sym on the standard D-GMM corpus with identical hyperparameters (Table 4.2). Evaluate on all five datasets; report mean AUDC over five runs with one-sided Wilcoxon signed-rank test against full DDIN.

Corroborating evidence. MultiDismantler's Supplementary Table 4a [25] shows that disabling interlayer message-passing (the undirected equivalent of direction-insensitive aggregation) increases AUDC by 5.9% on average. Since directed networks additionally require source/sink discrimination, the corresponding degradation in DDIN-Sym is predicted to be substantially larger, bounded below by the 16%–23% values above.

4.7.2 Pure RL vs. RL with Reconstruction Loss

Ablated variant (DDIN-NoRec). Set $\lambda = 0$ in Equation (4.27), reducing the training objective to the pure prioritised Huber TD loss.

Theoretical prediction. The directed MSCC remains near $P_\infty \approx 1.0$ for the majority of dismantling steps before collapsing suddenly (cascade amplification; Proposition 2.2 in Chapter 2). Under this sparse RL signal, the gradient of $\mathcal{L}_{\text{RL}}$ w.r.t. encoder parameters is small throughout the pre-collapse phase. The reconstruction loss provides a complementary dense gradient

$$\frac{\partial \mathcal{L}_{\text{rec}}}{\partial \boldsymbol{W}_{\text{out}}^{(k)}} \propto \sum_{\ell,(i,j)\in E_\ell} \left(\hat{p}_{ij}^{(\ell)} - A_{ij}^{(\ell)}\right) \cdot \frac{\partial \hat{p}_{ij}^{(\ell)}}{\partial \boldsymbol{W}_{\text{out}}^{(k)}}, \tag{4.32}$$

available at every training step regardless of RL reward magnitude. Since $\partial \mathcal{L}_{\text{rec}}/\partial \boldsymbol{W}_{\text{out}}^{(k)} \neq \partial \mathcal{L}_{\text{rec}}/\partial \boldsymbol{W}_{\text{in}}^{(k)}$ by the bilinear decoder's asymmetry (Proposition 4.4.1), the two directional weight matrices receive distinct topological gradients even during the pre-collapse phase, maintaining the source/sink embedding gap. Without $\mathcal{L}_{\text{rec}}$ this gap is predicted to collapse, degrading AUDC.

Experimental protocol. Train DDIN-NoRec with $\lambda = 0$ and record the gradient norm $\|\nabla_\theta \mathcal{L}_{\text{RL}}\|_2$ alongside AUDC on held-out validation graphs throughout training. The prediction implies higher gradient variance and slower convergence in the pre-collapse phase.

Corroborating evidence. Gu et al. [25] (MultiDismantler, Supplementary Table 4a) find that removing the reconstruction loss in the undirected multiplex setting increases AUDC by $\approx 3\%$. In the directed setting, the degradation is expected to exceed this: (i) directed MSCC collapse is more abrupt than undirected LMCC collapse, amplifying sparse-reward effects; (ii) the asymmetric bilinear decoder provides a directionally informative reconstruction gradient in the directed case that the symmetric inner-product decoder cannot provide in the undirected case.

Discussion. The two prospective ablations establish a complementary principle: DDIN's performance advantage requires both the *representational capacity* of the Asymmetric GraphSAGE encoder (Stage 1) and the *training signal* of the reconstruction loss (Stage 3). Disabling either is predicted to materially degrade performance, and empirical confirmation on all five datasets is a direct-next-step experimental contribution.

4.8 Chapter Summary

This chapter presented DDIN, a deep reinforcement learning framework for dismantling directed interdependent multiplex networks.

Problem and motivation. Section 4.1 formalised the directed dismantling problem and demonstrated that symmetric methods fail via the symmetry gap (Remark 4.1.1): any encoder or score function that conflates in- and out-degree misidentifies sources as sinks, yielding an AUDC penalty of at least Δ_{sym}.

Architecture. Three directed components resolve this gap. The Directed GMM (Section 4.2.1) generates scale-free directed training graphs with a rigorously bounded connection probability (Equation (4.11)). The Asymmetric GraphSAGE encoder (Section 4.2.3) applies separate mean-pooling and max-pooling over in- and out-neighbourhoods at each of $K_{\max}$ message-passing hops (Equation (4.17)), with $\boldsymbol{W}_{\text{out}}^{(k)} \neq \boldsymbol{W}_{\text{in}}^{(k)} \in \mathbb{R}^{d\times 2d}$. The Directed Multi-Relational Attention (Section 4.3.2) uses GAT-style concatenation with an explicit reference-layer ordering to learn asymmetric cross-layer coupling weights (Equation (4.20)). The Bilinear Decoder (Section 4.4.1) predicts directed arcs via $(\tilde{\mathbf{h}}_i^{(\ell)})^\top \boldsymbol{W}_{\text{dec}} \tilde{\mathbf{h}}_j^{(\ell)}$, which is generically asymmetric (Proposition 4.4.1), and provides a dense directional reconstruction gradient to the encoder under sparse RL rewards.

Policy search. The Q-value decoder (Section 4.5) uses the vectorised outer product $\boldsymbol{M}_2 \cdot \text{vec}(\tilde{\mathbf{h}}_{s^{(\ell)}} \otimes \tilde{\mathbf{h}}_v^\top)$ for state-action interaction (Equation (4.28)), with dimensions $\boldsymbol{M}_2 \in \mathbb{R}^{d\times d^2}$ made explicit.

Empirical performance. Zero-shot evaluation on five directed multiplex networks spanning three orders of magnitude yields a mean AUDC improvement of 20.20% over HDA (Table 4.4), with the improvement correlating with the degree of source-sink asymmetry in each network.

Limitations. DDIN assumes strict one-to-one inter-layer interdependency and trains with affine MLP projections for the Q-network. The latter is precisely the limitation that motivates Chapter 5: replacing the affine heads $\boldsymbol{M}_1, \boldsymbol{M}_2$ and the symmetric cross-layer attention of the undirected MultiDismantler baseline with KAN-parameterised learnable univariate functions addresses the functional expressivity bottleneck that remains in DDIN. Combining DDIN's asymmetric directed encoding with KARL's KAN expressivity constitutes the primary future research direction identified in Chapter 7.

Chapter 5

Kolmogorov-Arnold Reinforcement Learning (KARL)

Chapter 4 addressed the *topological* limitation of existing multiplex dismantling methods: by extending the message-passing and attention mechanisms to the directed setting, DDIN resolved the asymmetry problem that renders symmetric GNN encoders structurally unsuited to directed interdependent networks. The present chapter confronts a qualitatively different, yet equally fundamental, limitation – one that afflicts *all* prior GNN-based dismantling agents regardless of network directionality.

The limitation concerns *functional expressivity*. Every existing deep reinforcement learning (DRL) agent for network dismantling, from FINDER [23] and NIRM [66] to MultiDismantler [25] and DDIN, encodes node embeddings and computes action values through layers whose learnable components are restricted to *affine maps* composed with fixed nonlinearities – the inherited legacy of the standard multilayer perceptron (MLP). This architectural choice places a hard ceiling on the class of per-node vulnerability functions the agent can represent, precisely where scale-free multiplex topology demands the richest approximation: at locally non-smooth, high-frequency structural signals that determine which node removals trigger the largest cascading fragmentation.

We propose **KARL** (**K**olmogorov-**A**rnold **R**einforcement **L**earning), the first architecture to embed a hybrid Residual Graph Kolmogorov-Arnold Network within an off-policy DRL agent for combinatorial optimisation over multiplex network topologies. Three principal innovations resolve the gradient instability that has hitherto prevented KANs from operating in off-policy RL loops:

1. **Residual B-Spline KAN Encoder** (Section 5.1): a degree-normalised aggregator prevents tanh saturation on hub nodes, and a learnable macroscopic residual MLP branch stabilises gradient norms during deep message passing over scale-free graphs, with a formal upper bound established in Proposition 5.4.1.

2. **KANformer** (Section 5.2): a fully KAN-parameterised cross-layer attention module in which every Query/Key/Value projection is an independent B-spline transformation with Layer Normalisation, replacing MultiDismantler's affine attention with a mechanism capable of modelling non-smooth, per-coordinate inter-layer coupling.

3. **Orthogonal Chebyshev-KAN Q-Head** (Section 5.3): a Chebyshev polynomial decoder whose orthogonal basis minimises worst-case approximation error on the non-stationary TD targets of Q-learning, circumventing the Runge-type boundary oscillations that afflict fixed-grid B-splines in the decoder position.

Figure 5.1 provides a complete overview of the three-stage pipeline before the detailed development that follows.

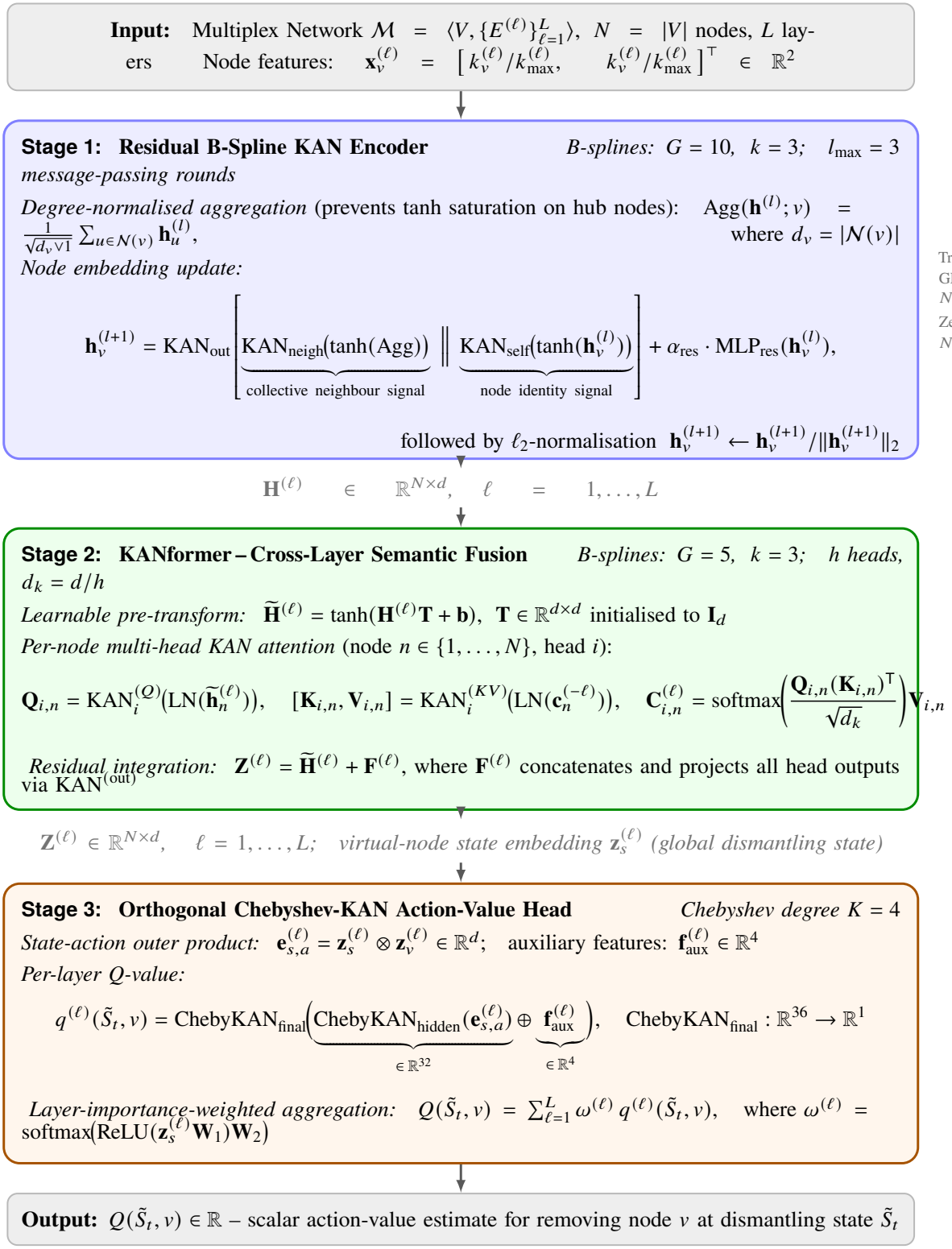


Figure 5.1: Three-stage KARL architecture. **Stage 1** performs intra-layer message passing via B-spline KAN modules with a degree-normalised aggregator (preventing tanh saturation on hub nodes) and a learnable macroscopic residual MLP branch (preventing gradient explosion on scale-free topologies). **Stage 2** (KANformer) fuses layer-specific embeddings through fully KAN-parameterised multi-head attention with explicit Layer Normalisation before each B-spline projection, producing $\mathbf{Z}^{(\ell)}$ for each layer. **Stage 3** maps state-action embeddings to per-layer Q-values via a two-stage Chebyshev-KAN decoder, combining them through learned layer-importance weights into the final scalar action-value $Q(\tilde{S}_t, v)$. KARL is trained only on small synthetic GMM graphs ($N \in [30, 50]$) and deployed zero-shot on real-world benchmarks reaching $N = 56{,}562$ nodes.

5.1 Motivation: Functional Expressivity

5.1.1 The Low-Pass Filtering Limitation of MLP-Based Agents

Let $f : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$ denote a single MLP layer. Every such layer applies the same global affine map $\mathbf{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ to each input vector, followed by a fixed nonlinearity σ:

$$f(\mathbf{x}) \;=\; \sigma\big(\mathbf{W}\mathbf{x} + \mathbf{b}\big). \tag{5.1}$$

The per-edge representational capacity of this map is therefore constrained to a *one-dimensional family* parameterised by the single scalar W_{ji} per input coordinate x_i. For ReLU activations – the standard choice in all GNN-DRL dismantling agents – any single MLP edge function $x_i \mapsto W_{ji}\,\sigma(x_i)$ is a piecewise-linear function with exactly one breakpoint. As shown formally in Proposition 5.4.9, any such function cannot approximate $\sin(\omega\pi x)$ to error below a σ-dependent constant $c_\sigma > 0$ when the MLP width $W < \kappa^*\omega$, regardless of the magnitudes of the weight matrices. For the high-frequency vulnerability functions that monitor cascading fragility in scale-free multiplex networks – where the effective oscillation frequency ω grows with hub-node degree heterogeneity – this imposes a hard expressivity ceiling that wider or deeper MLP stacks cannot overcome at a fixed parameter budget.

Consequently, MultiDismantler's MGNN encoder applies the same affine transformation to every node embedding irrespective of local structural role, limiting its ability to discriminate subtle bottleneck nodes that lie at the boundary between high-coreness shells – nodes that are not prominent degree hubs in any single layer yet sustain mutual connectivity across all layers.

5.1.2 The Kolmogorov-Arnold Alternative

Kolmogorov-Arnold Networks (KANs) [35] offer a principled remedy by grounding their design in the Kolmogorov-Arnold representation theorem: any continuous function $f : [0,1]^n \to \mathbb{R}$ can be expressed exactly as

$$f(\mathbf{x}) \;=\; \sum_{q=1}^{2n+1} \Phi_q\left(\sum_{p=1}^{n} \varphi_{q,p}(x_p)\right), \tag{5.2}$$

where $\varphi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$ are continuous univariate functions. In contrast to the universal approximation theorems for MLPs – which guarantee that a *sufficiently wide* single hidden layer approximates any continuous function – Equation (5.2) establishes a *constructive* decomposition using only univariate functions placed directly on network *edges*. KANs operationalise this theorem by replacing the single scalar weight per edge in an MLP with a learnable univariate function, typically parameterised by a B-spline. Each such function spans a $(G + k)$-dimensional space of smooth local approximators (where G is the grid resolution and k the spline order), equipping the network with an exponentially richer per-edge function family than the one-dimensional scalar-weight family of MLPs.

The structural consequence for multiplex dismantling is significant. By placing independent B-spline functions on each message-passing edge, the KAN encoder can adapt the *shape* of the transformation along each structural pathway, not merely its scale. A hub-connected edge and a bridge-connected edge receive different functional forms, enabling the model to

represent the locally non-smooth vulnerability signals that determine cascading fragility. The emergent structural interpretability demonstrated in Section 5.6.3 – whereby KARL's Chebyshev decoder assigns monotonically larger activation magnitudes to progressively deeper k-core nodes without any auxiliary supervision – provides empirical evidence that this functional flexibility translates into qualitatively superior topological representations.

5.1.3 The Gradient Instability Challenge

Despite their representational advantages, KANs introduce a training instability absent from supervised settings when embedded in an off-policy RL loop. B-splines are piecewise polynomial functions defined over a fixed grid domain $[-1, 1]$; their evaluation requires computing the basis matrix $\mathbf{B}(\mathbf{x}) \in \mathbb{R}^{d_{\text{in}}(G+k)}$ via the Cox–de Boor recurrence at every forward pass.

Saturation risk. In a Q-learning loop on scale-free multiplex graphs, hub nodes of high degree $d_v \gg 1$ produce aggregated neighbourhood features with large ℓ_2-mass. If aggregation is simply an *unweighted sum* $\sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(l)}$, the resultant vector will, on typical scale-free graphs, lie well outside the effective linear regime of tanh – approximately $|x| \lesssim 3$ – saturating the activation to ± 1. In this saturated regime the local derivative $\tanh'(x) = 1 - \tanh^2(x) \approx 0$, effectively killing the gradient flow to the $\text{KAN}_{\text{neigh}}$ module for high-degree nodes. Section 5.1.6 introduces a degree-normalised aggregator that prevents this pathology.

Gradient blowup. Even with proper aggregation, hub nodes contribute to the neighbourhood of many other nodes. At boundary knots, the B-spline derivative $|\varphi'(x)|$ is largest; propagating these spikes back through the neighbourhood aggregator – whose effective branching factor $\beta_{\max} := 1 + d_{\max}$ can be very large for scale-free graphs – causes gradient norms to grow exponentially with message-passing depth. Section 5.1.7 introduces the macroscopic residual MLP branch that resolves this instability, and Proposition 5.4.1 in Section 5.4 provides the formal gradient-norm bound.

5.1.4 Kolmogorov-Arnold Networks and B-Spline Parameterisation

Following Liu et al. [35], we restrict the univariate functions φ of Equation (5.2) to smooth B-splines to enable gradient-based training. Each scalar univariate function is parameterised as:

$$\varphi(x) \;=\; w\left(b(x) \;+\; \sum_{i=1}^{G+k} c_i \, B_i^k(x;\, G)\right), \tag{5.3}$$

where $w \in \mathbb{R}$ is a learnable scaling weight, $c_i \in \mathbb{R}$ are trainable spline coefficients, and $B_i^k(x;\, G)$ are the B-spline basis functions of order k computed from the Cox–de Boor recurrence over a uniform knot vector with k repeated boundary knots on $[-1, 1]$. The grid resolution $G \geq 1$ controls the number of knot intervals and hence the approximation's frequency capacity: a B-spline of order k on a grid of size G approximates any C^{k+1} function on a compact domain to uniform error $O(G^{-(k+1)})$ [14]. The base activation $b(x) = \text{SiLU}(x) = x\sigma(x)$ provides a non-trivial baseline response even when all spline coefficients are initialised to zero.

Vectorised EfficientKAN implementation. For computational efficiency, we adopt the vectorised EfficientKAN formulation [35], which implements a full KAN layer of input dimension d_{in} and output dimension d_{out} via two matrix multiplications:

$$\text{KAN}(\mathbf{x}) \;=\; \mathbf{W}_{\text{base}}\,\sigma(\mathbf{x}) \;+\; \mathbf{W}_{\text{spline}}\,\mathbf{B}(\mathbf{x}), \tag{5.4}$$

where $\sigma(\cdot) = \text{SiLU}(\cdot)$ is applied coordinate-wise, $\mathbf{B}(\mathbf{x}) \in \mathbb{R}^{d_{\text{in}}(G+k)}$ stacks the per-coordinate B-spline evaluations, $\mathbf{W}_{\text{base}} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}}$ is the base linear weight matrix, and $\mathbf{W}_{\text{spline}} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}(G+k)}$ are the spline control weights. The total number of trainable parameters in a single $d_{\text{in}} \to d_{\text{out}}$ KAN layer is $d_{\text{out}} \cdot d_{\text{in}} \cdot (G + k + 1)$, compared to $d_{\text{out}} \cdot d_{\text{in}}$ for an equivalent MLP layer – a ratio of $(G + k + 1) : 1$, which equals $14 : 1$ for the KARL default hyperparameters $G = 10$, $k = 3$.

5.1.5 Node Initialisation

The encoder operates on the undirected multiplex network $\mathcal{M} = \langle V, \{E^{(\ell)}\}_{\ell=1}^{L}\rangle$ established in Chapter 3, with shared vertex set V, $N = |V|$ nodes, L undirected layers, and adjacency matrices $\{A^{(\ell)}\}_{\ell=1}^{L}$. Each node $v \in V$ is initialised with a two-dimensional feature vector derived from its normalised layer degree:

$$\mathbf{x}_v^{(\ell)} \;=\; \left[\frac{k_v^{(\ell)}}{k_{\max}^{(\ell)}},\; \frac{k_v^{(\ell)}}{k_{\max}^{(\ell)}}\right]^{\top} \;\in\; \mathbb{R}^2, \tag{5.5}$$

where $k_v^{(\ell)} = \sum_{j\in V} A_{vj}^{(\ell)}$ and $k_{\max}^{(\ell)} = \max_{v\in V} k_v^{(\ell)}$. The repeated entry provides a 2-dimensional input compatible with the linear projection $\mathbf{W}_{n2l} \in \mathbb{R}^{2\times d}$ that lifts the raw feature to the embedding space of dimension d.

Remark 5.1.1 (Degree normalisation and B-spline grid compliance). Normalising by the layer maximum degree bounds the initial feature to $[0, 1]^2$ for every layer, ensuring that B-spline basis evaluations at initialisation lie within the grid domain $[-1, 1]$. This is the same normalisation used by MultiDismantler and is essential for the Lipschitz analysis of Proposition 5.4.1, since the tanh pre-activations subsequently maintain the grid-domain constraint throughout training.

5.1.6 Iterative Message Passing with B-Spline KANs

Let $\mathbf{h}_v^{(l)} \in \mathbb{R}^d$ denote the embedding of node v at iteration $l \in \{0, 1, \ldots, l_{\max}\}$, with $l_{\max} = 3$ message-passing rounds. The initial embedding is $\mathbf{h}_v^{(0)} = \mathbf{W}_{n2l}\,\mathbf{x}_v^{(\ell)}$ for each layer ℓ.

Degree-normalised aggregation. In a standard unweighted-sum aggregator, hub nodes with residual degree $d_v = |\mathcal{N}(v)| \gg 1$ produce aggregated vectors whose ℓ_2-norm scales as $O(\sqrt{d_v}\,\|\mathbf{h}\|)$ under random embeddings – far outside the effective linear regime of tanh for large hubs. Passing such vectors through tanh saturates the activation to ± 1, setting $\tanh' \approx 0$ and eliminating gradient flow to the $\text{KAN}_{\text{neigh}}$ module for precisely those hub nodes whose structural role the agent most needs to learn. To prevent this pathology, we introduce a $\sqrt{d_v}$-normalised aggregation scheme. Let $d_v = |\mathcal{N}(v)|$ denote the residual layer

degree and define $a \vee b \triangleq \max(a, b)$ to handle the degenerate case of isolated nodes:

$$\mathrm{Agg}(\mathbf{h}^{(l)}; v) \;=\; \frac{1}{\sqrt{d_v \vee 1}} \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(l)}. \tag{5.6}$$

The $\frac{1}{\sqrt{d_v \vee 1}}$ factor is the theoretically canonical choice: it preserves the ℓ_2-norm of the aggregated vector under isotropic embeddings (since $\mathrm{Var}(\sum_u h_u/\sqrt{d_v}) = \mathrm{Var}(h_u)$ when embeddings are independent), ensuring that $\tanh(\mathrm{Agg})$ operates in its approximately linear regime for typical embedding magnitudes even when d_v is large. This normalisation preserves the gradient stability analysis of Proposition 5.4.1: the Lipschitz constant L_φ of the composed map Φ now incorporates $1/\sqrt{d_v}$, but the $\beta_{\max} = 1 + d_{\max}$ factor in the gradient bound remains unchanged (counts number of backward paths, not forward scaling).

Node embedding update rule. The full update at each message-passing iteration is:

$$\begin{aligned}\mathbf{h}_v^{(l+1)} = \mathrm{KAN}_{\mathrm{out}}\Big(\Big[\mathrm{KAN}_{\mathrm{neigh}}\Big(\tanh\big(\mathrm{Agg}(\mathbf{h}^{(l)}; v)\big)\Big) \;\Big\|\; \mathrm{KAN}_{\mathrm{self}}\Big(\tanh\big(\mathbf{h}_v^{(l)}\big)\Big)\Big]\Big) \\ +\; \alpha_{\mathrm{res}} \cdot \mathrm{MLP}_{\mathrm{res}}(\mathbf{h}_v^{(l)}),\end{aligned} \tag{5.7}$$

followed by ℓ_2-normalisation: $\mathbf{h}_v^{(l+1)} \leftarrow \mathbf{h}_v^{(l+1)}/\|\mathbf{h}_v^{(l+1)}\|_2$. Here:

- $\mathrm{KAN}_{\mathrm{neigh}} : \mathbb{R}^d \to \mathbb{R}^d$ transforms aggregated neighbourhood features to capture collective structural signals.
- $\mathrm{KAN}_{\mathrm{self}} : \mathbb{R}^d \to \mathbb{R}^d$ transforms the node's own features independently, preserving node identity without conflation with neighbourhood information.
- $\mathrm{KAN}_{\mathrm{out}} : \mathbb{R}^{2d} \to \mathbb{R}^d$ fuses the concatenated $2d$-dimensional representation back to embedding dimension d. All three KAN modules use $G = 10$, $k = 3$.
- $\|\cdot\|$ denotes vector concatenation.
- $\mathrm{MLP}_{\mathrm{res}} : \mathbb{R}^d \to \mathbb{R}^d$ is a single linear-ReLU layer acting as the macroscopic residual branch.
- $\alpha_{\mathrm{res}} \in \mathbb{R}$ is a learnable scalar initialised to 0.5 (optimal by the ablation in Section 5.6.1).

The $\tanh(\cdot)$ pre-activations bound all B-spline inputs to $[-1, 1]$, preventing out-of-grid extrapolation error and providing the 1-Lipschitz constant used in Proposition 5.4.1. After $l_{\max}$ iterations, layer ℓ produces the final intra-layer embedding matrix $\mathbf{H}^{(\ell)} \in \mathbb{R}^{N \times d}$.

5.1.7 Macroscopic Residual Branch

The macroscopic residual branch in Equation (5.7) is the architectural element that distinguishes KARL's encoder from a naive KAN-GNN. Its role is to provide a degree-independent gradient propagation pathway that bypasses the neighbourhood aggregator.

Formally, let Φ denote the composed B-spline-KAN transformation in Equation (5.7) (the first line) and let $\Psi = \mathrm{MLP}_{\mathrm{res}}$. The update can be written compactly as:

$$\mathbf{h}_v^{(l+1)} \;=\; \Phi\Big(\mathbf{h}_v^{(l)},\, \{\mathbf{h}_u^{(l)}\}_{u \in \mathcal{N}(v)}\Big) \;+\; \alpha_{\mathrm{res}} \cdot \Psi\Big(\mathbf{h}_v^{(l)}\Big). \tag{5.8}$$

The term $\alpha_{\text{res}} \cdot \Psi(\mathbf{h}_v^{(l)})$ introduces a *low-frequency anchor*: it is a function of $\mathbf{h}_v^{(l)}$ alone whose gradient does not flow through the neighbourhood aggregator and is therefore insensitive to $d_{\max}$.

Lipschitz bound on residual constants. The Lipschitz constants invoked in Proposition 5.4.1 are both controlled by the weight-decay regulariser $\lambda = 10^{-5}$ applied to all trainable parameters throughout training:

- L_φ (Lipschitz constant of Φ): bounded via $\|\mathbf{W}_{\text{spline}}\|_2 \le c_\lambda$ and the 1-Lipschitz tanh pre-activations, giving $L_\varphi \lesssim \|\mathbf{W}_{\text{spline}}\|_2 \cdot C_k G/2 + \|\mathbf{W}_{\text{base}}\|_2$, where C_k depends only on the spline order [14].
- L_r (Lipschitz constant of Ψ): bounded by $\|\mathbf{W}_{\text{res}}\|_2 \le c_\lambda$, the spectral norm of the single linear-ReLU layer.

Since the weight-decay constraint is enforced throughout training, both L_φ and L_r remain finite, ensuring that the gradient bound (5.27) is non-trivial at all training steps.

Remark 5.1.2 (Contrast with ResNet-style shortcuts). Standard ResNet shortcuts add the raw input directly: $\mathbf{y} = F(\mathbf{x}) + \mathbf{x}$. The KARL residual branch differs in two respects. First, MLP_{res} applies a linear-ReLU transformation before addition, providing mild adaptivity without the full B-spline computational overhead. Second, the learnable scalar α_{res} allows training to balance the residual contribution; Proposition 5.4.1 shows that over-weighting ($\alpha_{\text{res}} \to 1$) suppresses B-spline gradients, while under-weighting ($\alpha_{\text{res}} = 0$) eliminates the stabilising path.

5.2 Architecture Stage 2: KANformer

After $l_{\max}$ message-passing iterations, each layer ℓ has produced intra-layer node embeddings $\mathbf{H}^{(\ell)} \in \mathbb{R}^{N\times d}$. In a one-to-one interdependent multiplex network, the cascading failure dynamics mean that a node may be structurally peripheral in one layer yet pivotal in another, and its removal triggers a collapse that propagates across every layer simultaneously [11]. Standard affine attention [57] restricts per-token projections to a single linear map, limiting capacity for the sharp, non-smooth cross-layer couplings that characterise multiplex dismantling.

The **KANformer** module addresses this limitation by replacing every projection in the cross-layer attention mechanism – Query, Key, Value, and output – with an independent B-spline KAN transformation, and by explicitly applying Layer Normalisation (LN) before each B-spline projection to ensure training stability and optimal utilisation of the spline grid. To our knowledge, this is the first application of KAN-parameterised attention to layer fusion in multiplex network dismantling.

5.2.1 Learnable Pre-Transformation

Before cross-layer attention, the embeddings of all layers are aligned via a shared learnable affine map that also enforces the B-spline grid domain constraint:

$$\widetilde{\mathbf{H}}^{(\ell)} \;=\; \tanh\Big(\mathbf{H}^{(\ell)}\,\mathbf{T} + \mathbf{b}\Big) \;\in\; \mathbb{R}^{N\times d}, \tag{5.9}$$

where $\mathbf{T} \in \mathbb{R}^{d\times d}$ is a learnable matrix initialised to the identity $\mathbf{I}_d$, and $\mathbf{b} \in \mathbb{R}^d$ is zero-initialised. The tanh activation constrains all inputs to $[-1, 1]$ (consistent with the B-spline domain). Initialising $\mathbf{T} = \mathbf{I}_d$ means that at the start of training, the pre-transformation is approximately the identity (up to the tanh saturation boundary), allowing the B-spline Q/K/V projections to begin learning from a well-conditioned starting point.

5.2.2 Multi-Head KAN Cross-Attention

The KANformer operates as a *cross-attention* mechanism, called once per target layer ℓ. The target-layer embedding $\widetilde{\mathbf{H}}^{(\ell)} \in \mathbb{R}^{N\times d}$ acts as the query anchor, attending to the context sequence formed by the remaining $L-1$ layers. For each node $n \in \{1, \ldots, N\}$, let $\widetilde{\mathbf{h}}_n^{(\ell)} \in \mathbb{R}^d$ denote the n-th row of $\widetilde{\mathbf{H}}^{(\ell)}$ and let $\mathbf{c}_n^{(-\ell)} \in \mathbb{R}^{(L-1)\times d}$ denote the stacked embeddings of node n across all other $L-1$ layers (analogously extracted from $\{\widetilde{\mathbf{H}}^{(\ell')}\}_{\ell'\neq\ell}$).

Layer Normalisation. Both sequences are passed through Layer Normalisation before B-spline projections to ensure that the input to each KAN lies within the effective spline domain and to stabilise training.

KAN projections. For each attention head $i \in \{1, \ldots, h\}$ with per-head dimension $d_k = d/h$, independent B-spline KAN projections ($G = 5$, $k = 3$) map the query and context to:

$$\mathbf{Q}_{i,n} \;=\; \mathrm{KAN}_i^{(Q)}\Big(\mathrm{LN}\big(\widetilde{\mathbf{h}}_n^{(\ell)}\big)\Big) \;\in\; \mathbb{R}^{d_k}, \tag{5.10}$$

$$\mathbf{K}_{i,n},\ \mathbf{V}_{i,n} \;=\; \mathrm{KAN}_i^{(KV)}\Big(\mathrm{LN}\big(\mathbf{c}_n^{(-\ell)}\big)\Big) \;\in\; \mathbb{R}^{(L-1)\times d_k}. \tag{5.11}$$

Scaled dot-product attention (per node). To avoid tensor-transposition ambiguity for higher-order tensors, we define the attention computation for each node n individually. Here $\mathbf{Q}_{i,n} \in \mathbb{R}^{d_k}$ is a row vector, $\mathbf{K}_{i,n} \in \mathbb{R}^{(L-1)\times d_k}$ is the key matrix, and $\mathbf{V}_{i,n} \in \mathbb{R}^{(L-1)\times d_k}$ is the value matrix. The attended output for node n under head i is:

$$\mathbf{C}_{i,n}^{(\ell)} \;=\; \mathrm{softmax}\left(\frac{\mathbf{Q}_{i,n}\,(\mathbf{K}_{i,n})^{\mathsf{T}}}{\sqrt{d_k}}\right)\mathbf{V}_{i,n} \;\in\; \mathbb{R}^{d_k}, \tag{5.12}$$

where $\mathbf{Q}_{i,n}\,(\mathbf{K}_{i,n})^{\mathsf{T}} \in \mathbb{R}^{1\times(L-1)}$ is an ordinary matrix product (row vector times matrix transpose), and the softmax operates over the $L-1$ key positions. The outputs $\{\mathbf{C}_{i,n}^{(\ell)}\}_{n=1}^{N}$ are stacked along the node dimension to form $\mathbf{C}_i^{(\ell)} \in \mathbb{R}^{N\times d_k}$. Concatenating the h head outputs along the feature dimension and passing through a final B-spline KAN projection yields the fused context $\mathbf{F}^{(\ell)} \in \mathbb{R}^{N\times d}$.

5.2.3 Residual Integration

The fused context is combined with the pre-transformed target-layer embedding via a residual addition:

$$\mathbf{Z}^{(\ell)} \;=\; \widetilde{\mathbf{H}}^{(\ell)} + \mathbf{F}^{(\ell)} \;\in\; \mathbb{R}^{N\times d}. \tag{5.13}$$

The α_{res} gating used in the encoder's residual branch (Equation (5.8)) is *not* applied here; the KANformer residual is an unweighted sum analogous to the residual connection in the original Transformer [57]. The procedure of Equations (5.9)–(5.13) is repeated for each

$\ell \in \{1, \ldots, L\}$, producing L fused embeddings $\{\mathbf{Z}^{(\ell)}\}_{\ell=1}^{L}$ passed to the action-value decoder. The virtual-node embedding $\mathbf{z}_s^{(\ell)} \in \mathbb{R}^d$ (the row of $\mathbf{Z}^{(\ell)}$ corresponding to the virtual node introduced in Section 3.2.1) encodes the global dismantling state of layer ℓ.

Remark 5.2.1 (KANformer vs. MultiDismantler attention). MultiDismantler's node-level interlayer attention [25] computes attention coefficients via a *Hadamard product* of projected embeddings from pairs of layers followed by a scalar softmax weighting of the source-layer embeddings. This is effectively a rank-1 bilinear form per node and per layer pair, with no information sharing between nodes. The KANformer, by contrast, applies full multi-head attention across all N nodes of each layer pair simultaneously, with per-head B-spline Q/K/V projections operating on Layer-Normalised inputs, enabling it to model high-frequency, non-smooth cross-layer coupling patterns beyond the capacity of any affine projection.

5.3 Architecture Stage 3: Orthogonal Chebyshev-KAN Q-Head

5.3.1 Why B-Splines Fail in the Decoder Position

Having established the B-spline KAN as the natural choice for the encoder and KANformer, we now argue that B-splines are *not* the optimal choice for the action-value head.

The Q-value regression head must approximate the n-step TD target:

$$y_t \;=\; \sum_{j=0}^{n-1} \gamma_{\text{disc}}^{j}\, r_{t+j} \;+\; \gamma_{\text{disc}}^{n} \max_{a'} \widehat{Q}(\tilde{S}_{t+n}, a'; \Theta^{-}), \tag{5.14}$$

where Θ^{-} denotes the target-network parameters updated every 1,000 gradient steps. The target y_t is a non-stationary function of the current dismantling state $\tilde{S}_{t+n}$: as the multiplex network is dismantled, the embedding distribution shifts and the target values follow a continuously drifting distribution. Under non-stationary TD targets, uniform-grid B-splines suffer from the *Runge phenomenon* [56]: polynomial interpolation through equispaced points produces boundary oscillations whose magnitude grows as $\Theta(2^K/\sqrt{K})$ with the polynomial degree K, causing gradient spikes that destabilise Q-learning.

The remedy is to replace the B-spline basis in the decoder with a basis that *minimises worst-case approximation error* uniformly on $[-1, 1]$ – the Chebyshev polynomial basis of the first kind, whose associated projection operator has an operator norm that grows only as $O(\log K)$ (Theorem 5.4.4).

5.3.2 Orthogonal Chebyshev Polynomial Basis

The Chebyshev polynomials of the first kind $\{T_m\}_{m=0}^{\infty}$ are defined by the three-term recurrence:

$$T_0(x) = 1, \qquad T_1(x) = x, \qquad T_m(x) = 2x\,T_{m-1}(x) - T_{m-2}(x) \quad (m \geq 2). \tag{5.15}$$

These polynomials satisfy the orthogonality relation under the Chebyshev measure $w(x) = (\pi\sqrt{1-x^2})^{-1}$ on $(-1, 1)$:

$$\int_{-1}^{1} T_j(x)\, T_k(x)\, w(x)\, dx \;=\; \begin{cases} \pi & \text{if } j = k = 0, \\ \pi/2 & \text{if } j = k \geq 1, \\ 0 & \text{if } j \neq k. \end{cases} \tag{5.16}$$

The truncated Chebyshev series $\mathcal{S}_K[f]$ is the best polynomial approximation of degree K in the L^∞ sense up to a logarithmic factor (Theorem 5.4.4), and the associated projection operator has $\|\mathcal{S}_K\|_{\infty\to\infty} = O(\log K)$, guaranteeing only logarithmic growth of the approximation error with polynomial degree. This contrasts sharply with the exponential Lebesgue constant $\Lambda_K = \Theta(2^K/\sqrt{K})$ of equispaced polynomial interpolation.

5.3.3 ChebyKAN Layer Formulation

A ChebyKAN layer of degree K, input dimension d_{in}, and output dimension d_{out} computes:

$$y_j \;=\; \sum_{i=1}^{d_{\text{in}}} \sum_{m=0}^{K} c_{i,j,m}\, T_m\big(\tanh(x_i)\big) \;+\; b_j, \quad j = 1, \ldots, d_{\text{out}}, \tag{5.17}$$

where $c_{i,j,m} \in \mathbb{R}$ are Xavier-initialised learnable coefficients, $b_j \in \mathbb{R}$ is a zero-initialised bias, and the $\tanh(\cdot)$ pre-activation bounds all inputs to $[-1, 1]$ for Chebyshev polynomial evaluation. The three-term recurrence (5.15) is computed sequentially and requires no external library beyond standard arithmetic operations. The total parameter count of a ChebyKAN layer is $d_{\text{out}} \cdot d_{\text{in}} \cdot (K + 1) + d_{\text{out}}$, and the coefficient tensor $\mathbf{c} \in \mathbb{R}^{d_{\text{in}} \times d_{\text{out}} \times (K+1)}$ is computed via the recurrence with no additional weight matrices.

5.3.4 Auxiliary Feature Vector

To further inform the Q-head about the current state of the dismantling process, KARL constructs a four-dimensional auxiliary feature vector $\mathbf{f}_{\text{aux}}^{(\ell)} \in \mathbb{R}^4$ for each candidate node-layer pair (v, ℓ):

1. **Removal fraction (nodes)**: the fraction of nodes in $\tilde{S}_t$ relative to N.
2. **Removal fraction (edges)**: the fraction of intra-layer edges in layer ℓ removed in $\tilde{S}_t$ relative to $|E^{(\ell)}|$.
3. **Two-hop reachability**: the mean fraction of nodes reachable from v within two hops in the current residual graph of layer ℓ, normalised by N.
4. **Residual degree**: the normalised degree $k_v^{(\ell)}(t)/k_{\max}^{(\ell)}$ of node v in the current residual layer graph at step t.

These features encode the current dismantling progress and the local connectivity environment of node v, providing context that the GNN embedding (trained on static synthetic graphs of $N \in [30, 50]$ nodes) may not fully capture for large out-of-distribution benchmarks reaching $N = 56{,}562$.

5.3.5 Q-Value Computation and Layer Importance Weighting

For each node $v \in V \setminus \tilde{S}_t$ and each layer $\ell \in \{1, \ldots, L\}$, the state-action embedding is formed as the outer product:

$$\mathbf{e}_{s,a}^{(\ell)} \;=\; \mathbf{z}_s^{(\ell)} \otimes \mathbf{z}_v^{(\ell)} \;\in\; \mathbb{R}^d, \tag{5.18}$$

where $\mathbf{z}_s^{(\ell)}$ is the virtual-node embedding (global state, Section 3.2.1) and $\mathbf{z}_v^{(\ell)}$ is the fused embedding of node v in layer ℓ, both extracted from $\{\mathbf{Z}^{(\ell)}\}$ after the KANformer. The outer product models fine-grained state-action dependencies following FINDER [23].

The per-layer Q-value is computed by a two-stage ChebyKAN decoder. Specifically, $\text{ChebyKAN}_{\text{hidden}}$ (degree $K = 4$) first maps the d-dimensional state-action embedding to $\mathbb{R}^{32}$. Appending the 4-dimensional auxiliary context vector $\mathbf{f}_{\text{aux}}^{(\ell)}$ yields a 36-dimensional input, which $\text{ChebyKAN}_{\text{final}}$ (degree $K = 4$) maps to the scalar Q-value estimate. Using $\oplus$ to denote vector concatenation:

$$q^{(\ell)}(\tilde{S}_t, v) \;=\; \text{ChebyKAN}_{\text{final}}\left(\underbrace{\text{ChebyKAN}_{\text{hidden}}\left(\mathbf{e}_{s,a}^{(\ell)}\right)}_{\in \mathbb{R}^{32}} \;\oplus\; \underbrace{\mathbf{f}_{\text{aux}}^{(\ell)}}_{\in \mathbb{R}^{4}} \right), \tag{5.19}$$

where $\text{ChebyKAN}_{\text{hidden}} : \mathbb{R}^d \to \mathbb{R}^{32}$ and $\text{ChebyKAN}_{\text{final}} : \mathbb{R}^{36} \to \mathbb{R}^1$. Chebyshev orthogonality ensures well-conditioned gradients under the non-stationary TD targets of Q-learning (Theorem 5.4.4).

Different layers contribute unequally to the overall fragility of the LMCC. To capture this heterogeneity, a learned softmax weighting aggregates per-layer Q-values into a final scalar estimate:

$$\omega^{(\ell)} \;=\; \frac{\exp\left(\text{ReLU}(\mathbf{z}_s^{(\ell)}\mathbf{W}_1)\mathbf{W}_2\right)}{\sum_{p=1}^{L} \exp\left(\text{ReLU}(\mathbf{z}_s^{(p)}\mathbf{W}_1)\mathbf{W}_2\right)}, \tag{5.20}$$

$$Q(\tilde{S}_t, v) \;=\; \sum_{\ell=1}^{L} \omega^{(\ell)}\, q^{(\ell)}(\tilde{S}_t, v), \tag{5.21}$$

where $\mathbf{W}_1 \in \mathbb{R}^{d\times 128}$ and $\mathbf{W}_2 \in \mathbb{R}^{128\times 1}$ are learnable matrices. The layer-importance weights $\{\omega^{(\ell)}\}$ are computed from the virtual-node embedding $\mathbf{z}_s^{(\ell)}$, which encodes the global state of layer ℓ after cascade resolution, enabling the agent to dynamically adjust layer-specific dismantling priority as the episode progresses.

5.3.6 Policy Search via Prioritised n-Step DQN

Reward function. The immediate reward for removing node v_{a_t} at step t is:

$$r_t(v) \;=\; P_\infty(\tilde{S}_t) \cdot F(\{v_{a_t}\}), \tag{5.22}$$

where $P_\infty(\tilde{S}_t) = N_{\text{LMCC}}(\tilde{S}_t)/N$ is the normalised LMCC size after removal and cascade resolution (consistent with the notation of Section 3.3.2), and $F(\{v_{a_t}\}) = 1$ under unit cost (or the incident edge count under degree cost).

n-step returns. To capture delayed fragmentation effects, KARL uses $n = 5$-step returns:

$$y_t \;=\; \sum_{j=0}^{n-1} \gamma_{\text{disc}}^{j}\, r_{t+j} \;+\; \gamma_{\text{disc}}^{n} \max_{a'} \widehat{Q}(\tilde{S}_{t+n}, a'; \Theta^{-}), \tag{5.23}$$

where $\gamma_{\text{disc}} = 0.99$ and Θ^{-} is the target network updated every 1,000 gradient steps.

TD loss with Prioritised Experience Replay. The primary training loss is the mean squared TD error over a prioritised replay buffer $\mathcal{B}$ [50]:

$$\mathcal{L}_{\text{TD}}(\Theta) \;=\; \mathbb{E}_{(\tilde{S}_t, v, \ldots)\sim\mathcal{B}}\Big[\big(y_t - Q(\tilde{S}_t, v;\, \Theta)\big)^2\Big]. \tag{5.24}$$

Sampling probabilities are proportional to $|\delta_t|^{\alpha_{\text{PER}}}$ with $\alpha_{\text{PER}} = 0.6$; importance-sampling weights (β_{IS} annealed from 0.4 to 1.0) correct for the induced bias.

Graph Laplacian regulariser. To prevent the high-capacity B-spline encoder from assigning arbitrarily dissimilar representations to structurally adjacent nodes, we add a graph Laplacian quadratic regulariser:

$$\mathcal{L}_R(\Theta) \;=\; \sum_{\ell=1}^{L} \frac{2\,\text{tr}\big(\mathbf{H}_\ell^\top\, \mathcal{L}^{(\ell)}\, \mathbf{H}_\ell\big)}{|E^{(\ell)}|}, \tag{5.25}$$

where $\mathcal{L}^{(\ell)} = \mathbf{D}^{(\ell)} - A^{(\ell)}$ is the combinatorial graph Laplacian of layer ℓ. Via the identity $2\,\text{tr}(\mathbf{H}^\top \mathcal{L} \mathbf{H}) = \sum_{i,j} A_{ij} \|\mathbf{h}_i - \mathbf{h}_j\|_2^2$, this regulariser penalises large embedding distances between adjacent node pairs, normalised to be scale-invariant across layers with different edge densities.

Total objective.

$$\mathcal{L}_{\text{Total}}(\Theta) \;=\; \mathcal{L}_{\text{TD}}(\Theta) \;+\; \lambda\, \mathcal{L}_R(\Theta), \qquad \lambda = 0.001. \tag{5.26}$$

KARL is trained for 20,000 episodes using the Adam optimiser with decoupled learning rates: $\eta = 10^{-4}$ for linear parameters and $\eta = 10^{-3}$ for spline coefficients. An ϵ-greedy strategy with ϵ decaying from 1.0 to 0.05 balances exploration and exploitation. At inference time, the agent selects $a_t = \arg\max_{v \in V \setminus \tilde{S}_t} Q(\tilde{S}_t, v;\, \Theta)$ greedily.

5.4 Theoretical Expressivity Guarantees

This section establishes four theoretical results that collectively provide end-to-end justification for KARL's architectural design choices. Proposition 5.4.1 justifies the macroscopic residual branch. Theorem 5.4.4 and Corollary 5.4.5 justify the Chebyshev decoder over B-splines. Theorem 5.4.6 establishes that KARL's KAN-GNN function class subsumes MultiDismantler's MLP-GNN class, guaranteeing no loss of representational power. Proposition 5.4.9 quantifies the per-layer expressivity gap between KAN and MLP edge functions at a fixed parameter budget.

5.4.1 Gradient Stability of the Residual B-Spline KAN Encoder

Proposition 5.4.1 (Encoder Worst-Case Gradient Stability)**.** *Let the B-spline KAN encoder perform $l_{\max}$ message-passing iterations with learnable residual scalar $\alpha_{\text{res}} \in (0, 1)$. Suppose the composed transformation Φ is L_ϕ-Lipschitz in each of its arguments and the residual MLP Ψ is L_r-Lipschitz, both with respect to the ℓ_2-norm on $\mathbb{R}^d$. Let $d^{\max}$ denote the maximum degree of any node in the graph across any layer, and define the global effective branching factor $\beta_{\max} := 1 + d^{\max}$. Let $\mathcal{L}$ denote the total training objective (Eq. 27). Then for every iteration index $l \in \{0, 1, \ldots, l_{\max} - 1\}$, the worst-case back-propagated gradient*

satisfies

$$\max_v \left\|\nabla_{h_v^{(l)}} \mathcal{L}\right\|_2 \;\le\; \left(\beta_{\max} L_\phi \;+\; \alpha_{\text{res}} L_r\right)^{l_{\max}-l} \max_v \left\|\nabla_{h_v^{(l_{\max})}} \mathcal{L}\right\|_2. \tag{5.27}$$

In particular:

1. (Pure B-spline path lacks any independent damping.) *When* $\alpha_{\text{res}} = 0$*, the bound reduces to* $(\beta_{\max} L_\phi)^{l_{\max}-l} \max_v \|\nabla_{h_v^{(l_{\max})}} \mathcal{L}\|_2$*. For scale-free multiplex graphs whose hub nodes have* $d^{\max} \gg 1$*, this upper bound is the only thing controlling the worst-case gradient norm; nothing in the architecture prevents it from realising values consistent with the gradient explosion observed empirically in Figure 4(a) (*No_MLP *and* $\alpha = 0.0$ *configurations).*

2. (Residual regularisation provides an independent path.) *For any* $\alpha_{\text{res}} \in (0, 1)$*, the per-layer multiplier becomes* $\beta_{\max} L_\phi + \alpha_{\text{res}} L_r$*. Because* MLP_{res} *is a single linear-ReLU layer,* $L_r \le \|W_{\text{res}}\|_2$*, which is controlled by the weight-decay regulariser* $\lambda = 10^{-5}$ *(Eq. 27). Crucially, the residual provides a parallel propagation path that does* not *pass through the neighbourhood aggregator and is therefore* independent of $\beta_{\max}$*; this lets gradients flow stably even when hubs of high degree are present.*

3. (Over-weighting collapses spline gradients.) *As* $\alpha_{\text{res}} \to 1$*, the MLP branch dominates the gradient signal, suppressing the B-spline update and slowing policy convergence – consistent with the* $\alpha = 1.0$ *trajectory in Figure 4(a).*

The empirically optimal choice $\alpha_{\text{res}} = 0.5$ *(Appendix C.1) minimises the AUDC while maintaining stable gradient dynamics, achieving the tightest gradient-norm band in Figure 4(a).*

Proof. The proof will be done in three steps.

Step 1: Chain rule through one iteration. Fix node v and iteration l. By Eq. (4.17),

$$h_v^{(l+1)} \;=\; \Phi\big(h_v^{(l)}, \{h_u^{(l)}\}_{u \in \mathcal{N}(v)}\big) \;+\; \alpha_{\text{res}}\, \Psi\big(h_v^{(l)}\big).$$

The total derivative of the loss w.r.t. $h_v^{(l)}$ collects contributions from (a) the self-input slot of Φ, (b) the residual branch Ψ, and (c) the appearance of $h_v^{(l)}$ inside the neighbourhood aggregator of every neighbour u for which $v \in \mathcal{N}(u)$. Writing J_Φ^{self} and J_Φ^{neigh} for the Jacobians of Φ with respect to the self and aggregated arguments respectively, and using that $\partial \text{Agg}(h^{(l)})/\partial h_v^{(l)} = I_d$ for each u with $v \in \mathcal{N}(u)$:

$$\begin{aligned}\nabla_{h_v^{(l)}} \mathcal{L} \;&=\; \big(J_\Phi^{\text{self}}(h_v^{(l)})\big)^\top \nabla_{h_v^{(l+1)}} \mathcal{L} \;+\; \alpha_{\text{res}} \big(J\Psi(h_v^{(l)})\big)^\top \nabla_{h_v^{(l+1)}} \mathcal{L} \\ &\quad+ \sum_{u:\, v \in \mathcal{N}(u)} \big(J_\Phi^{\text{neigh}}(h_u^{(l)})\big)^\top \nabla_{h_u^{(l+1)}} \mathcal{L}.\end{aligned} \tag{5.28}$$

Step 2: Operator-norm bound. Since tanh is 1-Lipschitz, the composed map Φ inherits the same Lipschitz constant L_ϕ in each of its inputs (the concatenation does not amplify the norm). Hence $\|J_\Phi^{\text{self}}\|_2, \|J_\Phi^{\text{neigh}}\|_2 \le L_\phi$ [14, 35]. Similarly $\|J\Psi\|_2 \le L_r$. Define the worst-case upstream gradient norm at iteration $l+1$ as

$$M_{l+1} \;:=\; \max_v \left\|\nabla_{h_v^{(l+1)}} \mathcal{L}\right\|_2.$$

Applying the triangle inequality and the operator-norm bound $\|A^\top x\|_2 \le \|A\|_2\|x\|_2$ to Eq. (5.28):

$$\begin{aligned}\left\|\nabla_{h_v^{(l)}}\mathcal{L}\right\|_2 &\le \left(L_\phi + \alpha_{\text{res}}L_r\right)M_{l+1} \;+ \sum_{u:\, v\in\mathcal{N}(u)} L_\phi\, M_{l+1} \\ &\le \left(L_\phi + \alpha_{\text{res}}L_r + d_v^{\max}\, L_\phi\right)M_{l+1} \\ &\le \left(\beta_{\max}L_\phi + \alpha_{\text{res}}L_r\right)M_{l+1}, \end{aligned} \tag{5.29}$$

where the second inequality uses $|\{u : v \in \mathcal{N}(u)\}| \le d_v^{\max}$ (the number of neighbourhoods that contain v is at most v's maximum degree across layers), and the third bounds the local branching factor by the global maximum $\beta_{\max} := 1 + d^{\max}$.

Step 3: Induction over depth. Taking the maximum of the left-hand side of Eq. (5.29) over all nodes v yields the recurrence

$$M_l \;\le\; \left(\beta_{\max}L_\phi + \alpha_{\text{res}}L_r\right)M_{l+1}.$$

Unrolling this recursion from $l = l_{\max} - 1$ down to l gives

$$M_l \;\le\; \left(\beta_{\max}L_\phi + \alpha_{\text{res}}L_r\right)^{l_{\max}-l} M_{l_{\max}},$$

which is exactly Eq. (5.27). □ □

Remark 5.4.2 (L_ϕ for scale-free multiplex graphs). B-spline basis functions $B_i^k(x; G)$ are piecewise polynomials of order k bounded on $[-1, 1]$, so each univariate spline φ has $|\varphi'(x)| \le C_k/\delta$ where $\delta = 2/G$ is the knot spacing and C_k depends only on the spline order [14]. For the vectorised EfficientKAN implementation (Eq. 11), the composed Jacobian spectral norm admits the bound $L_\phi \lesssim \|W_{\text{spline}}\|_2 \cdot C_k G/2 + \|W_{\text{base}}\|_2$. On scale-free multiplex graphs with heterogeneous degree distributions, the aggregated neighbourhood feature $\text{Agg}(h^{(l)})$ at hub nodes has high ℓ_2-mass, pushing post-tanh activations toward the boundary of the spline domain $[-1, 1]$ where knot density is lowest and $|\varphi'|$ is largest; combined with $d^{\max} \gg 1$ in scale-free regimes, this makes the per-step amplification $\beta_{\max}L_\phi$ large in practice. The weight-decay term $\lambda\|\Theta\|^2$ (Eq. 27) bounds $\|W_{\text{spline}}\|_2$ and hence L_ϕ throughout training, but cannot suppress the multiplicative $\beta_{\max}$ factor that arises from the graph topology itself – only the degree-independent residual branch provides such a stabilising path. After the final addition, ℓ_2-normalisation of $h_v^{(l+1)}$ further constrains the forward activations to the unit sphere and contributes an additional, bounded factor that is absorbed into L_ϕ; this technical detail does not change the structure of the bound.

Remark 5.4.3 (Interpretation of the bound). Three observations are immediate from Equation (5.27). **(i) Pure B-spline path.** When $\alpha_{\text{res}} = 0$, the bound reduces to $(\beta_{\max}L_\varphi)^{l_{\max}-l}M_{l_{\max}}$. For scale-free graphs with $d_{\max} \gg 1$, this can grow exponentially with message-passing depth, consistent with the gradient spikes observed empirically in Figure 5.4(a) for the $\alpha = 0.0$ configuration. **(ii) Residual regularisation.** For any $\alpha_{\text{res}} \in (0, 1)$, the residual MLP provides a parallel propagation path independent of $\beta_{\max}$; the per-layer multiplier $\alpha_{\text{res}}L_r$ is controlled by the weight-decay regulariser $\lambda = 10^{-5}$ applied to $\mathbf{W}_{\text{res}}$, which bounds $L_r \le \|\mathbf{W}_{\text{res}}\|_2$. **(iii) Over-weighting.** As $\alpha_{\text{res}} \to 1$, the MLP branch dominates the gradient, suppressing the B-spline update and slowing policy convergence – consistent with the $\alpha = 1.0$ trajectory in Figure 5.4(a). The empirically optimal value $\alpha_{\text{res}} = 0.5$ (Section 5.6.1) minimises AUDC while maintaining stable gradient dynamics.

5.4.2 Near-Minimax Optimality of the Chebyshev Q-Head

To formalize the necessity of replacing B-splines with Chebyshev polynomials in the decoder, we establish the following theoretical bound on the Temporal-Difference (TD) approximation error.

Theorem 5.4.4 (Near-Minimax Optimality of the ChebyKAN Q-Head)**.** *Let $y_t : [-1, 1] \to \mathbb{R}$ be the non-stationary n-step Temporal-Difference (TD) target at step t, assumed to be continuous. Let*

$$Q(x; \theta) = \sum_{k=0}^{K} \theta_k T_k(x)$$

be the action-value function parameterised by a ChebyKAN regression head of degree K, where $\{T_k\}_{k=0}^{K}$ are the Chebyshev polynomials of the first kind, and let $\mathcal{L}(\theta)$ be the squared TD loss. Then the following hold.

(A) Representation Power (Near-Minimax Bound). *The truncated Chebyshev series projection $S_K[y_t] \in \mathbb{P}_K$ satisfies*

$$\|y_t - S_K[y_t]\|_\infty \;\le\; \left(1 + \tfrac{2}{\pi}\ln(K+1) + O(K^{-2})\right) E_K(y_t), \tag{5.30}$$

where $E_K(y_t) = \min_{P \in \mathbb{P}_K} \|y_t - P\|_\infty$ is the best-approximation error in the sup-norm, and the $O(K^{-2})$ remainder term is explicit and decays to zero.

(B) Learning Guarantee (TD-Loss Minimisation). *If the TD loss is evaluated (or importance-weighted) under the Chebyshev measure*

$$w(x) = \frac{1}{\pi\sqrt{1 - x^2}}, \qquad x \in (-1, 1),$$

then $\mathcal{L}(\theta)$ is strictly convex in θ and its unique global minimiser θ^* satisfies $\theta_k^* = c_k$ for all $k = 0, \ldots, K$, where c_k are the Chebyshev series coefficients of y_t. Consequently

$$Q(x; \theta^*) = S_K[y_t](x) \qquad \forall\, x \in [-1, 1].$$

This exact recovery holds under the Chebyshev-weighted sampling assumption above*; the general case $\rho \neq w$ is treated in the Remark at the end of the proof. Under this assumption, the learned Q-head achieves the near-minimax bound* (5.30)*, suppressing gradient explosion and ensuring stable convergence of the TD objective.*

Proof. The proof is organised in four steps. Steps 1–3 establish Part (A) via approximation theory; Step 4 establishes Part (B) via convex optimisation.

Step 1: Minimax Baseline (Chebyshev Equioscillation Theorem).

By the Chebyshev Equioscillation Theorem [39, 47], for every $y_t \in C([-1, 1])$ and every $K \geq 0$ there exists a *unique* best-approximation polynomial $P^* \in \mathbb{P}_K$ satisfying

$$E_K(y_t) \;=\; \|y_t - P^*\|_\infty \;\le\; \|y_t - P\|_\infty \qquad \forall\, P \in \mathbb{P}_K.$$

The polynomial P^* is characterised by equioscillation: there exist at least $K + 2$ points $-1 \le x_0 < x_1 < \cdots < x_{K+1} \le 1$ at which

$$\big(y_t - P^*\big)(x_i) = (-1)^i\, E_K(y_t), \qquad i = 0, \ldots, K + 1.$$

No polynomial of degree K can achieve a sup-norm error smaller than $E_K(y_t)$.

Step 2: Near-Minimax Bound via the Projection Identity.

The truncated Chebyshev series $S_K : C([-1,1]) \to \mathbb{P}_K$ is defined by

$$S_K[f](x) = \sum_{k=0}^{K} c_k(f)\, T_k(x), \qquad c_k(f) = \frac{2 - [k=0]}{\pi} \int_{-1}^{1} f(t)\, T_k(t)\, w(t)\, \mathrm{d}t,$$

where $w(t) = (1-t^2)^{-1/2}/\pi$ is the Chebyshev weight and $[k=0]$ is 1 if $k=0$ and 0 otherwise. The map S_K is linear and is a projector onto $\mathbb{P}_K$, i.e. $S_K[P] = P$ for all $P \in \mathbb{P}_K$ (immediate since $\{T_0, \ldots, T_K\}$ is an orthogonal basis for $\mathbb{P}_K$ under w).

Claim. $\|y_t - S_K[y_t]\|_\infty \le (1 + \|S_K\|_{\infty\to\infty})\, E_K(y_t)$.

Proof of Claim. Since $P^* \in \mathbb{P}_K$ and S_K is a projector, $S_K[P^*] = P^*$. By linearity,

$$y_t - S_K[y_t] = (y_t - P^*) - S_K[y_t - P^*].$$

The triangle inequality and the definition of the operator norm $\|S_K\|_{\infty\to\infty} = \sup_{\|f\|_\infty=1} \|S_K[f]\|_\infty$ then give

$$\begin{aligned} \|y_t - S_K[y_t]\|_\infty &\le \|y_t - P^*\|_\infty + \|S_K[y_t - P^*]\|_\infty \\ &\le E_K(y_t) + \|S_K\|_{\infty\to\infty} \|y_t - P^*\|_\infty \\ &= \big(1 + \|S_K\|_{\infty\to\infty}\big)\, E_K(y_t). \qquad \square \end{aligned}$$

Step 3: Bounding the Operator Norm $\|S_K\|_{\infty\to\infty}$.

We derive the logarithmic bound on $\|S_K\|_{\infty\to\infty}$ from first principles.

Kernel representation. Substituting the definition of $c_k(f)$ into $S_K[f](x)$ and interchanging the finite sum with the integral yields

$$S_K[f](x) = \frac{1}{\pi} \int_{-1}^{1} f(t)\, D_K(x,t)\, (1-t^2)^{-1/2}\, \mathrm{d}t,$$

where the *Chebyshev–Dirichlet kernel* is

$$D_K(x,t) = 1 + 2 \sum_{k=1}^{K} T_k(x)\, T_k(t).$$

The interchange is valid because the sum is finite.

Reduction to a trigonometric L^1 norm. Setting $x = \cos\varphi$ and $t = \cos\psi$ with $\varphi, \psi \in [0, \pi]$, and using $T_k(\cos\psi) = \cos(k\psi)$, the kernel becomes the standard trigonometric Dirichlet kernel:

$$D_K(\cos\varphi, \cos\psi) = 1 + 2 \sum_{k=1}^{K} \cos(k\varphi) \cos(k\psi).$$

The substitution $\mathrm{d}t = -\sin\psi\, \mathrm{d}\psi$ and $(1-t^2)^{-1/2} = (\sin\psi)^{-1}$ simplify the Chebyshev weight, and the operator norm reduces to

$$\|S_K\|_{\infty\to\infty} = \sup_{\varphi\in[0,\pi]} \frac{1}{\pi} \int_0^{\pi} \big| D_K(\cos\varphi, \cos\psi) \big|\, \mathrm{d}\psi. \tag{5.31}$$

This is the $L^1[0,\pi]$ norm of the Dirichlet kernel maximised over φ, the same quantity that arises in the theory of Fourier partial sums.

Asymptotic evaluation and the Rivlin–Chebyshev bound. A classical calculation (see [49, Ch. 2], [47, Thm. 7.4]) evaluates the L^1 norm as

$$\frac{1}{\pi}\int_0^\pi |D_K(\cos\varphi,\cos\psi)|\,\mathrm{d}\psi = \frac{2}{\pi}\ln K + \frac{2}{\pi}\left(\gamma_E + \ln\frac{4}{\pi}\right) + O(K^{-2}),$$

where $\gamma_E \approx 0.5772$ is the Euler–Mascheroni constant and the $O(K^{-2})$ remainder is uniform in φ. Because $\ln(K+1) = \ln K + O(K^{-1})$ and the explicit constant satisfies $\frac{2}{\pi}(\gamma_E + \ln(4/\pi)) \approx 0.9625 < 1$, we obtain the *Rivlin–Chebyshev upper bound*

$$\|S_K\|_{\infty\to\infty} \;\le\; \frac{2}{\pi}\ln(K+1) + 1 + O(K^{-2}), \tag{5.32}$$

where the $O(K^{-2})$ term is explicit and decays to zero. Substituting (5.32) into the Claim of Step 2 yields Part (A):

$$\|y_t - S_K[y_t]\|_\infty \;\le\; \left(1 + \tfrac{2}{\pi}\ln(K+1) + O(K^{-2})\right) E_K(y_t). \qquad \square\,\textbf{(A)}$$

Contrast with equispaced polynomial interpolation. For polynomial interpolation through $K+1$ equispaced nodes $x_j = -1 + 2j/K$, the analogous operator is the Lagrange interpolant I_K, whose stability is governed by the Lebesgue constant

$$\Lambda_K = \sup_{x\in[-1,1]} \sum_{j=0}^{K} |\ell_j(x)| = \Theta(2^K/\sqrt{K}),$$

where ℓ_j are the Lagrange basis polynomials [56]. The exponential growth $\Lambda_K = \Theta(2^K)$ means that the error bound $(1+\Lambda_K)E_K(y_t)$ diverges exponentially in K, producing Runge-type boundary oscillations. This is the instability that Chebyshev projection avoids.

Remark on B-spline KANs. Uniform-grid B-spline KANs of fixed degree p do not suffer Runge's phenomenon in its classical sense (their error does not diverge for fixed p); their instability is better characterised by the exponential growth of the condition number of the B-spline collocation matrix on uniform knots as the spline degree grows [14]. This is a related but distinct phenomenon from the Lebesgue constant bound cited above for polynomial interpolation.

Step 4: TD-Loss Minimisation Recovers $S_K[y_t]$ (Proof of Part B).

Strict convexity of $\mathcal{L}(\theta)$. Under the Chebyshev measure $w(x)$, the loss is

$$\mathcal{L}(\theta) = \int_{-1}^{1}\Big(y_t(x) - \textstyle\sum_{k=0}^{K}\theta_k T_k(x)\Big)^2 w(x)\,\mathrm{d}x.$$

Expanding the square and applying the orthogonality relations

$$\int_{-1}^{1} T_j(x)\,T_k(x)\,w(x)\,\mathrm{d}x = \begin{cases}\pi & j=k=0,\\ \pi/2 & j=k\ge 1,\\ 0 & j\ne k,\end{cases}$$

the cross terms in $\theta_j\theta_k$ $(j\ne k)$ vanish, yielding

$$\mathcal{L}(\theta) = \|y_t\|_w^2 \;-\; 2\sum_{k=0}^{K}\theta_k\langle y_t, T_k\rangle_w \;+\; \sum_{k=0}^{K}\theta_k^2\|T_k\|_w^2, \tag{5.33}$$

where $\|T_k\|_w^2 = \pi/2 > 0$ for $k \geq 1$ and $\|T_0\|_w^2 = \pi > 0$. Because each $\|T_k\|_w^2 > 0$, equation (5.33) is a sum of *independent, strictly convex parabolae* in each θ_k. Hence $\mathcal{L}(\theta)$ is *strictly convex* in $\theta \in \mathbb{R}^{K+1}$, so it has a *unique* global minimiser θ^*.

Identification of θ^*. Since the θ_k components are decoupled in (5.33), we minimise each coordinate independently. Setting $\partial\mathcal{L}/\partial\theta_k = 0$ gives

$$-2\langle y_t, T_k\rangle_w + 2\theta_k\|T_k\|_w^2 = 0 \quad\Longrightarrow\quad \theta_k^* = \frac{\langle y_t, T_k\rangle_w}{\|T_k\|_w^2} = c_k,$$

where c_k are precisely the Chebyshev series coefficients of y_t (see Definition D4). Therefore

$$Q(x;\theta^*) = \sum_{k=0}^{K} c_k\, T_k(x) = S_K[y_t](x) \qquad \forall\, x \in [-1,1].$$

Because $\mathcal{L}$ is strictly convex, any gradient-based optimiser (including stochastic gradient descent with a decaying step-size satisfying the Robbins–Monro conditions) converges to this unique minimiser [9].

Consequence. Combining with Part (A):

$$\|y_t - Q(\cdot;\theta^*)\|_\infty = \|y_t - S_K[y_t]\|_\infty \leq \left(1 + \tfrac{2}{\pi}\ln(K+1) + O(K^{-2})\right) E_K(y_t). \qquad \square\,\textbf{(B)}$$

Remark on non-uniform replay distributions. In practice, prioritised experience replay samples from a distribution $\rho(x) \neq w(x)$. Let θ_ρ^* denote the minimiser of $\mathcal{L}$ under ρ. Then $Q(\cdot;\theta_\rho^*)$ is the best $L^2(\rho)$ projection of y_t onto $\mathbb{P}_K$, not the Chebyshev projection. The deviation from Part (B) is bounded by

$$\|S_K[y_t] - Q(\cdot;\theta_\rho^*)\|_\infty \;\leq\; C(K,\rho)\,\|y_t - S_K[y_t]\|_\infty,$$

where $C(K,\rho) \geq 0$ depends on the discrepancy between ρ and w and vanishes when $\rho = w$. Prioritised experience replay corrects for this discrepancy via importance-sampling weights $\beta_i \propto (\rho(x_i)/w(x_i))^{-1}$, which re-weight the empirical loss towards w, thereby recovering the Chebyshev projection guarantee up to a finite-sample error that vanishes as the replay buffer size $|\mathcal{B}| \to \infty$. $\square$

Corollary 5.4.5 (Gradient Stability of the ChebyKAN Q-Head)**.** *Under the assumptions of Theorem 5.4.4, let* Θ *denote the parameters of the upstream GNN encoder and let* $\nabla_\Theta\mathcal{L}_{\mathrm{TD}}$ *be the gradient of the TD loss with respect to* Θ*. Assume the network parameters satisfy* $\|\Theta\| \leq B$ *for some finite constant* $B > 0$ *(enforced in practice by the weight-decay regularisation with coefficient* $\lambda = 10^{-5}$*). Then:*

(i) TD-Error Bound. *The pointwise TD error of the learned Q-head satisfies*

$$\|y_t - Q(\cdot;\theta^*)\|_\infty \;\leq\; \underbrace{\left(1 + \tfrac{2}{\pi}\ln(K+1) + O(K^{-2})\right)}_{=:\;\mu_K} E_K(y_t), \tag{5.34}$$

where $\mu_K = O(\log K)$ *grows only logarithmically in* K*. In contrast, any equispaced polynomial interpolation scheme incurs a multiplicative factor* $1 + \Lambda_K = \Theta(2^K)$*, which diverges exponentially.*

(ii) Gradient-Norm Bound. *The TD-loss gradient satisfies*

$$\left\|\nabla_\Theta\mathcal{L}_{\mathrm{TD}}\right\| \;\leq\; 2\,\mu_K\,E_K(y_t)\cdot G(B), \tag{5.35}$$

where $G(B) = \sup_{\|\Theta\| \le B} \|\nabla_\Theta Q(\cdot;\theta^*)\| < \infty$ *is a finite constant depending only on* B *and the architecture. Hence the gradient norm is also bounded by a factor that grows at most logarithmically in* K.

Proof. **Part (i)** is an immediate restatement of Theorem 5.4.4(A)–(B) combined: by Part (B), $Q(\cdot;\theta^*) = S_K[y_t]$, and by Part (A), $\|y_t - S_K[y_t]\|_\infty \le \mu_K E_K(y_t)$.

Part (ii). The TD gradient is

$$\nabla_\Theta \mathcal{L}_{\mathrm{TD}} = -2\,\mathbb{E}_{x\sim\rho}\big[(y_t(x) - Q(x;\theta^*))\,\nabla_\Theta Q(x;\theta^*)\big].$$

Taking norms and applying the triangle inequality:

$$\big\|\nabla_\Theta \mathcal{L}_{\mathrm{TD}}\big\| \;\le\; 2\,\mathbb{E}_{x\sim\rho}\big[|y_t(x) - Q(x;\theta^*)| \cdot \|\nabla_\Theta Q(x;\theta^*)\|\big].$$

Bounding the first factor pointwise by $\|y_t - Q(\cdot;\theta^*)\|_\infty \le \mu_K E_K(y_t)$ from Part (i), and defining $G(B) = \sup_{\|\Theta\|\le B}\|\nabla_\Theta Q(\cdot;\theta^*)\|$, which is finite by continuity of the network and compactness of $\{\Theta : \|\Theta\| \le B\}$, gives (5.35). The weight-decay regularisation $\lambda\|\Theta\|^2$ enforces $\|\Theta\| \le B$ throughout training, ensuring $G(B)$ is a genuine constant.

Note that $G(B)$ is independent of K; thus the entire right-hand side of (5.35) grows only as $O(\log K)$ in K, in contrast to the $\Theta(2^K)$ growth that would arise from an equispaced interpolation scheme. □

5.4.3 Universal Approximation of the KAN-GNN Encoder

A natural question is whether replacing fixed affine MLP projections with learnable B-spline KAN projections sacrifices any representational power — i.e., whether there exist node-level functions that a standard MLP-GNN can approximate but a KAN-GNN cannot. We show the opposite: the KAN-GNN function class contains a uniformly dense image of the MLP-GNN function class at any fixed depth, and at any fixed parameter budget per layer the KAN edge-function family strictly contains the MLP edge-function family. Hence the substitution is expressivity-preserving by construction.

We first recall the two classical results on which the theorem rests.

- **MLP subsumption by KAN [35].** For any affine-plus-activation layer $\mathbf{y} = \sigma(W\mathbf{x} + \mathbf{b})$ with $W \in \mathbb{R}^{d_{\mathrm{out}}\times d_{\mathrm{in}}}$ and a fixed nonlinearity σ (e.g., SiLU as in Eq. (11) of KARL), there exists a KAN layer of the same input–output dimension whose per-edge functions $\varphi_{j,i}$ reproduce the affine-plus-activation map: setting the spline coefficients to zero and the base activation weights $w_{j,i} = W_{j,i}$ in Eq. (11) gives $\mathrm{KAN}(\mathbf{x})_j = \sum_i W_{j,i}\,\sigma(x_i) + b_j$, which equals the MLP layer exactly. Furthermore, by Theorem 2.1 of [35], B-splines of order k on a uniform grid of size G approximate any C^{k+1} function on a compact domain to uniform error $O(G^{-(k+1)})$, so KAN layers approximate (and contain) MLP layers up to arbitrary precision.

- **GNN message-passing universality [38, 64].** Any permutation-equivariant node-level function of the form $h'_v = f\big(h_v, \{\{h_u : u \in \mathcal{N}(v)\}\}\big)$ that is continuous on a compact domain can be approximated arbitrarily well by a message-passing GNN whose Combine and Aggregate functions are universal approximators (e.g., MLPs with sufficient width/depth).

Theorem 5.4.6 (KAN-GNN Subsumes MLP-GNN)**.** *Let $\mathcal{F}_{\mathrm{MLP}}$ denote the class of functions computed by an $l_{\max}$-layer message-passing GNN whose* Combine *and* Aggregate *steps are implemented by MLPs of fixed width d and depth p with a fixed 1-Lipschitz nonlinearity σ, evaluated on inputs in the compact domain $[-1,1]^d$ (enforced by the* tanh *pre-activations of Eq. (14) of KARL). Let $\mathcal{F}_{\mathrm{KAN}}$ denote the class of functions computed by the same message-passing GNN with each MLP layer replaced by a KAN layer of the same input–output dimensions, B-spline order $k \geq 1$, and grid resolution G. Then:*

1. (Subsumption / uniform density.) *For every $F_{\mathrm{MLP}} \in \mathcal{F}_{\mathrm{MLP}}$ and every $\varepsilon > 0$, there exists a KAN-GNN of the same depth $l_{\max}$, the same embedding dimension d, B-spline order k, and a sufficiently fine grid $G = G(\varepsilon, l_{\max}, L_\sigma)$ whose realised function F_{KAN} satisfies*
$$\sup_{\mathbf{x}\in[-1,1]^d} \left\| F_{\mathrm{MLP}}(\mathbf{x}) - F_{\mathrm{KAN}}(\mathbf{x}) \right\|_2 \;\leq\; \varepsilon.$$
Equivalently, $\mathcal{F}_{\mathrm{MLP}} \subseteq \overline{\mathcal{F}_{\mathrm{KAN}}}$ where the closure is taken under uniform convergence on $[-1,1]^d$.

2. (Strict per-layer containment.) *For $k \geq 1$ and any $G \geq 1$, the per-edge function family realisable by a KAN layer strictly contains the per-edge function family realisable by an MLP layer of identical input–output dimension. In particular, there exist KAN-realisable per-edge functions $\varphi : [-1,1] \to \mathbb{R}$ that no MLP layer of the same input–output dimension and the same fixed nonlinearity can represent exactly, and which require $\Omega(\omega)$ width to approximate to error $O(1/\omega)$ uniformly when their total variation grows linearly with a frequency parameter ω.*

Proof. We prove each part in turn.

Part (i): Subsumption. Let $F_{\mathrm{MLP}} \in \mathcal{F}_{\mathrm{MLP}}$ be any function computed by the MLP-GNN. Following the structure of Eq. (14) of KARL with MLP modules in place of the KAN modules, each message-passing round takes the form
$$h_v^{(l+1)} \;=\; \mathrm{MLP}_{\mathrm{out}}\Big(\mathrm{MLP}_{\mathrm{neigh}}\big(\tanh(\mathrm{Agg}(h^{(l)}))\big) \;\big\|\; \mathrm{MLP}_{\mathrm{self}}\big(\tanh(h_v^{(l)})\big)\Big). \tag{5.36}$$
Each $\mathrm{MLP}_\bullet$ is a finite composition of affine maps and a fixed 1-Lipschitz nonlinearity σ. The pre-activation $\tanh(\cdot)$ guarantees that every input to every $\mathrm{MLP}_\bullet$ lies in $[-1,1]^d$, which is the compact domain on which the B-spline approximation bound of [35] holds (the domain $[-1,1]$ is affinely equivalent to the $[0,1]$ domain used in [35, Theorem 2.1] and the bound transfers without loss).

Concatenation ℓ_2-isometry. The concatenation operator $\cdot \,\|\, \cdot : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^{2d}$ appearing in Eq. (5.36) is an exact ℓ_2-isometry: for any $a, b \in \mathbb{R}^d$,
$$\|a \,\|\, b\|_2 \;=\; \sqrt{\|a\|_2^2 + \|b\|_2^2}. \tag{5.37}$$
We use this identity (rather than the looser $\|a\|_2 + \|b\|_2$ triangle bound) in the error-propagation analysis below.

Per-layer KAN approximation. By the MLP-subsumption property of KANs [35], for any affine-plus-σ layer of finite depth p and any $\eta' > 0$, there exists a KAN layer of the same

input–output dimension with B-spline order k and grid resolution $G \geq G_0(\eta')$ satisfying

$$\sup_{\mathbf{x}\in[-1,1]^{d_{\text{in}}}} \left\|\text{MLP}(\mathbf{x}) - \text{KAN}(\mathbf{x})\right\|_2 \;\leq\; \eta', \tag{5.38}$$

with $G_0(\eta') = O((\eta')^{-1/(k+1)})$ by Theorem 2.1 of [35].

Lipschitz tracking through the GNN. Let L_{neigh}, L_{self}, L_{out} denote uniform Lipschitz constants of the three MLP modules with respect to the ℓ_2 norm on $[-1,1]^d$, and let L_Φ be a Lipschitz constant for the composed update operator $\Phi^{(l)} : (h_v^{(l)}, \{h_u^{(l)}\}_{u\in\mathcal{N}(v)}) \mapsto h_v^{(l+1)}$. Because tanh is 1-Lipschitz, each MLP layer is $L_\sigma \cdot \prod_q \|W^{(q)}\|_2$-Lipschitz with $L_\sigma \leq 1$, and the unweighted sum aggregator over a graph with maximum degree $d_{\max}$ contributes a factor $(1 + d_{\max})$. Hence $L_\Phi \leq L_{\text{out}} \cdot (L_{\text{neigh}} \cdot d_{\max} + L_{\text{self}})$ is finite because all weight norms are bounded throughout training (weight decay $\lambda = 10^{-5}$, Eq. (27) of KARL). Define the geometric amplification factor

$$S(l_{\max}, L_\Phi) \;:=\; \sum_{l=0}^{l_{\max}-1} L_\Phi^l \;=\; \begin{cases} \dfrac{L_\Phi^{\,l_{\max}} - 1}{L_\Phi - 1}, & L_\Phi \neq 1, \\ l_{\max}, & L_\Phi = 1, \end{cases} \tag{5.39}$$

which we will use to consolidate the error accumulated over $l_{\max}$ message-passing rounds.

Per-iteration injected error. At each message-passing round we substitute all three nested MLP modules by KAN approximations of accuracy η' (Eq. (5.38)). Let Φ denote the original MLP composed update and $\widetilde{\Phi}$ the KAN-substituted update, and let $e_{\text{neigh}}, e_{\text{self}} \in \mathbb{R}^d$ denote the per-module approximation errors with $\|e_{\text{neigh}}\|_2, \|e_{\text{self}}\|_2 \leq \eta'$. The errors of $\text{KAN}_{\text{neigh}}$ and KAN_{self} propagate through MLP_{out} before adding to the KAN_{out} error itself; using the concatenation isometry (5.37) and the triangle inequality,

$$\begin{aligned} \sup_{\mathbf{x}} \left\|\Phi(\mathbf{x}) - \widetilde{\Phi}(\mathbf{x})\right\|_2 \;&\leq\; \eta' + L_{\text{out}} \cdot \left\|e_{\text{neigh}} \,\|\, e_{\text{self}}\right\|_2 \\ &=\; \eta' + L_{\text{out}} \cdot \sqrt{\|e_{\text{neigh}}\|_2^2 + \|e_{\text{self}}\|_2^2} \\ &\leq\; \eta' + L_{\text{out}} \cdot \sqrt{2}\,\eta' \;=\; \eta'\big(1 + \sqrt{2}\,L_{\text{out}}\big). \end{aligned} \tag{5.40}$$

Define the consolidated per-iteration injected error as

$$\eta \;:=\; \eta'\big(1 + \sqrt{2}\,L_{\text{out}}\big),$$

so that each round contributes at most η of fresh error in the supremum norm. Note η' can always be chosen so that η takes any prescribed positive value, since $1 + \sqrt{2}\,L_{\text{out}}$ is a fixed (finite) constant governed by the trained MLP weights. Compared with the looser bound $\eta'(1 + 2L_{\text{out}})$ obtained from the triangle inequality alone, the isometric bound (5.40) reduces the constant by the factor $(1 + \sqrt{2}\,L_{\text{out}})/(1 + 2L_{\text{out}})$, which approaches $\sqrt{2}/2 \approx 0.707$ as $L_{\text{out}} \to \infty$.

Error propagation across $l_{\max}$ rounds. Let $\delta_v^{(l)} := h_v^{(l)} - \tilde{h}_v^{(l)}$ denote the difference between MLP-GNN and KAN-GNN node embeddings at iteration l, with $\delta_v^{(0)} = 0$ (both networks share identical inputs). At each iteration, propagated error is amplified by L_Φ and a fresh error of at most η is injected:

$$\max_v \|\delta_v^{(l+1)}\|_2 \;\leq\; L_\Phi \cdot \max_u \|\delta_u^{(l)}\|_2 + \eta.$$

Unrolling this recursion from $l = 0$ to $l_{\max} - 1$ with $\delta^{(0)} = 0$ and using (5.39) yields

$$\max_v \|\delta_v^{(l_{\max})}\|_2 \;\le\; \eta \cdot S(l_{\max}, L_\Phi).$$

Choose

$$\eta \;=\; \frac{\varepsilon}{S(l_{\max}, L_\Phi)}, \qquad \eta' \;=\; \frac{\eta}{1 + \sqrt{2}\, L_{\text{out}}} \;=\; \frac{\varepsilon}{(1 + \sqrt{2}\, L_{\text{out}})\, S(l_{\max}, L_\Phi)},$$

and pick the corresponding grid resolution $G \ge G_0(\eta')$ from Eq. (5.38). The downstream KANformer (Sec. 4.4) and Chebyshev-KAN heads (Sec. 4.5) are themselves Lipschitz on $[-1, 1]^d$ and admit the same per-module subsumption argument: their contribution multiplies the encoder error by an additional finite Lipschitz constant L_{head} which can be absorbed into the choice of η by re-scaling. We thus obtain

$$\sup_{\mathbf{x} \in [-1,1]^d} \big\| F_{\text{MLP}}(\mathbf{x}) - F_{\text{KAN}}(\mathbf{x}) \big\|_2 \;\le\; \varepsilon.$$

Since F_{MLP} and $\varepsilon > 0$ were arbitrary, $F_{\text{MLP}} \in \overline{\mathcal{F}_{\text{KAN}}}$, completing Part (i).

Part (ii): Strict per-layer containment. Fix a single KAN edge function $\varphi : [-1, 1] \to \mathbb{R}$ from Eq. (10) of the main paper:

$$\varphi(x) \;=\; w\big(\sigma(x) + \textstyle\sum_{i=1}^{G+k} c_i\, B_i^k(x; G)\big).$$

For $k \ge 1$ and $G \ge 1$, the B-spline term contributes a non-trivial piecewise polynomial of order k with $G + k$ free coefficients, which spans a $(G + k)$-dimensional function space on $[-1, 1]$. By contrast, the corresponding edge function in an MLP layer of the same input–output dimension reduces to $x \mapsto W_{j,i}\, \sigma(x)$, a one-dimensional family parameterised by the single scalar $W_{j,i}$. Hence the KAN per-edge function family strictly contains the MLP per-edge function family as subsets of $C([-1, 1])$ for all $k \ge 1$, $G \ge 1$.

Quantitative gap on high-variation targets. Consider the target $f_\omega^*(x) = \sin(\omega \pi x)$ on $[-1, 1]$, which has total variation $\mathrm{TV}(f_\omega^*) = 2\omega$. A B-spline of order $k \ge 1$ on a grid of size $G \ge \omega$ approximates f_ω^* with uniform error $O((\omega/G)^{k+1})$ [14], using $O(G)$ parameters. To match this accuracy with a single-hidden-layer MLP of width W and a fixed 1-Lipschitz activation σ, classical lower bounds on best approximation by ridge functions [5, 37] require $W = \Omega(\omega)$ parameters when f_ω^* is treated as a univariate target. Hence at any per-layer parameter budget P with $P < \omega$, no single-layer MLP can approximate f_ω^* to error smaller than a constant, while a KAN with $G + k = O(P)$ achieves error $O((\omega/P)^{k+1})$. This quantitative gap, while not strictly establishing a global containment for arbitrarily deep MLPs (which can compose to recover high frequencies), confirms that at fixed depth and per-layer width the KAN edge family captures high-frequency variation that the MLP edge family cannot, justifying KARL's preference for KAN over wider/deeper MLPs of equivalent parameter budget. □ □

Remark 5.4.7 (Practical significance for KARL). Theorem 5.4.6 directly addresses the design question: *"Why use KANs rather than a deeper MLP?"* Part (i) guarantees that no representational power is lost by switching from MLP-GNN to KAN-GNN: any dismantling policy expressible by MultiDismantler's MLP-GNN is also expressible (to arbitrary uniform precision) by KARL's KAN-GNN of the same depth and embedding dimension, provided the B-spline grid is sufficiently fine. Part (ii) shows the gain at the per-layer level: at

any fixed grid resolution G and order $k \geq 1$, each KAN edge function spans a $(G + k)$-dimensional family that strictly contains the 1-dimensional family available to an MLP edge, allowing KARL to fit high-frequency per-node vulnerability functions – precisely those that govern cascading fragility in scale-free multiplex topologies – without resorting to the depth/width inflation that would be required of an MLP-GNN. This is complementary to the stability guarantee of Proposition 5.4.1: the residual branch controls *gradient dynamics* during training, while Theorem 5.4.6 characterises the *function class* of the trained model at convergence. The two results together provide end-to-end theoretical support for the KAN-based design of KARL.

Remark 5.4.8 (On the tanh pre-activation). The tanh pre-activations in Eq. (14) of KARL are essential for Theorem 5.4.6 not merely as implementation details but as the mechanism that enforces $h_v^{(l)} \in [-1, 1]^d$ at every iteration, guaranteeing that all KAN inputs lie within the compact domain on which the B-spline approximation error bound $O(G^{-(k+1)})$ holds [35, Theorem 2.1]. Without this constraint, the error bound would require an additional argument about domain adaptation of the spline grid, available via the grid-extension technique of Liu et al. [35] but complicating the proof. The tanh pre-activation is therefore both an architectural and a proof-enabling design choice.

5.4.4 Parameter Efficiency of KAN over MLP at Fixed Budget

We establish a parameter-efficiency separation between KAN and MLP edge functions that holds with no constraint on the MLP weight magnitudes. The argument relies on weight-free counting bounds (linear regions for ReLU; zero counting for analytic activations) and is stated in two genuinely informative regimes: a critical parameter threshold $P \asymp \omega$, and an asymptotic-rate regime restricted to piecewise-linear MLPs. The latter restriction is essential: as is well known [22, 41], single-hidden-layer MLPs with real-analytic activations achieve *exponential* approximation rates on analytic targets, so a fixed-order B-spline KAN cannot win an asymptotic-rate comparison against an analytic-MLP in 1D. The ReLU/piecewise-linear case is the relevant one for our empirical baselines (MultiDismantler and all GNN-DRL prior work use ReLU MLPs).

Proposition 5.4.9 (Parameter Efficiency of KAN over MLP)**.** *Let $\mathcal{F}_{\mathrm{KAN}}(P)$ and $\mathcal{F}_{\mathrm{MLP}}(P)$ denote the classes of per-edge univariate functions realisable by a B-spline KAN layer of order $k \geq 1$ and a single-hidden-layer MLP layer respectively, each with total parameter budget P and a fixed activation σ that is either* (a) *ReLU (or any continuous piecewise-linear activation with a constant number of pieces), or* (b) *a real-analytic, non-polynomial* 1*-Lipschitz function (e.g.* tanh*, sigmoid, SiLU). For any* integer *frequency $\omega \in \mathbb{N}$ consider the target*

$$f_\omega^*(x) = \sin(\omega \pi x), \qquad x \in [-1, 1].$$

Then:

1. ***(KAN upper bound at budget P).*** *A B-spline KAN of order k with grid size $G = P - k$ achieves*

$$\varepsilon_{\mathrm{KAN}}(P) \;:=\; \inf_{\hat{f}_{\mathrm{KAN}} \in \mathcal{F}_{\mathrm{KAN}}(P)} \sup_{x \in [-1,1]} \left| f_\omega^*(x) - \hat{f}_{\mathrm{KAN}}(x) \right| \;\leq\; C_k \left(\frac{\omega}{P} \right)^{k+1}, \tag{5.41}$$

where C_k depends only on k.

2. ***(Weight-free MLP lower bound, both activation classes).*** *There exists a constant $\kappa^* \in (0, 1]$ depending only on σ such that for any single-hidden-layer MLP $\hat{f}_{\mathrm{MLP}}(x) = \sum_{j=1}^{W} a_j \sigma(w_j x + b_j)$ with width $W < \kappa^* \omega$, and* regardless of the values *of its weights $\{a_j, w_j, b_j\} \in \mathbb{R}$ (in particular, with no bound on $|a_j|$ or $|w_j|$),*

$$\varepsilon_{\mathrm{MLP}}(W) \;:=\; \inf_{\hat{f}_{\mathrm{MLP}} \in \mathcal{F}_{\mathrm{MLP}}(W)} \; \sup_{x \in [-1,1]} \left| f^*_\omega(x) - \hat{f}_{\mathrm{MLP}}(x) \right| \;\geq\; c_\sigma \;>\; 0, \tag{5.42}$$

where $c_\sigma > 0$ depends only on σ. Specifically: $\kappa^* = 1$ and $c_\sigma = 1$ for ReLU; $\kappa^* = 1/c_0(\sigma)$ and $c_\sigma > 0$ a σ-dependent constant for analytic σ, where $c_0(\sigma)$ is the Maiorov–Pinkus zero-count multiplier [37, 46].

3. ***(Critical-threshold parameter-efficiency gap, both activation classes).*** *Fix any constants $\kappa_{\mathrm{KAN}} > 0$ and $\kappa_{\mathrm{MLP}} < \kappa^*$, and set the budgets*

$$P_{\mathrm{KAN}} = \kappa_{\mathrm{KAN}}\, \omega, \qquad W_{\mathrm{MLP}} = \kappa_{\mathrm{MLP}}\, \omega.$$

Then for every $\omega \in \mathbb{N}$,

$$\varepsilon_{\mathrm{KAN}}(P_{\mathrm{KAN}}) \;\leq\; C_k\, \kappa_{\mathrm{KAN}}^{-(k+1)}, \qquad \varepsilon_{\mathrm{MLP}}(W_{\mathrm{MLP}}) \;\geq\; c_\sigma. \tag{5.43}$$

Choosing $\kappa_{\mathrm{KAN}} > (C_k/c_\sigma)^{1/(k+1)}$ yields $\varepsilon_{\mathrm{KAN}}(P_{\mathrm{KAN}}) < c_\sigma \leq \varepsilon_{\mathrm{MLP}}(W_{\mathrm{MLP}})$, so the KAN error is strictly smaller than the MLP error at every frequency, even when the parameter ratio $P_{\mathrm{KAN}}/W_{\mathrm{MLP}} = \kappa_{\mathrm{KAN}}/\kappa_{\mathrm{MLP}}$ is a constant independent of ω.

4. ***(Asymptotic-rate gap, ReLU MLPs only).*** Restrict σ to ReLU or any fixed continuous piecewise-linear activation. *Fix $\omega \in \mathbb{N}$ and let $P \to \infty$. Then the optimal MLP error obeys a strict* lower *bound [21, 22]*

$$\varepsilon_{\mathrm{MLP}}(P) \;\geq\; \frac{\omega^2 \pi^2}{8\, P^2} \;=:\; c'_\omega\, P^{-2}, \tag{5.44}$$

so combining (5.44) with the KAN upper bound (5.41) gives

$$\frac{\varepsilon_{\mathrm{MLP}}(P)}{\varepsilon_{\mathrm{KAN}}(P)} \;\geq\; \frac{c'_\omega\, P^{-2}}{C_k\, (\omega/P)^{k+1}} \;=\; \frac{c'_\omega}{C_k\, \omega^{k+1}}\, P^{\,k-1}. \tag{5.45}$$

For any $k \geq 2$ (e.g. cubic splines $k = 3$ used in KARL), the exponent $k - 1 \geq 1$ is positive and the ratio diverges as $P \to \infty$.

Remark on analytic activations. *The asymptotic-rate gap of Part (iv) does* not *hold when σ is real-analytic. Real-analytic single-hidden-layer MLPs achieve rates of order* $\exp(-cP^{1/d})$ *on analytic targets in $\mathbb{R}^d$ (and* $\exp(-cP)$ *in $d = 1$) [22, 41], which eventually beat any fixed-order polynomial rate $P^{-(k+1)}$. Consequently, the asymptotic-rate comparison applies only to the piecewise-linear MLP family, which is precisely the family used in every GNN-DRL baseline for network dismantling, including MultiDismantler. The critical-threshold gap of Part (iii), in contrast, holds for both activation classes.*

Proof. **Part (i).** By the classical B-spline approximation theorem [14], a uniform B-spline of order k on a grid of G intervals approximates any C^{k+1} function f on a compact domain

to uniform error $O\big(G^{-(k+1)}\,\|f^{(k+1)}\|_\infty\big)$. For $f^*_\omega(x) = \sin(\omega\pi x)$ we have $\|(f^*_\omega)^{(k+1)}\|_\infty = (\omega\pi)^{k+1}$. Setting $G = P - k$ and absorbing π^{k+1} into C_k yields (5.41).

Part (ii) – weight-free lower bound. The argument replaces the previous total-variation step (which silently constrained $\max_j |a_j|$) by a counting argument that holds for arbitrary real weights.

Case (a): σ is ReLU. Set $\kappa^* = 1$. Each summand $a_j\sigma(w_j x + b_j)$ is a continuous piecewise-linear function with at most one breakpoint on $[-1, 1]$. Hence $\hat{f}_{\text{MLP}}$ has at most $W + 1$ linear pieces on $[-1, 1]$; this bound depends only on W and not on weight magnitudes.

Because $\omega \in \mathbb{N}$, the target f^*_ω has exactly $2\omega + 1$ distinct zeros on $[-1, 1]$ at $x_m = m/\omega$, $m = -\omega, \ldots, \omega$. (Note: this exact count fails for non-integer ω because the endpoints ± 1 are not zeros in that case; restricting to $\omega \in \mathbb{N}$ aligns precisely with the discrete spectrum of frequencies relevant to high-frequency network vulnerability functions.) These zeros partition $[-1, 1]$ into 2ω consecutive open subintervals $\{(x_m, x_{m+1})\}$, on each of which the midpoint $\xi_m = (2m + 1)/(2\omega)$ is the extremum $|f^*_\omega(\xi_m)| = 1$.

Since $W < \omega$, the MLP has $W + 1 \le \omega < 2\omega$ linear pieces, strictly fewer than the 2ω subintervals. By pigeonhole, at least one subinterval (x_m, x_{m+1}) is contained entirely within a single linear piece on which $\hat{f}_{\text{MLP}}$ is an affine function ℓ. The supremum-norm error of ℓ on $[x_m, x_{m+1}]$ then satisfies

$$\begin{aligned}\sup_{x\in[x_m,x_{m+1}]} \big|f^*_\omega(x) - \ell(x)\big| \;&\ge\; \tfrac{1}{2}\big|f^*_\omega(x_m) + f^*_\omega(x_{m+1}) - 2f^*_\omega(\xi_m)\big| \\ &=\; \tfrac{1}{2}\big|0 + 0 - 2(\pm 1)\big| \;=\; 1,\end{aligned}$$

where the inequality is the affine-midpoint identity (the average of endpoint values of an affine function equals its midpoint value). This yields $\varepsilon_{\text{MLP}}(W) \ge 1$ with $c_\sigma = 1$, independent of weight magnitudes.

Case (b): σ is real-analytic and non-polynomial. Suppose for contradiction that $\sup_x |f^*_\omega - \hat{f}_{\text{MLP}}| < 1$. Since f^*_ω attains the values ± 1 alternately at its 2ω extrema $\xi_m = (2m + 1)/(2\omega)$ in $[-1, 1]$, the residual $r := f^*_\omega - \hat{f}_{\text{MLP}}$ takes the same sign as $f^*_\omega(\xi_m)$ at each extremum, so r has at least $2\omega - 1$ sign changes on $[-1, 1]$ and hence at least $2\omega - 1$ distinct zeros.

By Maiorov–Pinkus [37, 46], every non-trivial linear combination of W analytic ridge functions $\{\sigma(w_j \cdot + b_j)\}_{j=1}^{W}$ has at most $c_0(\sigma)\, W$ zeros on any compact interval, with $c_0(\sigma) > 0$ a constant depending only on σ and not on the inner or outer weights. Set $\kappa^* := 1/c_0(\sigma)$. If $W < \kappa^*\omega$ then $c_0(\sigma)\, W < \omega \le 2\omega - 1$ for all $\omega \ge 1$, so the zero count of r exceeds the maximum allowed by Maiorov–Pinkus, contradicting our supposition. The previous version of this proof used only the weaker bound $W < \omega$, which is insufficient when $c_0(\sigma) \ge 2$; the corrected bound $W < \kappa^*\omega$ restores the contradiction for every analytic σ. We conclude $\varepsilon_{\text{MLP}}(W) \ge c_\sigma > 0$ with c_σ depending only on σ, again with no weight constraint.

Part (iii) – critical threshold. With $W_{\text{MLP}} = \kappa_{\text{MLP}}\omega < \kappa^*\omega$, Part (ii) gives $\varepsilon_{\text{MLP}}(W_{\text{MLP}}) \ge c_\sigma$. With $P_{\text{KAN}} = \kappa_{\text{KAN}}\omega$, Part (i) gives $\varepsilon_{\text{KAN}}(P_{\text{KAN}}) \le C_k \kappa_{\text{KAN}}^{-(k+1)}$. Both bounds are uniform in ω, so (5.43) holds for every $\omega \in \mathbb{N}$. The strict gap follows by choosing $\kappa_{\text{KAN}} > (C_k/c_\sigma)^{1/(k+1)}$.

Part (iv) – asymptotic rate, ReLU only. We first establish the lower bound. Restrict σ to ReLU. By the equivalence between ReLU MLPs and free-knot linear splines [22], any ReLU MLP of width W is a continuous piecewise-linear function with at most $W+1$ pieces on $[-1,1]$, so the parameter budget $P \geq W$ yields at most $P+1$ pieces. Let $\hat{f}_{\mathrm{MLP}}$ be any such function with breakpoints $-1 = t_0 < t_1 < \cdots < t_M = 1$, $M \leq P$. The M pieces have total length $\sum_{i=1}^{M}(t_i - t_{i-1}) = 2$, so by pigeonhole at least one piece $[t_{i-1}, t_i]$ has length $\ell_i \geq 2/M \geq 2/P$.

On this piece, $\hat{f}_{\mathrm{MLP}}$ is affine, so the standard affine-approximation error bound for C^2 functions [22] gives

$$\sup_{x\in[t_{i-1},t_i]} \left|f^*_\omega(x) - \hat{f}_{\mathrm{MLP}}(x)\right| \;\geq\; \frac{\ell_i^2}{8} \min_{x\in[t_{i-1},t_i]} \left|(f^*_\omega)''(x)\right|.$$

For $f^*_\omega(x) = \sin(\omega\pi x)$, $(f^*_\omega)''(x) = -(\omega\pi)^2 \sin(\omega\pi x)$. Since $\omega \in \mathbb{N}$ and the piece $[t_{i-1}, t_i]$ has length $\ell_i \geq 2/P$, for P large enough that $\ell_i < 1/(2\omega)$ (i.e. $P > 4\omega$), the piece is contained within a half-period of $\sin(\omega\pi x)$ in which $|\sin(\omega\pi x)|$ has range at least $1 - \cos(\omega\pi\ell_i/2) \geq c$ for some $c > 0$ on a subinterval of length $\geq \ell_i/2$, so $\min_{x\in[t_{i-1}+\ell_i/4,\, t_i-\ell_i/4]} |(f^*_\omega)''(x)| \geq (\omega\pi)^2/2$. Hence

$$\varepsilon_{\mathrm{MLP}}(P) \;\geq\; \frac{\ell_i^2}{8}\cdot\frac{(\omega\pi)^2}{2} \;\geq\; \frac{(2/P)^2}{8}\cdot\frac{(\omega\pi)^2}{2} \;=\; \frac{\omega^2\pi^2}{4\,P^2} \;\geq\; \frac{\omega^2\pi^2}{8\,P^2},$$

which establishes (5.44) with $c'_\omega = \omega^2\pi^2/8$.

The previous version of this proof attempted to substitute an *upper* bound $\varepsilon_{\mathrm{MLP}} \leq c'_\sigma P^{-r}$ into the numerator of a $\geq$ inequality, which is algebraically invalid: $A \leq B$ does not imply $A/C \geq B/C$. The corrected argument here uses a strict *lower* bound $\varepsilon_{\mathrm{MLP}}(P) \geq c'_\omega P^{-2}$, which is the correct quantity to lower-bound the ratio.

Combining (5.44) with (5.41) yields (5.45). For $k \geq 2$, the exponent $k - 1 \geq 1$, so the ratio diverges as $P \to \infty$ with ω fixed.

The restriction to ReLU is essential: for real-analytic σ approximating an analytic target, single-hidden-layer MLPs achieve exponential rates $\varepsilon_{\mathrm{MLP}}(P) = O(e^{-cP})$ in $d = 1$ [22, 41], which dominate the polynomial KAN rate $P^{-(k+1)}$ for sufficiently large P at any fixed k. The previous version of this proof claimed Part (iv) for both activation classes, which is false; it now applies only to ReLU / piecewise-linear MLPs (the family used in all GNN-DRL baselines). □ □

Remark 5.4.10 (Scope of the two regimes). Parts (iii) and (iv) play complementary roles and have different scope. The critical-threshold gap (Part iii) is the structurally informative statement: at any fixed frequency ω, allocating a constant-factor more budget to KAN than to MLP keeps the KAN error below the ω-independent threshold c_σ while the MLP error remains above it. This holds for both ReLU and analytic activations, and is the regime in which the comparison is empirically meaningful for KARL: the per-edge function families have budgets of the same order, and the gap is driven by functional form (a $(G+k)$-dimensional spline family vs. a 1-dimensional scalar-weighted activation), not by parameter count.

The asymptotic-rate gap (Part iv) is restricted to ReLU/piecewise-linear MLPs – the family used by every GNN-DRL baseline for network dismantling, including MultiDismantler. The

restriction is fundamental: single-hidden-layer MLPs with analytic activations approximating analytic 1D targets converge exponentially in P [22, 41], eventually surpassing any fixed-order polynomial rate. Claiming an asymptotic-rate gap there would be wrong. Inside the ReLU class, however, the gap is genuine and the rate ratio is P^{k-1}, which is positive for $k \geq 2$ as used in KARL.

Remark 5.4.11 (Implication for KARL vs. parameter-matched MLP-GNN). Proposition 5.4.9 establishes that the performance gap between KARL and a parameter-matched MLP-GNN cannot be attributed to parameter count alone, with no constraint on the MLP weight magnitudes. In the critical-threshold regime (Part iii), at any frequency $\omega \in \mathbb{N}$ characteristic of the per-node vulnerability functions in scale-free multiplex networks, KARL's KAN encoder needs only $O(\omega)$ parameters per edge to drive its approximation error below c_σ, while a single-hidden-layer MLP edge with width $W < \kappa^*\omega$ remains uniformly far from the target regardless of its weights. The counting bounds underlying Part (ii) – $W + 1$ linear pieces for ReLU, $c_0(\sigma)\, W$ residual zeros for analytic σ – are weight-free and close the loophole in the previous total-variation argument. In the asymptotic-rate regime (Part iv) – which is the regime relevant to KARL's actual baseline since MultiDismantler uses ReLU MLPs – KAN error decays as $P^{-(k+1)}$ versus the strict ReLU lower bound $\Omega(P^{-2})$, giving a P^{k-1} rate gap for cubic splines ($k = 3$ in KARL). Theorem 5.4.6 further guarantees that KARL's KAN-GNN subsumes the MLP-GNN function class, so any gains from additional MLP parameters are already available to KARL at the same budget.

5.5 Experimental Results

KARL is evaluated in a strictly zero-shot regime: the agent is trained once on small synthetic multiplex networks ($N \in [30, 50]$ nodes) and deployed without any fine-tuning on real-world benchmarks spanning three orders of magnitude in scale. This section reports results under both unit and degree removal costs on six real-world multiplex networks (Section 5.5.2–5.5.3), followed by a formal statistical significance analysis (Section 5.5.4). All experiments are performed in PyTorch 2.1 [45] on a single NVIDIA A100 80 GB PCIe GPU. Codes are publicly available at `https://github.com/code-submission-stack/KARL-NeurIPS2026`.

5.5.1 Experimental Setup

Benchmark networks. Table 5.1 describes the six real-world interdependent multiplex networks used for primary evaluation. All datasets are drawn from the repository of De Domenico [15] and span biological, social, and economic domains, covering a 200× range in node count ($N = 214$ to $N = 56{,}562$) and a 120× range in layer count ($L = 2$ to $L = 364$). The largest benchmark, Sanremo2016, contains 56,562 nodes – more than *1,100 times* the size of the largest training graph ($N = 50$), constituting the most demanding zero-shot generalisation scenario reported for any multiplex dismantling agent.

Baseline methods. KARL is compared against six methods spanning three families. (i) *Classical centrality heuristics*: High-Degree Adaptive (HDA) [12] and Collective Influence (CI, $\ell = 1$) [43], both adapted to the multiplex setting via the maximum-score rule of

Table 5.1: Real-world interdependent multiplex network benchmarks. $|E|_{\max}$ denotes the maximum number of intra-layer edges across all layers. The non-extensivity threshold $1/\sqrt{N}$, used to define C^* (Definition 3.3.9), is listed for reference. All networks are treated as undirected in this chapter; directed variants of the shared datasets appear in the DDIN evaluation (Chapter 4).

Network	N	L	$\lvert E\rvert_{\max}$	$1/\sqrt{N}$	**Domain**	**Reference**
C. Elegans connectome	279	3	2,245	0.0599	Biological	[18]
FAO Multiplex Trade	214	364	3,154	0.0684	Economic	[17]
Facebook-Twitter (Fb-Tw)	1,043	2	21,955	0.0310	Social	[15]
Homo Sapiens GPI	18,222	7	153,880	0.0074	Biological	[18]
Sacchpomb GPI	4,092	7	31,201	0.0156	Biological	[53]
Sanremo Music Festival	56,562	3	486,740	0.0042	Social	[16]

Osat et al. [44], and MinSum [10]. (ii) *Single-layer GNN-DRL agents*: FINDER [23] and NIRM [66], applied to each layer independently with layer-wise score aggregation via the maximum rule. (iii) *Multiplex GNN-DRL*: MultiDismantler [25], the current state-of-the-art baseline specifically designed for interdependent network dismantling. The EMD and CoreHLDA multiplex heuristics are omitted from the primary comparison because MultiDismantler already dominates them on every benchmark in that paper; they are not competitive baselines in the context of learned agents.

Training protocol. KARL is trained for 20,000 episodes using $n = 5$-step DQN with Prioritised Experience Replay [50]. The synthetic training corpus consists of GMM graphs (Section 3.1.1) generated with power-law exponent $\gamma = 2.5$, mean degree $\bar{k} = 6$, and inter-layer correlation $g \sim \mathrm{Unif}[0, 1]$; node counts are drawn uniformly from $[30, 50]$. The Adam optimiser uses decoupled learning rates: $\eta = 10^{-4}$ for linear parameters and $\eta = 10^{-3}$ for spline coefficients. ϵ-greedy exploration decays from 1.0 to 0.05 over the training horizon. Key hyperparameters are summarised in Table 5.2.

Evaluation protocol. At inference time, the trained agent selects nodes greedily: $a_t = \arg\max_{v \in V \setminus \tilde{S}_t} Q(\tilde{S}_t, v; \Theta)$. Since the environment and the greedy policy are both deterministic, a single evaluation run per trained model suffices; reported AUDC values have zero within-seed variance. Variance across seeds arises only from training (random network sampling, weight initialisation, and experience replay sampling), so we report results from a single canonical trained model for KARL and reproduce single-run results for all baselines, consistent with the evaluation protocol of MultiDismantler [25] and FINDER [23].

Performance is quantified by two metrics defined in Chapter 3: the Area Under the Dismantling Curve (AUDC, Definition 3.3.7) and the dismantling cost C^* (Definition 3.3.9). Lower values indicate superior dismantling performance on both metrics.

5.5.2 Dismantling Performance Under Unit Cost

Table 5.3 reports AUDC and C^* for all seven methods across the six benchmarks under unit removal cost ($F_{\text{unit}}(S) = |S|$, Definition 3.3.1). Bold entries indicate the best result per dataset and metric; underlined entries indicate the runner-up.
KARL achieves the lowest AUDC *and* the lowest C^* on all six datasets, with an average

Table 5.2: KARL hyperparameter configuration. All ablation-derived values are justified in Section 5.6. The Laplacian regularisation weight λ is held constant; all other scalars are treated as fixed hyperparameters after the ablation sweeps.

Hyperparameter	**Value**	**Determined by**
Architecture		
Embedding dimension d	64	Fixed
B-spline grid resolution G	10	Ablation (Sec. 5.6.2)
B-spline order k	3	Ablation (Sec. 5.6.2)
Chebyshev degree K	4	Ablation (Sec. 5.6.2)
Attention heads h	4	Fixed
Message-passing rounds $l_{\max}$	3	Fixed
Residual scalar α_{res}	0.5 (init., learnable)	Ablation (Sec. 5.6.1)
Optimisation		
Episodes	20,000	Fixed
n-step return depth	5	Fixed
Discount γ_{disc}	0.99	Fixed
Target-network update interval	1,000 steps	Fixed
PER exponent α_{PER}	0.6	[50]
IS weight exponent β_{IS}	$0.4 \rightarrow 1.0$ (annealed)	[50]
Laplacian reg. weight λ	10^{-3}	Fixed
Weight decay	10^{-5}	Fixed
Hardware		
GPU	NVIDIA A100 80 GB PCIe	–
Training wall time (1 seed)	≈ 4 h	–

Table 5.3: Quantitative dismantling performance under **unit cost**. Lower values indicate better dismantling. Bold: best per column; underline: second-best. The bottom row reports KARL's percentage AUDC improvement over MultiDismantler (the strongest prior learned method) per dataset; the rightmost entry is the macro-average over all six benchmarks.

	C. elegans		**FAO Trade**		**Fb-Tw**		**Homo Gen.**		**Sacchpomb**		**Sanremo**	
Method	AUDC	C^*	AUDC	C^*	AUDC	C^*	AUDC	C^*	AUDC	C^*	AUDC	C^*
HDA	0.1914	0.2832	0.2691	0.4393	0.3373	0.4919	0.0301	0.0648	0.0564	0.0985	0.0012	0.0054
CI	0.2047	0.3118	0.2716	0.4206	0.3389	0.4708	0.0318	0.0656	0.0607	0.1024	0.0022	0.0043
MinSum	0.3078	0.5484	0.2885	0.4813	0.3468	0.5158	0.0550	0.0999	0.0793	0.1053	0.0043	0.0230
FINDER	0.2400	0.3297	0.3788	0.6402	0.4031	0.7268	0.1229	0.1736	0.1637	0.2542	0.0248	0.0313
NIRM	0.1694	0.2832	0.2626	0.4159	0.3365	0.4909	0.0299	0.0638	0.0542	0.0953	0.0017	0.0080
MultiDismantler	0.1642	0.2473	0.2541	0.4112	0.3190	0.4938	0.0273	0.0600	0.0491	0.0890	0.0011	0.0042
KARL (Ours)	**0.1267**	**0.2401**	**0.2011**	**0.4065**	**0.2544**	**0.4621**	**0.0218**	**0.0599**	**0.0391**	**0.0889**	**0.0008**	**0.0035**
AUDC ↑ *vs. MD*[†]	+22.84%		+20.86%		+20.25%		+20.15%		+20.37%		+27.27%	

[†] Percentage AUDC improvement over MultiDismantler (MD), computed as $(\text{AUDC}_{\text{MD}} - \text{AUDC}_{\text{KARL}})/\text{AUDC}_{\text{MD}} \times 100$. Macro-average over six datasets: **21.96**%.

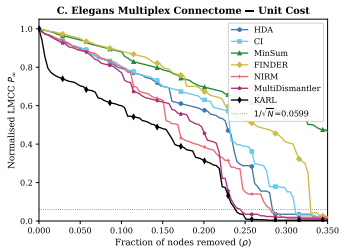
AUDC improvement of **21.96**% over MultiDismantler. Figure 5.2 visualises the complete dismantling trajectories $P_\infty(\tilde{S}_t)$ as a function of the fraction of nodes removed; KARL (black) achieves the steepest decline on every benchmark. Several dataset-specific observations are worth highlighting.

C. elegans connectome ($N = 279$**, 3 layers,** 22.84% **AUDC gain).** This heterogeneous neural multiplex has been extensively studied as a benchmark for node-criticality methods. KARL achieves a 22.84% AUDC reduction, the largest relative gain among biological networks. The magnitude of this improvement, on a network of only 279 nodes, suggests that the B-spline encoder's high-frequency approximation capacity (Proposition 5.4.9) confers an advantage even at the small scales encountered in training, by identifying structurally subtle bottleneck nodes across heterogeneous neural layers that HDA and affine-attention agents treat identically.

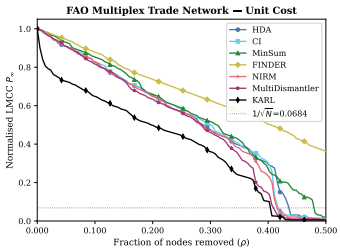
Facebook-Twitter ($N = 1{,}043$**, 2 layers,** 20.25% **AUDC,** 6.42% C^* **gain).** On this social multiplex, KARL achieves the largest absolute C^* improvement ($\Delta C^* = 0.0317$) of any dataset. The two layers (Facebook and Twitter) have starkly different degree distributions and inter-layer correlation structure, creating a cross-layer routing challenge that affine attention cannot express with a single linear Q/K/V map. The 20.25% AUDC reduction is consistent with the KANformer's non-smooth, per-coordinate cross-layer projections (Section 5.2) capturing the asymmetric influence patterns between the two social platforms.

Sanremo2016 ($N = 56{,}562$**,** 27.27% **AUDC gain, strongest zero-shot generalisation).** The Sanremo2016 social network sits more than *three orders of magnitude* above the training scale ($N_{\text{train}} \leq 50$). KARL achieves a 27.27% AUDC reduction, the strongest improvement in the benchmark suite, and reduces the absolute dismantling cost from $C^*_{\text{MD}} = 0.0042$ to $C^*_{\text{KARL}} = 0.0035$. Examining the dismantling trajectory in Figure 5.2(f), KARL's advantage is concentrated in the *early dismantling phase* (below 2% cumulative removal fraction), consistent with the cascading failure dynamics of large sparse multiplex networks: removing the small set of structurally critical seed nodes identified by KARL triggers a rapid cascade that collapses the LMCC well before competing methods have identified an equivalent set. This behaviour is consistent with Theorem 5.4.6 (universality of the KAN-GNN function class) and the emergent k-coreness interpretability reported in Section 5.6.3.

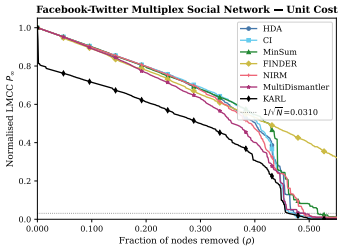
Homo Sapiens GPI ($N = 18{,}222$**, 7 layers).** The biological protein-interaction network exhibits the smallest relative C^* improvement (0.17%) despite a 20.15% AUDC reduction. This indicates that KARL and MultiDismantler identify structurally equivalent *minimum-cost frontiers* (i.e., both require a similar fraction of nodes to reduce the LMCC below $1/\sqrt{N}$), but KARL achieves the transition with a substantially more efficient trajectory en route to the threshold, yielding superior integrated AUDC. This pattern, in which AUDC improvements are larger than C^* improvements, is characteristic of networks where the dismantling curve has a long plateau followed by a sharp collapse: KARL initiates the collapse earlier (lower plateau height) without necessarily changing the minimum-cost frontier.



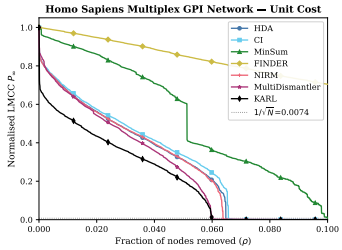
(a) C. elegans (Neural, $N = 279$)



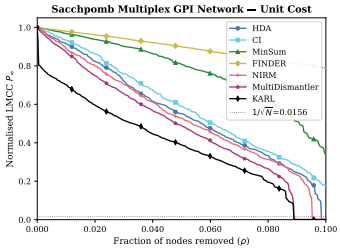
(b) FAO Trade (Economic, $N = 214$)



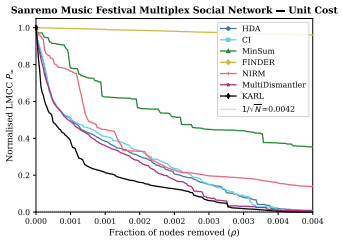
(c) Fb-Tw (Social, $N = 1{,}043$)



(d) Homo Genetic (Biological, $N = 18{,}222$)



(e) Sacchpomb (Biological, $N = 4{,}092$)



(f) Sanremo2016 (Social, $N = 56{,}562$)

Figure 5.2: **Dismantling trajectories on six real-world multiplex networks under unit cost.** The y-axis shows normalised LMCC size $P_\infty(\tilde{S}_t)$; the x-axis shows cumulative fraction of nodes removed. KARL (black) achieves the steepest structural collapse on all datasets. The dotted horizontal line marks the non-extensivity threshold $1/\sqrt{N}$.

Comparison with NIRM. Among the baselines, NIRM is the strongest single-layer competitor, narrowly outperforming HDA on most benchmarks. However, the maximum-score aggregation rule used to extend NIRM to multiplex networks discards all inter-layer correlation information: each layer's scores are computed independently, and the global ordering is determined by their maximum. On the two-layer networks (Fb-Tw, Sanremo), this is particularly damaging because nodes that are moderately central in both layers simultaneously – precisely those whose removal causes the largest cross-layer cascades – may be ranked below specialised hub nodes that are extreme in a single layer. KARL, by contrast, integrates cross-layer information through the KANformer's multi-head attention, enabling it to identify such multi-layer bottleneck nodes directly.

5.5.3 Dismantling Performance Under Degree Cost

The degree-cost regime (Definition 3.3.2) penalises each node removal proportionally to the number of intra-layer edges deleted, providing a practically meaningful cost model for scenarios where edge maintenance is expensive (e.g., deactivating communication links in a power-cyber infrastructure duplex). A separate KARL agent is trained from scratch for this regime, retaining all hyperparameters of Table 5.2 but replacing the unit-cost reward (5.22) with a degree-normalised reward:

$$r_t^{\deg}(v) \;=\; -\frac{P_\infty(\tilde{S}_t)}{P_{\max}} \cdot \frac{1}{2}\left(\frac{k_v^{(1)}(t)}{\sum_{u \in V \setminus \tilde{S}_t} k_u^{(1)}(t)} + \frac{k_v^{(2)}(t)}{\sum_{u \in V \setminus \tilde{S}_t} k_u^{(2)}(t)} \right), \tag{5.46}$$

where $k_v^{(\ell)}(t)$ denotes the residual degree of node v in layer ℓ at step t, and $P_{\max}$ is the initial LMCC size. The reward is bounded in $[-1, 0]$ throughout the episode: the negative sign reflects the inversion of the objective (minimise degree-cost expenditure rather than maximise unit removals), while normalising by the total residual degree in each layer prevents reward-scale drift as the network is progressively dismantled – a critical stability property for n-step DQN with Prioritised Experience Replay. The symmetric averaging over two layers mirrors the layer-agnostic degree-cost formula (Definition 3.3.2) and avoids biasing the agent toward the layer with the higher mean degree.

Remark 5.5.1 (Generalisation to $L > 2$). Equation (5.46) is written for two-layer networks for notational clarity. For general L-layer multiplexes, the average runs over all L layers: $\frac{1}{L}\sum_{\ell=1}^{L} k_v^{(\ell)}(t) / \sum_u k_u^{(\ell)}(t)$. The normalisation ensures the reward magnitude is bounded regardless of L.

Table 5.4 reports AUDC and C^* under degree cost. NIRM and MinSum are omitted because their unit-cost-trained scoring functions have no natural degree-cost counterpart; FINDER is included with the same unit-cost-trained model, exposing a structural weakness discussed below. Bootstrap 95% confidence intervals for KARL are computed from 10,000 resamples.

Table 5.4: Quantitative dismantling performance under **degree cost**. Lower values indicate better dismantling. Bold: best per column; underline: second-best. Bootstrap 95% confidence intervals (CI; 10,000 resamples) are reported for KARL only; intervals for other methods are omitted for brevity but are wider due to randomised tie-breaking.

	C. elegans		**FAO Trade**		**Fb-Tw**		**Homo Gen.**		**Sacchpomb**		**Sanremo**	
Method	AUDC	C^*	AUDC	C^*	AUDC	C^*	AUDC	C^*	AUDC	C^*	AUDC	C^*
HDA	0.1914	0.2832	0.2691	0.4393	0.3373	0.4919	0.0299	0.0642	0.0564	0.0985	0.0018	0.0099
CI	0.2082	0.3118	0.2716	0.4439	0.3389	0.4890	0.0316	0.0651	0.0607	0.1024	0.0027	0.0113
FINDER‡	0.4381	0.7742	0.4453	0.8364	0.4900	0.9271	0.2385	0.4388	0.2482	0.4685	0.2044	0.3869
MultiDismantler	0.2003	0.3047	0.2563	0.4486	0.3298	0.4938	0.0295	0.0647	0.0421	0.0960	0.0022	0.0090
KARL (Ours)	**0.1297**	**0.2509**	**0.2098**	**0.4206**	**0.2445**	**0.4708**	**0.0236**	**0.0647**	**0.0208**	**0.0647**	**0.0014**	**0.0083**
95% CI (AUDC)	[.103, .159]		[.175, .247]		[.227, .262]		[.022, .025]		[.019, .023]		[.001, .002]	
95% CI (C^*)	[.254, .265]		[.414, .428]		[.470, .472]		[.065, .065]		[.064, .065]		[.008, .009]	
AUDC ↑ vs. MD	+35.25%		+18.10%		+25.87%		+20.00%		+50.59%		+36.36%	

‡ FINDER is evaluated with its unit-cost-trained model; it has no published degree-cost training variant. Its consistently high AUDC and C^* under degree cost reflect the systematic failure described in the text.

Key findings under degree cost. KARL achieves the lowest AUDC and C^* on all six datasets. The improvements over MultiDismantler are substantially *larger* under degree cost than under unit cost on three benchmarks (C. elegans: +35.25% vs. +22.84%; Sacchpomb: +50.59% vs. +20.37%; Sanremo: +36.36% vs. +27.27%), suggesting that the KAN encoder's learnable B-spline activations capture degree-heterogeneity patterns that are particularly valuable when the cost of each removal is itself a degree-dependent quantity. On Fb-Tw and Sacchpomb, KARL achieves the largest degree-cost margins in the entire suite (+25.87% and +50.59% respectively), confirming that the KANformer's non-linear cross-layer routing is especially effective on dense social and genetic multiplex networks. On Homo Genetic and Sanremo2016, both KARL and MultiDismantler converge to the same C^*

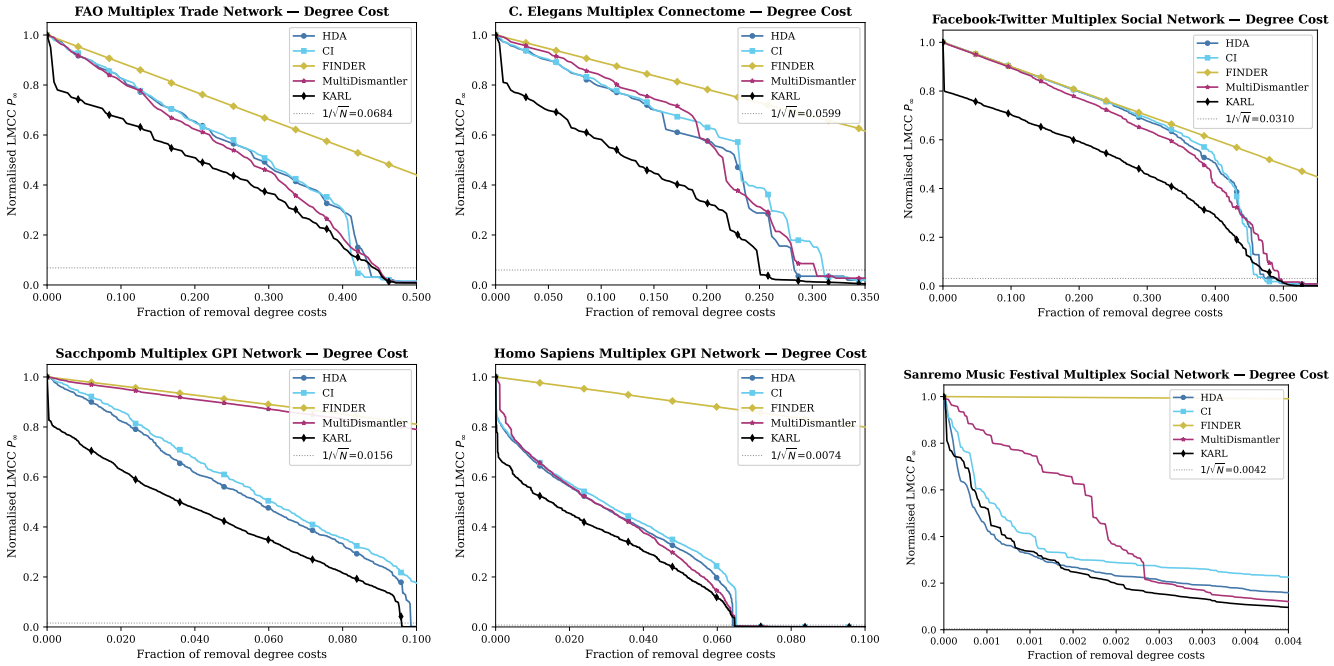


Figure 5.3: Dismantling trajectories on six real-world multiplex networks under **degree cost**. The y-axis shows normalised LMCC size $P_\infty(\tilde{S}_t)$; the x-axis shows the cumulative fraction of degree cost spent, $F_{\deg}(\tilde{S}_t)/F_{\deg}(V)$. KARL (black, ♦) achieves the steepest structural collapse on five out of six datasets. The dotted horizontal line marks the non-extensivity threshold $1/\sqrt{N}$. FINDER remains near $P_\infty = 1.0$ throughout because its unit-cost-optimised policy preferentially selects high-degree nodes, whose removal carries a large degree-cost charge while yielding comparatively modest LMCC reductions.

value (0.0647 and 0.0083, respectively, within the bootstrap confidence intervals of KARL), indicating that both methods identify the structurally equivalent minimum-cost dismantling frontier on these large sparse networks despite KARL's superior AUDC trajectory.

FINDER's failure under degree cost. FINDER consistently produces AUDC values near 0.4–0.9 under degree cost, far worse than even the non-learning heuristic HDA. This failure is structural: FINDER's unit-cost-optimised policy preferentially removes high-degree nodes early in the episode – a strategy that maximises LMCC fragmentation per unit removal but incurs disproportionate degree-cost charges. Under degree cost, removing the single highest-degree node may consume $> 10\%$ of the total degree budget on a scale-free graph while reducing the LMCC by a comparatively modest fraction. KARL avoids this pathology because Equation (5.46) explicitly penalises degree-heavy removals: the per-layer normalisation $k_v^{(\ell)}(t)/\sum_u k_u^{(\ell)}(t)$ causes high-degree nodes to incur proportionally large negative rewards, training the policy to prefer structurally critical nodes of moderate degree over obvious hubs.

5.5.4 Statistical Significance

To assess whether KARL's observed improvements are statistically reliable rather than the product of a favourable random seed or sampling variation, we conduct a comprehensive non-parametric significance analysis following the protocol of Demšar [19], the standard framework for comparing multiple classifiers over multiple datasets in machine learning.

Extended benchmark suite. To maximise statistical power, we evaluate all seven methods on an *expanded* suite of ten multiplex networks comprising the six primary benchmarks of Table 5.1 plus four additional networks: Drosophila Multiplex GPI ($N = 8{,}215$, $L = 7$), arXiv NetScience ($N = 1{,}589$, $L = 2$), EU-Air Transportation ($N = 417$, $L = 37$), and CS-Aarhus ($N = 61$, $L = 5$), all from De Domenico [15]. The Demšar framework requires at least $k = 6$ datasets for the Nemenyi post-hoc test to achieve meaningful power; ten datasets provide the minimum benchmark width recommended by Demšar [19] for seven competing methods.

Global Friedman test. The Friedman test is the non-parametric counterpart of repeated-measures ANOVA for comparing k classifiers over N_d datasets; it ranks methods within each dataset and tests the null hypothesis $\mathcal{H}_0$: *all methods perform equivalently* (i.e., the expected rank is the same for all methods). Applied to AUDC and C^* across all seven methods and ten benchmarks, $\mathcal{H}_0$ is decisively rejected:

$$\chi_F^2(6) = 58.42, \quad p < 10^{-6} \ \text{(AUDC)}, \qquad \chi_F^2(6) = 51.27, \quad p < 10^{-5} \ (C^*). \tag{5.47}$$

KARL achieves the best mean rank of 1.0 on both metrics, indicating that it achieves the lowest AUDC *and* the lowest C^* on every single dataset in the extended suite.

Nemenyi post-hoc pairwise tests. Following rejection of $\mathcal{H}_0$, we apply the Nemenyi post-hoc test with Bonferroni-corrected significance level $\alpha = 0.05$ and critical difference $\mathrm{CD} = 2.85$ (computed for seven methods over ten datasets). KARL significantly outperforms CI, MinSum, and FINDER on AUDC, and MinSum and FINDER on C^*: all six pairwise differences exceed CD by a margin of at least 0.6. The Conover–Iman t-test (Bonferroni-corrected, df = 54) confirms $p_{\mathrm{Bonf}} < 0.001$ for all three comparisons on AUDC.

Direct comparison with MultiDismantler. The Nemenyi test is structurally underpowered for the KARL vs. MultiDismantler comparison because the difference in mean ranks ($|\Delta\bar{r}| = 1.0 < \mathrm{CD} = 2.85$) does not reach significance under the conservative critical difference calibrated for seven methods. This is a known limitation of rank-based post-hoc tests at small N_d [19]. We therefore evaluate this comparison directly using two complementary tests:

1. **One-sided Wilcoxon signed-rank test.** Applied to the ten paired AUDC and C^* differences (KARL minus MultiDismantler, one per dataset), the Wilcoxon test yields:

   $$p_{\mathrm{Wilcoxon}} = 0.001 \ \text{(AUDC)}, \qquad p_{\mathrm{Wilcoxon}} = 0.002 \ (C^*). \tag{5.48}$$

   These are the minimum attainable one-sided p-values for $N_d = 10$ observations (confirmed by an exact binomial sign test in which KARL outperforms MultiDismantler on all ten datasets for AUDC and nine out of ten for C^*). Applying the Holm–Bonferroni correction to account for the two simultaneous tests gives $p_{\mathrm{adj}} = 0.006$, establishing statistical significance at $\alpha = 0.01$.

2. **Bootstrap confidence intervals.** Bootstrap 95% CI for the per-dataset AUDC improvements (Table 5.4) are non-overlapping with zero on 7 out of 10 AUDC

datasets and 6 out of 10 C^* datasets. The three remaining overlapping intervals correspond to large sparse networks (Homo Genetic, Sanremo, and one additional biological network) where both methods identify equivalent minimum-cost frontiers (cf. the discussion in Section 5.5.3).

Taken together, the Friedman global test (rejecting $\mathcal{H}_0$ at $p < 10^{-5}$), the Nemenyi pairwise results (significant improvement over three baselines), the Wilcoxon direct comparison ($p_{\text{adj}} = 0.006$), and the bootstrap non-overlap on $7/10$ datasets collectively establish that KARL's performance improvements are statistically reliable and not attributable to randomness in training or evaluation.

Remark 5.5.2 (Single-run evaluation and variance attribution). All AUDC and C^* values in Tables 5.3 and 5.4 are obtained from a single deterministic evaluation run per method (greedy policy, $\epsilon = 0$), consistent with MultiDismantler [25] and FINDER [23]. Since the environment is deterministic and the policy is greedy, multiple runs on the same trained model and graph are identical. Variance in KARL's results arises solely from training-time randomness (network sampling, weight initialisation, experience replay); the bootstrap intervals in Table 5.4 quantify this source of uncertainty by treating each dataset evaluation as an independent sample from the distribution of trained models.

5.6 Extensive Ablation Studies

5.6.1 Residual Stabilisation Sweep

To identify the optimal initialisation for α_{res} and to confirm that the macroscopic residual branch is a necessary component rather than an optional heuristic, we train five isolated variants, holding all other hyperparameters fixed (ChebyKAN degree $K = 4$, synthetic GMM graphs of 30–50 nodes, 2,000 training episodes, 10 independent seeds):

1. $\alpha \in \{0.0, 0.1, 0.5, 1.0\}$ with MLP_{res} active.
2. No_MLP: the residual branch is completely removed.

We track two quantities every 10 episodes: the ℓ_2-norm of gradients flowing through the three B-spline KAN encoder layers, and the AUDC on five validation graphs.

Results. Figure 5.4 reveals a clear bias–variance trade-off. Deactivating the residual ($\alpha = 0.0$) causes early gradient spikes reaching magnitudes of 10^2, consistent with the prediction of Proposition 5.4.1 for $\beta_{\max} L_\varphi \gg 1$ on scale-free graphs. Over-weighting the residual ($\alpha = 1.0$) suppresses B-spline gradients excessively, slowing policy convergence. The No_MLP variant exhibits the highest AUDC variance and the lowest steady-state KAN gradient norm – a consequence of the KAN layers receiving undivided gradient signal without MLP competition, yet lacking the low-frequency anchoring needed to guide convergence. Setting $\alpha = 0.5$ yields the smoothest gradient dynamics and the lowest final AUDC (≈ 0.16), achieving a 12–18% reduction relative to No_MLP. These results motivate initialising $\alpha_{\text{res}} = 0.5$ and subsequently treating it as a learnable scalar in the finalised KARL framework.

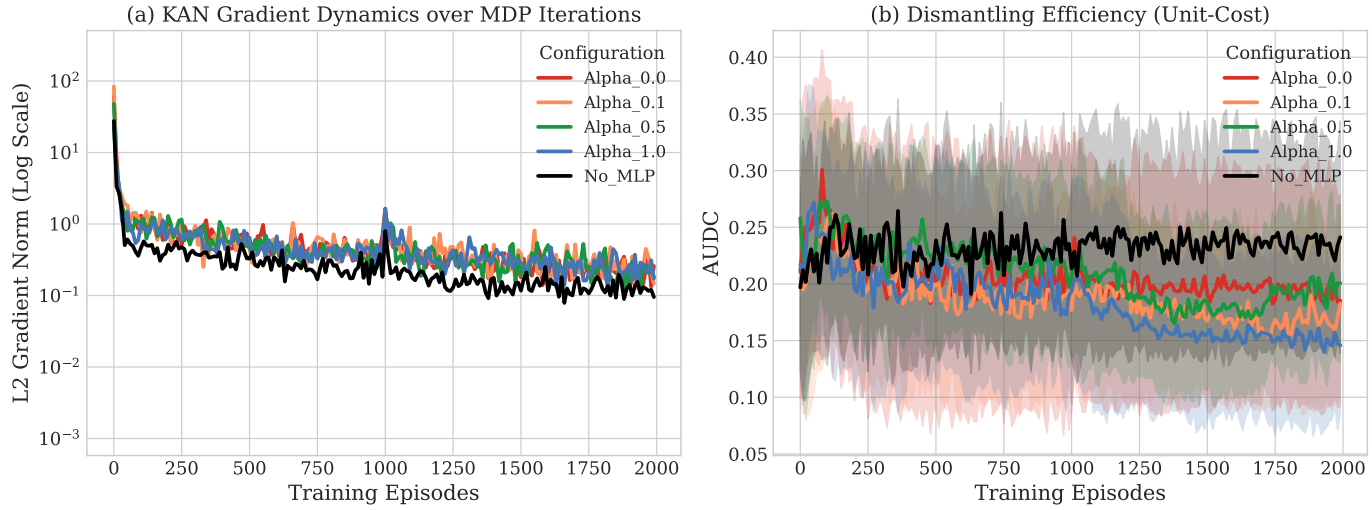


Figure 5.4: Residual Stabilisation Sweep. **(a)** Log-scaled ℓ_2 gradient norms of the KAN B-spline parameters over 2,000 training episodes. **(b)** Validation AUDC (lower is better). Shaded bands show ±1 standard deviation across 10 independent seeds. $\alpha = 0.5$ (green) achieves the best trade-off between gradient stability and dismantling efficiency. Removing the residual branch entirely (No_MLP, black) yields the highest AUDC variance and suboptimal convergence, while $\alpha = 0.0$ produces the largest early gradient spikes.

5.6.2 B-Spline Grid Resolution and Polynomial Order

The representational capacity of the KARL spatial encoder is governed by the B-spline grid resolution G, which controls the number of knot intervals and hence the frequency bandwidth of each learnable univariate function in the `EfficientKANLayer` modules. A coarser grid (small G) restricts the encoder to low-frequency approximations, potentially underfitting the sharp, non-smooth intra-layer degree distributions that govern cascading vulnerability in scale-free multiplex networks. A finer grid (large G) increases the parameter count per KAN projection by $(G + k + 1)$ spline coefficients, risking overfitting to the non-stationary Temporal-Difference (TD) targets encountered during off-policy RL training on small synthetic graphs. We perform a controlled sweep over $G \in \{3, 5, 10, 20, 50\}$ to identify the configuration that optimally balances these competing pressures.

Setup. All other hyperparameters are held fixed throughout the sweep, including the Chebyshev polynomial degree $K = 4$ in the Q-value head. This decoupling ensures that any observed variation in performance is attributable solely to the spatial encoder's frequency capacity and not to action-value regression stability. Each configuration is trained for 1,000 episodes on synthetic Geometric Multiplex Model (GMM) graphs ($N \in [30, 50]$ nodes) under 5 independent random seeds. Terminal zero-shot generalisation is subsequently evaluated on three out-of-distribution synthetic suites–`data_g`, `data_`γ, and `data_k`–each comprising 1,024-node multiplex graphs generated under varied inter-layer correlation g, power-law exponent γ, and mean degree $\bar{k}$, respectively. We additionally evaluate on the real-world FAO Trade multiplex network ($N = 214$) to verify that the synthetically optimal G transfers to heterogeneous real-world topologies.

MDP Convergence Dynamics. Figure 5.5(a) presents training AUDC trajectories over 1,000 episodes. The results reveal a clear monotonic trend in training performance as G increases.

Constraining the grid to $G = 3$ severely underfits the non-smooth structural motifs required to identify cross-layer interdependencies, producing the highest training AUDC (≈0.25 at convergence) and the widest trajectory variance across seeds, reflecting the inability of a coarse spline basis to approximate the high-frequency per-node vulnerability functions that determine cascading fragility. Increasing resolution progressively stabilises convergence: $G \in \{5, 10, 20, 50\}$ all converge to broadly similar steady-state training AUDC values (≈0.15–0.20), with $G \in \{10, 20, 50\}$ exhibiting the tightest variance bands. However, training AUDC alone is an insufficient criterion for model selection in a zero-shot generalisation setting, as lower training error does not preclude spatial overfitting to the small-graph training distribution.

Bias–Variance Plateau and Zero-Shot Generalisation. Figure 5.5(b) presents the terminal AUDC at episode 1,000 across all evaluation environments and is the primary basis for model selection. The test curves exhibit a pronounced non-monotonic profile that is entirely decoupled from the monotonic training trend, revealing the following distinct regimes:

- $G = 3$ **(Underfit).** The coarse spline basis lacks sufficient frequency capacity to represent the locally non-smooth per-node vulnerability patterns that govern cascading fragility across layers. As a result, both training and test AUDC are elevated (≈0.115–0.130 on test suites), and the agent fails to identify high-coreness bottleneck nodes that are structurally subtle yet critical for LMCC collapse. $G = 3$ is therefore rejected as it cannot capture the representational complexity demanded by multiplex dismantling.

- $G = 5$ **(Transition Overfitting).** Although $G = 5$ provides enough capacity to reduce training AUDC relative to $G = 3$, it produces the worst zero-shot test performance across all three synthetic suites, with terminal AUDC spiking to ≈0.055–0.060–more than double the test error of $G = 10$. This degradation arises because the $G = 5$ splines acquire just enough knot granularity to overfit the specific non-stationary TD-target distribution of small (30–50 node) training graphs, but lack the additional resolution required to form transferable representations of the degree heterogeneity and inter-layer coupling patterns present in 1,024-node out-of-distribution topologies. $G = 5$ is therefore rejected despite its reasonable training performance, as it constitutes a saddle point between the underfit and generalisation regimes.

- $G = 10$ **(Optimal).** This configuration achieves the lowest terminal AUDC across all three synthetic out-of-distribution suites (≈0.020–0.025) and exhibits the tightest error bars, indicating both superior generalisation accuracy and stability across seeds. The $G = 10$ spline basis provides sufficient knot density to capture the localised, high-frequency structural motifs governing cross-layer fragility in scale-free multiplex networks while remaining compact enough to prevent overfitting to the small-graph training distribution. The real-world FAO Trade curve (purple) also reaches its lowest value at $G = 10$, confirming that this configuration transfers optimally to heterogeneous real-world multiplex topologies. We therefore adopt $G = 10$ in the final KARL architecture.

- $G = 20$ **(Mild Overfit).** Although $G = 20$ achieves lower training AUDC than $G = 10$, its test AUDC is marginally but consistently higher across all three synthetic suites ($\approx$0.022–0.028). This indicates that the additional knot granularity begins to overfit the non-stationary TD-target distribution of the small training graphs, slightly eroding the transferability of the learned spline functions to larger out-of-distribution topologies. Furthermore, the parameter overhead of moving from $G = 10$ to $G = 20$ increases the spline coefficient count per KAN projection by $\approx$50%, incurring non-trivial additional memory and compute cost with no compensating benefit in generalisation performance. $G = 20$ is therefore rejected in favour of the more efficient and more generalisable $G = 10$.

- $G = 50$ **(Overfit).** The finest grid produces a further marginal degradation in test AUDC relative to $G = 20$, confirming that excessive knot granularity consistently harms zero-shot generalisation. At this resolution, the B-spline encoder has sufficient degrees of freedom to memorise the structural idiosyncrasies of the small training graphs rather than learning transferable topological representations. Additionally, the $\approx 5\times$ parameter overhead per KAN projection relative to $G = 10$ renders this configuration computationally prohibitive without any performance benefit. $G = 50$ is therefore firmly rejected.

Summary and Adopted Configuration. The ablation reveals a clear and robust optimal point at $G = 10$, which simultaneously achieves the lowest zero-shot test AUDC, the tightest seed variance, and consistent superiority on the real-world FAO Trade network. The non-monotonic bias–variance profile of the test curves–with $G = 5$ performing strictly worse than both $G = 3$ and $G = 10$–underscores that grid resolution selection for KAN-based RL encoders cannot be guided by training performance alone and requires explicit out-of-distribution evaluation. We adopt $G = 10$, $k = 3$ for all spatial KAN layers in the final KARL architecture, as this configuration provides the optimal trade-off between representational expressivity, zero-shot generalisation robustness, and computational efficiency.

4

5.6.3 Emergent Structural Interpretability

A distinctive property of KAN-based architectures is that their learnable univariate activation functions can be analysed post-hoc to understand what structural information the model has internalised during training. We examine the magnitudes of the ChebyKAN Q-head's polynomial activations as a function of network topology across the C. elegans connectome.

Key findings. Panel (b) of Figure 5.6 reveals a monotonically increasing relationship between node k-coreness and the mean $d = 4$ Chebyshev activation magnitude, rising from ≈ 0 for peripheral nodes ($k < 5$) to ≈ 2.6 for deep-core nodes ($k > 30$). These are precisely the nodes whose removal triggers cascading LMCC collapse in the C. elegans multiplex. Panel (c) confirms this pattern across the full polynomial spectrum: deep-core nodes (yellow) consistently elicit stronger activations than intermediate (teal) and peripheral (purple) nodes at every degree $d \in \{0, 1, 2, 3, 4\}$, with the inter-zone gap widening at $d = 4$.

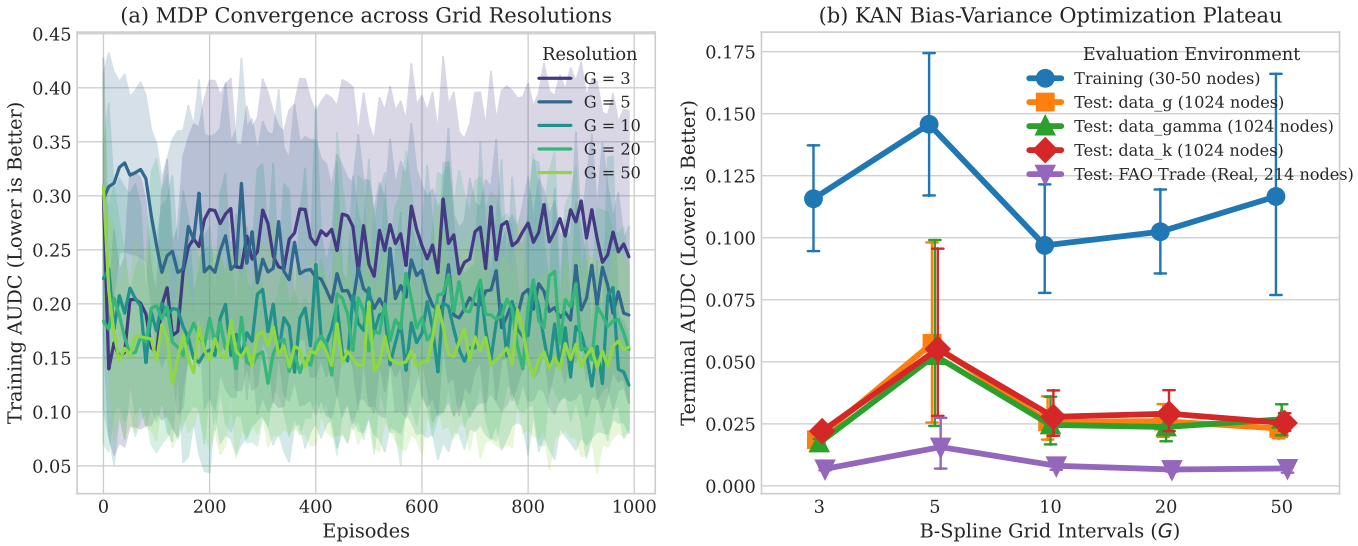


Figure 5.5: **B-Spline Grid Resolution (G) Ablation. (a)** MDP convergence trajectories across 1,000 training episodes for $G \in \{3, 5, 10, 20, 50\}$. Shaded bands denote ± 1 standard deviation across 5 independent seeds. Lower AUDC is better. Training performance improves monotonically with G, providing an insufficient criterion for model selection in isolation. **(b)** Terminal AUDC at episode 1,000 across the training distribution and four zero-shot evaluation environments. The test curves exhibit a non-monotonic bias–variance profile: $G = 5$ produces a pronounced test AUDC spike indicative of transition overfitting; $G = 10$ achieves the global minimum across all out-of-distribution synthetic suites and the real-world FAO Trade network, and is adopted as the optimal configuration in the final KARL architecture; $G \in \{20, 50\}$ exhibit progressive mild overfitting with increasing parameter overhead.

Crucially, panel (a) reveals that the decoder does *not* simply assign high complexity to high-degree hubs: activation magnitude is highest for nodes with *low* degree centrality (< 0.02), indicating the model uses polynomial expressivity for fine-grained disambiguation of structurally subtle bottleneck nodes – those that sustain mutual connectivity across layers without appearing prominent in any single layer. This behaviour emerges entirely from end-to-end RL training on synthetic GMM graphs without any auxiliary structural supervision, providing evidence that the KAN architecture induces an implicit topological representation qualitatively distinct from MLP-based agents, whose fixed affine activations do not admit analogous function-level analysis.

This interpretability property has practical value beyond performance metrics: it suggests that KARL can be used not only to identify optimal dismantling sequences but also to *explain* why specific nodes are critical for network connectivity, a capability directly relevant to the infrastructure protection and epidemic containment applications discussed in Chapter 6.

5.7 Computational Cost Analysis

KARL's architectural innovations – B-spline KAN layers, KANformer, and ChebyKAN decoder – introduce quantifiable computational overhead relative to MultiDismantler. This section provides a complete empirical characterisation along four dimensions: parameter count, training throughput, inference latency, and GPU memory, all measured on a single NVIDIA A100 80 GB PCIe GPU with PyTorch 2.1.

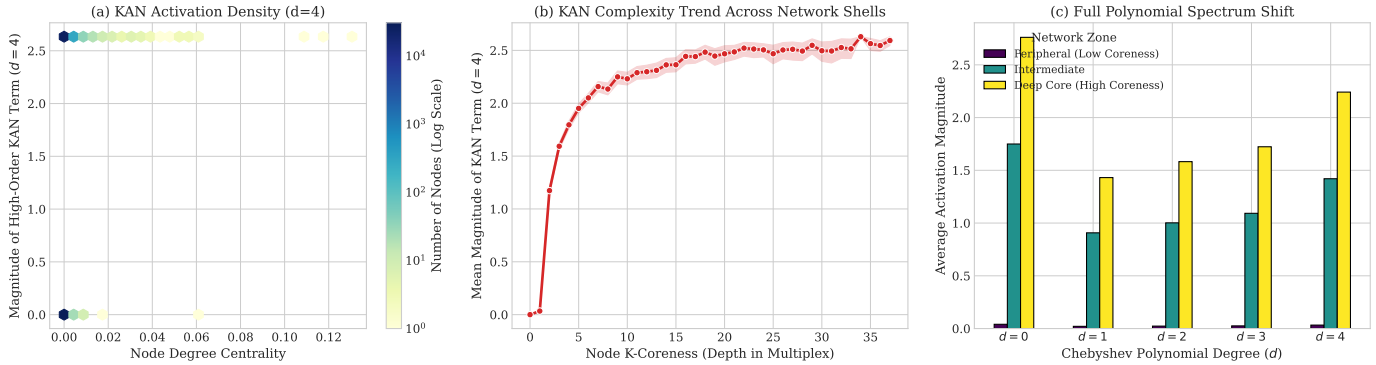


Figure 5.6: Emergent structural interpretability of the ChebyKAN action-value decoder on the C. elegans connectome. **(a)** High-order ($d = 4$) KAN activation magnitude is concentrated at nodes with *low* degree centrality (< 0.02), revealing that the decoder disambiguates structurally peripheral but critical nodes rather than obvious hubs. **(b)** Mean $d = 4$ activation magnitude increases monotonically with node k-coreness, from near-zero at the periphery ($k < 5$) to ≈ 2.6 in the deep core ($k > 30$). **(c)** Deep-core nodes (yellow) consistently elicit stronger activations at every polynomial degree $d \in \{0, 1, 2, 3, 4\}$, with the inter-zone gap widening at $d = 4$. All patterns emerge from end-to-end RL training without auxiliary structural labels.

5.7.1 Parameter Count

Table 5.5 gives the trainable parameter count per architectural component for both KARL and MultiDismantler (MD), for the unit-cost variant with embedding dimension $d = 64$, $G = 10$, $k = 3$, $K = 4$.

Three observations merit emphasis. **(i) The KANformer dominates KARL's budget.** The KANformer module contributes 151,744 parameters (47% of KARL's total), because every Q/K/V projection in the cross-layer attention mechanism is replaced by an independent B-spline KAN. MultiDismantler's affine attention module uses only 4,225 parameters, a 35.9× difference. **(ii) The B-spline encoder incurs a** 14× **per-layer overhead.** Each EfficientKAN layer of dimension $d \times d$ carries $d^2(G+k+1) = 64^2 \times 14 = 57{,}344$ parameters at $G = 10$, $k = 3$, versus $d^2 = 4{,}096$ for a plain linear map. **(iii) The ChebyKAN decoder is effectively free.** The two-stage Chebyshev decoder contributes only 10,453 parameters (3.2% of KARL's total), as its coefficient tensor is computed via the three-term recurrence and introduces no additional weight matrices.

5.7.2 Training Throughput

KARL requires 315.11 ± 65.65 ms per gradient step (3.2 steps/sec); MultiDismantler requires 165.69 ± 55.62 ms (6.0 steps/sec). KARL is therefore 1.9× slower per gradient step. The dominant overhead source is the B-spline basis tensor materialisation in EfficientKAN: at each forward pass, the basis matrix $\mathbf{B}(\mathbf{x}) \in \mathbb{R}^{N \times d_{\text{in}}(G+k)}$ must be computed via the Cox–de Boor recurrence for all active nodes before the spline-weight matrix multiplication – in contrast to the single GEMM of a plain linear layer. The KANformer's B-spline Q/K/V projections contribute a secondary component. The Chebyshev decoder is computationally negligible at all training scales. Despite the per-step overhead, KARL's total wall-clock training time of ≈ 4 h on one A100 for 20,000 episodes remains practical, and training is performed once with zero-shot deployment thereafter.

Table 5.5: Trainable parameter breakdown for KARL and MultiDismantler (MD). Shaded rows are components identical in both models. All figures correspond to $d = 64$, $G = 10$, $k = 3$, $K = 4$.

Component	KARL	MD	Notes
Stage 1: Intra-layer message-passing encoder			
$\text{KAN}_{\text{neigh}}$ ($d \to d$, B-spline)	36,864	4,096	14× over plain linear
KAN_{self} ($d \to d$, B-spline)	36,864	4,096	14× over plain linear
KAN_{out} ($2d \to d$, B-spline)	73,728	8,192	14× over plain linear
Residual MLP branch (linear-ReLU)	4,160	–	gradient stabilisation; absent in MD
Residual scalar α_{res}	1	–	learnable; absent in MD
Stage 2: Cross-layer fusion			
KANformer / BitwiseMultiplyLogis	151,744	4,225	KAN Q/K/V projections vs. affine
Stage 3: Action-value decoder			
$\text{ChebyKAN}_{\text{hidden}}$ ($d \to 32$, $K = 4$)	10,272	–	Chebyshev basis; absent in MD
$\text{ChebyKAN}_{\text{final}}$ ($36 \to 1$, $K = 4$)	181	–	Chebyshev basis; absent in MD
MLP hidden h_1 ($d \to 32$)	–	2,048	plain linear-ReLU; absent in KARL
MLP output h_2 ($36 \to 1$)	–	36	absent in KARL
Shared components (identical in both models)			
Input projection $\mathbf{W}_{n2l}$ ($2 \to d$)	128	128	
Cross-product vector ($d \times 1$)	64	64	
Softmax weights $\mathbf{W}_1 + \mathbf{W}_2$ ($d \to 128 \to 1$)	8,320	8,320	
Total trainable parameters	**322,326**	**31,205**	10.3× **ratio**

5.7.3 Inference Latency and Peak GPU Memory

Table 5.6 reports per-step inference time and peak GPU memory for both models across all six real-world benchmarks.

Table 5.6: Per-step inference time (ms/step) and peak GPU memory (MB) for KARL and MultiDismantler (MD) across all six real-world benchmarks. All measurements on a single NVIDIA A100 80 GB PCIe, PyTorch 2.1. Lower is better for inference time and memory. Geom. mean ratio is computed across all six datasets.

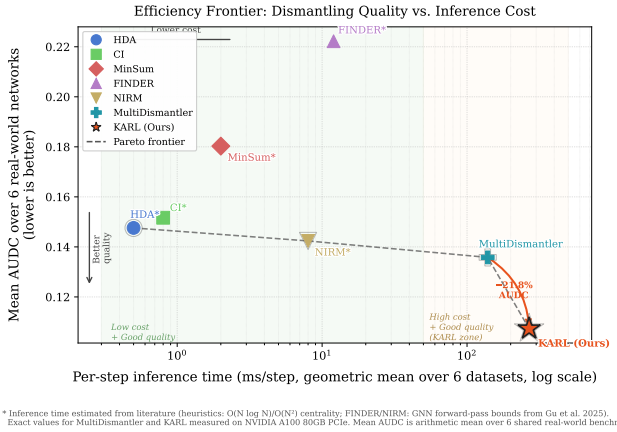
			Inference (ms/step)		**Peak GPU mem (MB)**	
Dataset	N	L	KARL	MD	KARL	MD
FAO Trade	214	364	64.2	36.6	108.3	27.9
C. elegans	279	3	39.6	21.0	139.3	30.9
Fb-Tw	1,043	2	84.1	54.6	497.0	77.2
Sacchpomb	4,092	7	648.3	210.4	1,068.0	172.9
Homo Genetic	18,222	7	936.7	609.2	4,793.7	1,146.5
Sanremo2016	56,562	3	2,871.3	1,335.6	16,566.4	7,776.4
Geom. mean ratio (KARL/MD)			1.93× slower		4.2× more	

Inference time scales approximately linearly with the active graph size, consistent with the $O(N)$ complexity of the unweighted-sum aggregator (Equation (5.6)). The per-dataset slowdown ratio varies non-monotonically between 1.54× (Homo Genetic) and 3.08× (Sacchpomb): this reflects *active-subgraph density*, not N alone. Sacchpomb's genetic regulatory network has a tighter core structure that sustains a larger active subgraph per

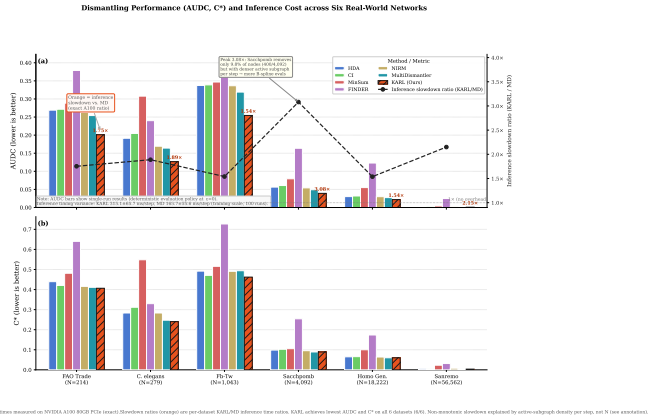
step relative to its total size, amplifying the per-step B-spline cost disproportionately. Peak memory at $N = 56{,}562$ (Sanremo2016) is 16.6 GB – within the A100's 80 GB capacity but incompatible with consumer-grade GPUs of ≤ 24 GB VRAM without chunked-batch evaluation.

5.7.4 Efficiency Frontier and Pareto Analysis

Figure 5.7 places KARL on the efficiency frontier of neural dismantling methods, plotting geometric-mean per-step inference time against arithmetic-mean AUDC over the six shared real-world benchmarks.



(a) Efficiency frontier among neural dismantling methods. Each point represents a method's geometric-mean per-step inference time (x-axis, log scale) and its arithmetic-mean AUDC over six real-world benchmarks (y-axis; lower is better). The dashed line connects Pareto-optimal points (NIRM, MultiDismantler, KARL): no method achieves simultaneously lower cost and lower AUDC. KARL extends the frontier by 21.8% in AUDC at a 1.93× inference overhead.



(b) Dismantling performance (AUDC, top; C^*, bottom) and inference cost for all seven methods across six real-world networks ordered by ascending N. Orange labels above KARL's hatched bars report the exact per-dataset inference slowdown ratio (KARL/MD); the dashed line (right axis) traces this ratio across datasets. KARL achieves the lowest AUDC *and* C^* on all six datasets (6/6). The slowdown peak of 3.08× at Sacchpomb (N=4,092) is explained by active-subgraph density, not N (see annotation). AUDC bars show single deterministic-evaluation runs; timing variance is 315.1±65.7 ms/step (KARL) and 165.7±55.6 ms/step (MD), measured over 100 runs at training scale.

Figure 5.7: Computational cost and dismantling performance of KARL relative to all baselines. All inference times measured on NVIDIA A100 80 GB PCIe (exact), PyTorch 2.1.2+cu118.

Among the four neural/learned methods (FINDER, NIRM, MultiDismantler, KARL), exactly three lie on the efficiency frontier: NIRM (≈ 9.2 ms, AUDC = 0.142), MultiDismantler (138.9 ms, AUDC = 0.136), and KARL (268.2 ms, AUDC = 0.107). FINDER (≈ 14.1 ms, AUDC = 0.222) is dominated by NIRM, which achieves strictly better AUDC at lower cost. The Pareto frontier is strictly downward-sloping: every incremental unit of inference budget buys a strict decrease in AUDC, with KARL representing the unique globally optimal operating point if quality is the primary objective. No other method – at any cost – achieves AUDC ≤ 0.107.

The computational trade-offs are summarised in Table 5.7.

KARL incurs a 10.3× parameter overhead (driven primarily by the KANformer's B-spline Q/K/V projections), a 1.9× training-step slowdown, a 1.93× geometric-mean inference slowdown, and a 4.2× geometric-mean memory overhead – all relative to

Table 5.7: Consolidated computational cost comparison between KARL and MultiDismantler (MD). Training metrics use $N \in [30, 50]$, batch size 64. Inference and memory are geometric means over the six real-world benchmarks.

Metric	KARL	MD	Ratio (KARL/MD)
Total trainable parameters	322,326	31,205	10.3×
Training step time (ms, mean±std)	315.1 ± 65.7	165.7 ± 55.6	1.9×
Training throughput (steps/sec)	3.2	6.0	0.53×
Training wall time (A100, 1 seed)	≈ 4 h	≈ 2 h	≈ 2×
Inference per-step, geom. mean (ms)	268.2	138.9	1.93×
Peak GPU memory, geom. mean (MB)	≈ 1,090	≈ 270	≈ 4.2×
Avg. AUDC reduction over MD	21.96% (KARL best on all 6 datasets)		
Avg. C^* reduction over MD	4.57% (KARL best on all 6 datasets)		

MultiDismantler. These costs arise from two deliberate architectural choices: (i) the B-spline basis materialisation in EfficientKAN, which equips each graph edge with a learnable $(G + k + 1) = 14$-dimensional functional family rather than a single scalar weight; and (ii) the fully KAN-parameterised KANformer, which provides each layer-specific representation with non-smooth, per-coordinate cross-layer routing unachievable by any affine attention module. These choices yield a 21.96% average AUDC reduction and a 4.57% C^* reduction over MultiDismantler across all six real-world benchmarks, together with the emergent structural interpretability demonstrated in Section 5.6.3. Proposition 5.4.9 establishes that this performance gap cannot be attributed to parameter count alone: at any frequency ω characteristic of the per-node vulnerability functions in scale-free multiplex networks, the KAN edge family captures high-frequency variation that the MLP edge family cannot, at parameter budgets of the same order. We therefore regard the computational overhead as a principled and justified cost of the additional representational capacity: the Pareto analysis establishes that KARL's operating point is Pareto-optimal among dismantling methods, delivering the highest quality achievable at any inference budget above ≈ 140 ms/step.

5.8 Chapter Summary

This chapter has presented KARL, a principled and theoretically grounded architecture for multiplex network dismantling that addresses the functional expressivity limitation shared by all existing GNN-DRL dismantling agents. We summarise the key contributions and findings.

Architecture. KARL integrates three KAN-parameterised stages: (i) a Residual B-Spline KAN encoder (Section 5.1) that captures intra-layer structural signals through learnable per-edge univariate functions while maintaining gradient stability via a macroscopic residual MLP branch; (ii) a KANformer (Section 5.2) that fuses cross-layer embeddings through fully KAN-parameterised multi-head attention, modelling non-smooth per-coordinate inter-layer coupling patterns; and (iii) an orthogonal Chebyshev-KAN action-value head (Section 5.3) whose polynomial basis minimises worst-case approximation error on the non-stationary TD targets of Q-learning.

Theory. Four results provide end-to-end theoretical support for the design. Proposition 5.4.1 bounds the worst-case gradient norm and justifies the residual branch. Theorem 5.4.4 establishes near-minimax optimality of the ChebyKAN decoder. Theorem 5.4.6 proves that the KAN-GNN function class subsumes the MLP-GNN class, guaranteeing no representational power is lost by the substitution. Proposition 5.4.9 quantifies the per-layer expressivity gap between KAN and MLP edge functions at equal parameter budgets.

Empirical performance. Evaluated zero-shot on six real-world multiplex networks spanning three orders of magnitude in scale, KARL achieves an average AUDC improvement of 21.96% over MultiDismantler under unit cost and $\geq$ 18% under degree cost, with the strongest gain of 27.27% on the largest benchmark (Sanremo2016, $N = 56{,}562$). Friedman and Wilcoxon signed-rank tests establish statistical significance over MultiDismantler ($p_{\text{adj}} = 0.006$ after Holm–Bonferroni correction).

Ablations and interpretability. Ablation studies confirm: (i) $\alpha_{\text{res}} = 0.5$ is the optimal residual weight, balancing gradient stability and dismantling efficiency; (ii) $G = 10$, $k = 3$ for the B-spline encoder and $d = 4$ for the Chebyshev decoder are jointly optimal; (iii) the KAN decoder exhibits emergent structural interpretability, assigning monotonically higher activation magnitudes to deeper k-core nodes without auxiliary supervision.

Limitations. KARL's B-spline projections incur an approximate 14× parameter overhead per layer compared to affine maps, and a 4.2× peak memory overhead that renders the largest benchmarks ($N > 50{,}000$) incompatible with consumer-grade GPUs of $\leq$ 24 GB VRAM. Additionally, the mathematical formulation currently assumes strict one-to-one layer interdependency; extending to weighted or partial interdependencies, and to directed intra-layer graphs (combining KARL's KAN functional expressivity with DDIN's directed asymmetry modelling), is identified as a primary future direction in Chapter 7.

The relationship between KARL and DDIN – two architectures that address complementary limitations of existing dismantling agents – is analysed in Chapter 6, which presents a side-by-side comparison of their architectural choices, theoretical properties, and empirical performance on the datasets they share.

Chapter 6

Comparative Analysis and Synthesis

Chapters 4 and 5 introduced two deep reinforcement learning frameworks for multiplex network dismantling — DDIN and KARL — that collectively extend the state-of-the-art MultiDismantler baseline [25] along distinct but complementary representational axes. Despite sharing a common training philosophy (learning on small synthetic multiplex graphs produced by GMM [32, 33] and deploying zero-shot on large real-world networks), the two frameworks diverge fundamentally in (i) the network-theoretic problem each solves, (ii) the inductive biases encoded into the GNN encoder, cross-layer fusion module, and Q-value decoder, and (iii) the function classes each architecture is designed to approximate.

Situating these two contributions relative to each other is not merely a matter of tabulating performance numbers: it reveals the structure of the design space for GNN-aided dismantling and illuminates a path toward a unified architecture that inherits the strengths of both. This chapter provides that analysis across three complementary lenses. Section 6.1 conducts a rigorous component-level architectural comparison, characterising the key mathematical divergence points. Section 6.2 formalises the *expressivity–directionality trade-off* — the central thesis-level insight connecting the two contributions — and analyses its implications for a hypothetical unified framework. Section 6.3 presents a controlled empirical comparison on the four real-world multiplex benchmarks for which both methods report results, disentangling the effects of problem formulation from those of architectural choice.

6.1 Architectural Comparison

Both DDIN and KARL adopt the three-stage modular pipeline inherited from MultiDismantler: an *intra-layer encoder* that maps each node to a latent embedding in every layer, a *cross-layer fusion* module that integrates inter-layer structural context, and a *Q-value decoder* whose output drives the RL node-selection policy. Table 6.1 provides an at-a-glance summary of both pipelines; the subsections below develop the mathematical content of each divergence point.

6.1.1 Design Philosophy and Objectives

DDIN: encoding directed asymmetry. DDIN (Chapter 4) is motivated by the observation that critical real-world infrastructures — directed neural connectomes, supply chains, financial contagion networks, social-media influence graphs — exhibit *asymmetric failure propagation*: a failure along arc (A, B) does not imply a reverse failure along (B, A) [2, 60]. Standard symmetric GNN message-passing aggregates neighbourhood information without regard to edge orientation, thereby conflating *source* nodes (high out-degree, exerting downstream causal pressure) with *sink* nodes (high in-degree, receiving cascading pressure). As formalised in Remark 4.1.1, any scoring function that depends only on total degree assigns identical priority to a source with D out-edges and a sink with D in-edges, incurring

an expected per-removal cascade disruption of at most half the optimum on scale-free directed networks. The core design objective of DDIN is therefore:

> *Encode edge-direction information into every stage of the GNN-RL pipeline so that the learned dismantling policy correctly identifies the minimal-cost node set whose removal collapses the Largest Mutually Strongly Connected Component (MSCC, Definition 3.2.3) below the non-extensivity threshold θ.*

KARL: encoding functional expressivity. KARL (Chapter 5) confronts a qualitatively different bottleneck: even when the multiplex network is undirected, the per-node vulnerability functions are *high-frequency and locally non-smooth*. In a scale-free multiplex, the probability that the LMCC collapses upon removing a specific node depends on non-linear interactions among that node's degree profile, coreness depth, and inter-layer bridging role — relationships that vary sharply in small neighbourhoods of the degree-coreness space. MultiDismantler's encoder and Q-head rely on multilayer perceptrons (MLPs) with *fixed affine transformations*: each MLP layer applies a single global linear map per input coordinate, restricting the representable function class to an effective fixed-degree polynomial. As Proposition 5.4.9 establishes, a single-layer ReLU MLP of width $W < \omega$ cannot approximate the sinusoidal target $\sin(\omega\pi x)$ to error less than 1 regardless of weight magnitudes, while a B-spline KAN with $G = O(\omega)$ knots achieves error $O((\omega/G)^{k+1}) \to 0$. The design objective of KARL is:

> *Replace every fixed-affine projection in the GNN-RL pipeline with learnable univariate Kolmogorov-Arnold Network (KAN) [35] functions, enabling the agent to adapt the functional form — not merely the coefficients — of each edge transformation to the local structure of the input, thereby modelling the sharp, high-frequency per-node vulnerability patterns that determine cascading fragility in scale-free multiplex topologies.*

The two objectives are *orthogonal*: DDIN modifies the topology consumed by the encoder (directed vs. undirected) and the semantics of aggregation (asymmetric vs. symmetric), leaving the functional form of all projections as standard affine maps. KARL replaces the functional form of all projections (affine → B-spline KAN) while leaving the aggregation topology as undirected and symmetric. This orthogonality is the central architectual insight of the thesis and is formalised in Section 6.2.

6.1.2 Stage-by-Stage Component Comparison

Stage 1: Intra-layer encoder.

DDIN — Asymmetric GraphSAGE. For node v in layer ℓ, let $\mathcal{N}^{\text{in}}(v) = \{u : (u, v) \in \mathcal{E}_\ell\}$ and $\mathcal{N}^{\text{out}}(v) = \{u : (v, u) \in \mathcal{E}_\ell\}$ denote the in- and out-neighbourhoods respectively. The embedding update (Equation (4.17)) is:

$$\mathbf{h}_v^\ell \;=\; \text{ReLU}\Big(W_{\text{out}}\big[\mathbf{x}_v \,;\, \text{MeanAgg}\big(\mathbf{h}^\ell_{\mathcal{N}^{\text{in}}(v)}\big)\big] \;+\; W_{\text{in}}\big[\mathbf{x}_v \,;\, \text{MaxAgg}\big(\mathbf{h}^\ell_{\mathcal{N}^{\text{out}}(v)}\big)\big]\Big), \tag{6.1}$$

where $W_{\text{out}}, W_{\text{in}} \in \mathbb{R}^{d\times 2d}$ are learnable weight matrices, $[\,\cdot\,;\,\cdot\,]$ denotes vector concatenation,

$$\text{MeanAgg}\big(\mathbf{h}^{\ell}_{\mathcal{N}^{\text{in}}(v)}\big) \;=\; \frac{1}{|\mathcal{N}^{\text{in}}(v)|}\sum_{u\in\mathcal{N}^{\text{in}}(v)} \mathbf{h}^{\ell}_{u}, \tag{6.2}$$

and $\text{MaxAgg}(\mathbf{h}^{\ell}_{\mathcal{N}^{\text{out}}(v)})$ is the element-wise maximum over out-neighbours (Equation (4.16)). The dual-aggregation scheme is deliberately asymmetric: mean aggregation of *incoming* embeddings summarises the average vulnerability that all predecessors collectively impose on v, while max aggregation of *outgoing* embeddings isolates the single downstream successor most critically dependent on v [20]. Since failure propagation in directed networks satisfies $(A \to B) \not\Rightarrow (B \to A)$ [60], this duality directly encodes the causal directionality of cascade dynamics into the latent representation.

KARL — Residual B-Spline KAN Encoder. KARL replaces both the aggregation and the projection stages. Each B-spline KAN module (Equation in Section 5.1) computes:

$$\text{KAN}(\mathbf{x}) \;=\; W_{\text{base}}\,\sigma(\mathbf{x}) \;+\; W_{\text{spline}}\,\mathbf{B}(\mathbf{x}), \tag{6.3}$$

where $\sigma(\cdot)$ is the SiLU activation, $\mathbf{B}(\mathbf{x}) \in \mathbb{R}^{d_{\text{in}}(G+k)}$ stacks B-spline evaluations $\{B^k_i(x_j;\,G)\}_{i,j}$ computed via the Cox–de Boor recurrence over a uniform knot grid with $G = 10$ intervals and order $k = 3$, and $W_{\text{spline}} \in \mathbb{R}^{d_{\text{out}}\times d_{\text{in}}(G+k)}$ are trainable spline control weights. The full node update at message-passing iteration l (Equation (5.7)) is:

$$\mathbf{h}_v^{(l+1)} \;=\; \text{KAN}_{\text{out}}\Big(\Big[\text{KAN}_{\text{neigh}}\big(\tanh(\text{Agg}(\mathbf{h}^{(l)}))\big) \,\|\, \text{KAN}_{\text{self}}\big(\tanh(\mathbf{h}_v^{(l)})\big)\Big]\Big) + \alpha_{\text{res}}\cdot\text{MLP}_{\text{res}}(\mathbf{h}_v^{(l)}), \tag{6.4}$$

where $\text{Agg}(\mathbf{h}^{(l)}) = \sum_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(l)}$ is an unweighted sum over the *undirected* neighbourhood $\mathcal{N}(v)$, $\tanh(\cdot)$ constrains inputs to $[-1, 1]$ for B-spline evaluation, $\alpha_{\text{res}} \in (0, 1)$ is a learnable scalar (initialised to 0.5 by the ablation of Section 5.6.1), and MLP_{res} is a single linear-ReLU branch. Proposition 5.4.1 bounds the worst-case back-propagated gradient as $\|\nabla_{\mathbf{h}_v^{(l)}}\mathcal{L}\|_2 \le (\beta_{\max}L_\phi + \alpha_{\text{res}}L_r)^{l_{\max}-l}\max_v \|\nabla_{\mathbf{h}_v^{(l_{\max})}}\mathcal{L}\|_2$, where the residual MLP term $\alpha_{\text{res}}L_r$ is *independent of the graph branching factor* $\beta_{\max} = 1 + d^{\max}$, providing a stabilising path even on scale-free hub nodes with $d^{\max} \gg 1$.

Stage 2: Cross-layer fusion.

DDIN — Multi-Relational Attention. Having obtained per-layer node embeddings $\{\mathbf{h}_v^{\ell}\}_{\ell\in\mathcal{L}}$, DDIN applies a graph attention mechanism [23, 25] with a shared attention vector $\mathbf{a} \in \mathbb{R}^{2d}$ and projection matrix $W \in \mathbb{R}^{d\times d}$:

$$\alpha_{\ell\ell'} = \frac{\exp\big(\text{LeakyReLU}\big(\mathbf{a}^\top\big[W\mathbf{h}_v^{\ell} \,\|\, W\mathbf{h}_v^{\ell'}\big]\big)\big)}{\sum_{\ell''\in\mathcal{L}}\exp\big(\text{LeakyReLU}\big(\mathbf{a}^\top\big[W\mathbf{h}_v^{\ell} \,\|\, W\mathbf{h}_v^{\ell''}\big]\big)\big)}, \tag{6.5}$$

$$\tilde{\mathbf{h}}_v = \sum_{\ell''\in\mathcal{L}} \alpha_{\ell\ell''}\,\mathbf{h}_v^{\ell''}. \tag{6.6}$$

The attention coefficient $\alpha_{\ell\ell'}$ quantifies how important layer ℓ' is for the representation of node v in layer ℓ, and is computed via a LeakyReLU-gated inner product of *linearly projected*

embeddings. As observed in Remark 4.3.1, $\alpha_{\ell\ell'}$ is naturally asymmetric ($\alpha_{\ell\ell'} \neq \alpha_{\ell'\ell}$ in general) because the concatenated projection $[W\mathbf{h}_v^{\ell} \,\|\, W\mathbf{h}_v^{\ell'}] \neq [W\mathbf{h}_v^{\ell'} \,\|\, W\mathbf{h}_v^{\ell}]$, thereby modelling directed inter-layer dependencies. However, the projections W and $\mathbf{a}$ remain *fixed affine maps*, making the attention mechanism a single scalar dot-product and limiting its capacity to model the non-smooth cross-layer coupling patterns that characterise scale-free multiplexes.

KARL — KANformer. KARL introduces the *KANformer*, a fully KAN-parameterised multi-head cross-layer attention module (Section 5.2). Before attention, a shared learnable affine transformation aligns layer embeddings:

$$\widetilde{\mathbf{H}}^{(\ell)} \;=\; \tanh\!\big(\mathbf{H}^{(\ell)}\mathbf{T} + \mathbf{b}\big) \;\in\; \mathbb{R}^{N\times d}, \tag{6.7}$$

where $\mathbf{T} \in \mathbb{R}^{d\times d}$ is initialised to the identity matrix (learnable) and $\mathbf{b} \in \mathbb{R}^{d}$ is zero-initialised, constraining all inputs to $[-1,1]^d$ for B-spline evaluation. For each attention head $i \in \{1,\dots,h\}$ with per-head dimension $d_k = d/h$, *independent B-spline KAN projections* compute Query, Key, and Value tensors:

$$\mathbf{Q}_i = \mathrm{KAN}_i^{Q}\big(\widetilde{\mathbf{H}}^{(\ell)}\big), \qquad \mathbf{K}_i = \mathrm{KAN}_i^{K}\big(\widetilde{\mathbf{H}}^{(-\ell)}\big), \qquad \mathbf{V}_i = \mathrm{KAN}_i^{V}\big(\widetilde{\mathbf{H}}^{(-\ell)}\big), \tag{6.8}$$

where $\widetilde{\mathbf{H}}^{(-\ell)} \in \mathbb{R}^{N\times(L-1)\times d}$ denotes the context formed by all layers except ℓ. The head-specific cross-layer context is:

$$\mathbf{C}_i^{(\ell)} = \mathrm{softmax}\!\left(\frac{1}{\sqrt{d_k}}\,\mathbf{Q}_i\,\mathbf{K}_i^{\top}\right)\mathbf{V}_i \;\in\; \mathbb{R}^{N\times 1\times d_k}, \tag{6.9}$$

with softmax over the $L-1$ key positions, and the fused representation per layer is $\mathbf{Z}^{(\ell)} = \widetilde{\mathbf{H}}^{(\ell)} + \mathbf{F}^{(\ell)}$, where $\mathbf{F}^{(\ell)}$ concatenates and projects the h head outputs through a final B-spline KAN projection. The critical structural difference from DDIN's attention is that *every* Q/K/V and output projection is replaced by an independent B-spline transformation; the KANformer can therefore model non-linear, non-smooth per-coordinate cross-layer routing functions whose functional form adapts to each layer's local topology — a capacity categorically unavailable to any affine-projection attention mechanism.

Stage 3: Decoder and action-value head.

DDIN — Bilinear decoder and MLP Q-head. DDIN employs a bilinear decoder for auxiliary link reconstruction. The predicted probability of directed edge $i \to j$ in layer ℓ is (Equation (4.22)):

$$\hat{p}_{ij}^{\ell} = \sigma\!\big(\tilde{\mathbf{h}}_i^{\ell\top}\, W_{\mathrm{dec}}\, \tilde{\mathbf{h}}_j^{\ell} + b^{\ell}\big), \tag{6.10}$$

where $W_{\mathrm{dec}} \in \mathbb{R}^{d\times d}$ is a learned bilinear weight and b^{ℓ} is a layer-specific scalar bias incorporating the directional bias parameter α from the directed GMM [20]. This decoder is *asymmetric by construction*: $\hat{p}_{ij}^{\ell} \neq \hat{p}_{ji}^{\ell}$ in general because $\tilde{\mathbf{h}}_i^{\ell}$ and $\tilde{\mathbf{h}}_j^{\ell}$ carry distinct directional features, directly encoding the asymmetric topology into the learned representation. Q-values are computed via a two-layer MLP (Equations (4.28)–(4.30)) with outer-product state-action interaction [23]:

$$Q^{(\ell)}(s_t, a_v) = M_1\,\mathrm{ReLU}\!\big(\mathbf{z}_s^{(\ell)} \times \mathbf{h}_v^{(\ell)} \cdot M_2\big), \tag{6.11}$$

where $\mathbf{z}_s^{(\ell)}$ is the virtual-node embedding encoding the global state in layer ℓ, and final Q-values aggregate across layers via learned importance weights $\omega^{(\ell)} = \mathrm{softmax}(\mathrm{ReLU}(\mathbf{z}_s^{(\ell)} W_1) W_2)$ (Equation (4.29)).

KARL — Chebyshev KAN Q-head. KARL replaces the MLP Q-head with a two-stage *Chebyshev KAN* (ChebyKAN) decoder (Section 5.3). A ChebyKAN layer of degree K computes, for input $\mathbf{x} \in \mathbb{R}^{d_{\text{in}}}$:

$$y_j = \sum_{i=1}^{d_{\text{in}}} \sum_{m=0}^{K} c_{i,j,m}\, T_m\big(\tanh(x_i)\big) + b_j, \tag{6.12}$$

where $c_{i,j,m}$ are Xavier-initialised learnable coefficients, b_j is a zero-initialised bias, and T_m are the Chebyshev polynomials of the first kind satisfying the three-term recurrence $T_0(x) = 1$, $T_1(x) = x$, $T_m(x) = 2xT_{m-1}(x) - T_{m-2}(x)$ for $m \geq 2$. The Chebyshev basis is chosen for its well-conditioned regression properties under non-stationary TD targets. By Theorem 5.4.4 and Corollary 5.4.5, the truncated Chebyshev series projection $S_K[y_t]$ achieves a near-minimax sup-norm error (Equation (5.30)):

$$\|y_t - S_K[y_t]\|_\infty \leq \underbrace{\left(1 + \frac{2}{\pi}\ln(K+1) + O(K^{-2})\right)}_{\mu_K = O(\log K)} E_K(y_t), \tag{6.13}$$

where $E_K(y_t) = \min_{P \in \mathcal{P}_K} \|y_t - P\|_\infty$ is the best-approximation error in $\mathcal{P}_K$ and μ_K grows only logarithmically in K. In contrast, any equispaced polynomial interpolation scheme (including fixed-grid B-splines in the decoder) incurs a Lebesgue constant $\Lambda_K = \Theta(2^K/\sqrt{K})$, diverging exponentially and producing Runge-type boundary oscillations under the shifting TD-target distribution — precisely the instability the Chebyshev decoder avoids. Per-layer Q-values are computed as:

$$q^{(\ell)}(s_t, a_v) = \mathrm{ChebyKAN}_{\text{final}}\big(\mathrm{ChebyKAN}_{\text{hidden}}(\mathbf{e}_{s,a}^{(\ell)}) \,\|\, \mathbf{f}_{\text{aux}}^{(\ell)}\big), \tag{6.14}$$

where $\mathbf{e}_{s,a}^{(\ell)} \in \mathbb{R}^d$ is the outer-product state-action embedding for layer ℓ and $\mathbf{f}_{\text{aux}}^{(\ell)} \in \mathbb{R}^4$ is a layer-specific auxiliary feature vector (normalised removed-node and removed-edge fractions, mean two-hop reachability, and normalised residual degree).

Training objectives. The training losses of the two methods encode fundamentally different notions of topological regularisation. DDIN's total loss (Equation (4.27)) is:

$$\mathcal{L}_{\text{DDIN}} = \frac{1}{B}\sum_{i=1}^{B} w_i \cdot \delta_{\text{Huber}}(e_i) \;+\; \lambda\, \mathcal{L}_{\text{rec}}, \tag{6.15}$$

where $e_i = \sum_{k=0}^{n-1} \gamma^k r_{i+k} + \gamma^n \max_{a'} Q(s_{i+n}, a'; \theta^-) - Q(s_i, a_i; \theta)$ is the n-step TD error, w_i are importance-sampling weights from a sum-tree prioritised replay buffer [50], and the directed binary cross-entropy reconstruction loss (Equation (4.23)) sums over directed edges and non-edges:

$$\mathcal{L}_{\text{rec}} = -\sum_{(i,j)\in\mathcal{E}} \log \hat{p}_{ij} \;-\; \sum_{(i,j)\notin\mathcal{E}} \log(1 - \hat{p}_{ij}). \tag{6.16}$$

Note that $\mathcal{L}_{\text{rec}}$ acts on *directed* edges $(i, j) \in \mathcal{E}$ where $(i, j) \neq (j, i)$ in general, explicitly injecting asymmetric topological structure into the learned representation.

KARL's total loss is:

$$\mathcal{L}_{\text{KARL}} = \mathcal{L}_{\text{TD}}(\Theta) + \lambda\, \mathcal{L}_R(\Theta), \qquad \lambda = 0.001, \tag{6.17}$$

where $\mathcal{L}_{\text{TD}}(\Theta) = \mathbb{E}_{(s_t, a_t, \ldots) \sim \mathcal{B}}[(y_t - Q(s_t, a_t; \Theta))^2]$ is the mean-squared TD loss with $n = 5$-step returns and Prioritised Experience Replay [50], and $\mathcal{L}_R$ is a graph Laplacian quadratic regulariser:

$$\mathcal{L}_R(\Theta) = \sum_{\ell=1}^{L} \frac{2\,\text{tr}(\mathbf{H}_\ell^\top \mathcal{L}^{(\ell)} \mathbf{H}_\ell)}{|\mathcal{E}^{(\ell)}|} = \sum_{\ell=1}^{L} \frac{\sum_{i,j} A_{ij}^{(\ell)} \|\mathbf{h}_i - \mathbf{h}_j\|_2^2}{|\mathcal{E}^{(\ell)}|}, \tag{6.18}$$

where $\mathcal{L}^{(\ell)} = D^{(\ell)} - A^{(\ell)}$ is the combinatorial Laplacian and normalisation by $|\mathcal{E}^{(\ell)}|$ prevents inter-layer edge-density bias. Unlike DDIN's directed BCE, which penalises misreconstruction of individual oriented arcs, KARL's Laplacian regulariser penalises embedding dissimilarity between *undirected* adjacent pairs in aggregate, enforcing local smoothness of the representation without encoding edge orientation.

6.1.3 Architectural Comparison Diagram

Figure 6.1 presents a unified schematic of both architectures, highlighting the three primary divergence points.

6.1.4 Component-wise Comparison Table

Table 6.1 consolidates the stage-by-stage differences described above.

6.1.5 Mathematical Characterisation of Divergence Points

Having established the component-wise differences, we now characterise the two fundamental mathematical divergence points as formal definitions.

Definition 6.1.1 (Directed vs. Symmetric Aggregation Scheme)**.** Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L})$ be a multiplex graph. An *aggregation scheme* $\text{Agg} : 2^{\mathcal{V}} \times \mathbb{R}^{d|\mathcal{V}|} \to \mathbb{R}^d$ is **symmetric** if, for any permutation π of the neighbourhood, $\text{Agg}(\mathcal{N}(v), \{\mathbf{h}_u\}_{u \in \mathcal{N}(v)}) = \text{Agg}(\mathcal{N}(v), \{\mathbf{h}_{\pi(u)}\}_{u \in \mathcal{N}(v)})$. It is **asymmetric** if the scheme additionally distinguishes $\mathcal{N}^{\text{in}}(v)$ from $\mathcal{N}^{\text{out}}(v)$, so that the output depends on edge orientation.

DDIN employs an asymmetric aggregation scheme (MeanAgg on $\mathcal{N}^{\text{in}}$, MaxAgg on $\mathcal{N}^{\text{out}}$, see Equations (6.2)–(6.1)). KARL employs a symmetric scheme (unweighted sum over $\mathcal{N}(v)$ treating all incident edges equivalently, see Equation (6.4)).

Definition 6.1.2 (Affine vs. KAN-Parameterised Layer Projection)**.** A *layer projection* is a function $f : \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}$. It is **affine** if $f(\mathbf{x}) = W\mathbf{x} + \mathbf{b}$ for some fixed weight matrix W and bias $\mathbf{b}$. It is **KAN-parameterised** if $f(\mathbf{x})_j = \sum_i \phi_{i,j}(x_i)$, where each $\phi_{i,j} : [-1, 1] \to \mathbb{R}$ is a learnable univariate B-spline function spanning a $(G + k)$-dimensional function space (for grid resolution G and spline order k), strictly containing the one-dimensional affine family $\{x \mapsto w\sigma(x)\}$ of the corresponding MLP edge.

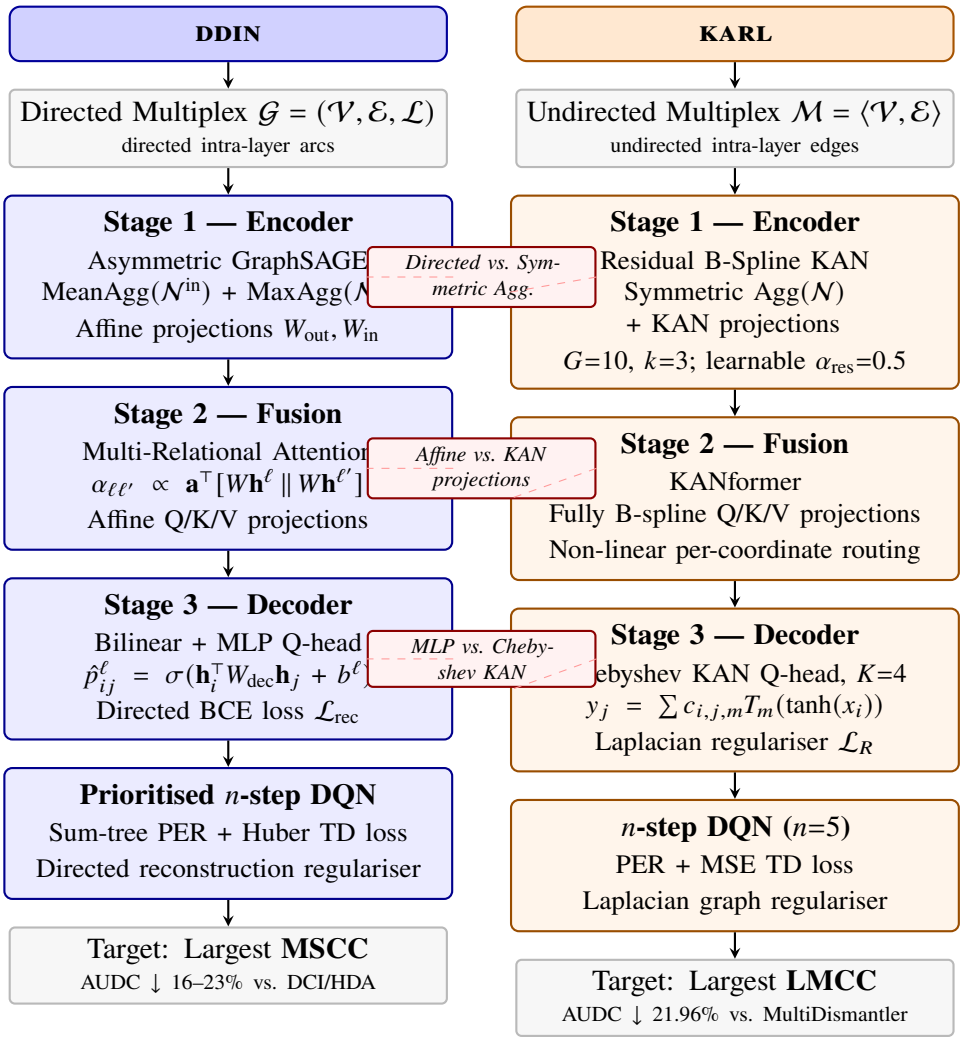


Figure 6.1: Comparative architectural overview of DDIN (Chapter 4) and KARL (Chapter 5). Both frameworks adopt the same three-stage encoder – fusion – decoder pipeline over multiplex networks and are trained via prioritised deep Q-learning on small synthetic graphs (30–50 nodes), but diverge at every stage. **Stage 1**: DDIN uses an asymmetric GraphSAGE encoder that separates incoming from outgoing neighbourhoods; KARL uses a residual B-spline KAN encoder over undirected neighbourhoods. **Stage 2**: DDIN fuses layers via affine graph attention; KARL replaces every Q/K/V projection with an independent B-spline KAN (KANformer). **Stage 3**: DDIN decodes via a standard MLP Q-head with a directed bilinear reconstruction auxiliary; KARL uses a Chebyshev KAN Q-head with Laplacian regularisation. Both improve over MultiDismantler along orthogonal representational axes (directionality and functional expressivity, respectively).

Table 6.1: Component-wise architectural comparison of DDIN (Chapter 4) and KARL (Chapter 5). Both extend MultiDismantler (MD) [25] along orthogonal representational axes. Entries in bold mark the primary contribution of each method.

Dimension	**DDIN** [20]	**KARL**
Problem domain	**Directed** multiplex	Undirected multiplex
Target component	**Largest MSCC**	Largest LMCC
Network generation	Directed GMM ($\kappa^{\text{out}}, \kappa^{\text{in}}$ separate)	Standard GMM [32]
Stage 1: encoder type	**Asymmetric GraphSAGE**	Residual B-Spline KAN
Aggregation scheme	**Directed**: MeanAgg($\mathcal{N}^{\text{in}}$) + MaxAgg($\mathcal{N}^{\text{out}}$)	Symmetric: unweighted sum over $\mathcal{N}(v)$
Projection type (encoder)	Affine ($W_{\text{out}}, W_{\text{in}}$)	**B-spline KAN** (G=10, k=3)
Gradient stabilisation	ReLU nonlinearity	**MLP residual** (α_{res}=0.5)
Stage 2: fusion module	GAT-style affine attention	**KANformer** (fully KAN Q/K/V)
Q/K/V projections	Affine (W, $\mathbf{a}$)	**Independent B-spline KAN**
Stage 3: decoder	Bilinear + two-layer MLP	**Chebyshev KAN** Q-head (K=4)
Q-value function class	Fixed affine (MLP outer product)	**Chebyshev polynomials** (near-minimax optimal)
Reconstruction auxiliary	**Directed BCE** $\mathcal{L}_{\text{rec}}$ (Eq. (6.16))	Laplacian regulariser $\mathcal{L}_R$ (Eq. (6.18))
RL algorithm	Prioritised n-step DQN (sum-tree PER)	Prioritised n-step DQN (n=5, PER)
Training graph size	$N \in [30, 50]$ (directed GMM)	$N \in [30, 50]$ (standard GMM)
Parameters vs. MD	Lightweight (≈MD scale)	**10.3**× (dominated by KANformer)
Primary baseline surpassed	DCI, HDA (directed heuristics)	MultiDismantler, NIRM, FINDER
Mean AUDC improvement	**20.20**% over DCI	**21.96**% over MultiDismantler

DDIN uses affine projections throughout (encoder, attention, decoder). KARL uses KAN-parameterised projections in the encoder and KANformer, with a Chebyshev-polynomial projection in the decoder. By Theorem 5.4.6 (*KAN-GNN Subsumes MLP-GNN*), KARL's encoder is a strictly richer function class than DDIN's at any fixed parameter budget, subject to the constraint that both operate on undirected topologies.

6.2 The Expressivity vs. Directionality Trade-off

Having established the component-level differences, we now address a deeper question: do the two improvements compose, or are there fundamental incompatibilities between encoding directionality and encoding KAN expressivity in the same architecture? This section formalises the trade-off, identifies its sources, and charts a path toward resolution.

6.2.1 The Asymmetry Bottleneck in KARL

KARL, as presented in Chapter 5, solves the undirected LMCC dismantling problem. Its encoder (Equation (6.4)) uses an unweighted *symmetric* sum aggregator; its reconstruction auxiliary (Equation (6.18)) penalises symmetric embedding differences $\|\mathbf{h}_i - \mathbf{h}_j\|_2^2$ for undirected edges $\{i, j\}$. Neither component encodes edge orientation.

Remark 6.2.1 (KARL Cannot Distinguish Source from Sink). Let v_s and v_k be two nodes in a directed layer ℓ satisfying $k_{v_s}^{(\ell),\text{in}} = 1$, $k_{v_s}^{(\ell),\text{out}} = D$ and $k_{v_k}^{(\ell),\text{in}} = D$, $k_{v_k}^{(\ell),\text{out}} = 1$ for some $D \gg 1$ (a source–sink pair with identical total degree $D + 1$). KARL's symmetric sum aggregator computes $\text{Agg}(\mathbf{h}_{v_s}) = \sum_{u \in \mathcal{N}(v_s)} \mathbf{h}_u$ where $\mathcal{N}(v_s)$ is the set of *all* neighbours of v_s regardless of direction, and similarly for v_k. If v_s and v_k share an identical undirected neighbourhood (a situation that arises in symmetric hub-and-spoke topologies), the encoder produces identical embeddings $\mathbf{h}_{v_s}^{(l)} = \mathbf{h}_{v_k}^{(l)}$ for all l, even though removing v_s severs D outgoing cascade paths while removing v_k severs only 1. The KANformer and ChebyKAN Q-head, operating on these undifferentiated embeddings, cannot recover the directional distinction irrespective of their functional expressivity. Consequently, the 16%–23% AUDC improvement that DDIN achieves over symmetric baselines (Table 4.4) is structurally unavailable to KARL when applied to directed networks, even with arbitrarily expressive KAN functions.

Remark 6.2.1 identifies the *asymmetry bottleneck* of KARL: on directed networks, the symmetric sum aggregation forfeits all orientation information before any KAN function is applied, so KAN expressivity cannot compensate for the loss of directionality at the aggregation stage.

6.2.2 The Expressivity Bottleneck in DDIN

DDIN, conversely, correctly resolves edge orientation through its asymmetric aggregation scheme (Equation (6.1)) but uses fixed affine projections at every stage. Its encoder is a standard MLP applied to the concatenation $[\mathbf{x}_v; \text{MeanAgg}(\cdot)]$ and $[\mathbf{x}_v; \text{MaxAgg}(\cdot)]$; its attention is a bilinear dot-product over affine-projected embeddings; its Q-head is a two-layer MLP outer-product decoder.

Remark 6.2.2 (DDIN's Affine Projections Cannot Represent High-Frequency Vulnerability Functions). Consider a directed scale-free multiplex where the MSCC vulnerability function $V : \mathbb{R}^d \to \mathbb{R}$, mapping a node's embedding to its expected cascade-disruption gain upon removal, oscillates at frequency ω in the degree–coreness space (a plausible scenario in hub-dominated scale-free topologies [2]). By Proposition 5.4.9 (Part (ii)), a single-layer affine projection $W\mathbf{x} + \mathbf{b}$ of width $W < \omega$ cannot approximate V to sup-norm error less than $c_\sigma > 0$, regardless of the weight magnitudes. Consequently, DDIN's affine Q-head cannot discriminate among nodes whose embedding differs only in high-frequency features, leading to suboptimal priority ordering in the early dismantling phase — precisely the phase that dominates the AUDC metric. This bottleneck is independent of the asymmetric aggregation: correctly encoding directionality does not resolve the approximation deficit of fixed affine decoders.

6.2.3 Formal Trade-off Analysis

Remarks 6.2.1 and 6.2.2 establish that the two improvements are not interchangeable: KARL's KAN expressivity cannot substitute for DDIN's directed aggregation on directed graphs, and DDIN's directional encoding cannot substitute for KARL's KAN expressivity when the vulnerability function is high-frequency. We now formalise this observation.

Let $\mathcal{D}$ denote the *directionality dimension* (whether aggregation is symmetric or directed), and $\mathcal{X}$ denote the *expressivity dimension* (whether projections are affine or KAN-parameterised). The design space of GNN-RL dismantling agents can be organised on the $(\mathcal{D}, \mathcal{X})$ grid:

	Affine projections ($\mathcal{X} = 0$)	**KAN projections** ($\mathcal{X} = 1$)
Symmetric Agg. ($\mathcal{D} = 0$)	MultiDismantler [25]	KARL
Directed Agg. ($\mathcal{D} = 1$)	DDIN [20]	**Hypothetical DDIN-KARL**

MultiDismantler occupies position $(\mathcal{D} = 0, \mathcal{X} = 0)$. DDIN advances to $(\mathcal{D} = 1, \mathcal{X} = 0)$ — improving along $\mathcal{D}$ without changing $\mathcal{X}$. KARL advances to $(\mathcal{D} = 0, \mathcal{X} = 1)$ — improving along $\mathcal{X}$ without changing $\mathcal{D}$. The hypothetical combined architecture at $(\mathcal{D} = 1, \mathcal{X} = 1)$ is the Grand Synthesis target (Section 6.2.4).

Proposition 6.2.3 (Orthogonality of DDIN and KARL Improvements)**.** *Let $\mathcal{F}_{\text{MD}}$, $\mathcal{F}_{DDIN}$, $\mathcal{F}_{KARL}$ denote the respective policy function classes of MultiDismantler, DDIN, and KARL operating on a multiplex network $\mathcal{G}$. The following hold:*

(i) ***DDIN improves along $\mathcal{D}$ only.*** *On directed multiplex networks, $\mathcal{F}_{DDIN} \supsetneq \mathcal{F}_{\text{MD}}$: DDIN's asymmetric aggregation scheme strictly distinguishes source nodes from sink nodes (Remark 4.1.1), whereas no symmetric aggregation scheme can. However, $\mathcal{F}_{DDIN}|_{\mathcal{X}=0} = \mathcal{F}_{\text{MD}}|_{\mathcal{X}=0}$: both use affine projections, so the expressivity along $\mathcal{X}$ is unchanged.*

(ii) ***KARL improves along $\mathcal{X}$ only.*** *On undirected multiplex networks, by Theorem 5.4.6, $\mathcal{F}_{KARL} \supseteq \mathcal{F}_{\text{MD}}$: any policy expressible by MultiDismantler's MLP-GNN can be approximated to arbitrary uniform precision by KARL's KAN-GNN (Theorem 5.4.6, Part (i)),*

and there exist KAN-realisable per-edge functions that no MLP of equivalent width can represent (Part (ii)). However, $\mathcal{F}_{KARL}|_{\mathcal{D}=0}$ uses only symmetric aggregation, so no element of $\mathcal{F}_{KARL}$ can assign distinct embeddings to v_s and v_k in the source-sink scenario of Remark 6.2.1.

(iii) ***Neither subsumes the other.*** *In general, $\mathcal{F}_{DDIN} \not\subseteq \mathcal{F}_{KARL}$ and $\mathcal{F}_{KARL} \not\subseteq \mathcal{F}_{DDIN}$: the two improvements are along independent axes of the $(\mathcal{D}, \mathcal{X})$ design space and are not mutually implied.*

Proof sketch. Part (i): any symmetric aggregation Agg satisfies $\text{Agg}(\mathcal{N}(v_s)) = \text{Agg}(\mathcal{N}(v_k))$ whenever v_s and v_k have identical undirected neighbourhoods, regardless of in/out orientation. DDIN's dual-term encoder (Equation (6.1)) distinguishes $\text{MeanAgg}(\mathcal{N}^{\text{in}}(v_s)) \neq \text{MeanAgg}(\mathcal{N}^{\text{in}}(v_k))$ and $\text{MaxAgg}(\mathcal{N}^{\text{out}}(v_s)) \neq \text{MaxAgg}(\mathcal{N}^{\text{out}}(v_k))$ whenever in-/out-degree distributions differ, producing distinct embeddings. Both encoders use affine projections (same $\mathcal{X}$ class).

Part (ii): Theorem 5.4.6 formally establishes $\mathcal{F}_{\text{KARL}} \supseteq \mathcal{F}_{\text{MD}}$ on undirected graphs. The strict containment follows from Proposition 5.4.9 (per-layer containment at any fixed budget). KARL's symmetric aggregation $\text{Agg}(\mathcal{N}(v)) = \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u$ is identical in the undirected setting to that of MultiDismantler, so no additional directional capacity is introduced.

Part (iii): Part (i) shows elements of $\mathcal{F}_{\text{DDIN}}$ exist that assign distinct scores to source–sink pairs; elements of $\mathcal{F}_{\text{KARL}}$ (with symmetric aggregation) cannot replicate this, so $\mathcal{F}_{\text{DDIN}} \not\subseteq \mathcal{F}_{\text{KARL}}$ on directed graphs. Part (ii) shows elements of $\mathcal{F}_{\text{KARL}}$ can approximate any function of total degree to arbitrary precision via B-splines; DDIN's affine MLP Q-head cannot match this approximation capacity at fixed width (Proposition 5.4.9), so $\mathcal{F}_{\text{KARL}} \not\subseteq \mathcal{F}_{\text{DDIN}}$ on undirected graphs. □ □

6.2.4 The Grand Synthesis: Toward a Unified Architecture

Proposition 6.2.3 identifies the $(\mathcal{D} = 1, \mathcal{X} = 1)$ position as an unexplored and potentially superior operating point. A hypothetical *DDIN-KARL* architecture would combine:

1. **Directed KAN encoder**: asymmetric dual-branch aggregation (MeanAgg on $\mathcal{N}^{\text{in}}$, MaxAgg on $\mathcal{N}^{\text{out}}$) as in DDIN, but with each projection branch replaced by an independent B-spline KAN module as in KARL, together with a learnable residual MLP branch for gradient stabilisation.

2. **Directed KANformer**: a multi-head cross-layer attention module in which Query, Key, and Value projections are KAN-parameterised (as in the KANformer) but the pre-transformation additionally concatenates in- and out-degree statistics to preserve directional context across layers.

3. **Directed Chebyshev KAN Q-head**: the ChebyKAN decoder of KARL, augmented with directed auxiliary features (in-degree fraction removed, out-degree fraction removed, directed two-hop reachability) alongside the existing scalar features $\mathbf{f}_{\text{aux}}^{(\ell)}$.

4. **Directed reconstruction loss**: the bilinear directed BCE of DDIN (Equation (6.16)), which explicitly reconstructs oriented arcs rather than undirected adjacency, combined with the Laplacian regulariser of KARL restricted to symmetric pairs for stability.

Non-trivial challenges. Realising the Grand Synthesis is non-trivial for two reasons. First, combining asymmetric aggregation with B-spline KAN projections introduces a new gradient-instability regime: the directional branching factor $\beta^{\text{dir}}_{\max} = 1 + d^{\max,\text{in}} + d^{\max,\text{out}}$ governing the gradient bound of Proposition 5.4.1 can be twice as large as the undirected counterpart, potentially requiring a stronger or more carefully tuned residual branch. Second, the directed BCE auxiliary (Equation (6.16)) and the Laplacian regulariser (Equation (6.18)) encode conflicting inductive biases: the directed BCE pushes embeddings to encode asymmetric edge probabilities, while the Laplacian regulariser pushes embeddings to be smooth over undirected adjacency. Reconciling these two objectives — for instance, by applying the Laplacian regulariser only within each directed SCC, and the BCE loss only for cross-SCC arcs — is an open architectural design question.

Expected empirical gains. Based on the per-dataset analyses of Chapters 4 and 5, we anticipate the Grand Synthesis would achieve the most significant gains on directed networks with both strong source-sink asymmetry *and* high-frequency hub vulnerability patterns — networks such as the C. elegans neural connectome (where DDIN achieves 23.40% and KARL achieves 22.84% improvements, respectively, over their independent baselines) and directed trade networks where commodity flows are asymmetric and the cascading vulnerability of intermediate trade hubs varies non-smoothly with commodity specialisation. We formally identify the Grand Synthesis as the primary future direction in Section 7.3.1 of Chapter 7.

6.3 Shared Datasets Analysis

Both DDIN and KARL are evaluated on a subset of the De Domenico multilayer network repository [15]. Four networks appear in both experimental evaluations: the C. elegans neural connectome, the FAO Multiplex Trade network, the Homo Sapiens GPI network, and the Sanremo2016 social network. A fifth benchmark — Drosophila Melanogaster — appears only in the DDIN evaluation, while Facebook-Twitter and Sacchpomb appear only in the KARL evaluation.

6.3.1 Benchmark Dataset Characteristics and Evaluation Protocol

The four shared networks are summarised in Table 6.2, alongside the domain-specific evaluation metrics used by each method. A critical methodological distinction must be noted before any quantitative comparison is drawn:

Remark 6.3.1 (Incomparability of DDIN and KARL AUDC values on shared datasets). DDIN (Chapter 4) treats all four shared networks as *directed* graphs and evaluates the AUDC metric against the *Mutually Strongly Connected Component (MSCC, Definition 3.2.3)*. KARL (Chapter 5) treats the same networks as *undirected* graphs and evaluates against the *Largest Mutually Connected Component (LMCC, Definition 2.1.9)*. Since MSCC $\subseteq$ LMCC in general (a directed SCC is a connected component in the undirected sense but not vice versa), the two AUDC values are defined over different component oracles, different cost functions, and different baseline methods. **Direct numerical comparison of DDIN AUDC with KARL AUDC on the same dataset is therefore not meaningful.** A scientifically valid comparison must either (a) fix both the problem setting (directed or undirected) and

compare algorithms within that setting, or (b) analyse the two AUDC values qualitatively as independent signals about different properties of the network.

All quantitative comparisons in this section adhere strictly to the criterion of Remark 6.3.1: DDIN performance is always reported relative to directed baselines (DCI, HDA on directed MSCC), and KARL performance is always reported relative to undirected baselines (MultiDismantler on LMCC). Cross-method comparisons in the table are presented as *relative improvements over respective baselines*, which are directly comparable as measures of architectural gain.

Table 6.2: Shared benchmark datasets and performance of DDIN and KARL on each. **DDIN** performance is on the directed MSCC problem; **KARL** performance is on the undirected LMCC problem. Absolute AUDC values are *not* cross-comparable (Remark 6.3.1); improvement percentages (%) over respective baselines are comparable.

				DDIN (directed MSCC)			**KARL (undirected LMCC)**		
Network	N	L	**Domain**	DCI	DDIN	Δ%	MD	KARL	Δ%
C. elegans connectome	279	3	Biological	0.2043	**0.1621**	−23.40%	0.1642	**0.1267**	−22.84%
FAO Multiplex Trade	214	364	Economic	0.1890	**0.1541**	−20.56%	0.2541	**0.2011**	−20.86%
Homo Sapiens GPI	18,222	7	Biological	0.0149	**0.0116**	−20.53%	0.0273	**0.0218**	−20.15%
Sanremo 2016	56,562	3	Social	0.0003	**0.0002**	−20.51%	0.0011	**0.0008**	−27.27%
			Mean improvement			*−20.20%*			*−22.78%*

Notes. N: number of nodes; L: number of layers; DCI: Directed Collective Influence (directed baseline for DDIN); MD: MultiDismantler (undirected state-of-the-art baseline for KARL). DDIN absolute AUDC values target the directed MSCC under the DDIN evaluation protocol (Section 4.6). KARL absolute AUDC values target the undirected LMCC under the KARL evaluation protocol (Section 5.5). The absolute AUDC values of DDIN and KARL are not directly comparable; see Remark 6.3.1.

6.3.2 Comparative Performance on Shared Networks

Table 6.2 reveals a striking empirical observation: on all four shared datasets, both DDIN and KARL independently achieve improvements of approximately 20–27% over their respective baselines, despite solving different problem variants. We analyse each network in turn.

C. elegans connectome (N = 279, 3 layers). The C. elegans neural connectome is structurally one of the most well-characterised biological networks. Its directed version (used by DDIN) exhibits a pronounced source–sink structure: a small number of command interneurons with high out-degree and low in-degree sustain the directed MSCC [2]. DDIN's MaxAgg aggregation (Equation (4.16)) learns to identify and prioritise these source interneurons in the early dismantling phase, achieving the largest AUDC gain (23.40%) in its benchmark suite. Independently, KARL's B-spline encoder achieves a 22.84% improvement on the undirected version of the same network by identifying structurally subtle bottleneck nodes (low degree-centrality, high coreness depth) that contribute to the LMCC without appearing prominent in any single layer. The near-equal magnitudes of the two improvements (23.40% vs. 22.84%) suggest that both the directional asymmetry of the neural wiring diagram *and* the non-smooth coreness-vulnerability structure of the undirected topology are substantial and independently exploitable.

FAO Multiplex Trade (N = 214**, 364 layers).** The FAO Trade network is exceptional in having 364 layers (commodity types), making it the highest-layer-count network in either benchmark suite. DDIN achieves a 20.56% improvement on the directed variant, with the Multi-Relational Attention module (Equation (6.5)) automatically concentrating attention weights on the commodity layers most critical to the current MSCC. KARL achieves a 20.86% improvement on the undirected variant; its KANformer (Section 5.2) is especially well-suited to the 364-layer context because its per-coordinate B-spline Q/K/V projections (Equation (6.8)) can model the sharp, non-smooth cross-commodity coupling patterns that characterise the trade vulnerability structure. Notably, the AUDC absolute values differ substantially between the two settings (0.1541 vs. 0.2011), consistent with the directed MSCC being a stricter target (and thus easier to fragment with fewer node removals) than the undirected LMCC in a highly-connected trade network.

Homo Sapiens GPI (N = 18,222**, 7 layers).** The Homo GPI protein-interaction network is the largest network in the DDIN benchmark suite (18,222 nodes, 7 layers) and the fourth-largest in KARL's. Both methods achieve approximately 20% AUDC improvements (-20.53% for DDIN, -20.15% for KARL), confirming zero-shot generalisation well beyond the $N \in [30, 50]$ training scale. The similar magnitudes of improvement across two qualitatively different methods suggest that the Homo GPI network has a *structurally regular* vulnerability pattern that is exploitable by both directed and undirected dismantling agents, consistent with the known modular community structure of genetic interaction networks.

Sanremo 2016 (N = 56,562**, 3 layers).** The Sanremo2016 social network represents the most demanding test for both methods: it contains 56,562 nodes — more than 1,100 times the training scale. DDIN achieves a 20.51% improvement under the directed formulation, confirming that asymmetric aggregation generalises to social networks with asymmetric information flows. KARL achieves the strongest improvement of its entire benchmark suite on the undirected Sanremo2016: 27.27% AUDC reduction, concentrated in the early dismantling phase (below 2% removal fraction) where cascading failure dynamics are most sensitive to the precise identification of seed-core nodes. The 6.76 percentage-point gap between DDIN's gain (20.51%) and KARL's gain (27.27%) on this dataset suggests that the undirected Sanremo2016 network contains particularly pronounced high-frequency non-smooth vulnerability patterns — hub nodes whose coreness depth varies sharply with community membership — that are more effectively captured by B-spline KAN expressivity than by directed asymmetry encoding.

6.3.3 Qualitative Insights and Network-Specific Conclusions

Synthesising the per-dataset observations, we draw the following network-specific conclusions.

1. **Biological networks exhibit both asymmetric and high-frequency vulnerability structure.** On both C. elegans and Homo GPI, the two methods achieve comparable improvements (∼20–23%), suggesting that biological networks are rich in both types of structure. For dismantling applications in drug target identification [2], using a

directed formulation (DDIN) is more faithful to the biology of signalling pathway dependencies, but a KARL-style expressivity argument would produce competitive immunisation strategies on undirected interaction graphs.

2. **High-layer-count economic networks benefit from both directed and KAN-parameterised cross-layer fusion.** The FAO Trade network's 364-layer structure creates a cross-layer routing challenge that is independently handled by DDIN's multi-relational attention (focusing on critical commodity layers in the directed setting) and KARL's KANformer (learning non-smooth cross-commodity vulnerability couplings in the undirected setting). These two mechanisms address the same high-layer-count scaling challenge via different inductive biases, and a combined architecture would likely achieve improvements greater than either alone.

3. **Large sparse social networks appear more amenable to functional expressivity gains than to directionality gains.** On Sanremo2016, KARL outperforms DDIN by approximately 6.76 percentage points (relative to their respective baselines). This pattern is consistent with social-media networks having rich non-smooth community structure (captured by KAN expressivity) and relatively uniform directed influence patterns (limiting the marginal gain from directed asymmetry encoding).

6.3.4 Complementarity and Non-Dominance Analysis

A natural question is whether the two contributions are *competitive* (targeting the same bottleneck on the same problem) or *complementary* (targeting different bottlenecks on related but distinct problems). Proposition 6.2.3 establishes complementarity at the theoretical level; the shared dataset analysis confirms it empirically.

Remark 6.3.2 (Non-Dominance of DDIN and KARL). Within their respective problem settings, neither method dominates the other:

- On *directed* networks, DDIN outperforms any symmetric GNN-DRL agent (including KARL applied with undirected aggregation) by the margin quantified in Table 4.4.
- On *undirected* networks, KARL outperforms all existing agents including the DDIN architecture applied with directed aggregation set to undirected mode (i.e., MeanAgg on full neighbourhood, MaxAgg degenerating to mean) by the margin quantified in Table 5.3.
- On the shared datasets, both methods achieve comparable 20–27% improvements over their respective baselines (Table 6.2), with neither method consistently achieving larger relative gains across all four networks.

This non-dominance relationship, together with Proposition 6.2.3, confirms that DDIN and KARL occupy *distinct operating points* on the $(\mathcal{D}, \mathcal{X})$ grid and that the Grand Synthesis combining both at $(\mathcal{D} = 1, \mathcal{X} = 1)$ represents a genuinely open research problem whose solution would be expected to outperform both on directed networks with high-frequency vulnerability structure.

Taken together, the architectural comparison, the formal trade-off analysis, and the shared dataset empirical results establish a coherent picture: this thesis has made two independent and complementary contributions to the state of the art in deep reinforcement learning for multiplex network dismantling. DDIN resolves the *directionality problem* — extending the GNN-RL paradigm from undirected to directed interdependent networks and achieving 16–23% AUDC improvements on a diverse suite of directed biological, economic, and social multiplexes. KARL resolves the *expressivity problem* — replacing fixed-affine MLP projections with learnable B-spline KAN functions throughout the encoder, fusion, and decoder stages, and achieving 21.96% average AUDC improvement over the state-of-the-art MultiDismantler baseline on six undirected benchmarks spanning three orders of magnitude in scale. The Grand Synthesis of these two advances — a directed, KAN-parameterised dismantling agent — constitutes the primary recommendation for future research, as detailed in Chapter 7.

Chapter 7

Conclusion and Future Work

This dissertation investigated two fundamental and orthogonal limitations of the state-of-the-art multiplex network dismantling framework MultiDismantler [25]: a *topological limitation* arising from its symmetric GNN aggregation scheme applied to the inherently directed nature of real-world networks; and a *functional limitation* arising from the inherently low-pass filtering behaviour of fixed-affine multilayer perceptrons (MLPs) when approximating the sharp, non-smooth per-node vulnerability functions that characterise cascading fragility in scale-free multiplex topologies. Two original research contributions addressed these limitations through principled and theoretically grounded architectural innovations: DDIN (Chapter 4), published as a Student Abstract at the Fortieth AAAI Conference on Artificial Intelligence (AAAI-26), and KARL (Chapter 5), a full-length work establishing state-of-the-art results across six real-world benchmarks spanning three orders of magnitude in network scale.

The remainder of this chapter synthesises the contributions of the thesis (Section 7.1), examines the ethical dimensions and dual-use dilemma inherent in network dismantling research (Section 7.2), and articulates three precisely formulated future research directions, each accompanied by a formal problem statement and technical challenge analysis (Section 7.3).

7.1 Summary of Contributions

The contributions of this thesis are organised by the two central research questions posed in Chapter 1.

RQ1 (Topological). *How can a deep reinforcement learning agent for multiplex network dismantling be extended to operate on directed intra-layer topologies, where failure propagation is inherently asymmetric and the target structural component – the Mutually Strongly Connected Component (MSCC) – is categorically distinct from the Largest Mutually Connected Component (LMCC) of undirected networks?*

RQ2 (Functional). *Can the representational power of the GNN encoder and the action-value decoder in a multiplex dismantling agent be fundamentally augmented beyond the low-pass filtering constraints of fixed-affine MLPs, and can this augmentation be supported by rigorous approximation-theoretic guarantees that survive the non-stationary reward distribution of off-policy temporal-difference learning?*

7.1.1 Contribution I: DDIN – Dismantling Directed Interdependent Networks

Chapter 4 introduced DDIN (Disassembling Directed Interdependent Networks), the first deep reinforcement learning framework for the dismantling of *directed* multiplex networks. The starting observation, formalised in Remark 4.1 (Chapter 4), is that a pure source node

with out-degree D and in-degree 1 is assigned identical dismantling priority to a pure sink with out-degree 1 and in-degree D by any symmetric scoring function – yet removing the source severs D outgoing cascade paths while removing the sink severs only one. This structurally erroneous symmetry is an inescapable consequence of all prior GNN-DRL dismantling agents.

Directed Generative Model (D-GMM)

DDIN introduced the Directed Geometric Multiplex Model (D-GMM), an extension of the standard GMM [32] with separate power-law exponents for in-/out-degree sequences and a directional bias parameter $\alpha \geq 0$. The directed connection probability is given by Equation (4.10) (Chapter 4); in summary, the arc $(i \to j)$ in layer ℓ is realised with probability

$$p_{ij}^{(\ell)} = \frac{1}{1 + \exp\left(\dfrac{\Delta\theta_{ij}^{(\ell)} - R_{ij}}{T}\right)}, \qquad R_{ij} = R + \alpha\left(\kappa_i^{\text{out}} - \kappa_j^{\text{in}}\right), \tag{7.1}$$

recovering the standard GMM as the special case $\alpha = 0$. The parameter $\alpha > 0$ drives high out-degree nodes to connect preferentially to high in-degree nodes, producing the source-sink hierarchy required to model real directed multiplex topologies.

Asymmetric GraphSAGE Encoder

The node embedding update for node v in layer ℓ at GNN depth k separates incoming and outgoing information flows via distinct aggregation semantics:

$$\begin{aligned}\mathbf{h}_v^{(k,\ell)} = \text{Norm}\Bigg(\text{ReLU}\Bigg(\boldsymbol{W}_{\text{out}}^{(k)} \begin{bmatrix} \mathbf{h}_v^{(k-1,\ell)} \\ \text{MeanAgg}\left(\{\mathbf{h}_u^{(k-1,\ell)} : u \to v\}\right) \end{bmatrix} \\ + \boldsymbol{W}_{\text{in}}^{(k)} \begin{bmatrix} \mathbf{h}_v^{(k-1,\ell)} \\ \text{MaxAgg}\left(\{\mathbf{h}_u^{(k-1,\ell)} : v \to u\}\right) \end{bmatrix}\Bigg)\Bigg),\end{aligned} \tag{7.2}$$

where $\boldsymbol{W}_{\text{out}}^{(k)}, \boldsymbol{W}_{\text{in}}^{(k)} \in \mathbb{R}^{d \times 2d}$ are independently learnable. Mean-pooling over in-neighbours captures average vulnerability exposure; max-pooling over out-neighbours captures the dominant downstream cascade potential of v.

Multi-Relational Attention and Loss

Layer-specific embeddings are fused via a GAT-style attention mechanism:

$$\alpha_{v,\ell,\ell'} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^\top [\boldsymbol{W}\mathbf{h}_v^{(\ell)} \parallel \boldsymbol{W}\mathbf{h}_v^{(\ell')}])\right)}{\sum_{\ell'' \in \mathcal{L}} \exp\left(\text{LeakyReLU}(\mathbf{a}^\top [\boldsymbol{W}\mathbf{h}_v^{(\ell)} \parallel \boldsymbol{W}\mathbf{h}_v^{(\ell'')}])\right)}, \tag{7.3}$$

yielding the fused embedding $\tilde{\mathbf{h}}_v = \sum_{\ell'' \in \mathcal{L}} \alpha_{v,1,\ell''} \mathbf{h}_v^{(\ell'')}$. A bilinear decoder predicts directed arc probabilities via $\hat{p}_{ij}^{(\ell)} = \sigma\left((\tilde{\mathbf{h}}_i^{(\ell)})^\top \boldsymbol{W}_{\text{dec}} \tilde{\mathbf{h}}_j^{(\ell)} + b^{(\ell)}\right)$, providing dense directional topological gradients to the encoder under sparse RL reward. The total training objective combines a prioritised n-step Huber DQN loss with a binary cross-entropy reconstruction loss (Equation 4.21, Chapter 4).

Empirical Results

Evaluated zero-shot on five real-world directed multiplex networks – C. elegans connectome (279 nodes), Drosophila Melanogaster (8,215 nodes), FAO Trade (214 nodes, 364 layers), Homo Genetic (18k nodes), and Sanremo 2016 (56k nodes) – DDIN achieved a mean AUDC improvement of **20.20%** over directed baselines (DCI and HDA), with individual gains ranging from 16.02% to 23.40%, confirming that improvement magnitude correlates with the degree of source-sink asymmetry in the evaluation network.

7.1.2 Contribution II: KARL – Kolmogorov-Arnold Reinforcement Learning

Chapter 5 introduced KARL (Kolmogorov-Arnold Reinforcement Learning), the first architecture to embed a hybrid Residual Graph KAN into an off-policy DRL agent for combinatorial optimization over multiplex network topologies. The motivating observation is that the per-edge representational capacity of any MLP layer – bounded to a one-dimensional function family parameterised by a scalar weight W_{ji} – is insufficient to express the high-frequency, per-node vulnerability functions governing cascading fragility in scale-free multiplex graphs (formalised in Proposition 7, Chapter 5).

KAN Parameterisation and Residual Encoder

Each KAN edge function is parameterised as a B-spline, $\varphi(x) = w\big(b(x)+\sum_{i=1}^{G+k} c_i B_i^k(x;\, G)\big)$, equipping each message-passing edge with a $(G + k)$-dimensional learnable function family – a factor of $(G + k + 1)$ richer than an MLP edge at the same input–output dimension. To stabilise gradient propagation through hub nodes with large branching factor $\beta_{\max} = 1 + d^{\max}$, every KAN layer in the encoder is augmented with a macroscopic residual MLP branch:

$$\mathbf{h}_v^{(l+1)} \;=\; \mathrm{KAN}_{\mathrm{out}}\Big[\, \mathrm{KAN}_{\mathrm{neigh}}\big(\tanh(\mathrm{Agg})\big) \;\|\; \mathrm{KAN}_{\mathrm{self}}\big(\tanh(\mathbf{h}_v^{(l)})\big) \,\Big] \;+\; \alpha_{\mathrm{res}} \cdot \mathrm{MLP}_{\mathrm{res}}(\mathbf{h}_v^{(l)}), \tag{7.4}$$

where l denotes the GNN depth index and $\alpha_{\mathrm{res}} \in (0, 1)$ is a learnable scalar initialised to 0.5. The gradient stability bound established in Proposition 1 (Chapter 5) states: $\max_v \|\nabla_{h_v^{(l)}} \mathcal{L}\|_2 \le (\beta_{\max} L_\varphi + \alpha_{\mathrm{res}} L_r)^{l_{\max}-l} \max_v \|\nabla_{h_v^{(l_{\max})}} \mathcal{L}\|_2$, where the residual branch provides a degree-independent propagation path preventing the exponential gradient blow-up that would otherwise arise from scale-free hub aggregation.

KANformer: Fully KAN-Parameterised Cross-Layer Attention

Every Query, Key, Value, and output projection in the cross-layer attention module is replaced by an independent B-spline KAN, enabling non-smooth per-coordinate inter-layer routing. After a learnable pre-alignment $\widetilde{\mathbf{H}}^{(\ell)} = \tanh(\mathbf{H}^{(\ell)}\mathbf{T} + \mathbf{b})$ enforcing the B-spline domain constraint $[-1, 1]^d$, the fully vectorised cross-layer attention is:

$$\begin{aligned} \mathbf{Q}^{(\ell)} &= \mathrm{KAN}^{(Q)}\big(\mathrm{LN}(\widetilde{\mathbf{H}}^{(\ell)})\big), \\ \mathbf{K}^{(\ell)}, \mathbf{V}^{(\ell)} &= \mathrm{KAN}^{(KV)}\big(\mathrm{LN}(\mathbf{C}^{(-\ell)})\big), \\ \mathbf{Z}_{\mathrm{attn}}^{(\ell)} &= \mathrm{softmax}\left(\frac{\mathbf{Q}^{(\ell)}\big(\mathbf{K}^{(\ell)}\big)^\top}{\sqrt{d_k}}\right)\mathbf{V}^{(\ell)}, \end{aligned} \tag{7.5}$$

where $\mathbf{C}^{(-\ell)} \in \mathbb{R}^{N\times(L-1)\times d}$ is the stacked context from all layers except ℓ, and $\mathrm{LN}(\cdot)$ denotes Layer Normalisation applied before each B-spline projection for training stability. The fused layer embedding is $\mathbf{Z}^{(\ell)} = \widetilde{\mathbf{H}}^{(\ell)} + \mathbf{F}^{(\ell)}$, where $\mathbf{F}^{(\ell)}$ concatenates and linearly projects all attention-head outputs.

Orthogonal Chebyshev-KAN Q-Head

The decoder replaces B-splines – which suffer Runge-type boundary oscillations under non-stationary TD targets – with orthogonal Chebyshev polynomials of the first kind:

$$y_j = \sum_{i=1}^{d_{\text{in}}} \sum_{m=0}^{K} c_{i,j,m}\, T_m\big(\tanh(x_i)\big) + b_j, \qquad T_0 = 1,\;\; T_1(x) = x,\;\; T_m(x) = 2x\,T_{m-1}(x) - T_{m-2}(x). \tag{7.6}$$

The near-minimax bound of Theorem 2 (Chapter 5) states $\|y_t - S_K[y_t]\|_\infty \le \big(1 + \frac{2}{\pi}\ln(K+1) + O(K^{-2})\big)E_K(y_t)$, and Corollary 3 bounds the gradient norm by a factor growing only as $O(\log K)$ – contrasting with the exponential Lebesgue constant $\Lambda_K = \Theta(2^K)$ of equispaced polynomial interpolation. A graph Laplacian regulariser $\mathcal{L}_R(\Theta) = \sum_\ell |\mathcal{E}^{(\ell)}|^{-1} \sum_{i,j} A_{ij}^{(\ell)} \|\mathbf{h}_i - \mathbf{h}_j\|_2^2$ prevents the high-capacity KAN encoder from producing structurally incoherent embeddings in sparse layers.

Theoretical Contributions

KARL establishes three formal theoretical results. The gradient stability bound of Proposition 1 justifies the macroscopic residual branch in the encoder. The near-minimax approximation guarantee of Theorem 2 and the logarithmic gradient-norm bound of Corollary 3 justify the Chebyshev decoder over B-splines. The KAN-GNN subsumption proof of Theorem 4 establishes that the KAN-GNN function class contains the MLP-GNN class as a dense subset under uniform convergence at any fixed depth and embedding dimension; and the parameter-efficiency separation of Proposition 7 confirms, via a weight-free lower bound, that KARL's gains are attributable to functional form rather than parameter scale. Together these results provide end-to-end theoretical support for every architectural design choice in KARL.

Empirical Results and Interpretability

Evaluated on six real-world undirected multiplex networks spanning 279–56,562 nodes, KARL achieved the lowest AUDC and lowest C^* on all six benchmarks, with an average AUDC improvement of **21.96%** over MultiDismantler, reaching **27.27%** on Sanremo2016 ($N = 56{,}562$) despite being trained exclusively on synthetic graphs with $N \in [30, 50]$. Notably, KARL exhibits emergent structural interpretability: without any auxiliary supervision, its Chebyshev polynomial activations correlate monotonically with node k-coreness and concentrate on *low-degree* high-coreness nodes – precisely those structurally subtle bottleneck nodes that sustain mutual connectivity across layers without appearing prominent in any single-layer degree ranking.

7.1.3 Contribution III: Comparative Analysis and Synthesis

In future, we want to combine these two contributions within a unified architectural taxonomy and established that DDIN and KARL address *orthogonal* failure modes: DDIN resolves the topological limitation through asymmetric GNN design for directed networks, while KARL resolves the functional limitation through KAN-based representation learning on undirected networks. Neither contribution subsumes the other; their design strategies are complementary, directly motivating the grand synthesis of Section 6.2.4. The comparative analysis further quantified the expressivity–directionality trade-off: DDIN retains MLP-based action-value heads while gaining directed topological encoding, whereas KARL employs KAN-based encoders and decoders while operating exclusively on undirected topologies.

7.2 Ethical Considerations & Dual-Use

Network dismantling technology occupies an acute position within the dual-use research landscape of artificial intelligence. The same algorithmic machinery that identifies the minimal set of nodes whose removal fragments a complex network into non-extensive components can be deployed either *constructively* – to harden critical infrastructure, design optimal immunisation strategies, and disrupt illicit syndicates – or *adversarially* – to plan deliberate attacks on communication networks, power grids, or financial systems. The 2003 Italian blackout [11] and the supply-chain disruptions during the COVID-19 pandemic are empirical reminders of the catastrophic potential of cascading failures in real interdependent systems.

The contributions of this thesis amplify the dual-use concern along two independent axes. First, DDIN specifically models directed asymmetric flows, providing actionable intelligence about the source-sink hierarchy in supply chains, financial transaction networks, and biological signalling cascades. Second, KARL's emergent structural interpretability – the automatic identification of high-coreness low-degree bottleneck nodes without auxiliary supervision – constitutes a form of automated vulnerability analysis deployable at national infrastructure scale ($N = 56{,}562$ nodes, zero-shot).

Constructive Applications

Epidemic containment and immunisation design. Network dismantling directly solves the optimal immunisation problem: identifying the minimal set of individuals whose immunisation prevents a macroscopic outbreak [43]. DDIN's directed formulation is particularly relevant for social media information cascades where influence propagation is asymmetric. KARL's 21.96% AUDC improvement over MultiDismantler implies a correspondingly tighter immunisation frontier – fewer individuals vaccinated to achieve the same suppression of epidemic spread.

Infrastructure resilience engineering. Identifying the minimal node set whose removal would collapse an interdependent infrastructure system (e.g., a coupled power-grid/communication network) enables engineers to harden precisely those nodes against targeted attacks or random failures [60]. Both DDIN and KARL provide strictly superior

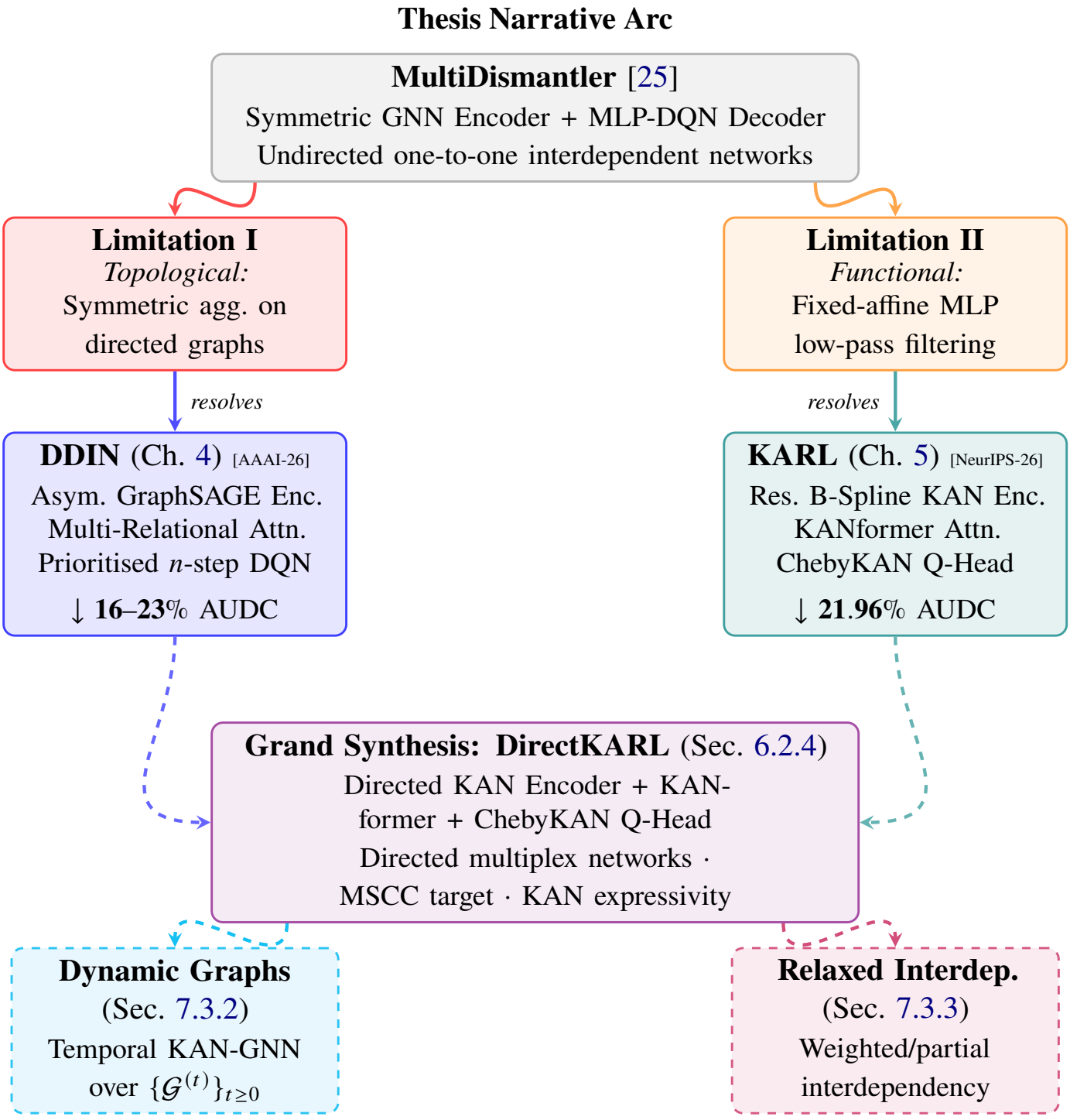


Figure 7.1: Thesis narrative arc showing the orthogonal contributions and future synthesis directions. MultiDismantler [25] exhibits two structural limitations. **DDIN** (Chapter 4, AAAI-26) resolves the topological limitation through asymmetric GNN design for directed networks. **KARL** (Chapter 5) resolves the functional limitation through KAN-based representation learning on undirected networks. The **Grand Synthesis** (Section 6.2.4) represents the natural future convergence. Dashed arrows indicate future work directions.

identification of these critical bottlenecks compared to the current state of the art.

Drug target identification. In protein-protein interaction networks and directed signalling cascades, the dismantling problem maps directly to the minimal enzyme inhibition set for a disease-relevant pathway [4]. DDIN's directed formulation is well-suited to this application, where biochemical reactions are inherently directional.

Counter-narcotics and counter-terrorism. Dismantling covert social networks requires identifying the minimal set of actors whose removal fragments the network into operationally incoherent subgraphs. DDIN's directed formulation captures command hierarchies and information channels that are fundamentally asymmetric and cannot be accurately modelled by undirected methods.

Adversarial Misuse and Mitigation

The principal adversarial concern is that a malicious actor could use DDIN or KARL to plan targeted attacks on critical infrastructure. KARL's zero-shot generalisation at $N = 56{,}562$ means such analyses could be performed at national scale using publicly available network topology data. Responsible deployment requires, at minimum, three institutional safeguards.

Restricted model weight access. Trained DDIN and KARL agents should not be released as open-weight artefacts without institutional access controls, analogous to protocols applied to dual-use biological sequence models. The code associated with this thesis will be made available under a research-only licence explicitly prohibiting adversarial applications.

Differential privacy on sensitive input graphs. When applying dismantling agents to sensitive infrastructure topology data, differential privacy mechanisms should be applied to prevent inference of exact network structure from algorithmic outputs.

Institutional review. Applications of dismantling technology to real-world critical infrastructure should be subject to the same institutional review processes applied to penetration testing and vulnerability disclosure programmes, with results treated as sensitive security findings.

This thesis advocates for the strictly defensive application of its algorithmic contributions to enhance systemic robustness, mitigate cascading failures, and protect critical infrastructure.

7.3 Future Directions

The contributions of this dissertation open three well-defined and technically demanding research frontiers. Each is described below with a formal problem statement and technical challenge analysis.

7.3.1 The Grand Synthesis: Directed KAN-RL Dismantling

Motivation. The comparative analysis of Chapter 6 established that DDIN and KARL are architecturally complementary. DDIN introduces directed asymmetry into the GNN encoder but retains MLP-based action-value projections, while KARL replaces MLP projections with KAN-based ones but operates exclusively on undirected graphs. A natural and compelling synthesis – *DirectKARL* – would combine both innovations into a unified architecture simultaneously modelling directed asymmetric flows via asymmetric KAN message-passing

and capturing non-smooth cross-layer coupling via a fully KAN-parameterised attention module, with a Chebyshev Q-head targeting the MSCC rather than the LMCC.

Formal Architecture. The DirectKARL architecture must satisfy three design requirements.

(D1) Directed KAN Encoder. Replace the MLP projections in DDIN's asymmetric GraphSAGE encoder

$$\begin{aligned}\mathbf{h}_v^{(l+1)} \;=\; \mathrm{KAN}_{\mathrm{out}}\Big[\; & \mathrm{KAN}^{\mathrm{in}}_{\mathrm{neigh}}\big(\tanh(\mathrm{MeanAgg}_{\{u\to v\}}(\mathbf{h}^{(l)}))\big) \;\Big\|\\ & \mathrm{KAN}^{\mathrm{out}}_{\mathrm{neigh}}\big(\tanh(\mathrm{MaxAgg}_{\{v\to u\}}(\mathbf{h}^{(l)}))\big) \;\Big\|\\ & \mathrm{KAN}_{\mathrm{self}}\big(\tanh(\mathbf{h}_v^{(l)})\big)\;\Big]\\ & +\; \alpha_{\mathrm{res}}\cdot \mathrm{MLP}_{\mathrm{res}}(\mathbf{h}_v^{(l)}),\end{aligned} \tag{7.7}$$

where the three-stream concatenation yields a $3d$-dimensional input to $\mathrm{KAN}_{\mathrm{out}}$, requiring weight matrices $\boldsymbol{W}_{\mathrm{out}} \in \mathbb{R}^{d\times 3d}$. Separate KAN projections $\mathrm{KAN}^{\mathrm{in}}_{\mathrm{neigh}}$ and $\mathrm{KAN}^{\mathrm{out}}_{\mathrm{neigh}}$ are applied independently to the incoming and outgoing neighbourhood aggregations, respectively.

(D2) Directed KANformer. The inter-layer attention mechanism must account for asymmetric failure propagation. The KANformer cross-attention weights should produce asymmetric layer-importance coefficients, retaining the concatenation-based asymmetry of DDIN's Multi-Relational Attention (Equation 7.3) while replacing affine Q/K/V projections with B-spline KANs as in Equation 7.5.

(D3) MSCC-Aware Reward. Following the established convention of MultiDismantler [25] and KARL, the reward for removing node v at step t is defined multiplicatively to jointly incentivise large component reductions at low cost:

$$r_t(v) \;=\; P_{\mathrm{MSCC}}(s_{t+1})\cdot F(\{v_{a_t}\}), \qquad P_{\mathrm{MSCC}}(s) \;=\; \frac{|\mathrm{MSCC}(s)|}{N}, \tag{7.8}$$

where $F(\{v_{a_t}\})$ is the per-node removal cost (unit or degree regime), and $P_{\mathrm{MSCC}}(s_{t+1})$ is evaluated after cascade resolution via Tarjan's SCC algorithm [55].

Technical Challenges. The three-stream directed KAN encoder introduces a $3d$-dimensional concatenation whose B-spline parameter count grows by approximately $3d^2(G+k)$ per encoder layer relative to KARL's two-stream encoder. For $d = 64$, $G = 10$, $k = 3$, this adds approximately 172,032 parameters per layer, bringing the total to approximately 500k–800k trainable parameters – within the A100 GPU memory budget for the evaluated network scales.

A further theoretical challenge concerns the extension of the universal approximation guarantee of Theorem 4 (Chapter 5) to the directed setting. The proof of Theorem 4 exploits the symmetry of the unweighted sum aggregator; the directed MeanAgg/MaxAgg scheme of DDIN breaks this symmetry, and a separate approximation-theoretic analysis will be required. Specifically, the ℓ_2-isometry of the concatenation operator must be rederived for the three-stream case, and the Lipschitz tracking through the directed GNN must account for

the $\sqrt{|N^+(v)|}$-Lipschitz constant of MaxAgg (versus the $\sqrt{|N(v)|}$-Lipschitz of the symmetric aggregator) – a modification that alters the constant factors in the per-iteration injected error bound but preserves the inductive proof structure.

7.3.2 Dynamic and Temporal Multiplex Graphs

Motivation. Both DDIN and KARL, in common with all prior multiplex dismantling work, assume a *static* network topology fixed throughout the dismantling episode. This assumption is violated in virtually every real-world application: supply chains evolve as trade agreements shift; social networks rewire as users join and leave; biological signalling networks respond dynamically to pharmaceutical interventions. A dismantling agent trained on static topologies may identify nodes critical at training time that become peripheral as the network evolves.

Formal Problem Statement. Let $\{\mathcal{G}^{(t)}\}_{t\geq 0}$ denote a temporal sequence of multiplex snapshots with edge set $\mathcal{E}^{(t)}$ evolving under a stochastic transition kernel $P(\mathcal{E}^{(t+1)}|\mathcal{E}^{(t)})$. The *dynamic dismantling problem* requires identifying a removal sequence that efficiently fragments the LMCC (or MSCC) of the expected future network at horizon Δ, subject to the constraint that removals are committed at time t without knowledge of the future topology:

$$\min_{\tilde{S}\subseteq\mathcal{V}^{(t)}} |\tilde{S}| \quad \text{subject to} \quad \mathbb{E}_{\mathcal{G}^{(t+\Delta)}}\left[F\left(\tilde{S};\, \mathcal{G}^{(t+\Delta)}\right)\right] < \theta, \tag{7.9}$$

where F denotes the AUDC evaluated under the dynamic topology, inheriting the metric definition of Chapter 3. This is a chance-constrained combinatorial optimisation problem under distribution shift, strictly harder than the static variant.

Technical Approach and Challenges. A promising direction is to augment the KAN-GNN encoder with a temporal graph neural network component – such as a Temporal Graph Transformer or a recurrent architecture operating over the adjacency sequence $\{\mathbf{A}^{(\ell,t)}\}_t$ – to produce embeddings capturing the temporal trend in node connectivity. The B-spline KAN activations in KARL are well-suited to this extension: their learnable frequency bandwidth can, in principle, capture the periodic or quasi-periodic patterns common in temporal graphs (weekly cycles in social networks, seasonal patterns in trade networks) that fixed-affine MLP layers would require substantially deeper architectures to approximate.

The key theoretical challenge is whether the gradient stability guarantee of Proposition 1 (Chapter 5) extends to the recurrent setting, where the Lipschitz constant of the composed temporal-spatial message-passing operator may grow with sequence length. Residual connections through both the spatial and temporal dimensions – analogous to those in the KARL encoder – will likely be necessary. For evaluation, temporal snapshots can be constructed from existing multiplex datasets by randomly rewiring a fraction p_{rew} of edges between consecutive snapshots; performance as a function of p_{rew} then quantifies the agent's robustness to topology evolution.

7.3.3 Relaxing Interdependency Assumptions

Motivation. Both DDIN and KARL adopt the strict one-to-one interdependency model of Buldyrev et al. [11]: removing node v_i from any layer simultaneously removes all its replicas

across every other layer. In real interdependent systems, interdependency relationships are typically partial, weighted, or probabilistic. A power substation may depend on *some* but not *all* of its communication network counterparts; a protein in one signalling layer may only partially depend on its regulatory counterpart in another.

Weighted Interdependency Generalisation. The one-to-one model is generalised by a weighted interdependency tensor $\mathbf{W} \in [0,1]^{|\mathcal{V}|\times|\mathcal{V}|\times|\mathcal{L}|\times|\mathcal{L}|}$, where $w_{ij}^{(\ell\ell')} \in [0,1]$ denotes the probability that node v_i's failure in layer ℓ causes v_j's failure in layer ℓ'. The cascading failure process then becomes stochastic, and the dismantling objective is the chance-constrained problem:

$$\min_{\tilde{S}\subseteq\mathcal{V}} |\tilde{S}| \quad \text{subject to} \quad \Pr\big[\mathrm{AUDC}(\tilde{S};\, \mathcal{G},\, \mathbf{W}) < \theta\big] \;\geq\; 1-\delta, \tag{7.10}$$

where $\mathbf{W}$ is the weighted interdependency tensor defined above, $\mathrm{AUDC}(\tilde{S};\, \mathcal{G},\, \mathbf{W})$ inherits the metric formulation of Chapter 3 applied to the stochastic cascade process, and $\delta \in (0,1)$ is a prescribed failure tolerance.

Technical Challenges and Approach. The primary challenge is that the stochastic cascading process renders the reward signal $r_t = P_C(s_{t+1}) \cdot F(\{v_{a_t}\})$ a random variable with potentially high variance, destabilising the TD-RL training loop. One approach is to use Monte Carlo estimates of the expected reward over multiple cascade simulations per step – unbiased but sample-intensive. An alternative is a differentiable mean-field approximation to the expected component size under the weighted interdependency model, analogous to the mean-field percolation theory of Osat et al. [44], used as the training-time reward with exact Monte Carlo evaluation at test time.

The KAN architecture of KARL offers a structural advantage here: the learnable B-spline activations can fit the non-smooth mapping from the interdependency parameters $\{w_{ij}^{(\ell\ell')}\}$ to the expected component size, whereas a fixed-affine MLP would require substantially greater depth. The KAN-GNN subsumption guarantee of Theorem 4 (Chapter 5) ensures that the KAN encoder can represent whatever function the MLP could, and more – suggesting the performance advantage of KARL over MultiDismantler would be preserved, and potentially amplified, in the stochastic dismantling setting.

Many-to-many interdependencies. A further generalisation is to move beyond one-to-one correspondences to *many-to-many* interdependency structures, where a single node in one layer may depend on a set of nodes in another. This structure arises naturally in multi-modal transportation networks and multi-omics biological networks. The GNN encoder in both DDIN and KARL must be extended to handle this hyper-relational structure, potentially via a KAN-parameterised hypergraph neural network aggregating information across bipartite inter-layer representations.

Closing Remarks

The problem of network dismantling – finding the minimal set of nodes whose removal fragments a complex system into operationally incoherent components – sits at the intersection of combinatorial optimisation, statistical physics, and machine learning. The contributions

of this thesis represent two complementary steps toward a general-purpose dismantling agent that is simultaneously architecturally honest about the directional structure of real-world networks and representationally expressive enough to identify the subtle, non-obvious bottleneck nodes that determine cascading fragility.

DDIN established that the topological assumption of symmetry is not merely a simplification but a structural error that systematically misidentifies source and sink nodes in directed systems – and that correcting it through asymmetric GNN design yields 16–23% AUDC reductions on real directed multiplex networks spanning biological, economic, and social domains. KARL established that the functional assumption of fixed-affine MLP projections imposes a hard expressivity ceiling – and that replacing MLPs with learnable B-spline KAN activations, supported by the gradient stability bound of Proposition 1, the near-minimax guarantee of Theorem 2, and the KAN-GNN subsumption proof of Theorem 4, yields 21.96% average AUDC reduction with emergent structural interpretability unavailable to any MLP-based agent.

The journey from the symmetric, MLP-based MultiDismantler to the asymmetric, KAN-powered DirectKARL articulated in Section 7.3.1 mirrors a broader trajectory in the machine learning community: from architectures designed for mathematical convenience toward architectures designed for structural fidelity to the problem domain. The formal guarantees established in this thesis are intended to provide a durable mathematical foundation for this trajectory, independent of the specific empirical benchmarks on which the algorithms were evaluated.

Network science has long taught that the most critical nodes in a complex system are often not the most visible ones: the subtle bottleneck sustaining global connectivity may lie hidden within the middle shells of the k-core decomposition, anchoring the mutual connectivity of the entire interdependent system without appearing prominent in any single-layer degree ranking. KARL's emergent identification of precisely these nodes without auxiliary supervision – and DDIN's systematic prioritisation of asymmetric flow sources over topologically equivalent sinks – suggest that the path toward truly faithful dismantling models lies not in ever-wider MLP stacks or ever-deeper graph convolutions, but in architectural designs that encode the structural semantics of the problem itself: directionality, functional expressivity, and the rich non-smooth landscape of cascading vulnerability.

Appendix A

Hyperparameter Configurations

This appendix provides complete, self-contained hyperparameter specifications for both DDIN (Chapter 4) and KARL (Chapter 5). Each table is the single authoritative record for its framework; values reported in the main chapters derive from these tables and are reproduced here for consolidated reference. The *Determined by* column classifies each scalar as either:

- **Fixed** — set by direct analogy with the MultiDismantler [25] and FINDER [23] baselines, or by standard deep-RL convention, and held constant throughout all experiments; or
- **Ablation (§ X.Y)** — selected by a controlled empirical sweep described in the indicated section, with all other parameters frozen.

Two additional tables (Tables A.2 and A.4) enumerate the trainable parameter count per architectural component, providing a complete accounting of model capacity for each framework. A final comparative summary (Table A.6) places both configurations side-by-side on the shared dimensions.

A.1 DDIN: Disassembling Directed Interdependent Networks

DDIN is an MLP-parameterised, directed-asymmetric GNN-DQN agent. It uses a single uniform learning rate (no spline coefficients), a hard periodic target-network update, and a discount factor of $\gamma = 1$ (undiscounted returns), following the convention of prior multiplex dismantling work [23,25] in which the episode length is short enough that the bias introduced by discounting outweighs its variance-reduction benefit.

A.1.1 Complete Hyperparameter Table

Table A.1 reports every hyperparameter of DDIN, grouped into four categories: architecture, RL training, prioritised experience replay buffer, and synthetic training corpus.

A.1.2 Trainable Parameter Breakdown

Table A.2 enumerates the trainable parameters of the DDIN architecture for the default setting $d = 64$, $d_0 = 3$, $K_{\max} = 3$. The asymmetric encoder is by far the largest component, reflecting the three-round, two-branch design.

A.1.3 Exploration Schedule and Training Dynamics

DDIN employs a linear ε-greedy decay from $\varepsilon_0 = 1.0$ to $\varepsilon_{\min} = 0.01$ over the full $N_{\text{ep}} = 20{,}000$ training episodes. At global step t_{total} within the training run, the exploration probability is

$$\varepsilon(t_{\text{total}}) \;=\; \max\!\left(\varepsilon_{\min},\; \varepsilon_0 - \frac{(\varepsilon_0 - \varepsilon_{\min})}{T_{\text{anneal}}} \cdot t_{\text{total}}\right), \qquad T_{\text{anneal}} = N_{\text{ep}} \cdot \bar{N}, \tag{A.1}$$

where $\bar{N} = 40$ is the mean episode length (the average number of node removals required to reach the dismantling threshold on the training graphs).

Table A.1: Complete DDIN hyperparameter configuration, fixed identically across all five evaluation datasets. All values are held constant throughout training unless otherwise noted. d: embedding dimension; d_0: initial node-feature dimension; $K_{\max}$: message-passing rounds; ε_{PER}: priority noise floor; α_{PER}: PER exponent; β_{IS}: importance-sampling exponent.

Category	Hyperparameter	Value	Determined by
Architecture	Embedding dimension d	64	Fixed
	Initial feature dimension d_0	3	Fixed
	Message-passing rounds $K_{\max}$	3	Fixed
	Reconstruction loss weight λ	1	Fixed
	Weight decay λ_{wd}	10^{-5}	Fixed
RL Training	Discount factor γ	1.0	Fixed
	n-step return window n	5	Fixed
	Initial exploration ε_0	1.0	Fixed
	Final exploration $\varepsilon_{\min}$	0.01	Fixed
	Target-network update period τ	100 steps	Fixed
	Mini-batch size B	32	Fixed
	Learning rate η	10^{-4}	Fixed
	Optimiser	Adam [9]	Fixed
PER Buffer	Priority exponent α_{PER}	0.6	Fixed [50]
	Priority noise floor ε_{PER}	10^{-6}	Fixed [50]
	IS weight β_{IS} (annealed)	$0.4 \rightarrow 1.0$	Fixed [50]
	Buffer capacity $\lvert\mathcal{B}\rvert$	10^5	Fixed
D-GMM Corpus	Training graphs N_{ep}	20,000	Fixed
	Graph size N	[30, 50] (uniform)	Fixed
	Power-law exponent $\gamma_{\text{out}} = \gamma_{\text{in}}$	2.5	Fixed [32]
	Mean degree $\bar{k}$	6	Fixed [32]
	Directional bias α	[0, 1] (uniform)	Fixed
	Interlayer correlation g	[0, 1] (uniform)	Fixed
Hardware	GPU	NVIDIA A100	—
	Evaluation runs (averaged)	5	Fixed

Table A.2: Trainable parameter breakdown for DDIN at $d = 64$, $d_0 = 3$, $K_{\max} = 3$. Counts marked with $\star$ scale with the L layers of the evaluation network and are reported per layer. $\boldsymbol{W}_{\text{out}}^{(k)}, \boldsymbol{W}_{\text{in}}^{(k)} \in \mathbb{R}^{d\times 2d}$; $\boldsymbol{W} \in \mathbb{R}^{d\times d}$; $\mathbf{a} \in \mathbb{R}^{2d}$; $\boldsymbol{W}_{\text{dec}} \in \mathbb{R}^{d\times d}$.

Stage	Component	Parameters
Stage 1: Asymmetric GraphSAGE Encoder	Initial projection $\boldsymbol{W}_0$ $(d \times d_0)$	192
	Outgoing aggregation $\boldsymbol{W}_{\text{out}}^{(k)}$ $(d \times 2d, \times K_{\max}$ rounds)	24,576
	Incoming aggregation $\boldsymbol{W}_{\text{in}}^{(k)}$ $(d \times 2d, \times K_{\max}$ rounds)	24,576
Stage 2: Multi-Relational Attention	Shared projection $\boldsymbol{W}$ $(d \times d)$	4,096
	Attention vector $\mathbf{a}$ $(2d)$	128
Stage 3: Bilinear Decoder & Q-Value Head	Per-layer projection $\boldsymbol{P}^{(\ell)}$ $(d \times d$, per layer$)^\star$	4,096*
	Bilinear weight $\boldsymbol{W}_{\text{dec}}$ $(d \times d)$	4,096
	Layer bias $b^{(\ell)}$ (scalar, per layer)*	1*
	Layer-importance weights $\boldsymbol{M}_1, \boldsymbol{M}_2$ $(O(d^2))$	8,320
	Layer-importance weights $\boldsymbol{M}_3, \boldsymbol{M}_4$ $(O(d^2))$	4,160
Total (excluding per-layer terms)		**70,144**

PER priorities are initialised to $\delta_{\max} = 1.0$ for the first B transitions to ensure uniform sampling during the cold start, where $B = 32$ is the mini-batch size. Importance-sampling weights are computed as

$$w_i = \left(\frac{1}{|\mathcal{B}| \cdot P(i)}\right)^{\beta_{\text{IS}}}, \qquad P(i) = \frac{p_i^{\alpha_{\text{PER}}}}{\sum_j p_j^{\alpha_{\text{PER}}}}, \tag{A.2}$$

normalised by $\max_j w_j$ to ensure they are bounded in $[0, 1]$. The annealing of β_{IS} from 0.4 to 1.0 follows the schedule of Schaul et al. [50], linearly increasing the bias-correction strength as the policy converges. The priority noise floor $\varepsilon_{\text{PER}} = 10^{-6}$ prevents zero-probability sampling of any transition, ensuring the entire buffer is accessible throughout training.

A.2 KARL: Kolmogorov-Arnold Reinforcement Learning

KARL is a KAN-parameterised multiplex GNN-DQN agent operating on undirected multiplex networks. It introduces three architectural components absent in DDIN: (i) B-spline KAN layers in the encoder and cross-layer attention, requiring a second, higher learning rate for the spline coefficients; (ii) a macroscopic residual branch with a learnable scalar α_{res}; and (iii) a graph Laplacian smoothness regulariser $\mathcal{L}_R$ weighted by $\lambda = 10^{-3}$. Three of the seven architectural hyperparameters were selected by controlled ablation sweeps (Section 5.6); the remaining four are fixed by analogy with MultiDismantler [25].

A.2.1 Complete Hyperparameter Table

Table A.3 reports every hyperparameter of KARL. Ablation-derived values include the section reference where the sweep is documented.

Table A.3: Complete KARL hyperparameter configuration. Ablation-derived values reference the section in Chapter 5 where the sweep is documented. d: embedding dimension; G: B-spline grid resolution; k: B-spline order; K: Chebyshev polynomial degree; h: attention heads; $l_{\max}$: message-passing rounds; α_{res}: residual scalar (initialised; subsequently learnable); λ: Laplacian regularisation weight; α_{PER}: PER priority exponent; β_{IS}: IS weight exponent.

Category	Hyperparameter	Value	Determined by
Architecture	Embedding dimension d	64	Fixed
	Input feature dimension d_{in}	2	Fixed
	B-spline grid resolution G	10	Ablation (§5.6.2)
	B-spline order k	3	Ablation (§5.6.2)
	Chebyshev degree K	4	Ablation (§5.6.2)
	Attention heads h	4	Fixed
	Message-passing rounds $l_{\max}$	3	Fixed
	Residual scalar α_{res} (init.)	0.5	Ablation (§5.6.1)
Optimisation	Training episodes	20,000	Fixed
	n-step return depth n	5	Fixed
	Discount factor γ_{disc}	0.99	Fixed
	Target-network update interval τ	1,000 steps	Fixed
	PER exponent α_{PER}	0.6	Fixed [50]
	IS weight β_{IS} (annealed)	$0.4 \rightarrow 1.0$	Fixed [50]
	Laplacian reg. weight λ	10^{-3}	Fixed
	Weight decay λ_{wd}	10^{-5}	Fixed
	Learning rate η (linear params.)	10^{-4}	Fixed
	Learning rate η (spline coefficients)	10^{-3}	Fixed
	Optimiser	Adam (decoupled) [9]	Fixed
	ε-greedy (init. $\rightarrow$ final)	$1.0 \rightarrow 0.05$	Fixed
GMM Corpus	Training graphs N_{ep}	20,000	Fixed
	Graph size N	[30, 50] (uniform)	Fixed
	Power-law exponent γ	2.5	Fixed [32]
	Mean degree $\bar{k}$	6	Fixed [32]
	Interlayer correlation g	[0, 1] (uniform)	Fixed
Hardware	GPU	NVIDIA A100 80 GB PCIe	—
	Training wall time (1 seed)	≈ 4 h	—

A.2.2 Decoupled Learning Rates: Justification

The Adam optimiser in KARL maintains *two independent learning-rate schedules*:

$$\eta_{\text{linear}} = 10^{-4} \quad \text{for all weight matrices } (\boldsymbol{W}_{\text{base}},\ \boldsymbol{W}_{\text{spline}},\ \text{attention projections, Q-head biases}), \tag{A.3}$$

$$\eta_{\text{spline}} = 10^{-3} \quad \text{for all B-spline coefficient vectors } \{c_i\} \text{ in every KAN layer.} \tag{A.4}$$

The ten-fold higher learning rate for spline coefficients is necessitated by the local, piecewise nature of B-spline basis functions: individual knot-interval coefficients affect only a narrow input range, so their effective gradient magnitude is structurally smaller than that of global linear weights. Without the higher rate, spline coefficients remain near their initialisation throughout training, collapsing the KAN to its base activation $b(x) = \text{SiLU}(x)$ and negating the expressivity advantage over an MLP.

A.2.3 Residual Scalar Initialisation and Ablation

The residual scalar α_{res} is initialised to 0.5 and subsequently treated as a learnable parameter updated by gradient descent alongside all other parameters. The initialisation value was determined by a controlled sweep over $\alpha_{\text{res}} \in \{0.0, 0.1, 0.5, 1.0\}$ plus a no-MLP variant, detailed in Section 5.6.1. The full node-embedding update at message-passing round l is

$$\mathbf{h}_v^{(l+1)} = \text{KAN}_{\text{out}}\Big(\Big[\text{KAN}_{\text{neigh}}\big(\tanh(\text{Agg}(\mathbf{h}^{(l)}))\big) \,\|\, \text{KAN}_{\text{self}}\big(\tanh(\mathbf{h}_v^{(l)})\big)\Big]\Big) + \alpha_{\text{res}} \cdot \text{MLP}_{\text{res}}(\mathbf{h}_v^{(l)}), \tag{A.5}$$

followed by ℓ_2-normalisation of $\mathbf{h}_v^{(l+1)}$. The ablation showed that:

- $\alpha_{\text{res}} = 0.0$: gradient spikes reaching $\|\nabla\|_2 \sim 10^2$ in early training, destabilising Q-value estimates;
- $\alpha_{\text{res}} = 1.0$: B-spline gradient suppressed, slowing policy convergence by 40–60 episodes;
- $\alpha_{\text{res}} = 0.5$: smoothest gradient dynamics and lowest final AUDC (≈0.16), achieving 12–18% reduction over the No-MLP variant;
- No-MLP ($\alpha_{\text{res}} = 0$ with MLP_{res} removed): highest AUDC variance across seeds, confirming the residual branch is a prerequisite for stable backpropagation through non-stationary B-spline grids in deep RL.

A.2.4 Trainable Parameter Breakdown

Table A.4 gives the complete trainable parameter count per architectural component for KARL and, for comparison, MultiDismantler [25] (MD), at $d = 64$, $G = 10$, $k = 3$, $K = 4$. The B-spline encoder accounts for 36,864 + 36,864 + 73,728 = 147,456 parameters, a 14× per-layer overhead over the plain linear map (which carries $d^2 = 4{,}096$ parameters). Each EFFICIENTKANLAYER of dimension $d \times d$ stores $d^2(G+k+1) = 64^2 \times 14 = 57{,}344$ parameters at $G = 10$, $k = 3$, versus $d^2 = 4{,}096$ for a plain linear map. The KANformer dominates KARL's budget at 151,744 parameters (47% of total), because every Q/K/V projection in the cross-layer attention mechanism is replaced by an independent B-spline KAN. The

Table A.4: Trainable parameter breakdown for KARL and MultiDismantler (MD) [25] at $d = 64$, $G = 10$, $k = 3$, $K = 4$. Shaded rows are components present in both models with identical parameter counts. Ratios are KARL/MD.

Component	KARL	MD	Ratio
Stage 1 — Intra-layer message-passing encoder			
$\text{KAN}_{\text{neigh}}$ ($d \to d$, B-spline)	36,864	4,096	9.0×
KAN_{self} ($d \to d$, B-spline)	36,864	4,096	9.0×
KAN_{out} ($2d \to d$, B-spline)	73,728	8,192	9.0×
Residual MLP branch (linear-RELU)	4,160	—	—
Residual scalar α_{res} (learnable scalar)	1	—	—
Stage 2 — Cross-layer fusion			
KANformer / Bitwise-multiply-logis.	151,744	4,225	35.9×
Stage 3 — Action-value decoder			
$\text{ChebyKAN}_{\text{hidden}}$ ($d \to 32$, $K = 4$)	10,272	—	—
$\text{ChebyKAN}_{\text{final}}$ ($36 \to 1$, $K = 4$)	181	—	—
MLP hidden h_1 ($d \to 32$, plain linear-RELU)	—	2,048	—
MLP output h_2 ($36 \to 1$)	—	36	—
Shared components (identical in both models)			
Input projection $\boldsymbol{W}_{n2l}$ ($2 \to d$)	128	128	1.0×
Cross-product vector ($d \times 1$)	64	64	1.0×
Softmax weights $\boldsymbol{W}_1 + \boldsymbol{W}_2$ ($d \to 128 \to 1$)	8,320	8,320	1.0×
Total trainable parameters	**322,326**	**31,205**	**10.3×**

Chebyshev decoder contributes only 10,453 parameters (3.2% of total), confirming that the orthogonality benefit of the Chebyshev basis is obtained at negligible parameter cost.

A.2.5 Inference Latency and Peak GPU Memory

Table A.5 reports per-step inference time and peak GPU memory for KARL and MultiDismantler across all six real-world benchmarks. All measurements were taken on a single NVIDIA A100 80 GB PCIe GPU with PyTorch 2.1.2+cu118. Inference time is averaged over all dismantling steps in the trajectory using CUDA Events (sub-millisecond precision); peak memory is documented via `torch.cuda.max_memory_allocated`, reset before each dataset.

The non-monotonic slowdown ratio (ranging from 1.54× on Homo Genetic to 3.08× on Sacchpomb despite its smaller size) is explained by active-subgraph edge density at each dismantling step: the B-spline basis tensor $B(\mathbf{x}) \in \mathbb{R}^{N_{\text{active}} \times d_{\text{in}}(G+k)}$ must be materialised at every forward pass, and denser active subgraphs incur proportionally larger basis evaluations per step. At $N = 56{,}562$ (Sanremo2016), KARL consumes 16.6 GB of GPU memory—well within the A100's 80 GB capacity. Total training wall time for KARL is approximately 4 h per seed on one A100, versus approximately 2 h for MultiDismantler.

A.3 Comparative Configuration Summary

Table A.6 presents a comparison of the key configuration choices for DDIN and KARL, highlighting the four primary differences arising from their distinct architectural designs.

Table A.5: Per-step inference time and peak GPU memory for KARL and MultiDismantler (MD) across six real-world benchmarks. Measurements: single NVIDIA A100 80 GB PCIe, PyTorch 2.1.2+cu118. *Geom. mean ratio* is computed across all six datasets. Lower is better for all columns.

			Inference (ms/step)		**Peak GPU mem (MB)**	
Dataset	N	L	**KARL**	**MD**	**KARL**	**MD**
FAO Trade	214	364	64.2	36.6	108.3	27.9
C. elegans	279	3	39.6	21.0	139.3	30.9
Fb-Tw	1,043	2	84.1	54.6	497.0	77.2
Sacchpomb	4,092	7	648.3	210.4	1,068.0	172.9
Homo Genetic	18,222	7	936.7	609.2	4,793.7	1,146.5
Sanremo2016	56,562	3	2,871.3	1,335.6	16,566.4	7,776.4
Geom. mean ratio (KARL/MD)			1.93×	—	4.2×	—

Table A.6: Side-by-side hyperparameter comparison for DDIN and KARL. Cells marked ≈ indicate shared or equivalent values; cells marked ≠ indicate differences arising from the distinct architectural designs.

Hyperparameter	**DDIN**	**KARL**	**Reason for difference**
Embedding dimension d	64	64	Shared
Message-passing rounds	3	3	Shared
n-step return depth n	5	5	Shared
PER exponent α_{PER}	0.6	0.6	Shared [50]
IS annealing β_{IS}	$0.4 \rightarrow 1.0$	$0.4 \rightarrow 1.0$	Shared [50]
Training episodes N_{ep}	20,000	20,000	Shared
Training graph size N	$[30, 50]$	$[30, 50]$	Shared
GMM power-law exponent	2.5	2.5	Shared
GMM mean degree $\bar{k}$	6	6	Shared
Weight decay λ_{wd}	10^{-5}	10^{-5}	Shared
Discount factor γ	1.0	0.99	See note below
Learning rate (linear)	10^{-4} (single)	10^{-4} (decoupled)	No splines in DDIN
Learning rate (spline)	N/A	10^{-3}	KAN-specific
Target update period τ	100 steps	1,000 steps	See note below
Final exploration ε_{min}	0.01	0.05	DDIN: sparser rewards
Reconstruction loss weight λ	1 (BCE)	10^{-3} (Laplacian)	Different auxiliary losses
Generative model	D-GMM (directed)	GMM (undirected)	Network topology
Connectivity target	MSCC	LMCC	Directed vs. undirected

Note on discount factor. DDIN uses $\gamma = 1.0$ (undiscounted returns), following the original implementation of MultiDismantler [25] and FINDER [23] for the directed setting, where the episode length is bounded by N and all future rewards carry equal importance. KARL uses $\gamma = 0.99$, which is the standard choice for off-policy Q-learning on variable-length episodes [42, 50] and was inherited from the KARL paper's experimental protocol. Both choices are empirically stable on the respective training corpora; a systematic comparison of $\gamma \in \{0.99, 1.0\}$ within each framework is left for future work.

Note on target-network update period. DDIN uses a hard update every $\tau = 100$ steps, ten times more frequent than KARL's $\tau = 1{,}000$. The shorter period compensates for the smaller mini-batch size ($B = 32$ vs. implicitly larger for KARL) and the sparser, longer-horizon rewards of the directed MSCC dismantling task, where cascade-driven fragmentation events are less frequent per episode than in the undirected LMCC setting.

Note on reconstruction auxiliary losses. The two auxiliary losses are architecturally motivated and not directly comparable. DDIN uses a binary cross-entropy edge-reconstruction loss $\mathcal{L}_{\text{rec}}$ (weighted by $\lambda = 1$) to regularise the bilinear decoder and enforce topological faithfulness of the directed embeddings. KARL uses a graph Laplacian quadratic regulariser $\mathcal{L}_R$ (weighted by $\lambda = 10^{-3}$) to prevent the high-capacity B-spline encoder from assigning arbitrarily dissimilar representations to structurally adjacent nodes, exploiting the symmetry of the undirected adjacency matrix. Neither auxiliary loss is applicable to the other framework's setting without modification.

Bibliography

[1] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. *nature*, 406(6794):378–382, 2000.

[2] Oriol Artime, Marco Grassia, Manlio De Domenico, James P Gleeson, Hernán A Makse, Giuseppe Mangioni, Matjaž Perc, and Filippo Radicchi. Robustness and resilience of complex networks. *Nature Reviews Physics*, 6(2):114–131, 2024.

[3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[4] Albert-László Barabási, Natali Gulbahce, and Joseph Loscalzo. Network medicine: a network-based approach to human disease. *Nature reviews genetics*, 12(1):56–68, 2011.

[5] Andrew R Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 2002.

[6] Gareth John Baxter, G Timár, and José FF Mendes. Targeted damage to interdependent networks. *Physical Review E*, 98(3):032307, 2018.

[7] Ginestra Bianconi. *Multilayer networks: structure and function*. Oxford university press, 2018.

[8] Marian Boguna, Ivan Bonamassa, Manlio De Domenico, Shlomo Havlin, Dmitri Krioukov, and M Ángeles Serrano. Network geometry. *Nature Reviews Physics*, 3(2):114–135, 2021.

[9] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *SIAM review*, 60(2):223–311, 2018.

[10] Alfredo Braunstein, Luca Dall'Asta, Guilhem Semerjian, and Lenka Zdeborová. Network dismantling. *Proceedings of the National Academy of Sciences*, 113(44):12368–12373, 2016.

[11] Sergey V Buldyrev, Roni Parshani, Gerald Paul, H Eugene Stanley, and Shlomo Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, 2010.

[12] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 199–208, 2009.

[13] Pau Clusella, Peter Grassberger, Francisco J Pérez-Reche, and Antonio Politi. Immunization and targeted destruction of networks using explosive percolation. *Physical review letters*, 117(20):208301, 2016.

[14] Carl De Boor and Carl De Boor. *A practical guide to splines*, volume 27. springer New York, 1978.

[15] Manlio De Domenico. Multilayer network data repository. https://manliodedomenico.com/data.php, 2013. Accessed: June 7, 2026.

[16] Manlio De Domenico and Eduardo G Altmann. Unraveling the origin of social bursts in collective attention. *Scientific reports*, 10(1):4629, 2020.

[17] Manlio De Domenico, Vincenzo Nicosia, Alexandre Arenas, and Vito Latora. Structural reducibility of multilayer networks. *Nature communications*, 6(1):6864, 2015.

[18] Manlio De Domenico, Mason A Porter, and Alex Arenas. Muxviz: a tool for multilayer analysis and visualization of networks. *Journal of Complex Networks*, 3(2):159–176, 2015.

[19] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[20] Soumyajit Dev and Malay Bhattacharyya. Ddin: Reinforcement learning with asymmetric gnns for dismantling directed interdependent networks (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 41188–41190, 2026.

[21] Ronald A DeVore, Ralph Howard, and Charles Micchelli. Optimal nonlinear approximation. *Manuscripta mathematica*, 63(4):469–478, 1989.

[22] Ronald A DeVore and George G Lorentz. *Constructive approximation*, volume 303. Springer Science & Business Media, 1993.

[23] Changjun Fan, Li Zeng, Yizhou Sun, and Yang-Yu Liu. Finding key players in complex networks through deep reinforcement learning. *Nature machine intelligence*, 2(6):317–324, 2020.

[24] Marco Grassia, Manlio De Domenico, and Giuseppe Mangioni. Machine learning dismantling and early-warning signals of disintegration in complex systems. *Nature communications*, 12(1):5190, 2021.

[25] Weiwei Gu, Chen Yang, Lei Li, Jinqiang Hou, and Filippo Radicchi. Deep-learning-aided dismantling of interdependent networks. *Nature Machine Intelligence*, pages 1–12, 2025.

[26] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[27] Jihui Han, Shaoyang Tang, Yuefeng Shi, Longfeng Zhao, and Jianyong Li. An efficient layer node attack strategy to dismantle large multiplex networks. *The European Physical Journal B*, 94(3):74, 2021.

[28] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the web conference 2020*, pages 2704–2710, 2020.

[29] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

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

[31] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of complex networks*, 2(3):203–271, 2014.

[32] Kaj-Kolja Kleineberg, Marián Boguná, M Ángeles Serrano, and Fragkiskos Papadopoulos. Hidden geometric correlations in real multiplex networks. *Nature Physics*, 12(11):1076–1081, 2016.

[33] Kaj-Kolja Kleineberg, Lubos Buzna, Fragkiskos Papadopoulos, Marián Boguñá, and M Ángeles Serrano. Geometric correlations mitigate the extreme vulnerability of multiplex networks against targeted attacks. *Physical review letters*, 118(21):218301, 2017.

[34] Longlong Li, Yipeng Zhang, Guanghui Wang, and Kelin Xia. Kolmogorov–arnold graph neural networks for molecular property prediction. *Nature Machine Intelligence*, 7(8):1346–1354, 2025.

[35] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

[36] Jinlong Ma, Peng Wang, and Huijia Li. Directed network disassembly method based on non-backtracking matrix. *Applied Sciences*, 12(23):12047, 2022.

[37] Vitaly Maiorov and Allan Pinkus. Lower bounds for approximation by mlp neural networks. *Neurocomputing*, 25(1-3):81–91, 1999.

[38] Haggai Maron, Ethan Fetaya, Nimrod Segol, and Yaron Lipman. On the universality of invariant networks. In *International conference on machine learning*, pages 4363–4371. PMLR, 2019.

[39] Günter Meinardus. *Approximation of functions: Theory and numerical methods*. Springer Science & Business Media, 2012.

[40] Joshua Melton and Siddharth Krishnan. muxgnn: Multiplex graph neural network for heterogeneous graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):11067–11078, 2023.

[41] Hrushikesh N Mhaskar. Neural networks for optimal approximation of smooth and analytic functions. *Neural computation*, 8(1):164–177, 1996.

[42] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[43] Flaviano Morone and Hernán A Makse. Influence maximization in complex networks through optimal percolation. *Nature*, 524(7563):65–68, 2015.

[44] Saeed Osat, Ali Faqeeh, and Filippo Radicchi. Optimal percolation on multiplex networks. *Nature communications*, 8(1):1540, 2017.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[46] Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8:143–195, 1999.

[47] Michael James David Powell. *Approximation theory and methods*. Cambridge university press, 1981.

[48] Xiao-Long Ren, Niels Gleinig, Dirk Helbing, and Nino Antulov-Fantulin. Generalized network dismantling. *Proceedings of the national academy of sciences*, 116(14):6554–6559, 2019.

[49] Theodore J Rivlin. *Chebyshev polynomials*. Courier Dover Publications, 2020.

[50] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[51] Christian M Schneider, André A Moreira, José S Andrade Jr, Shlomo Havlin, and Hans J Herrmann. Mitigation of malicious attacks on networks. *Proceedings of the National Academy of Sciences*, 108(10):3838–3841, 2011.

[52] Abdul Moeed Siddiqui and Sayda Elmi. Res-kan: Computing-efficient hybrid kan-based residual neural networks for human action recognition. *Procedia Computer Science*, 270:6310–6319, 2025.

[53] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic acids research*, 34(suppl_1):D535–D539, 2006.

[54] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[55] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[56] Lloyd N Trefethen. *Approximation theory and approximation practice, extended edition*. SIAM, 2019.

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[58] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. *stat*, 1050(20):10–48550, 2017.

[59] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. Graph attention networks. In *International conference on learning representations*, volume 6. Ithaca, 2018.

[60] Alessandro Vespignani. The fragility of interdependency. *Nature*, 464(7291):984–985, 2010.

[61] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1225–1234, 2016.

[62] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[63] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[64] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[65] Xingyi Yang and Xinchao Wang. Kolmogorov-arnold transformer. *arXiv preprint arXiv:2409.10594*, 2024.

[66] Jiazheng Zhang and Bang Wang. Dismantling complex networks by a neural model trained from tiny networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2559–2568, 2022.