



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
IFCE *CAMPUS* FORTALEZA
PRÓ-REITORIA DE PESQUISA, PÓS-GRADUAÇÃO E INOVAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

GABRIEL JONAS DA SILVA DUARTE

MÉTODOS ESCALÁVEIS E ELEVAÇÃO DIFERENCIÁVEL PARA APRENDIZADO
PROFUNDO TOPOLÓGICO

FORTALEZA - CEARÁ

2025

GABRIEL JONAS DA SILVA DUARTE

MÉTODOS ESCALÁVEIS E ELEVAÇÃO DIFERENCIÁVEL PARA APRENDIZADO
PROFUNDO TOPOLÓGICO

Dissertação apresentada ao curso de Mestrado
Acadêmico em Ciência da Computação do
PRÓ-REITORIA DE PESQUISA, PÓS-
GRADUAÇÃO E INOVAÇÃO

Programa de Pós-Graduação em Ciência da
Computação do *Campus* Fortaleza da Instituto
Federal de Educação, Ciência e Tecnologia do
Ceará, como requisito parcial à obtenção do
título de mestre em Ciência da Computação.
Área de Concentração: Ciência da Computação.

Orientador: Prof. Dr. Amauri Holanda de
Souza Junior.

FORTALEZA - CEARÁ

2025

Dados Internacionais de Catalogação na Publicação
Instituto Federal do Ceará - IFCE
Sistema de Bibliotecas - SIBI
Ficha catalográfica elaborada pelo SIBI/IFCE, com os dados fornecidos pelo(a) autor(a)

D812m Duarte, Gabriel.
MÉTODOS ESCALÁVEIS E ELEVAÇÃO DIFERENCIÁVEL PARA APRENDIZADO PROFUNDO
TOPOLÓGICO / Gabriel Duarte. - 2025.
67 f. : il. color.

Dissertação (Mestrado) - Instituto Federal do Ceará, Mestrado em Ciência da Computação, Campus
Fortaleza, 2025.

Orientação: Prof. Dr. Amauri Holanda de Souza Junior.

1. Redes neurais para grafos. 2. Aprendizado profundo topológico. 3. Elevação diferenciável. I. Título.
CDD 004



INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO CEARÁ
Av. 13 de maio, 2081 - Bairro Benfica - CEP 60040-531 - Fortaleza - CE - www.ifce.edu.br

FOLHA DE APROVAÇÃO

Processo: 23256.010129/2025-96

Interessado: Gabriel Jonas da Silva Duarte

GABRIEL JONAS DA SILVA DUARTE

MÉTODOS ESCALÁVEIS E ELEVAÇÃO DIFERENCIÁVEL PARA APRENDIZADO PROFUNDO TOPOLÓGICO

Dissertação apresentada ao Curso de MESTRADO ACADÊMICO EM CIÊNCIA DA COMPUTAÇÃO do Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE) *Campus* de Fortaleza, como requisito parcial para obtenção do Título de Mestre em em Ciência da Computação.

Aprovada em: 31/07/2025.

BANCA EXAMINADORA

Prof. Dr. Amauri Holanda de Souza Junior (Orientador)

Instituto Federal de Educação, Ciência e Tecnologia do Ceará - IFCE

Prof. Dr. Helio Pedrini

Universidade Estadual de Campinas

Prof. Dr. Erickson Rangel do Nascimento

Universidade Federal de Minas Gerais

*Este trabalho é dedicado aos meu avós,
Francisco de Assis Frutuoso Duarte e Joana So-
ares Campelo Duarte, em que mesmo diante das
dificuldades, nunca deixaram que faltasse nada.*

AGRADECIMENTOS

A conclusão deste trabalho representa não apenas o encerramento de uma etapa trilhada na academia, mas também o resultado de uma rede de apoio, incentivo e colaboração de diversas pessoas. E a instituição que, direta ou indiretamente, me ajudaram a conseguir essa conquista. A todos, meus mais sinceros agradecimentos.

Agradeço ao professor Amauri Holanda, que me ensinou praticamente tudo o que sei e continua me ensinando todos os dias. A ele deixo minha profunda gratidão por toda a dedicação e paciência demonstradas ao longo desses anos. Cada orientação e conselho contribuíram significativamente para minha evolução, tanto acadêmica quanto profissional.

Agradeço também ao professor Diego Mesquita, que, durante todo esse tempo, esteve ao nosso lado, virando noites, escrevendo, *debugando* e teorizando conosco.

Ao CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), agradeço pelo suporte financeiro e institucional. O incentivo à pesquisa proporcionado foi essencial para que eu continuasse o desenvolvimento deste projeto, seu suporte garantiu condições adequadas para que eu conseguisse realizar alguns experimentos.

Aos meus familiares, meu pai, mãe, irmão, tias, tios e minha avó, minha eterna e profunda gratidão. Vocês foram parte do meu alicerce em todos os momentos, oferecendo suporte emocional e material. A paciência, compreensão e palavras de encorajamento foram fundamentais para que eu seguisse em frente, mesmo diante das maiores dificuldades. Cada conquista minha é também de vocês.

À minha amiga Emanuelle Jessica, agradeço pelas palavras encorajadoras, que, desde a graduação, me motivaram a não desistir da área.

Aos meus amigos Pedro Alisson e Isaac Freitas, que, como uma forma de escape, compartilhavam comigo discussões sobre implementações, *queries* ou qualquer outra coisa, apenas por diversão. Aos meus amigos, Matheus Teixeira, Rubens e Iolanda pelas jogatinas e karaokês durante a noite.

Ao meu amigo Acélio Sousa, que me ensinou muito por meio de conversas sobre tecnologia, matemática e assuntos afins — tudo isso teve grande valor na formação de quem sou hoje.

Às minhas amigas Mayanna Rodrigues, Helaine Pontes e Emilly Maia, e aos meus amigos Steve Nascimento, Felipe, Marcos Túlio, José João, Marília Abreu e João, minha sincera gratidão. Com vocês vivi momentos incríveis. Jamais esquecerei o carinho, o companheirismo e

o apoio que me deram ao longo de toda esta jornada.

Aos meus companheiros de time de pesquisa — Erik Jhones, Tamara Arruda, Jorge Luis Franco e Acélio Sousa —, meu agradecimento por todas as contribuições ao longo desta pesquisa, seja por meio de discussões, sugestões ou trocas de experiências.

Agradeço também aos meus amigos e companheiros de equipe Helio Lopes, Walber Florencio, Herik Costa, Werbster Marques e Talisson Freitas, que sempre me cobriram no trabalho. Eles me apoiaram e me deram suporte nas ocasiões em que precisei me ausentar por conta deste projeto. Jamais esquecerei o que fizeram por mim.

A todos citados acima agradeço por estarem sempre por perto, nos momentos de alegria e nas fases mais difíceis. Pelas conversas inspiradoras, pelas palavras de apoio, pelas risadas nos momentos de tensão e pelo companheirismo constante. A presença de vocês tornou essa jornada mais leve, mais humana e muito mais significativa.

Por fim, agradeço àqueles que acreditaram em mim mesmo quando eu duvidei. Este trabalho é resultado de um esforço coletivo, e levo comigo a certeza de que nenhuma conquista é feita sozinho. A todos vocês, o meu mais sincero e afetuoso obrigado.

“Não existe triunfo sem perda,
não há vitória sem sofrimento,
não há liberdade sem sacrifício.”

(J.R.R. Tolkien)

RESUMO

Redes neurais topológicas (*topological neural networks*, TNNs) compreendem uma classe de redes neurais capazes de processar dados com estrutura relacional, como grafos, complexos simpliciais e complexos celulares. De modo similar às redes neurais para grafos, TNNs normalmente empregam camadas de passagem de mensagem onde elementos da entrada (e.g., nós e faces) atualizam suas representações com base nas representações de seus vizinhos topológicos. Para tarefas preditivas em grafos, essa etapa de passagem de mensagem é precedida por um procedimento não-supervisionado (i.e., definido *a priori*) de *lifting* (elevação ou levantamento), no qual grafos de entrada são transformados em objetos relacionais de ordem superior (e.g., hipergrafos). Apesar do sucesso inicial das TNNs, esses modelos sofrem de duas limitações cruciais, que pretendemos aliviar com este trabalho: o alto custo computacional da etapa de passagem de mensagem e a natureza não-supervisionada de *liftings*. Para aliviar o custo de TNNs, este trabalho investiga o efeito de removermos não-linearidades em TNNs para complexos simpliciais, o que torna possível pré-computarmos o processo de passagem de mensagem. O modelo linear resultante, chamado de SimpleXCN, demonstrou desempenho preditivo comparável às TNNs enquanto permitiu uma redução de até 25% no tempo de execução e uma diminuição de até 50% no uso de memória, com a utilização de até 10 vezes menos parâmetros. Além disso, este trabalho propõe ∂ lift (DiffLift), uma nova abordagem para aprendizado de elevações de grafos para hipergrafos, complexos celulares e complexos simpliciais, habilitando aprendizagem de ponta a ponta (*end-to-end learning*) para TNNs em tarefas sobre grafos. De maneira geral, ∂ lift utiliza representações de vértices para identificar e parametrizar distribuições sobre células de ordem superior candidatas à inclusão. Isso resulta em um modelo escalável, que pode ser facilmente integrado a qualquer TNN. Nos experimentos, o ∂ lift superou métodos de *lifting* existentes em múltiplos *benchmarks* de classificação de grafos e nós, com ganhos de até 45% em acurácia.

Palavras-chave: Redes neurais para grafos. Aprendizado profundo topológico. Elevação diferenciável.

ABSTRACT

Topological neural networks (TNNs) comprise a class of neural networks capable of processing data with relational structure, such as graphs, simplicial complexes, and cellular complexes. Akin to graph neural networks, typical TNNs employ message-passing layers where each element of the input (e.g., nodes or cells) updates its representation based on those of its topological neighbors. For predictive tasks on graphs, this message-passing step is preceded by an unsupervised (i.e., defined *a priori*) lifting procedure, in which input graphs are transformed into higher-order relational objects (e.g., hypergraphs). Despite the initial success of TNNs, these models suffer from two crucial limitations, which we aim to alleviate with this work: the high computational cost of the message-passing step and the unsupervised nature of graph liftings. To alleviate the cost of TNNs, this work investigates the effect of removing nonlinearities in TNNs for simplicial complexes, which makes it possible to pre-compute the message-passing process. The resulting linear model, called `SimpleXCN`, demonstrated predictive performance comparable to TNNs while allowing for a reduction of up to 25% in runtime and a reduction of up to 50% in memory usage, using up to 10 times fewer parameters. Furthermore, this work proposes ∂lift (`DiffLift`), a new approach for learning liftings from graphs to hypergraphs, cellular complexes, and simplicial complexes, enabling end-to-end learning for TNNs in graph tasks. In general, ∂lift uses vertex representations to identify and parameterize distributions over higher-order cells that are candidates for inclusion. This results in a scalable model that can be easily integrated into any TNN. In experiments, ∂lift outperformed existing lifting methods on multiple graph and node classification benchmarks, with accuracy gains of up to 45%.

Keywords: Graph Neural Networks. Topological Deep Learning. Differentiable Lifting.

LISTA DE FIGURAS

Figura 1 – Ilustração de um complexo celular, e os homeomorfismos correspondentes ϕ_i a três bolas fechadas de diferentes dimensões no complexo.	24
Figura 2 – Exemplos de domínios topológicos e relacionais.	26
Figura 3 – Visão geral de como um vértice agrega mensagens de sua vizinhança local. .	34
Figura 4 – Duas iterações do ∂ lift para complexos de células.	49

LISTA DE TABELAS

Tabela 1 – Propriedades dos conjuntos de dados utilizados nos experimentos do SimpleXCN.	43
Tabela 2 – Comparação do SimpleXCN com outras arquiteturas para complexos simpliciais em múltiplos bancos de dados.	44
Tabela 3 – Os resultados do teste de Wilcoxon, comparando as acurácias do SimpleXCN a dois métodos distintos, SCN e SCCN.	45
Tabela 4 – Comparação de tempo por iteração entre o SimpleXCNe outras arquiteturas	45
Tabela 5 – O número total de parâmetros para cada arquitetura e o pico de memória alocada para sua execução são apresentados.	46
Tabela 6 – Estatísticas dos conjuntos de dados para tarefas em nível de grafo.	52
Tabela 7 – Principais hiperparâmetros presentes no TopoBenchmarkX para CWN, CXN e UniGCNII	53
Tabela 8 – Classificação de grafos: ∂ lift vs liftings estáticos.	54
Tabela 9 – Efeito da GNN escolhida no desempenho do ∂ lift.	55
Tabela 10 – Estatísticas para os conjuntos de dados utilizados para classificação de nós. .	56
Tabela 11 – Comparação do DCM e ∂ lift na classificação de nós.	56
Tabela 12 – Comparação do ∂ lift e dos <i>baselines</i> de <i>lifting</i> estáticos para hipergrafos na classificação de nós.	57

LISTA DE SIGLAS

AUROC	<i>Area under the ROC curve</i>
CNN	<i>Convolutional Neural Networks</i>
CWN	<i>CW Networks</i>
CXN	<i>Cellular Complex neural networks</i>
DCM	<i>Differentiable Cell Complex Module</i>
DGM	<i>Differentiable Graph Module)</i>
GCN	<i>Graph Convolutional Network</i>
GCNII	<i>Graph Convolution Network II</i>
GIN	<i>Graph Isomorph Network</i>
GNN	<i>Graph Neural Network</i>
GPS	<i>Graph Transformer</i>
GPU	<i>Graphics Processing Unit</i>
IA	<i>Artificial Intelligence</i>
kNN	<i>K-nearest neighbors</i>
ML	<i>Machine Learning</i>
MLP	<i>Multilayer Perceptron</i>
PIM	<i>Polygon Inference Module</i>
RBF	<i>Radial Basis Function</i>
RNN	<i>Recurrent Neural Network</i>
SCCN	<i>Simplicial Complex Convolutional Network</i>
SCN	<i>Simplex Neural Network</i>
SGC	<i>Simple Graph Convolution</i>
SNN	<i>Simplicial Neural Networks</i>
STE	<i>Straight Through Estimator</i>
TDL	<i>Topological Deep Learning</i>
TNN	<i>Topological Neural Network</i>
UniGNN	<i>Unified Framework for Graph and Hypergraph Neural Networks</i>

LISTA DE SÍMBOLOS

$\in$	Pertence
$\forall$	Para todo
$\exists$	Existe
G	Grafo
V	Conjunto de Vértices
$\subseteq$	Inclusão ou subconjunto em teoria dos conjuntos
$\times$	Produto Cartesiano
$\mathbb{R}$	Conjunto dos números reais
E	Conjunto de Arestas
A	Matriz de Adjacência
$\mathcal{N}(u)$	Conjunto dos vizinhos de u em um grafo
$\mathcal{N}_B(\cdot)$	Conjunto de elementos pertencentes à vizinhança de fronteira
$\mathcal{N}_C(\cdot)$	Conjunto de elementos pertencentes à vizinhança de co-fronteira
$\mathcal{N}_\uparrow(\cdot)$	Conjunto de elementos pertencentes à adjacência superior
$\mathcal{N}_\downarrow(\cdot)$	Conjunto de elementos pertencentes à adjacência inferior
Δ	Matriz Laplaciana
σ, τ, δ	Simplexo ou Célula
x_σ	Caraterísticas da Célula σ
K	Complexo Simplicial ou Celular
∂	Operador da função de fronteira (<i>Boundary</i>)
$\mathbb{G}$	Domínio do espaço de grafos
$\mathbb{T}$	Domínio de um espaço topológico
$Z_n(K)$	<i>Kernel</i> dos operadores de fronteira

B	Matriz de Incidência ou Matriz de Fronteira (Boundary)
$\mathbb{Z}/2\mathbb{Z}$	Notação para os inteiros módulo 2
$Cl(\mathcal{G})$	Conjunto de cliques em um grafo G
ℓ	Camada
H	Saída
ψ	Função de ativação
ϕ	Representação de uma rede neural de passagem de mensagem
$\otimes_i$	Operação de agregação que combina as mensagens de diferentes vizinhanças
$\oplus$	Função de agregação invariável a permutações
Θ	Matriz de parâmetros (pesos) aprendíveis
$\parallel$	Concatenação
$\{\cdot\}$	Multiconjunto
r	Dimensão(ranking) de uma simplexo (ou célula)
S_r	Matriz de propagação (ou de difusão) de dimensão r
$\mathcal{L}$	Função de perda
$Ber(\cdot)$	Distribuição de Bernoulli
$\mathcal{C}$	Conjunto de células
z_v	Representação vetorial (<i>embedding</i>) para um nó v
z_c	Representação vetorial (<i>embedding</i>) para uma célula c

SUMÁRIO

1	INTRODUÇÃO	17
1.1	Motivação e Visão Geral	17
1.2	Hipóteses de Pesquisa	19
1.3	Objetivos	20
1.4	Produção Científica	21
1.5	Estrutura da Dissertação	21
2	DOMÍNIOS TOPOLÓGICOS E RELACIONAIS	22
2.1	Espaços Topológicos	22
2.2	Grafos	23
2.2.1	<i>Hipergrafos</i>	<i>23</i>
2.3	Complexos Simpliciais	23
2.4	Complexos Celulares	24
2.5	Complexos combinatórios	25
2.6	Atributos	26
2.7	Estruturas de Vizinhança	26
2.8	Cadeias, Operadores de Fronteira e Ciclos	27
3	REDES NEURAIIS TOPOLÓGICAS	30
3.1	Elevando Grafos a Objetos de Ordem Superior	30
3.1.1	<i>Elevação por Cliques</i>	<i>30</i>
3.1.2	<i>Elevação por Ciclos</i>	<i>31</i>
3.1.3	<i>Elevação por k-saltos</i>	<i>31</i>
3.1.4	<i>Elevação por k Vizinhos mais Próximos</i>	<i>32</i>
3.1.5	<i>Elevação por Kernel</i>	<i>32</i>
3.1.6	<i>Differentiable Cell Complex Module</i>	<i>32</i>
3.1.7	<i>Elevação de Atributos (features)</i>	<i>33</i>
3.2	Redes Neurais Topológicas	33
3.2.1	Redes Neurais para Grafos	33
3.2.1.1	<i>Graph Convolutional Network (GCN)</i>	<i>34</i>
3.2.1.2	<i>Simple Graph Convolution (SGC)</i>	<i>35</i>
3.2.1.3	<i>Graph Isomorphism Network (GIN)</i>	<i>35</i>
3.2.2	Redes de Passagem de Mensagem de Alta Ordem	37

3.2.2.1	<i>Simplex Convolutional Network (SCN)</i>	38
3.2.2.2	<i>CW Networks (CWN)</i>	39
3.2.2.3	<i>Cell Complex Neural Networks (CXN)</i>	39
3.2.2.4	<i>Unified Framework for Graph and Hypergraph Neural Networks (UniGNN)</i>	40
4	ACELERANDO REDES CONVOLUCIONAIS PARA COMPLEXOS	
	SIMPLICIAIS	41
4.1	Simple simplex convolutional network (SimpleXCN)	41
4.2	Experimentos e Resultados	42
4.2.1	<i>Conjuntos de Dados</i>	42
4.2.2	<i>Configuração Experimental</i>	43
4.2.3	<i>Resultados</i>	44
5	LIFTING DIFERENCIÁVEL	47
5.1	Formulação Geral	47
5.1.1	<i>Lifting de Grafo para Hipergrafo</i>	48
5.1.2	<i>Elevação de Características (Feature Lifting)</i>	49
5.1.3	<i>Elevação de Grafo para complexo de células</i>	49
5.1.3.1	<i>Caso $D = 1$: Aprendendo arestas</i>	50
5.1.3.2	<i>Caso $D \geq 2$: Aprendendo células D-dimensões</i>	50
5.2	Experimentos e Resultados	51
5.2.1	Classificação de grafos	51
5.2.1.1	<i>Conjuntos de dados</i>	51
5.2.1.2	<i>Baselines</i>	52
5.2.1.3	<i>Configuração da avaliação</i>	52
5.2.1.4	<i>Resultados</i>	54
5.2.1.5	<i>Impacto da escolha da GNN no desempenho</i>	55
5.2.2	Classificação de Nós	55
5.2.2.1	<i>Conjuntos de dados</i>	55
5.2.2.2	<i>Baselines</i>	56
5.2.2.3	<i>Configuração da avaliação</i>	56
5.2.2.4	<i>Resultados</i>	57
6	CONCLUSÃO	58
6.1	Principais achados	58
6.2	Limitações	59

6.3	Potenciais aplicações	59
6.4	Direções para trabalhos futuros	59
	REFERÊNCIAS	60

1 INTRODUÇÃO

À medida que os dados se tornam mais interconectados, as abordagens tradicionais de aprendizado de máquina mostram suas limitações, o que levou ao desenvolvimento de Redes Neurais para Grafos (GNNs) e, subsequentemente, Redes Neurais Topológicas (TNNs) para lidar com interações de alta ordem. Apesar de seu potencial, essa área enfrenta dois obstáculos principais: o elevado custo computacional associado à passagem de mensagem em estruturas topológicas e a predominância de métodos de elevação (*lifting*) de grafos que são estáticos e não supervisionados, nem sempre sendo ideais para a tarefa em questão. Em resposta a essas lacunas, esta dissertação apresenta duas contribuições centrais: o SimpleXCN, um modelo linear para complexos simpliciais que reduz o custo computacional mantendo um desempenho competitivo, e o ∂ lift (DiffLift), um *framework* diferenciável que aprende a mapear grafos para domínios de ordem superior de forma integrada ao treinamento do modelo. Este capítulo introduz a aplicação da inteligência artificial a dados com estruturas complexas e multidimensionais. Por fim, são detalhados os objetivos específicos, a produção científica resultante e a estrutura dos capítulos que compõem a dissertação.

1.1 Motivação e Visão Geral

Nos últimos anos, diversas áreas do conhecimento têm sido impactadas pelos avanços sem precedentes em inteligência artificial (IA) / aprendizado de máquina (*machine learning*, ML). De modo simples, ML é uma subárea da IA que permite que computadores aprendam a partir de dados, identificando padrões e tomando decisões com base em algoritmos que se aprimoram ao longo do tempo (ALPAYDIN, 2014).

O impacto de ML na sociedade moderna é inegável. Ele transformou a forma como as empresas operam, como os governos tomam decisões, e até mesmo como indivíduos interagem com a tecnologia no dia a dia. Sistemas de recomendação (YING *et al.*, 2018; GAO *et al.*, 2022), como os usados por plataformas de *streaming* de vídeo e música, dependem fortemente de ML para oferecer conteúdo personalizado. Além disso, ML desempenha um papel crucial em áreas críticas como a saúde, onde algoritmos são usados para prever doenças (KAVITHA *et al.*, 2021; SUN; SUN; ZHAO, 2022), identificar padrões em exames médicos, e até mesmo sugerir tratamentos personalizados (ZHANG; SHI; WANG, 2023). Na indústria financeira, ML é utilizado para detectar fraudes (LIU *et al.*, 2020; ALARFAJ *et al.*, 2022), prever flutuações de mercado e otimizar portfólios de investimento (PAIVA *et al.*, 2019; MA; HAN; WANG, 2021).

Diversos sistemas reais, como redes sociais (FREEMAN, 2004), redes de transporte (RAHMANI *et al.*, 2023) e proteínas (JHA; SAHA; SINGH, 2022), exibem estruturas relacionais complexas que descrevem a conexão e interação entre os componentes do sistema. Uma forma comum de representar tais estruturas é por meio de grafos — objeto matemático composto por conjuntos de vértices e arestas (*links* entre os vértices). Naturalmente, a ausência de uma ordem canônica para os vértices/arestas, as simetrias que grafos podem codificar e a forte interdependência entre os dados desses sistemas limitam a utilização de modelos clássicos de aprendizado de máquina para dados relacionais. Por exemplo, redes neurais do tipo perceptron multicamadas (*multilayer perceptron*, MLP) operam sobre entradas com estrutura tabular fixa e normalmente assumem que os dados são amostras independentes e identicamente distribuídas de uma distribuição de probabilidade alvo desconhecida (SHALEV-SHWARTZ; BEN-DAVID, 2014).

A área de aprendizado profundo geométrico (BRONSTEIN *et al.*, 2021) é a espinha dorsal por trás de recentes esforços bem-sucedidos para analisar e modelar dados com estrutura relacional. Em particular, redes neurais para grafos (*graph neural networks*, GNNs) (SCARSELLI *et al.*, 2009) têm levado a avanços em diversas áreas como design de antibióticos (JIANG *et al.*, 2021), sistemas de recomendação (YING *et al.*, 2018), tradução automática (WU *et al.*, 2020; BASTINGS *et al.*, 2017), simulação de sistemas físicos (SANCHEZ-GONZALEZ *et al.*, 2020) e redes sociais (SHARMA *et al.*, 2024).

Apesar do recente sucesso de GNNs, esses modelos exploram exclusivamente interações entre pares de entidades, potencialmente negligenciando relações de alta ordem relevantes para a tarefa de interesse. Além disso, GNNs possuem expressividade limitada pelo teste de isomorfismo de Weisfeiler-Lehman (WL), conhecido também como algoritmo de refinamento de cor (WEISFEILER; LEMAN, 1968). Nesse contexto, a área de aprendizado profundo topológico (*topological deep learning*, TDL) emerge como a nova fronteira do conhecimento no que se refere à aprendizagem de representações para dados relacionais (PAPAMARKOU *et al.*, 2024).

Principais representantes de TDL, as redes neurais topológicas (*topological neural networks*, TNNs) (BODNAR *et al.*, 2021a; HENSEL; MOOR; RIECK, 2021) são modelos baseados em passagem de mensagem que processam dados com estruturas relacionais de ordem superior, indo além de grafos. Assim, TNNs não só generalizam GNNs (i.e., toda GNN pode ser considerada uma TNN) mas também fornecem um arcabouço notacional para modelos neurais que operam sobre dados em diferentes domínios, incluindo complexos simpliciais (BODNAR *et al.*, 2021b), complexos celulares (BODNAR *et al.*, 2021a) e hipergrafos (HUANG; YANG, 2021). Como contrapartida ao poder representacional superior das TNNs, a lacuna de complexidade

computacional entre TNNs e GNNs aumenta à medida que a ordem do domínio relacional aumenta — decorrente da passagem de mensagens entre estruturas de ordem superior.

Para utilizarmos TNNs em tarefas preditivas em grafos, os grafos de entrada devem ser inicialmente submetidos a um procedimento responsável por mapeá-los para o domínio no qual a TNN opera, um processo conhecido como elevação ou levantamento (*lifting*). Métodos de elevação podem explorar tanto a conectividade quanto os atributos do grafo (e.g., atributos de vértices) para obter estruturas relacionais de ordem superior. Alguns exemplos incluem *elevação por cliques* (BODNAR *et al.*, 2021b), onde um complexo simplicial é produzido com base nos cliques no grafo de entrada, e *elevação por ciclos* (HAJIJ *et al.*, 2022b), que detecta ciclos para obter as células de dimensão 2 de um complexo celular.

Surpreendentemente, apesar do alto impacto da operação de elevação nas TNNs, a maioria dos métodos de elevação é não-supervisionada e, portanto, não é informada pela tarefa em questão, o que pode levar a escolhas subótimas. Até o momento, elevação diferenciável só foi explorada no contexto de complexos celulares (BATTILORO *et al.*, 2024).

1.2 Hipóteses de Pesquisa

A partir destas lacunas identificadas, as seguintes hipóteses de pesquisa guiaram o desenvolvimento deste trabalho:

- **Hipótese 1:** É possível reduzir significativamente o custo computacional das redes neurais topológicas por meio da simplificação de suas arquiteturas, preservando ou mantendo competitivo o desempenho em tarefas de classificação.
- **Hipótese 2:** Métodos de elevação aprendíveis e supervisionados podem superar abordagens estáticas, produzindo estruturas de ordem superior mais adequadas às tarefas de aprendizado.

Nesse contexto, esta dissertação investiga estratégias para lidar com duas limitações de TNNs para tarefas em grafos: *i*) o alto custo computacional do processo de mensagem de ordem superior e *ii*) a natureza estática / não-supervisionada dos métodos de elevação existentes.

Mais especificamente, este trabalho propõe o SimpleXCN um modelo linear para complexos simpliciais, que reduz o custo computacional ao remover as não-linearidades entre as camadas de uma SCN (YANG; SALA; BOGDAN, 2022). Apesar de sua simplicidade, SimpleXCN é competitivo com TNNs para complexo simplicial populares (como SCNs

e SCCNs) para tarefas de classificação de nós, além de reduzir significativamente o uso de memória, o número de parâmetros e o tempo de execução por época.

Além disso, a segunda parte deste trabalho apresenta o ∂ lift (DiffLift), uma estrutura geral de elevação diferenciável e aprendível de ponta a ponta, aplicável a diversos domínios, como hipergrafos, complexos simpliciais e complexos celulares. Esse método utiliza uma abordagem probabilística para amostrar células candidatas de tamanhos adaptativos. Ou seja, utiliza distribuições parametrizadas sobre células usando representações de nós derivados de modelos neurais para grafos — por exemplo, GNNs ou *Graph Transformers* (RAMPÁŠEK *et al.*, 2022). Para cada célula candidata, sua representação (*embedding*) é calculada e uma rede MLP é aplicada para estimar a probabilidade de uma célula ser ou não incluída na estrutura de saída.

Nossos resultados mostram que ∂ lift supera consistentemente os métodos de elevação não supervisionados em quase todos os *benchmarks* de classificação de grafos, alcançando desempenho superior em 22 de 24 experimentos, frequentemente por uma margem substancial. Esses ganhos são robustos em todas as arquiteturas de TNNs avaliadas.

1.3 Objetivos

O objetivo geral deste trabalho é desenvolver arquiteturas de TNNs escaláveis e aprendíveis de ponta a ponta, avançando o estado da arte em redes topológicas de alta ordem para tarefas preditivas em grafos. Para alcançar este objetivo, elencamos os seguintes objetivos específicos:

- Avaliar o impacto de camadas não-lineares no desempenho preditivo de redes neurais topológicas.
- Analisar experimentalmente o desempenho computacional das abordagens propostas em tarefas de predição a nível de grafo e nó.
- Desenvolver um método de elevação diferenciável para diferentes domínios topológicos, demonstrando os benefícios de tornar o processo de elevação aprendível e “supervisionado”.
- Comparar o desempenho do método de elevação diferenciável com abordagens tradicionais de elevação em tarefas de classificação de nós e de grafos.

1.4 Produção Científica

Durante este projeto de mestrado, os seguintes trabalhos científicos foram submetidos ou publicados:

- **FRANCO, Jorge; DUARTE, Gabriel; NIKITIN, Alexander V.; PONTI, Moacir A.; MESQUITA, Diego; SOUZA, Amauri H. Differentiable Lifting for Topological Neural Networks.** Submetido a: Thirty-Ninth Annual Conference on Neural Information Processing Systems (NeurIPS 2025), 2025, San Diego, EUA.
- **DUARTE, Gabriel; ARRUDA, Tamara; MESQUITA, Diego; SOUZA, Amauri H.** 2024. **Simplifying Simplex Convolutional Networks.** In: SYMPOSIUM ON KNOWLEDGE DISCOVERY, MINING AND LEARNING (KDMILE), 12. , 2024, Belém/PA. Anais. Porto Alegre: Sociedade Brasileira de Computação, 2024 . p. 137-144.

1.5 Estrutura da Dissertação

Com a Introdução, a dissertação apresentada é dividida em 6 capítulos. No Capítulo 2, é apresentada uma base teórica sobre domínios relacionais. No Capítulo 3, são expostos alguns métodos de elevação e algumas das principais redes neurais topológicas propostas na literatura. O Capítulo 4 descreve a formulação do método `SimpleXCN` para redução do custo computacional de redes neurais para complexos simpliciais. O Capítulo 5 apresenta o método ∂ lift, com sua formulação geral para elevação de grafos para hipergrafos e complexos de células. Além disso são relatadas as configurações experimentais e as bases de dados para avaliação do método proposto. Finalmente, no Capítulo 6, é apresentada a conclusão.

2 DOMÍNIOS TOPOLÓGICOS E RELACIONAIS

Este capítulo aborda conceitos de topologia necessários para entender redes neurais topológicas. A ênfase está na definição dos diferentes domínios topológicos que servem como entrada para essas redes. Mais especificamente, vamos sucintamente apresentar grafos, hipergrafos, complexos simpliciais, complexos celulares e complexos combinatórios.

2.1 Espaços Topológicos

Neste trabalho, estamos interessados em problemas cujos dados estão estruturados como espaços topológicos. Seja X um conjunto não vazio. Uma *topologia* $\mathcal{T}$ em X é uma coleção de subconjuntos de X (chamados de *conjuntos abertos*) que satisfaz as seguintes propriedades:

- 1) $\emptyset$ e X pertencem a $\mathcal{T}$;
- 2) A união arbitrária de subconjuntos de $\mathcal{T}$ pertence a $\mathcal{T}$;
- 3) A interseção finita de subconjuntos de $\mathcal{T}$ pertence a $\mathcal{T}$.

O par ordenado $(X, \mathcal{T})$ é denominado *espaço topológico* (MUNKRES, 2018, p. 74). Quando a topologia está subentendida, utilizamos somente X para se referir ao espaço topológico.

Um outro conceito básico que vamos utilizar a seguir é o de homeomorfismo entre espaços topológicos. Sejam $(X, \mathcal{T}_X)$ e $(Y, \mathcal{T}_Y)$ dois espaços topológicos. Uma função $f : X \rightarrow Y$ é chamada de *homeomorfismo* se satisfizer as seguintes condições:

- 1) f é bijetiva;
- 2) f é contínua, ou seja, a imagem de qualquer conjunto aberto de X é um conjunto aberto em Y ;
- 3) f^{-1} (a função inversa de f) é contínua.

Quando tal função existe, dizemos que X e Y são *homeomorfos* (MUNKRES, 2018, p. 103). Intuitivamente, um homeomorfismo é uma transformação que distorce o espaço sem rasgar ou colar, preservando as propriedades topológicas essenciais, como conectividade e número de buracos (*holes*).

2.2 Grafos

Grafos são estruturas de dados com a capacidade de representar redes complexas do mundo real, compostas por objetos e suas relações. Formalmente, um grafo não-direcionado é definido como uma tupla $G = (V, E)$, onde $v_i \in V$ representa um vértice, e E é um conjunto de pares não-ordenados de vértices distintos — chamados de arestas (HAMILTON, 2020). Um grafo é geralmente representado por uma matriz de adjacência A , em que $a_{i,j} = 1$ significa haver uma aresta e_{ij} que conecta v_i e v_j , e 0 caso contrário. A matriz diagonal de graus D de G é formada pelos elementos $d_{i,i} \doteq \sum_j a_{i,j}$, onde $d_{i,i}$ denota o grau do nó v_i (BARABÁSI, 2013).

Além disso, a matriz Laplaciana de G é dada por $\Delta = D - A$, e sua versão normalizada é definida como $\tilde{\Delta} = D^{-1/2} \Delta D^{-1/2}$. Essas matrizes são fundamentais para GNNs, uma vez que são utilizadas como operadores de difusão em grafos. Ainda nesse contexto, é muito comum utilizarmos matrizes de adjacências normalizadas com *self-loops*, i.e., $\tilde{A} = (D + I)^{-\frac{1}{2}}(A + I)(D + I)^{-\frac{1}{2}}$, em que I é a matriz identidade (KIPF; WELLING, 2016). O conjunto de vizinhos de um nó v em G é denotado por $\mathcal{N}(v) = \{u \in V : \{v, u\} \in E\}$.

2.2.1 Hipergrafos

Um hipergrafo é uma estrutura de dados que generaliza os grafos, permitindo que as arestas (ou hiperarestas) conectem múltiplos vértices simultaneamente. Formalmente, um hipergrafo é definido como uma tupla (V, E) , onde E é um conjunto de subconjuntos não vazios de V , representando as hiperarestas. Ou seja, cada $e \in E$ é um subconjunto de V que pode conter dois ou mais vértices (HAJIJ *et al.*, 2022b).

2.3 Complexos Simpliciais

Complexos simpliciais também generalizam a ideia de grafos. Formalmente, um *complexo simplicial abstrato* sobre um conjunto de vértices V é um conjunto K de subconjuntos de V (chamados *símplices* ou *simplexos*) tal que, para todo $\sigma \in K$ e todo $\tau \subset \sigma$ não-vazio, temos que $\tau \in K$. Seja σ um símplice (ou *simplex*), então seus subconjuntos não-vazios $\tau \subset \sigma$ são chamados de *faces*, e σ é uma *coface* de τ (VIRK, 2022, p. 37). A dimensão de um símplice é igual à sua cardinalidade menos 1, e a dimensão de um complexo simplicial é a dimensão máxima de seus símplices. Além disso, utilizamos $K_{(i)}$ para denotar o subconjunto de símplices de dimensão i de K .

Exemplo 2.1. Seja $V = \{0, 1, 2, 3\}$. A seguinte família de subconjuntos de V

$$K = \{\{0\}, \{1\}, \{2\}, \{3\}, \{0, 1\}, \{1, 2\}, \{2, 3\}, \{3, 0\}, \\ \{0, 1, 2\}, \{0, 1, 3\}, \{0, 2, 3\}, \{1, 2, 3\}, \{0, 1, 2, 3\}\}$$

forma um complexo simplicial de dimensão 3.

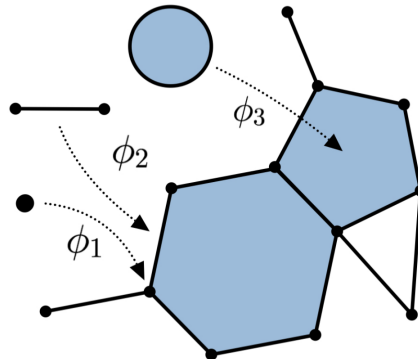
Note que todo grafo pode ser visto como um complexo simplicial de dimensão 1. Em particular, considere o grafo $G = (V, E)$, onde $V = \{0, 1, 2\}$ e $E = \{\{0, 1\}, \{1, 2\}\}$. Nesse caso, G corresponde ao complexo simplicial $K = \{\{0\}, \{1\}, \{2\}, \{0, 1\}, \{1, 2\}\}$.

Podemos ainda definir uma relação de fronteira (*boundary relation*) $\prec$ para símplies. Em particular, dizemos que $\tau \prec \sigma$ se, e somente se, $\tau \subset \sigma$ e não existe nenhum símlice δ tal que $\tau \subset \delta \subset \sigma$ (BODNAR *et al.*, 2021b). No Exemplo 2.1, temos que os vértices $\{0\}$ e $\{1\}$ estão na fronteira da aresta $\{0, 1\}$.

2.4 Complexos Celulares

Complexos celulares generalizam a ideia de complexos simpliciais (HATCHER, 2002) através da construção mais flexível de espaços topológicos por meio de células de dimensões variadas. A Figura 1 ilustra um complexo celular regular. Note que enquanto símplies de dimensão 2 devem possuir três vértices, células de dimensão 2 em um complexo celular podem conectar um número arbitrário de vértices.

Figura 1 – Ilustração de um complexo celular, e os homeomorfismos correspondentes ϕ_i a três bolas fechadas de diferentes dimensões no complexo.



Fonte: (BODNAR *et al.*, 2021a)

Formalmente, segundo Bodnar *et al.* (2021a), um complexo celular regular é um espaço topológico X com uma partição $\{X_\sigma\}_{\sigma \in P_X}$ de subespaços X_σ de X chamados células, tal que

1. Para cada ponto $x \in X$, existe uma vizinhança aberta (conjunto aberto) de x que intersecta um número finito de células.
2. Para todos $\sigma, \tau \in P_X$, a interseção $X_\tau \cap \overline{X_\sigma}$ (onde $\overline{X_\sigma}$ é o fecho de X_σ) é não-vazia se, e somente se, $X_\tau \subseteq \overline{X_\sigma}$. Esta condição implica que a interseção de duas células X_τ e X_σ só ocorre se X_τ estiver contido no fecho de X_σ . O fecho é composto pela célula X_σ juntamente com todos os seus limites (ou fronteiras, bordas).
3. Cada célula X_σ é homeomorfa a $\mathbb{R}^{D_\sigma}$, para algum valor de D_σ , que é chamado de a dimensão de X_σ ;
4. Para todo $\sigma \in P_X$, existe um homeomorfismo ϕ de uma bola fechada em $\mathbb{R}^{D_\sigma}$ para $\overline{X_\sigma}$, de modo que a restrição de ϕ à parte interior da bola seja um homeomorfismo sobre X_σ .

É importante notar que as condições (2) e (4) impõem uma estrutura de conjunto parcialmente ordenado (poset) sobre o conjunto P_X . Mais especificamente, dizemos que $\tau \leq \sigma$ se, e somente se, $X_\tau \subseteq \overline{X_\sigma}$, onde $\overline{X_\sigma}$ é o fecho de X_σ . Essa relação de ordem reflete como as células de dimensão inferior estão contidas nas de dimensão superior e forma o **poset das faces** de X .

Intuitivamente, isso significa que as células do complexo são organizadas de maneira hierárquica: células de dimensão inferior (como vértices) estão "abaixo" das células de dimensão superior (como faces), e essa relação de inclusão determina como as células se conectam entre si para formar o complexo de células.

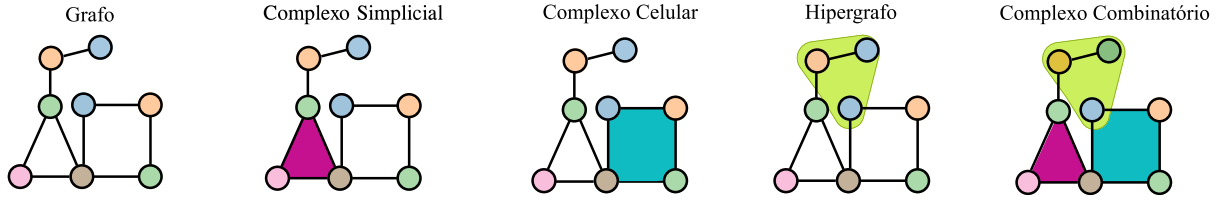
A condição de regularidade (4) adiciona mais detalhes a essa estrutura, introduzindo a *relação de fronteira* $\prec$. Dizemos que $\sigma \prec \tau$ se $\sigma < \tau$ e não existe nenhuma célula δ tal que $\sigma < \delta < \tau$, onde o símbolo $<$ denota a versão estrita da relação de ordem.

Essa relação de fronteira descreve como as células se "tocam" ou se "encostam". Por exemplo, no caso de um cubo, os vértices formam as bordas das arestas, e as arestas formam as bordas das faces. A relação de fronteira captura essas interações e conexões entre as células, organizando a topologia do complexo. Portanto, toda a informação topológica sobre X está codificada na estrutura de poset de P_X , e a relação de fronteira nos permite entender como as células estão relacionadas umas às outras no espaço topológico.

2.5 Complexos combinatórios

Complexos combinatórios (CCs) buscam definir objetos que contemplem tanto estruturas hierárquicas (como complexos celulares e simpliciais) quanto estruturas relacionais mais gerais, como hipergrafos. Ao invés de uma hierarquia rígida entre as células, os CCs permitem uma

Figura 2 – Exemplos de domínios topológicos e relacionais.



Fonte: De autoria própria.

organização mais flexível, estabelecendo uma relação de ordem (ou *ranking*) entre elas. Mais especificamente, um *complexo combinatório* (CC) (HAJII *et al.*, 2022b) é uma tupla (V, K, rk) , onde V é um conjunto finito, $K \subseteq 2^V \setminus \emptyset$ é um conjunto de células e $\text{rk} : K \rightarrow \mathbb{Z}_{\geq 0}$ é uma função de ordenação, tal que:

1. Para todo $v \in V$, temos que $v \in K$;
2. Para todas as células $\sigma, \sigma' \in K$, se $\sigma \subseteq \sigma'$, então $\text{rk}(\sigma) \leq \text{rk}(\sigma')$.

Note que os CCs exigem apenas a propriedade de preservação da ordem na condição (2) ao mesmo tempo que permitem flexibilidade para acomodar estruturas não hierárquicas, como hipergrafos. A Figura 2 ilustra as estruturas relacionais mais populares em TDL.

2.6 Atributos

Neste trabalho, consideramos estruturas relacionais equipadas com *features* ou atributos. Seja K um conjunto de células ou hiperarestas de um domínio relacional. A versão com atributos de K é uma tupla (K, x) em que $x : K \rightarrow \mathbb{R}^d$ atribui um vetor de características $x(\sigma)$ a cada célula σ . Daqui em diante, designamos as características de σ por x_σ .

2.7 Estruturas de Vizinhaça

Podemos explorar as relações de fronteira (ou funções de ordenação) para especificar vizinhos locais para cada célula. Em particular, Bodnar *et al.* (2021b) introduzem quatro estruturas de vizinhaça $\mathcal{N}$: fronteira, co-fronteira, adjacência superior e adjacência inferior.

A vizinhaça do tipo fronteira $\mathcal{N}_B(\sigma)$ de uma célula σ é o conjunto de todos os elementos τ que pertencem à fronteira de σ , ou seja:

$$\mathcal{N}_B(\sigma) = \{\tau : \tau \prec \sigma\}.$$

A vizinhaça de co-fronteira $\mathcal{N}_C(\sigma)$ de um elemento σ é o conjunto de todos os elementos

τ tal que σ pertence a sua fronteira, ou seja:

$$\mathcal{N}_C(\sigma) = \{\tau : \sigma \prec \tau\}.$$

A adjacência superior $\mathcal{N}_\uparrow(\sigma)$ de um elemento σ é o conjunto de todos os elementos τ que estão relacionados a σ por meio de algum elemento δ , de tal forma que $\tau \prec \delta$ e $\sigma \prec \delta$. Em termos simples, é a relação de adjacência de σ em direção "acima" ou para "dimensões superiores", onde existe uma conexão comum a σ e τ em uma célula δ , ou seja:

$$\mathcal{N}_\uparrow(\sigma) = \{\tau : \exists \delta \text{ tal que } \tau \prec \delta, \sigma \prec \delta\}.$$

A adjacência inferior $\mathcal{N}_\downarrow(\sigma)$ de um elemento σ é o conjunto de todos os elementos τ que estão relacionados a σ por meio de algum elemento δ , de tal forma que $\delta \prec \tau$ e $\delta \prec \sigma$. Ou seja, σ e τ são “conectados” por um elemento comum que é “inferior” a ambos:

$$\mathcal{N}_\downarrow(\sigma) = \{\tau : \exists \delta \text{ tal que } \delta \prec \tau, \delta \prec \sigma\}.$$

Exemplo 2.2. *Seja um complexo K composto pelas seguintes células $\{\{0\}, \{1\}, \{2\}, \{0, 1\}, \{0, 2\}, \{1, 2\}, \{0, 1, 2\}\}$ e $\sigma = \{0, 1\}$. Nesse caso, temos que :*

$$\mathcal{N}_B(\sigma) = \{\{0\}, \{1\}\}, \quad (2.1)$$

$$\mathcal{N}_C(\sigma) = \{\{0, 1, 2\}\}, \quad (2.2)$$

$$\mathcal{N}_\uparrow(\sigma) = \{\{0, 2\}, \{1, 2\}\}, \quad (2.3)$$

$$\mathcal{N}_\downarrow(\sigma) = \{\{0, 2\}, \{1, 2\}\}. \quad (2.4)$$

Análogos dessas vizinhanças e outras estruturas também podem ser obtidos por meio de funções de ordenação (HAJII *et al.*, 2022b).

Uma noção instrumental para utilizar essas estruturas é a de matrizes de incidência, uma vez que elas são utilizadas para definir estruturas de vizinhança e Laplacianas de Hodge (GOLDBERG, 2002). Formalmente, para toda dimensão (ou rank) $r > 1$, $B_r \in \{-1, 0, 1\}^{N_{r-1} \times N_r}$ é tal que $(B_r)_{ij}$ é ± 1 se o (r) -símplice j é *coface* do $(r-1)$ -símplice i e 0 caso contrário. A escolha do sinal das entradas não-zero é arbitrária e tipicamente dependente da aplicação.

2.8 Cadeias, Operadores de Fronteira e Ciclos

Nesta seção serão abordadas algumas noções básicas de topologia algébrica. Para simplificar a exposição, consideramos complexos simpliciais abstratos equipados com coeficientes no corpo finito $\mathbb{Z}/2\mathbb{Z} = \{0, 1\}$.

O espaço das n -cadeias é o espaço vetorial formado por todas as somas formais dos símlices de dimensão n de um complexo simplicial abstrato K . Formalmente, seja $n \geq 0$ e

$$K_{(n)} = \{\sigma \in K : \dim(\sigma) = n\},$$

o n -esqueleto de K , ou seja, o conjunto dos símlices de dimensão n . As n -cadeias de K são o conjunto $C_n(K)$ cujos elementos possuem a forma

$$c = \sum_{\sigma \in K_{(n)}} \epsilon_\sigma \sigma, \quad (2.5)$$

onde para todo $\sigma \in K_{(n)}$, $\epsilon_\sigma \in \mathbb{Z}/2\mathbb{Z}$ representa o coeficiente associado ao símlice σ (VIRK, 2022, p. 85).

Sejam c e c' duas n -cadeias. A soma de duas cadeias $(c + c')$ e o produto de uma cadeia por um escalar (λc) são respectivamente definidos por

$$c + c' = \sum_{\sigma \in K_{(n)}} (\epsilon_\sigma + \epsilon'_\sigma) \sigma, \quad (2.6)$$

$$\lambda c = \sum_{\sigma \in K_{(n)}} (\lambda \cdot \epsilon_\sigma) \sigma, \quad (2.7)$$

onde as operações de soma e multiplicação são realizadas módulo 2, isto é, na aritmética do corpo $\mathbb{Z}/2\mathbb{Z}$.

Definimos a fronteira de um n -simplex σ , denotada por $\partial_n \sigma$, como a soma de seus $(n - 1)$ -símlices constituintes, isto é,

$$\partial_n \sigma = \sum_{\tau \subset \sigma : |\tau| = |\sigma| - 1} \tau, \quad (2.8)$$

onde a soma é feita módulo 2 e a notação $\tau \subset \sigma$ indica que τ é uma face de σ de dimensão imediatamente inferior.

Esse operador de fronteira se estende linearmente aos espaços de cadeias. Em particular, o operador de fronteira ∂_n é uma função linear $\partial_n : C_n(K) \rightarrow C_{n-1}(K)$ que é definida pela seguinte equação:

$$\partial_n c = \partial_n \left(\sum_{\sigma \in K_{(n)}} \epsilon_\sigma \sigma \right) = \sum_{\sigma \in K_{(n)}} \epsilon_\sigma \partial_n \sigma. \quad (2.9)$$

Um fato fundamental na topologia algébrica é que a composição de operadores de fronteira consecutivos é nula, ou seja,

$$\partial_{n-1} \circ \partial_n = 0 \quad \text{para todo } n \geq 1. \quad (2.10)$$

Dessa forma, as imagens dos operadores ∂_n estão contidas nos núcleos (kernel) dos operadores ∂_{n-1} . Isto permite definir o conjunto dos n -ciclos de K que é o núcleo do operador de fronteira ∂_n ,

$$Z_n(K) = \ker \partial_n = \{c \in C_n(K) : \partial_n c = 0\}. \quad (2.11)$$

Exemplo 2.3. Considere o complexo simplicial $K = \{\{v_0\}, \{v_1\}, \{v_2\}, \{v_0, v_1\}, \{v_1, v_2\}, \{v_2, v_0\}\}$. Esse complexo forma um ciclo triangular de arestas. A 1-cadeia

$$c = \{v_0, v_1\} + \{v_1, v_2\} + \{v_2, v_0\}, \quad (2.12)$$

obtida somando formalmente as três arestas é um exemplo de 1-ciclo em K . De fato, aplicando o operador de fronteira ∂_1 a c , obtemos:

$$\partial_1(c) = \partial_1(\{v_0, v_1\}) + \partial_1(\{v_1, v_2\}) + \partial_1(\{v_2, v_0\}) = (v_0 + v_1) + (v_1 + v_2) + (v_0 + v_2). \quad (2.13)$$

Agrupando os termos iguais e lembrando que a soma é feita módulo 2, cada vértice aparece duas vezes na expressão acima. Assim, $\partial_1(c) = 2v_0 + 2v_1 + 2v_2 = 0$ em $\mathbb{Z}/2\mathbb{Z}$. Como $\partial_1(c) = 0$, concluímos que $c \in Z_1(K)$, isto é, c é um ciclo de dimensão 1.

3 REDES NEURAIIS TOPOLÓGICAS

Este capítulo apresenta as principais arquiteturas de redes neurais topológicas, incluindo redes neurais para grafos. Além disso, serão discutidos os métodos mais populares de elevação de grafos para domínios relacionais de ordem superior.

3.1 Elevando Grafos a Objetos de Ordem Superior

A maioria dos *benchmarks* para avaliação de TNNs compreende tarefas preditivas em grafos. Para que seja possível aplicar TNNs a esses problemas, é necessário um processo de transformação conhecido como elevação (*lifting*), que mapeia grafos para domínios topológicos de ordem superior (TELYATNIKOV *et al.*, 2024; HAJIJ *et al.*, 2022b; BERNÁRDEZ *et al.*, 2024). Esse processo de elevação pode ser pré-definido com base em características estruturais, como proximidade de nós, presença de ciclos, ou aprendido diretamente dos dados (BATTILORO *et al.*, 2024; KAZI *et al.*, 2022).

Um *lifting* é um mapeamento $\text{lift} : \mathbb{G} \rightarrow \mathbb{T}$ do espaço de grafos com atributos, $\mathbb{G}$, para um domínio alvo, $\mathbb{T}$, tal que $G \cong_{\mathbb{G}} G'$ implica que $\text{lift}(G) \cong_{\mathbb{T}} \text{lift}(G')$, onde $\cong_{\mathbb{T}}$ denota a relação de *isomorfismo* no domínio $\mathbb{T}$. Uma lista de diferentes tipos de *liftings* pode ser encontrada em Telyatnikov *et al.* (2024).

3.1.1 Elevação por Cliques

O conjunto de cliques em um grafo G é dado por

$$Cl(G) = \{c \subseteq V(G) : u \neq v \in c \implies \{u, v\} \in E(G)\},$$

isto é, cada elemento de $Cl(G)$ é um subgrafo completo de G . Os k -cliques de G são elementos de $Cl(G)$ de tamanho k , para $k > 1$, e são denotados por $Cl_k(G)$. Formalmente, a operação de elevação por k -cliques é dada por

$$\text{lift}_{\text{clique},k}(G) = V(G) \cup \bigcup_{i=2}^k Cl_i(G), \quad (3.1)$$

sendo $V(G)$ o conjunto de vértices do grafo. Note que a inclusão de todas as cliques de tamanho menor ou igual a k garante a propriedade de fechamento por inclusão necessária a um complexo simplicial (qualquer face de uma face pertence também ao complexo).

3.1.2 Elevação por Ciclos

A ideia da elevação por ciclos é identificar elementos em uma base de ciclos no grafo de entrada e usar o conjunto de vértices em cada ciclo como uma célula de dimensão 2 do complexo celular resultante.

Consideremos somas módulo 2 para vértices e arestas. Seja ∂_1 o operador de fronteira para arestas do grafo G , ou seja, $\partial_1(\{u, v\}) = \{u\} + \{v\}$ para qualquer aresta $\{u, v\} \in E(G)$. Os ciclos de G são dados por

$$L(G) = \left\{ l \subseteq E(G) : \sum_{e \in l} \partial_1(e) = 0 \right\}.$$

Uma base cíclica de G é uma coleção mínima de ciclos tal que qualquer ciclo em G possa ser escrito como soma desses ciclos, isto é, o menor conjunto $B \subseteq L$ tal que, para todo $l \in L$, exista um subconjunto $B' \subseteq B$ com

$$l = \sum_{b \in B'} b.$$

A função de elevação por ciclos é

$$\text{lift}_{\text{cycle}}(G) = V(G) \cup E(G) \cup \{V(b) : b \in B(G)\}, \quad (3.2)$$

sendo $V(b)$ o conjunto de vértices do ciclo b , e $\{V(b) : b \in B(G)\}$ é o conjunto das células de dimensão 2.

3.1.3 Elevação por k -saltos

A vizinhança de k -saltos de um nó $v \in V(G)$ é definida por

$$N_k[v] = \{u \in V(G) : \text{dist}(u, v) \leq k\}, \quad (3.3)$$

sendo $\text{dist}(u, v)$ a distância do caminho mais curto no grafo G , medida pelo número de arestas no caminho. Para construir o hipergrafo de k -saltos H a partir de G , forma-se uma hiperaresta para cada nó $v \in V(G)$ baseada em sua vizinhança de k -saltos:

$$\text{lift}_{k\text{-hop}}(G) = \{N_k[v] : v \in V(G)\}.$$

O parâmetro k controla a abrangência das vizinhanças incluídas como hiperarestas, com valores maiores de k incorporando progressivamente nós mais distantes.

3.1.4 Elevação por k Vizinhos mais Próximos

A elevação k -NN (*nearest neighbors*) constrói hiperarestas identificando os k vizinhos mais próximos com base nas características dos nós (espaço de *features*). Para cada nó, forma-se uma hiperaresta que inclui o próprio nó e seus k vizinhos mais próximos.

3.1.5 Elevação por Kernel

A elevação por *kernel* é um procedimento que constrói hiperarestas baseado em medidas de similaridade derivadas de kernels aplicados aos nós do grafo. Esses kernels podem ser definidos de três formas:

- i) Sobre a estrutura do grafo
- ii) sobre as características dos nós, ou
- iii) como uma composição que incorpora conjuntamente ambas as informações.

Para um nó de referência v , calcula-se a similaridade entre v e todos os demais nós v' usando uma função de kernel. Uma hiperaresta é então formada selecionando uma fração fixa (tipicamente 0,5) dos nós mais similares a v segundo o kernel escolhido. Esse processo é repetido para cada nó para construir o conjunto de hiperarestas.

Os kernels podem ser definidos nas seguintes formas: sobre nós $K_g(v, v^*)$, sobre características $K_x(x, x^*)$, ou sobre nós e características conjuntamente $C(K(x, x^*, v, v^*))$, onde C é uma função de composição válida. Kernels sobre características são calculados por kernels RBF ou exponenciais (DUVENAUD, 2014), enquanto kernels sobre grafos podem ser calculados por kernels de temperatura ou Matérn (SCHÖLKOPF; SMOLA, 2002; BOROVITSKIY *et al.*, 2021; NIKITIN *et al.*, 2022).

3.1.6 Differentiable Cell Complex Module

Battiloro *et al.* (2024) propuseram uma camada inovadora composta por vários módulos, sendo o Differentiable Cell Complex Module (DCM) o elemento central para a inferência de topologia latente. O DCM começa amostrando o 1-esqueleto do complexo celular latente por meio do α -Differentiable Graph Module (α -DGM). Em seguida, identifica polígonos — que representam interações de ordem superior — ao detectar ciclos no grafo amostrado, utilizando o Polygon Inference Module (PIM). Para uma explicação detalhada dos módulos α -DGM e PIM,

remetemos o leitor à Seção 3 de (BATILORO *et al.*, 2024). A topologia inferida é então usada em uma rede de passagem de mensagem de complexo celular, operando nos níveis de nó e aresta. Toda a arquitetura é treinada de forma ponta a ponta.

3.1.7 Elevação de Atributos (*features*)

Em geral, atributos são definidos somente para os nós do grafo. Assim, na elevação de grafos para objetos de ordem superior, é necessário definir como os atributos dos vértices serão propagados para obter atributos das células de dimensão maior que 0.

Seja σ uma célula/simplex de dimensão maior que 0. Na abordagem mais comum, chamada aqui de *lifting de soma*, o valor de $x(\sigma)$ (vetor de features de σ) é dado pela soma das features dos elementos na vizinhança de fronteira de σ , i.e., $x(\sigma) = \sum_{\tau \in \mathcal{N}_B(\sigma)} x(\tau)$. Assim, features de células de dimensão arbitrária podem ser obtidas recursivamente a partir das features dos vértices do grafo.

3.2 Redes Neurais Topológicas

Redes neurais topológicas (*topological neural networks*, TNNs) generalizam redes para grafos ao estender sua aplicação para estruturas de ordem superior, como complexos simpliciais e complexos celulares. A seguir, serão discutidas redes neurais para grafos e suas extensões topológicas, com ênfase em variantes populares e projetadas para complexos simpliciais e complexos celulares.

3.2.1 Redes Neurais para Grafos

De modo similar às redes convolucionais para imagens, redes neurais para grafos (*graph neural networks*, GNNs) (SCARSELLI *et al.*, 2009; XU *et al.*, 2019) combinam sequencialmente operações de filtragem local, não linearidade e, opcionalmente, *pooling* para capturar representações do grafo em cada camada. No paradigma dominante, GNNs aprendem por meio de um mecanismo de passagem de mensagem, no qual as mensagens vetoriais são trocadas entre os nós e atualizadas como redes neurais (HAMILTON; YING; LESKOVEC, 2017).

A cada iteração/camada ℓ da GNN, aplicamos uma função de agregação *Agg* que recebe como entrada o conjunto de *embeddings* dos nós na vizinhança de cada nó $v \in V$, $\mathcal{N}(v)$, e gera uma mensagem $m_v^{(\ell)}$ com base nessas informações agregadas da vizinhança de v . Em seguida, aplicamos uma função de atualização *Upd* que recebe $m_v^{(\ell)}$ e o *embedding* de v na

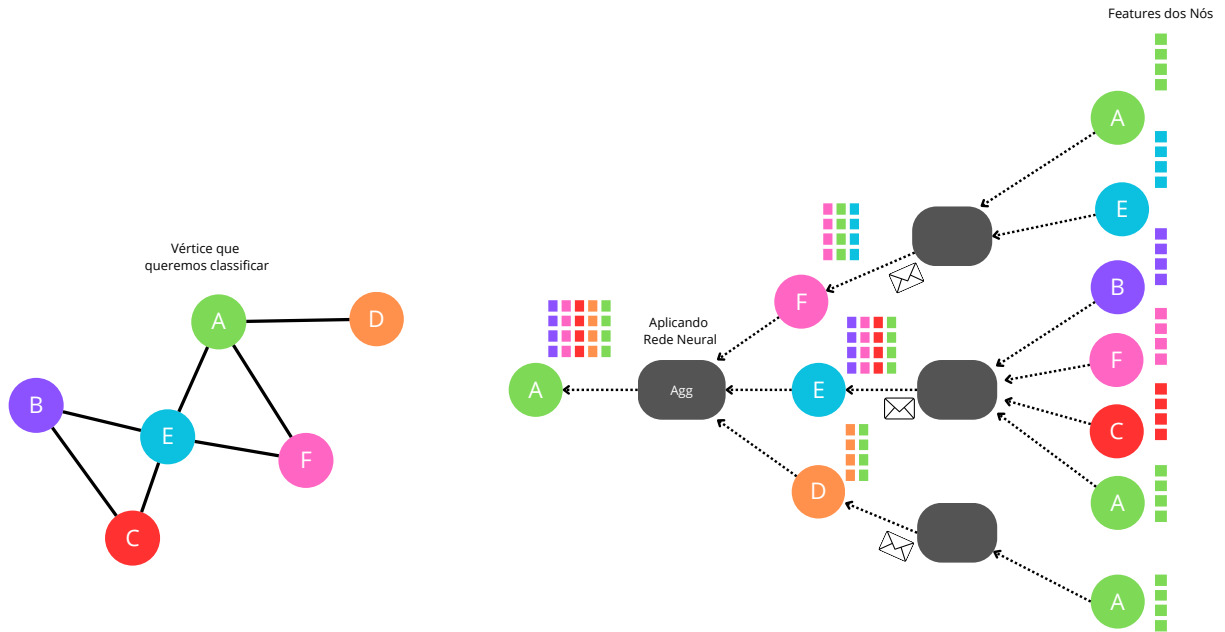
camada anterior. Matematicamente, o processo de passagem de mensagem pode ser descrito recursivamente como

$$m_v^{(\ell)} = \text{Agg}^{(\ell)}(\{h_u^{(\ell-1)} : u \in \mathcal{N}(v)\}), \quad \forall v \in V, \quad (3.4)$$

$$h_v^{(\ell)} = \text{Upd}^{(\ell)}(h_v^{(\ell-1)}, m_v^{(\ell)}), \quad \forall v \in V, \quad (3.5)$$

onde Upd é uma função arbitrária diferenciável (por exemplo, uma rede MLP) e Agg é uma função arbitrária invariante à ordem. Além disso, temos que $h_v^{(0)} = x_v \quad \forall v \in V$. A Figura 3 ilustra o processo de passagem de mensagem em GNNs.

Figura 3 – Visão geral de como um vértice agrega mensagens de sua vizinhança local. O modelo agrega mensagens dos vizinhos de A (ou seja, E , F e D) e, por sua vez, as mensagens vindas desses vizinhos são baseadas em informações agregadas de suas respectivas vizinhanças, e assim por diante. Esta visualização mostra uma versão de duas camadas de um modelo de passagem de mensagens.



Fonte: De autoria própria.

3.2.1.1 Graph Convolutional Network (GCN)

O modelo *Graph Convolutional Network* (GCN, (KIPF; WELLING, 2016)) foi provavelmente o principal responsável por popularizar GNNs, possivelmente devido à sua simplicidade. A função Agg da GCN é uma soma ponderada dos vizinhos do nó $u \in \mathcal{N}(v)$, de maneira que o embedding do nó u recebe peso inversamente proporcional ao produto das raízes de $\bar{d}_u = |\mathcal{N}(u)| + 1$ e $\bar{d}_v = |\mathcal{N}(v)| + 1$. Já a função Upd consiste em somar o embedding de cada

nó v dividido por $\bar{d}_v$ com a mensagem proveniente de seus vizinhos e aplicar uma camada linear seguida de uma função de ativação ψ (tipicamente uma função *Rectified Linear Unit*, RELU). Esse processo pode ser escrito como:

$$m_v^{(\ell)} = \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{\bar{d}_u \bar{d}_v}} h_u^{(\ell-1)} \quad \forall v \in V, \quad (3.6)$$

$$h_v^{(\ell)} = \psi \left(\left(\frac{1}{\bar{d}_v} h_v^{(\ell-1)} + m_v^{(\ell)} \right) \Theta_\ell \right) \quad \forall v \in V, \quad (3.7)$$

onde $\Theta_\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ é uma matriz de parâmetros/pesos. Vale ressaltar que as equações acima podem ser concisamente representadas por $H^{(\ell)} = \psi \left(\tilde{A} H^{(\ell-1)} \Theta_\ell \right)$. Note que GCNs utilizam uma única matriz de pesos por camada, e de forma geral, o processamento é bem parecido com o de uma rede neural MLP, com a principal diferença sendo a etapa de difusão de embeddings de nós no grafo através da multiplicação por $\tilde{A}$.

3.2.1.2 Simple Graph Convolution (SGC)

O modelo *simple graph convolution* (SGC) (WU *et al.*, 2019) consiste em uma simplificação da GCN. Note que podemos expandir a recursão da GCN para obter

$$H^{(\ell)} = \psi \left(\tilde{A} \psi \left(\dots \psi \tilde{A} \left(\tilde{A} X \Theta^{(1)} \right) \Theta^{(2)} \dots \right) \Theta^{(\ell)} \right). \quad (3.8)$$

Para reduzir o custo computacional das GCNs, o modelo SGC remove as não linearidades entre as camadas da GCN. Como resultado, é possível colapsar os parâmetros do modelo com L camadas em uma única matriz $\Theta = \Theta^{(1)} \Theta^{(2)} \dots \Theta^{(L)}$. Desta forma, a função de atualização para calcular a matriz de representações do SGC é definida como:

$$H = \tilde{A}^L X \Theta. \quad (3.9)$$

Surpreendentemente, o modelo SGC mostrou desempenho igual ou superior às GCNs para a tarefa de classificação de nós, mas é mais eficiente e possui menos parâmetros ajustáveis.

3.2.1.3 Graph Isomorphism Network (GIN)

Xu *et al.* (2019) exploram o fato de que o processo de passagem de mensagem se assemelha fortemente a uma heurística para detecção de isomorfismo em grafos chamada de teste de Weisfeiler-Lehman (WL) (WEISFEILER; LEMAN, 1968). Nessa heurística, inicializamos os nós de ambos os grafos que queremos verificar com cores que representam seus atributos (ou uma cor padrão, caso atributos de nó não estejam disponíveis). Nos passos seguintes do teste

WL, verificamos se os multisets de cores relativos a cada grafo são idênticos. Caso positivo, os nós atualizam suas cores em paralelo, aplicando uma função de HASH que toma como entrada a sua cor e o multiset com as cores de seus vizinhos. Caso contrário, o teste para indicando que os grafos não são isomórficos. Se os multisets não mudam entre duas iterações, o algoritmo encerra com resultado inconclusivo.

Note que a função HASH age de maneira análoga à composição das funções AGG e UPD em uma GNN, com a restrição de que HASH é uma função injetiva. Intuitivamente, então, basta que a composição $UPD \circ AGG$ seja injetiva para que uma GNN seja tão poderosa quanto o teste WL. A fim de alcançar esse objetivo, *graph isomorphism networks* (GINs) (XU *et al.*, 2019) usam a seguinte implementação de passagem de mensagem:

$$m_v^{(\ell)} = \sum_{u \in \mathcal{N}(v)} h_u^{(\ell-1)} \quad \forall v \in V, \quad (3.10)$$

$$h_v^{(\ell)} = \text{MLP}^{(\ell)} \left((1 + \epsilon^{(\ell)}) h_v^{(\ell-1)} + m_v^{(\ell)} \right) \quad \forall v \in V, \quad (3.11)$$

onde $\text{MLP}^{(\ell)}(\cdot)$ representa um perceptron multicamada. Se os atributos iniciais de cada nó forem do tipo *one-hot encodings*, então podemos fazer $h_v^{(0)} = x_v$; caso contrário, utilizamos $h_v^{(0)} = \text{MLP}(x_v)$. O primeiro caso acontece pois a soma de vetores *one-hot* já é injetiva. O *embedding* final de cada nó após a última camada é $h^{(L)}$.

Utilizando $h^{(L)}$ para tarefas de predição. No contexto de aprendizado em grafos, neste trabalho, estamos principalmente interessados em tarefas preditivas a nível de nó e grafo:

- *Classificação de nós:* Essa tarefa envolve a determinação da categoria ou classe de cada nó em um grafo, sendo útil para problemas como a identificação de comunidades em redes sociais ou a classificação de documentos em redes de citações acadêmicas (KIPF; WELLING, 2016).
- *Classificação de grafo:* Trata-se da tarefa de classificar uma característica discreta ou contínua de um grafo ou subgrafo. A previsão de características de grafos é particularmente útil em áreas onde cada entidade é representada por um grafo separado, e.g., classificação de propriedades de moléculas (XU *et al.*, 2019).

As subseções acima descreveram como extrair *embeddings* para cada nó, condensando informação sobre a estrutura do grafo ao redor do mesmo. Quando estamos interessados em classificar ou prever uma propriedade de um nó $v \in V$, usamos seu *embedding* final $h_v^{(L)}$ como entrada para uma rede neural feedforward (e.g., MLP) para obter predições da variável de saída.

É importante ressaltar que as GNNs são treinadas de maneira supervisionada (*end-to-end*). Ou seja, tanto os parâmetros da GNN quanto da rede feedforward são treinados de maneira conjunta, propagando gradientes a partir de uma função de perda supervisionada (e.g., entropia cruzada ou erro quadrático médio). No caso em que queremos classificar ou prever uma propriedade do grafo G , combinamos os embeddings $(h_v^{(L)})_{v \in V}$ de todos os seus nós usando uma função invariante a permutação (também chamada de *pooling* ou *readout*) para obter um embedding $h_G^{(L)}$ para o grafo. Novamente, usamos $h_G^{(L)}$ como entrada para uma rede feedforward que produz as predições do nosso modelo.

3.2.2 Redes de Passagem de Mensagem de Alta Ordem

Devido ao fato de o poder expressivo das GNNs baseadas em message passing ser, no máximo, equivalente ao do teste de Weisfeiler–Leman de 1-cor (1-WL), tais redes não conseguem, em geral, resolver certas tarefas nos grafos — por exemplo, detectar ou contar estruturas locais como triângulos ou cliques — sem recorrer a extensões arquiteturais. Essa limitação inerente tem motivado o desenvolvimento de modelos mais expressivos, explorando, por exemplo, redes neurais topológicas (TNNs) (BODNAR *et al.*, 2021b), features aleatórias (SATO; YAMADA; KASHIMA, 2021) e codificações estruturais/posicionais (LI *et al.*, 2020; YOU; YING; LESKOVEC, 2019).

Como as GNNs, a maioria das TNNs emprega mecanismos de passagem de mensagens para obter representações de nível de célula (PAPILLON *et al.*, 2023). Em particular, seja $(\mathcal{N}_i)_i$ uma sequência finita de estruturas de vizinhança, que representam as interações entre os elementos no espaço relacional. Na sua forma geral, começando com $h_\sigma^0 = x_\sigma$ para todas as células σ , onde x_σ é o vetor de características associado à célula σ , uma TNN de passagem de mensagens (BODNAR *et al.*, 2021a) calcula recursivamente o estado das células em camadas sucessivas. O cálculo das mensagens $m_{i,\sigma}^\ell$ para cada célula σ é dado pelos seguintes multiconjuntos:

$$m_{i,\sigma}^\ell = \begin{cases} \{\!\!\{ \phi_{\ell,i}(h_\tau^\ell, h_\sigma^\ell, h_\delta^\ell) : \tau \in \mathcal{N}_i(\sigma), \delta \in \mathcal{N}_B(\sigma, \tau) \}\!\!\}, & \text{se } \mathcal{N}_i = \mathcal{N}_\downarrow \\ \{\!\!\{ \phi_{\ell,i}(h_\tau^\ell, h_\sigma^\ell, h_\delta^\ell) : \tau \in \mathcal{N}_i(\sigma), \delta \in \mathcal{N}_C(\sigma, \tau) \}\!\!\}, & \text{se } \mathcal{N}_i = \mathcal{N}_\uparrow \\ \{\!\!\{ \phi_{\ell,i}(h_\tau^\ell, h_\sigma^\ell) : \tau \in \mathcal{N}_i(\sigma) \}\!\!\}, & \text{caso contrário.} \end{cases} \quad (3.12)$$

onde $\mathcal{N}_C(\sigma, \tau) = \mathcal{N}_C(\sigma) \cap \mathcal{N}_C(\tau)$ e $\mathcal{N}_B(\sigma, \tau) = \mathcal{N}_B(\sigma) \cap \mathcal{N}_B(\tau)$. Além disso, a função $\phi_{\ell,i}$ representa o cálculo das mensagens entre as células τ , σ e δ , dependendo do tipo de vizinhança envolvida, seja ela inferior ($\mathcal{N}_\downarrow$), superior ($\mathcal{N}_\uparrow$) ou baseadas em fronteira. A atualização do estado

das células h_σ^ℓ na camada ℓ para a camada seguinte $\ell + 1$ é:

$$h_\sigma^{\ell+1} = \varphi \left(h_\sigma^\ell, \bigotimes_i \text{Agg}_\ell(m_{i,\sigma}^\ell) \right) \quad (3.13)$$

onde h_σ^ℓ é a representação da célula σ na camada ℓ . A operação de agregação $\bigotimes$ combina as mensagens de diferentes vizinhanças e a função de agregação Agg_ℓ aplica uma agregação sobre as mensagens recebidas de todas as células vizinhas. A função φ é responsável por atualizar a representação da célula, geralmente utilizando uma rede neural do tipo MLP.

3.2.2.1 Simplex Convolutional Network (SCN)

O modelo SCN (YANG; SALA; BOGDAN, 2022) é um método de aprendizado de representação para complexos simpliciais. Ele pode ser visto como uma extensão natural do modelo GCN, onde a difusão no grafo com a matriz de adjacência normalizada é substituída por adjacências baseadas em Laplacianas de Hodge.

Para matriz de fronteira B com *ranking* $r > 0$, denotamos por $\Delta_r = B_{r-1}^T B_{r-1} + B_r B_r^T$ a matriz Laplaciana de Hodge de dimensão r . A matriz de adjacência correspondente com *self-loops* adicionados é dada por $A_r = \Delta_r + 2I_{N_r}$. Também definimos as matrizes de adjacência normalizadas como $\tilde{A}_r = D_r^{-1/2} A_r D_r^{-1/2}$ para $r = 0 \dots R$, onde $(D_r)_{ii} = \sum_j (A_r)_{ij}$ é a matriz diagonal de graus para os r -símplices. Na camada ℓ e para a dimensão r , a saída da camada do SCN é dada por:

$$H_r^{(\ell)} = \psi(\tilde{A}_r H_r^{(\ell-1)} W_r^{(\ell)}) \quad (3.14)$$

com $H_r^{(0)} = X_r$ — a matriz X_r contém os atributos dos símplices de dimensão r ao longo das linhas. Para a tarefa de classificação de nós é aplicada uma camada de *readout* feita recursivamente sobre a saída calculada para cada dimensão. Seja $Z_R = H_R^{(L)}$, realizamos o *readout* de dimensões superiores para inferiores da seguinte forma:

$$Z_{r-1} = \left[\psi(B_r Z_r \Theta_r^{(1)}) \parallel H_{r-1}^{(L)} \right] \Theta_r^{(2)} \quad \text{para } 1 \leq r \leq R \quad (3.15)$$

onde $\Theta_r^{(1)}$ e $\Theta_r^{(2)}$ são matrizes de parâmetros, $\psi(\cdot)$ uma função de ativação não linear e $\parallel$ a operação de concatenação. Por fim, a predição para nós é dada por

$$Y_0 = \text{softmax}(Z_0 \Theta_0) \quad (3.16)$$

Em que Θ_0 mapeia os vetores-linha de Z_0 para $\mathbb{R}^{C_0}$ com C_0 denotando o número de classes para os 0-símplices.

Na literatura também é proposta uma variante chamada de *simplicial complex convolutional networks* (SCCN) que considera troca de mensagem entre s mplices de dimens es adjacentes (HAJJI *et al.*, 2022a). Nesse caso, SCCN utiliza uma matriz de adjac ncia completa A dada por

$$A = \begin{bmatrix} \alpha A_0 & \beta B_0 & 0 \\ \beta B_0^T & \alpha A_1 & \beta B_1 \\ 0 & \beta B_1^T & \alpha A_2 \end{bmatrix} \quad (3.17)$$

A matriz A tem as matrizes de r -adjac ncia A_r na diagonal principal e as matrizes de incid ncia B_r nas diagonais superiores e inferiores. As vari veis α e β s o par metros ajust veis. A matriz A_r captura rela  es dentro da mesma dimens o, enquanto B_r mostra conex es entre um r -simplex e um $(r - 1)$ -simplex que   uma face do r -simplex. O SCCN usa a opera  o convolucional na matriz de adjac ncia completa A conforme a Equa  o 3.14.

3.2.2.2 CW Networks (CWN)

Bodnar *et al.* (2021a) prop em um framework geral para passagem de mensagem em complexos celulares, denominado CW networks (CWNs). Dentre os modelos propostos, *cell isomorphism networks* (CINs) seguem um desenho arquitetural baseado em GINs (XU *et al.*, 2019). Em particular, CINs empregam

$$m_B^{\ell+1}(\sigma) = \text{MLP}_{B,r(\sigma)}^\ell \left((1 + \epsilon_1) h_\sigma^\ell + \sum_{\tau \in \mathcal{N}_B(\sigma)} h_\tau^\ell \right) \quad (3.18)$$

$$m_\uparrow^{\ell+1}(\sigma) = \text{MLP}_{\uparrow,r(\sigma)}^\ell \left((1 + \epsilon_2) h_\sigma^\ell + \sum_{\tau \in \mathcal{N}_\uparrow(\sigma)} \sum_{\delta \in \mathcal{N}_C(\sigma) \cap \mathcal{N}_C(\tau)} \text{MLP}_{r(\sigma)}^\ell(h_\delta^\ell \parallel h_\tau^\ell) \right) \quad (3.19)$$

$$h_\sigma^{\ell+1} = \text{MLP}_{\text{upd}}^\ell (m_B^{\ell+1}(\sigma) \parallel m_\uparrow^{\ell+1}(\sigma)), \quad (3.20)$$

onde ϵ_1, ϵ_2 s o constantes, $\text{MLP}_{B,r(\sigma)}^\ell$ e $\text{MLP}_{\uparrow,r(\sigma)}^\ell$ s o MLPs de duas camadas, e $\text{MLP}_{\text{upd}}^\ell$ e $\text{MLP}_{r(\sigma)}^\ell$ s o camadas densas seguidas por fun  es de ativa  o do tipo ReLU.

Para tarefas de predi  o a n vel de complexo, as representa  es finais das c lulas s o combinadas usando uma camada de *readout* (por exemplo, m dia ou soma) antes de passar por um “cabe a” de classifica  o (por exemplo, MLP).

3.2.2.3 Cell Complex Neural Networks (CXN)

Hajji, Istvan e Zamzmi (2020) tamb m exploram redes de passagem de mensagem para complexos celulares. A abordagem proposta   muito similar ao modelo CWN — comparativa-

mente CXNs utilizam um número menor de estruturas de vizinhança. Em particular, em uma das variantes de CXNs¹, dado um *rank* r , o processo de passagem de mensagem é dado por:

$$h_{\sigma}^{\ell+1} = \begin{cases} \text{ReLU} \left(\gamma_{\sigma} h_{\sigma}^{\ell} \Theta_0^{\ell} + \sum_{\tau \in \mathcal{N}_{\uparrow}(\sigma)} \gamma_{\sigma, \tau} h_{\tau}^{\ell} \Theta_0^{\ell} \right), & \text{se } r(\sigma) = 0 \\ \text{ReLU} \left(\sum_{\tau \in \mathcal{N}_B(\sigma)} h_{\tau}^{\ell} \Theta_2^{\ell} \right), & \text{se } r(\sigma) = 2 \\ h_{\sigma}^0, & \text{se } r(\sigma) = 1 \end{cases} \quad (3.21)$$

onde γ_{σ} e $\gamma_{\sigma, \tau}$ são coeficientes que ajustam a influência das células vizinhas na célula alvo — podem ser aprendidos (como mecanismo de atenção) ou fixos (como em GCNs).

3.2.2.4 Unified Framework for Graph and Hypergraph Neural Networks (UniGNN)

UniGNN (HUANG; YANG, 2021) denota uma classe de modelos que visa generalizar GNNs para hipergrafos. Ela permite que qualquer GNN possa ser aplicada a hipergrafos. Mais especificamente, UniGNN emprega duas agregações. Seja $E_v = \{e \in E | v \in e\}$ conjunto de hiperarestas incidentes no vértice v onde E é o conjunto de hiperarestas. Então, UniGNN recursivamente computa:

$$m_e^{\ell} = \phi_1(\{h_u^{\ell}\}_{u \in e}) \quad \forall e \in E \quad (3.22)$$

$$h_v^{\ell+1} = \phi_2(h_v^{\ell}, \{m_e^{\ell}\}_{e \in E_v}) \quad \forall v \in V \quad (3.23)$$

onde ϕ_1 e ϕ_2 denotam funções de agregação invariantes à ordem. É possível mostrar que UniGNNs generalizam GNNs (HUANG; YANG, 2021).

¹ Ver <https://github.com/pyt-team/TopoModelX/blob/main/topomodelx/nn/cell/ccxn_layer.py>

4 ACELERANDO REDES CONVOLUCIONAIS PARA COMPLEXOS SIMPLICIAIS

As redes neurais topológicas (TNNs) têm mostrado grande potencial na modelagem de dados que possuem uma estrutura de relacionamentos complexas, como grafos e complexos simpliciais. No entanto, uma das principais limitações dessa abordagem está no custo computacional elevado associado ao treinamento dessas redes. A necessidade de manipular múltiplas dimensões e grandes matrizes de propagação torna o treinamento dessas redes extremamente intensivo em termos computacionais, o que limita sua aplicabilidade em cenários com grandes conjuntos de dados ou com estruturas topológicas de alta ordem.

Neste contexto, a proposta deste capítulo é apresentar uma versão simplificada e linear das redes convolucionais para complexos simpliciais (SCN), chamada de *simple simplex convolutional network* ou SimpleXCN. Além disso, o objetivo é de reduzir o custo computacional associado ao treinamento dessas redes.

4.1 Simple simplex convolutional network (SimpleXCN)

O SimpleXCN adota uma abordagem similar à apresentada no SGC (WU *et al.*, 2019), onde as não-linearidades são removidas para obtermos modelos lineares. Simplificando o SCN, SimpleXCN remove as não-linearidades entre as camadas intermediárias e colapsa as matrizes de pesos/parâmetros em uma única matriz Θ_r para cada dimensão do complexo simplicial — reduzindo substancialmente o número de parâmetros a serem aprendidos. Além disso, tratamos as matrizes utilizadas na propagação como hiper-parâmetros. Ou seja, assumamos que S_r é uma matriz de propagação (ou de difusão) de dimensão r . Então, podemos escolher $S_r = \Delta_r$ (Laplaciana de Hodge), $S_r = A_r = \Delta_r + 2I_{N_r}$ (adjacência) ou $S_r = \tilde{A}_r = D_r^{-1/2} A_r D_r^{-1/2}$ (adjacência normalizada).

Seguindo o modelo SCN, a propagação entre células de diferentes dimensões ocorre separadamente, seguido por uma etapa de *down-lifting*. Considere um complexo simplicial de dimensão $R = 2$ e uma SCN com $L = 1$ camada. Então, a representação a nível de nó obtida é:

$$H_r^{(1)} = \psi(\tilde{A}_r X_r W_r^{(1)}) \quad (4.1)$$

$$Z_0 = \left[\underbrace{\psi(B_1 [\underbrace{\psi(B_2 \underbrace{H_2^{(1)} \Theta_2^{(1)}}_{Z_2}) \| H_1^{(1)}] \Theta_2^{(2)} \Theta_1^{(1)})}_{Z_1} \| H_0^{(1)} \right] \Theta_1^{(2)}, \quad (4.2)$$

em que $\Theta_\ell \in \mathbb{R}^{d_\ell \times d_{\ell+1}}$ é uma matriz de parâmetros/pesos ajustáveis e $\parallel$ denota a operação de concatenação. Removendo todas as não-linearidades, obtemos

$$Z_0 = \left[B_1 [B_2 \tilde{A}_2 X_2 W_2^{(1)} \Theta_2^{(1)} \parallel \tilde{A}_1 X_1 W_1^{(1)}] \Theta_2^{(2)} \Theta_1^{(1)} \parallel \tilde{A}_0 X_0 W_0^{(1)} \right] \Theta_1^{(2)}, \quad (4.3)$$

$$Z_0 = \left[[B_1 B_2 \tilde{A}_2 X_2 W_2^{(1)} \Theta_2^{(1)} \parallel B_1 \tilde{A}_1 X_1 W_1^{(1)}] \Theta_2^{(2)} \Theta_1^{(1)} \parallel \tilde{A}_0 X_0 W_0^{(1)} \right] \Theta_1^{(2)}, \quad (4.4)$$

e colapsando as matrizes de parâmetros em uma matriz Θ_0 , temos:

$$Z_0 = \left[B_1 B_2 \tilde{A}_2 X_2 \parallel B_1 \tilde{A}_1 X_1 \parallel \tilde{A}_0 X_0 \right] \Theta_0. \quad (4.5)$$

Com isso, podemos agora generalizar essa abordagem para diferentes matrizes de difusão S_r e qualquer dimensão r . Para obtermos representações dos símlices de dimensão r , primeiro calculamos o resultado $S_{r'}^L X_{r'}$ de L passos de difusão de $\tilde{X}_{r'}$ para toda ordem r' , incluindo r . Então, concatenamos às *features* $(S_r^L X_r)_\sigma$ do r -símlice σ pós-difusão as somas das features de todas as cofaces de σ , também após difusão e separadas por ordem. Finalmente, obtemos os logits para classificação do r -símlice σ usando uma transformação linear Θ_r . De modo geral, o processo pode ser descrito sucintamente para todos os símlices de dimensão r como:

$$Z_r = \left[\left(\left\| \prod_{m=r+1}^R \left(\prod_{j=r+1}^m B_j \right) S_m^L X_m \right) \right\| S_r^L X_r \right] \Theta_r, \quad (4.6)$$

onde R é a dimensão do maior símlice do complexo simplicial. Para classificação de nós computamos $Y_0 = \text{softmax}(Z_0)$.

4.2 Experimentos e Resultados

Nesta seção serão detalhados o *setup* experimental, as etapas de pré-processamento aplicadas, as configurações da arquitetura das redes neurais, a métrica de avaliação e os conjuntos de dados utilizados para avaliar o SimpleXCN. A avaliação empírica se concentra no desempenho de redes neurais convolucionais para complexos simpliciais para classificação de nós. Serão apresentados e discutidos os resultados quanto à acurácia, e ao desempenho computacional de forma empírica.

4.2.1 Conjuntos de Dados

Para a análise, foram considerados oito bancos de dados provenientes de *benchmarks* amplamente utilizados, sendo quatro deles representações de redes de citações: Cora, Citeseer, Pubmed e DBLP (YANG; COHEN; SALAKHUDINOV, 2016; BOJCHEVSKI; GÜNNEMANN,

2018). Nestes casos, os nós representam artigos acadêmicos, enquanto as arestas indicam as citações entre os mesmos. As características dos nós são representadas por vetores de *bag of words*, e os rótulos dos nós correspondem aos tópicos acadêmicos dos artigos associados. Além disso, foram utilizados outros quatro bancos de dados caracterizados por conter heterofilia: *Amazon-ratings*, onde os nós representam produtos e as arestas conectam produtos, frequentemente comprados juntos; *Minesweeper* é um conjunto de dados sintético onde o grafo é uma grade regular de 100×100 , cada nó está conectado a oito nós vizinhos, e a tarefa é prever quais desses nós são minas; *Questions* é baseado em um site de perguntas e respostas, onde os nós representam usuários, e uma aresta conecta dois nós se um usuário respondeu à pergunta do outro. A tarefa é prever quais usuários permaneceram ativos no site (PLATONOV *et al.*, 2023). Cada banco de dados foi dividido na proporção de 50% para os dados de treinamento, 25% para validação e 25% para teste. A Tabela 1 apresenta as propriedades dos conjuntos de dados.

Tabela 1 – Propriedades dos conjuntos de dados. É apresentada a quantidade aproximada de nós, arestas, características e classes para cada conjunto de dados. Também descrevemos como os dados são distribuídos entre os conjuntos de teste, validação e treinamento. A proporção de divisão para cada conjunto de dados: metade dos dados é usada para treinamento, 25% é usada para validação, e 25% é usada para teste.

Conjuntos de dados	#Vertices	#Arestas	# Características	#Classes	Treino%	Validação%	Teste%
Cora	~ 2,7K	~ 10K	1,433	7	50	25	25
Citeseer	~ 3,3K	~ 9K	3,703	6	50	25	25
Minesweeper	~ 10,0K	~ 39K	7	2	50	25	25
DBLP	~ 17,7K	~ 105K	1,639	4	50	25	25
Pubmed	~ 19,7K	~ 88K	500	3	50	25	25
Amazon-ratings	~ 24,5K	~ 93K	300	5	50	25	25
Questions	~ 48,9K	~ 153,5K	301	2	50	25	25

4.2.2 Configuração Experimental

Para a comparação de desempenho, foram escolhidos os modelos SCN e SCCN (YANG; SALA; BOGDAN, 2022), pois eles estendem diretamente a abordagem da GCN, proporcionando um cenário ideal para a realização de experimentos com um método de simplificação semelhante ao do SGC. Todos os modelos foram treinados com pelo menos duas camadas de convolução. Também foi realizado um *grid search* utilizando o Optuna (AKIBA *et al.*, 2019) para encontrar os melhores hiperparâmetros. Definimos o espaço de busca para a taxa de aprendizado (de 0.001 a 0.01) com passo de 0.001, o decaimento de pesos entre (0, 5e-4, 5e-6) e a ordem de elevação (2), considerando apenas triângulos. Especificamente para o SimpleXCN, definimos um intervalo

de (2 a 7) para o grau de propagação.

Para as implementações, foi utilizado o *framework* TopoBenchmarkX (PYT-TEAM, 2024), que fornece todos os recursos necessários para calcular o complexo simplicial através da elevação por clique e da elevação de *features* via projeção (*Projection Sum*). Para cada conjunto de dados, o SimpleXCN foi executado por 5 vezes.

4.2.3 Resultados

A análise dos resultados apresentados na Tabela 2 revela que, embora o SimpleXCN não seja consistentemente o melhor modelo em todas as métricas, ele se mantém competitivo, apresentando um bom desempenho em relação aos outros métodos, SCN e SCCN. A média e o desvio padrão das acurácias mostram uma variação de desempenho entre os diferentes conjuntos de dados, indicando que não há uma arquitetura que se destaque de forma clara e definitiva.

Tabela 2 – A comparação de várias arquiteturas em múltiplos bancos de dados. Cada entrada da tabela apresenta a média e o desvio-padrão das acurácias. Apesar de não ter uma melhora significativa o SimpleXCN é competitivo comparativamente com as outras duas arquiteturas, embora seja bem mais simples.

	SCN	SCCN	SimpleXCN
Cora	86.3 $\pm$ 1.2	85.9 $\pm$ 0.3	86.1 $\pm$ 0.6
Citeseer	73.3 $\pm$ 1.7	73.0 $\pm$ 1.1	73.9 $\pm$ 1.6
PubMed	87.2 $\pm$ 0.4	86.5 $\pm$ 0.6	86.9 $\pm$ 0.7
DBLP	82.0 $\pm$ 0.2	82.9 $\pm$ 0.9	81.9 $\pm$ 0.3
Amazon-ratings	46.9 $\pm$ 0.3	44.7 $\pm$ 0.9	42.4 $\pm$ 0.7
Minesweeper	89.5 $\pm$ 0.3	89.8 $\pm$ 0.3	87.4 $\pm$ 0.3
Questions	65.1 $\pm$ 2.4	68.5 $\pm$ 2.5	68.0 $\pm$ 0.7

O SimpleXCN, apesar de ser uma arquitetura mais simples, apresenta um desempenho notável em comparação com o SCN e o SCCN, mostrando que sua simplicidade não compromete sua capacidade de entrega. Por exemplo, no conjunto de dados Cora, o SimpleXCN obteve uma acurácia de 86.1 ± 0.6 , ficando apenas ligeiramente atrás do SCN, que obteve 86.3 ± 1.2 . Em outros casos, como no Citeseer, o SimpleXCN obteve 73.9 ± 1.6 , superando o SCN, que teve 73.3 ± 1.7 .

Com o objetivo de avaliar se as diferenças observadas entre as acurácias do SimpleXCN e das outras arquiteturas são estatisticamente significativas, foi realizado o teste de Wilcoxon (WILCOXON, 1945) — um método não-paramétrico para comparação de duas amostras pareadas.

Tabela 3 – Os resultados do teste de Wilcoxon, comparando as acurácias do SimpleXCN a dois métodos distintos, SCN e SCCN. São mostrados em diversos conjuntos de dados: Cora, Citeseer, PubMed, DBLP, Amazon, Question e Minesweeper. Este teste foi utilizado para verificar se existem diferenças estatisticamente significativas entre as acurácias.

		Cora	Citeseer	PubMed	DBLP	Amazon	Questions	Minesweeper
SCN	Statistic	3.0	5.0	2.0	7.0	0.0	2.0	0.0
	p-value	0.4652	0.6250	0.1875	1.0000	0.0625	0.1875	0.0625
SCCN	Statistic	3.0	7.0	5.0	2.0	0.0	6.0	0.0
	p-value	0.4652	1.0000	0.6250	0.1875	0.0625	0.8125	0.0625

Tabela 4 – Comparação de tempo por iteração entre as arquiteturas. Foram realizados testes em quatro conjuntos de dados que contêm o maior número de nós e arestas; esses conjuntos também foram escolhidos porque, devido à elevada quantidade desses atributos, é mais fácil evidenciar diferenças de desempenho entre as arquiteturas. Nota-se que houve uma redução significativa no tempo de execução do conjunto Questions.

	SCN	SCCN	SimpleXCN
DBLP	0.49 seg/it	0.60 seg/it	0.47 seg/it
PubMed	0.11 seg/it	0.11 seg/it	0.11 seg/it
Questions	1.41 seg/it	1.52 seg/it	0.28 seg/it
Amazon	0.27 seg/it	0.26 seg/it	0.21 seg/it

Os resultados que estão presentes na Tabela 3, indicam que, em geral, não há diferenças estatisticamente significativas nos conjuntos de dados analisados. A maioria dos p -valores está acima de 0,05, sugerindo que as diferenças observadas podem ser atribuídas ao acaso e não representam um efeito real significativo. Apenas alguns conjuntos de dados apresentaram p -valores próximos ao limiar de significância, indicando uma tendência marginal para diferenças, mas sem evidência suficiente para rejeitar a hipótese nula com um nível de confiança padrão.

Porém, é importante destacar que o objetivo principal do SimpleXCN não é ser o modelo com melhor desempenho, mas sim ser mais leve e escalável. Portanto, mesmo que o SimpleXCN não consiga superar consistentemente os outros modelos em termos de acurácia, ele pode ser mais adequado para cenários com limitações de recursos computacionais ou a necessidade de lidar com grandes volumes de dados.

Foram realizados experimentos focados no desempenho computacional de forma empírica, utilizando uma GPU RTX4060. Neste teste, mediu-se o pico de memória alocada, o número de parâmetros e o tempo médio de execução por época para cada arquitetura. Para esses experimentos, selecionamos alguns dos conjuntos de dados com maior quantidade de nós, arestas e *features*, permitindo assim destacar de forma mais evidente a relação entre o desempenho e a

complexidade dos dados. Os resultados são mostrados nas Tabelas 4 e 5.

Tabela 5 – O número total de parâmetros para cada arquitetura e o pico de memória alocada para sua execução são apresentados.

	SCN	SCCN	SimpleXCN
DBLP	173 K / 1.29 GB	178 K / 1.29 GB	22.9 K / 1.11 GB
PubMed	64.3 K / 0.46 GB	69.3 K / 0.46 GB	5.5 K / 0.31 GB
Questions	45.2 K / 4.98 GB	50.1 K / 4.98 GB	2.4 K / 0.61 GB
Amazon	45.2 K / 0.99 GB	50.1 K / 0.98 GB	5.1 K / 0.40 GB

Pode-se observar uma redução clara, que chega a ser de 10 vezes no número de parâmetros e, conseqüentemente, uma redução significativa no uso de memória, que chega a ser de 50%. Essa redução, evidenciada pelas tabelas acima, é particularmente notável no conjunto de dados Questions. Isso ocorre porque esse conjunto possui a maior quantidade de nós e arestas, o que, durante o processo de lifting, resulta na formação de um número maior de cliques. Consequentemente, são geradas matrizes de incidência e Laplacianas de Hodge maiores para representar a estrutura do complexo simplicial. Isso proporciona um cenário favorável, evidenciando a redução tanto no tempo de execução quanto no número de parâmetros.

5 LIFTING DIFERENCIÁVEL

Nesta seção, será apresentado o ∂lift (lido como DiffLift), uma estrutura geral para aprender funções de elevação de grafos para estruturas de ordem superior. A Seção 5.1 fornece uma descrição iterativa do método, permitindo o aprendizado de estruturas de ordem cada vez mais altas. As Subseções 5.1.1 e 5.1.3 utilizam ∂lift para elevação de grafo para hipergrafo e grafo para complexo de celulares, respectivamente.

5.1 Formulação Geral

Consiste em determinar quais células de ordem superior devem ser adicionadas a um grafo de entrada G , de forma a satisfazer as restrições do domínio alvo. Para isso, foi proposta a seguinte formulação.

∂lift : Formulação geral para elevações diferenciáveis de grafos

Entrada: Grafo $G = (V, E, x)$, domínio alvo $\mathbb{T}$, e dimensão máxima $D_{\max}$.

Passo 1: Calcular as incorporações dos nós. Use uma GNN arbitrária para calcular uma representação vetorial (*embedding*) z_v para cada nó $v \in V$. Esta GNN pode ser um modelo pré-treinado ou aprendido de ponta a ponta. Defina a dimensão do domínio atual como $D = 1$.

Passo 2: Selecionar células candidatas. Dadas as representações dos nós $\{z_v\}_{v \in V}$, defina um conjunto de células candidatas $\mathcal{C} \subseteq 2^V$ de dimensão D . Para cada célula $C \in \mathcal{C}$, calcule uma representação $z_C = \bigoplus_{v \in C} z_v$, onde $\bigoplus$ é uma função de agregação invariante a permutações. Observe que o procedimento exato para definir células candidatas depende do domínio alvo $\mathbb{T}$, pois os candidatos devem respeitar possíveis restrições de hierarquia entre células.

Passo 3: Aceitar/rejeitar células candidatas. Aplique uma rede neural ϕ (por exemplo, uma MLP) que define uma probabilidade de aceitação $\phi(z_C)$ para cada célula candidata C . Finalmente, gere uma amostra y_C de uma distribuição de Bernoulli com parâmetro $\phi(z_C)$, indicando se a célula C é aceita ou não. O domínio resultante é então dado por $V \cup E \cup \{C \in \mathcal{C} : y_C = 1 \text{ com } y_C \sim \text{Ber}(\phi(z_C))\}$.

Passo 4: Verificação de término. Se $D = D_{\max}$, pare; caso contrário, $D \leftarrow D + 1$ e retorne ao Passo 2.

O ∂ lift é aprendido de ponta a ponta, utilizando o estimador *straight-through* (BENGIO; LÉONARD; COURVILLE, 2013) para propagar gradientes através das amostras no Passo 3. Para hipergrafos, assumimos que as hiperarestas possuem dimensão um, fazendo com que o ∂ lift pare assim que chega ao Passo 4. Observamos que os Passos 2 e 3 são os únicos dependentes do domínio. A seguir, é explicado como esses passos podem ser adaptados a domínios específicos.

5.1.1 *Lifting de Grafo para Hipergrafo*

Passo 1. De forma similar a formulação geral do ∂ lift: utilizamos uma GNN arbitrária para calcular uma representação z_v para cada nó $v \in V$, mantendo a dimensão inicial $D = 1$.

Passo 2. Suponha que desejamos aprender uma hiperaresta para cada nó. Para cada nó v , definimos uma hiperaresta candidata $C(v)$ como um conjunto $S \subseteq V$ que contém exatamente k_v nós, escolhidos de acordo com a seguinte condição:

$$C(v) = \left\{ S \subseteq V : |S| = k_v \text{ e } \forall w \notin S, \text{dist}(z_w, z_v) \geq \max_{u \in S} \text{dist}(z_u, z_v) \right\},$$

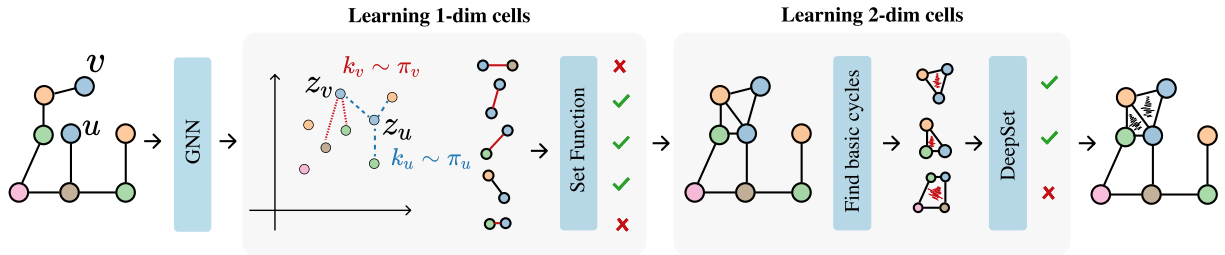
sendo $\text{dist}(z_u, z_v)$ a distância Euclidiana entre as representações vetoriais z_u e z_v dos nós u e v , respectivamente. A condição garante que, para cada nó w que não está em S , a distância $\text{dist}(z_w, z_v)$ será sempre maior ou igual à maior distância entre v e os nós presentes em S . Isso implica que os nós em S são os k_v nós mais próximos de v , e quaisquer nós adicionais não podem ser mais próximos de v do que os nós já selecionados.

Para permitir tamanhos adaptativos de hiperarestas, k_v é amostrado de acordo com uma distribuição de probabilidade parametrizada por uma função z_v de v . Mais especificamente, definimos o vetor de probabilidade π_v , de dimensão $k_{\max} - k_{\min} + 1$, tal que:

$$\pi_v \propto \exp \circ \text{MLP}(z_v), \quad (5.1)$$

sendo MLP uma rede neural de múltiplas camadas aplicada à representação z_v , e $\exp \circ$ denota a aplicação da função exponencial aplicada ao resultado da rede neural de multi-camadas. Em seguida, fazemos $k_v \sim \text{Categorical}(\pi_v)$, ou seja, k_v é amostrado a partir de uma distribuição categórica com probabilidade dada por π_v . Os valores $k_{\min}$ e $k_{\max}$ definem os limites inferior e superior para k_v , respectivamente.

Figura 4 – Duas iterações do ∂ lift para complexos de células. Na primeira iteração, calculamos as representações através de uma GNN para cada nó u , z_u , em que estas representações são utilizadas para delimitar as células candidatas de 1 dimensão. Especificamente, consideramos o equivalente das células de aresta que conectam u a cada um de seus k_u vizinhos mais próximos no espaço de representações, onde k_u é uma variável aleatória parametrizada por z_u . Uma função é aplicada sobre o conjunto de representações dos nós dentro de cada célula para calcular suas probabilidades de aceitação. A partir da segunda iteração, usamos a elevação por ciclos no nosso complexo de células aumentado para gerar células candidatas, cujas probabilidades de aceitação são calculadas de forma semelhante ao primeiro passo.



Passo 3. É definida a probabilidade de aceitação (ou seja, de $b_v = 1$) para $C(v)$ como uma função dos *embeddings* dos nós em $C(v)$. Mais especificamente, b_v é dado por:

$$b_v \sim \text{Ber}(\Psi(\{z_u : u \in C(v)\})), \quad (5.2)$$

onde Ψ é aprendível e mapeia de multiconjuntos (ou seja, Ψ é invariável à ordem) de elementos em $\mathbb{R}^d$ para $(0, 1)$.

5.1.2 Elevação de Características (Feature Lifting)

Cada hiperaresta aceita $C(v)$ recebe um vetor de características $x_{C(v)}$ calculado como uma operação de multiconjunto sobre $\{x_u : u \in C(v)\}$. Especificamente, utilizamos uma projeção de soma escalada:

$$x_{C(v)} = \frac{1}{k_v} \sum_{u \in C(v)} x_u, \quad \forall v \text{ tal que } b_v = 1. \quad (5.3)$$

5.1.3 Elevação de Grafo para complexo de células

Por razões computacionais, o procedimento de elevação para complexos de células é dividido em dois casos (modificando somente o **Passo 2** e **Passo 3**). Uma visão geral do método de elevação proposto de grafo para complexo de células está representado na Figura 4.

5.1.3.1 Caso $D = 1$: Aprendendo arestas

Passo 1. Novamente utilizamos uma GNN arbitrária para calcular uma representação z_v para cada nó $v \in V$, mantendo a dimensão inicial $D = 1$.

Passo 2. De forma similar à elevação de grafo para hipergrafo, para cada nó v , é amostrada um tamanho de vizinhança k_v e um conjunto $C(v) \subseteq V \setminus \{v\}$ contendo os nós associados aos k_v vizinhos mais próximos de v é definido no espaço de representações (*embeddings*).

Passo 3. Construimos as arestas candidatas (células de 1-dimensão) considerando cada par (v, v') e sua probabilidade de aceitação é definida (ou seja, de $b_{v,v'} = 1$) como uma função das representações de v e v' .

Especificamente, fazemos $b_{v,v'} \sim \text{Ber}(\Psi(\{z_v, z_{v'}\}))$, onde Ψ é uma função invariável à ordem. Ao final desta iteração, o complexo de células obtido é dado por:

$$K^1 = V \cup E \cup \{\{v, v'\} : v \in V, v' \in C(v) \text{ com } b_{v,v'} = 1\}. \quad (5.4)$$

Quanto ao *feature lifting*, a projeção de soma escalada é aplica, de forma idêntica ao caso do hipergrafo.

5.1.3.2 Caso $D \geq 2$: Aprendendo células D -dimensões

Para selecionar células candidatas de dimensão arbitrária, precisa-se ter noção de ciclos n -dimensões de um complexo de células. Seja $C_n(K)$ os n -ciclos do complexo de células K , equipados com a estrutura de espaço vetorial $\mathbb{Z}/2\mathbb{Z}$. Também seja $\partial_n : C_n(K) \rightarrow C_{n-1}(K)$ o a função de fronteira linear de K . Então, os n -ciclos de K são dados por $Z_n(K) = \ker(\partial_n)$ (ver Seção 2.8, pág 28).

Passo 2. Seja K^{D-1} o complexo de células no final da iteração $D - 1$. É selecionada uma base para o conjunto de ciclos $(D - 1)$ -dimensões $Z_{D-1}(K^{D-1})$ em K^{D-1} para servir como células candidatas. Uma base de ciclos é uma coleção mínima de ciclos tal que qualquer ciclo pode ser escrito como uma soma módulo-2 dos ciclos na base.

Passo 3. Seja $\mathcal{C}$ o conjunto de ciclos $(D - 1)$ -candidatos do **Passo 2**. A definição da probabilidade de aceitação de $C \in \mathcal{C}$ (ou seja, $b_C = 1$) usando um modelo DeepSet (ZAHEER *et al.*, 2017) sobre o multiconjunto de representações $\{z_v\}_{v \in C}$ de todos os nós v em C . O complexo de células obtido nesta iteração é então:

$$K^D = K^{D-1} \cup \{C \in \mathcal{C} : b_C = 1\}. \quad (5.5)$$

Para simplificação, novamente, as características das células D -dimensões são obtidas através da elevação de projeção de soma.

Observação 5.1. *Apesar da generalidade do ∂ lift, nos experimentos, são considerados apenas complexos de células 2-dimensionais ($D_{\max} = 2$) e utiliza-se o algoritmo de (PATON, 1969), disponível em bibliotecas como NetworkX (HAGBERG; SCHULT; SWART, 2008), para identificar ciclos básicos 1-dimensionais em grafos. Isso se deve principalmente ao fato de que implementações atuais de TNNs para complexos de células só suportam objetos 2-dimensionais — por exemplo, veja TopoBenchmarkX (TELYATNIKOV et al., 2024).*

Observação 5.2. *Uma versão determinística do ∂ lift pode ser feita usando um limiar de probabilidade, ou seja, simplesmente definimos $b_C = \mathbb{1}[\Psi(\cdot) > \gamma]$ com, por exemplo, $\gamma = 0.5$, para todas as células candidatas C .*

5.2 Experimentos e Resultados

Nesta seção, o ∂ lift é avaliado em duas tarefas complementares: classificação de grafos e classificação de nós. O ∂ lift é comparado com esquemas de elevação amplamente utilizados para hipergrafos e complexos de células. Também são relatados resultados em diferentes TNNs dentro de cada um desses domínios. Os experimentos foram realizados utilizando implementações desenvolvidas com as bibliotecas PyTorch (PASZKE et al., 2017) e PyTorch Geometric (FEY; LENSSEN, 2019), e executados em clusters de alto desempenho compostos por unidades de processamento gráfico (GPUs) dos modelos NVIDIA A100 e V100.

5.2.1 Classificação de grafos

5.2.1.1 Conjuntos de dados

O desempenho do método foi avaliado em seis conjuntos de dados de *benchmark* amplamente utilizados para previsão de propriedades moleculares: NCI1, NCI109, MUTAG, MOLHIV, PROTEINS e ZINC (KERSTING et al., 2016; DWIVEDI et al., 2023; HU et al., 2020). Esses conjuntos de dados são *benchmarks* padrão na literatura para avaliar a eficácia de modelos baseados em grafos (DWIVEDI et al., 2023; TELYATNIKOV et al., 2024). Todas as tarefas são problemas de classificação binária, com exceção do ZINC, que é uma tarefa de regressão.

Os conjuntos de dados NCI1, NCI109, PROTEINS e MUTAG fazem parte do TUDatasets (KERSTING et al., 2016) — uma coleção de conjuntos de dados amplamente utilizada como

benchmark em tarefas de aprendizado sobre grafos, onde cada grafo representa uma molécula; ou seja, os nós correspondem aos átomos e as arestas, às ligações químicas. Consequentemente, os rótulos codificam os tipos de átomos e ligações. Também utilizamos os conjuntos ZINC-12K e MOLHIV (HU *et al.*, 2020), *benchmarks* populares para previsão de propriedades moleculares. As estatísticas de cada conjunto de dados são apresentadas na Tabela 6.

Tabela 6 – Estatísticas dos conjuntos de dados para tarefas em nível de grafo. É apresentado a quantidade de grafos, classes, média de nós e arestas por grafo, além da porcentagem dividida para treino, validação e teste.

Conjunto de Dados	#Grafos	#Classes	Média #Nós	Média #Arestas	Treino%	Val%	Teste%
MUTAG	188	2	17.93	19.79	80	10	10
PROTEINS	1113	2	39.06	72.82	80	10	10
NCI1	4110	2	29.87	32.30	80	10	10
NCI109	4127	2	29.68	32.13	80	10	10
ZINC	12000	–	23.16	49.83	Divisão Pública		
MOLHIV	41127	2	25.5	27.5	Divisão Pública		

5.2.1.2 Baselines

O ∂ lift foi comparado com quatro métodos de elevação de grafos: *lifting* de ciclos, *lifting* de k -hops (k -saltos), *lifting* de k -vizinhos mais próximos (k -NN), e *lifting* de *kernel*. Entre esses, o *lifting* de ciclos é a estratégia mais amplamente adotada para *liftings* de grafo para complexo de células e se tornou o padrão em praticamente todas as TNNs que operam em complexos de células (TELYATNIKOV *et al.*, 2024; BODNAR *et al.*, 2021a). De forma similar, o *lifting* de k -hops é a abordagem predominante para construir hipergrafos a partir de grafos e frequentemente é o único método considerado em *benchmarks* recentes, como (TELYATNIKOV *et al.*, 2024). O *lifting* de k -NN também foi considerado, pois ele compartilha semelhanças com a abordagem proposta devido ao uso de k -NN. Finalmente, considerou-se o *lifting* de *kernel*, uma das abordagens mais bem-sucedidas no desafio ICML TDL (BERNÁRDEZ *et al.*, 2024). Notavelmente, essa escolha de *baselines* abrange métodos de *lifting* baseados em conectividade (*lifting* de ciclos e k -hops), características (k -NN *lifting*) e tanto conectividade quanto características (*lifting* de *kernel*).

5.2.1.3 Configuração da avaliação

Inicialmente para os conjuntos de dados ZINC e MOLHIV, foram utilizadas as divisões de dados públicas de treino/validação/teste, e para os conjuntos de dados restantes, utilizamos uma divisão aleatória de 80/10/10%. Seleccionamos duas classes de TNNs de domínios diferentes: redes neurais baseadas em complexos celulares — entre elas, a *CW Network* (CWN) (BODNAR

et al., 2021a) e a *Cell Complex Neural Network* (CXN) (HAJII; ISTVAN; ZAMZMI, 2020) — e redes para hipergrafos, provenientes do *framework* unificado para redes neurais de grafos e hipergrafos (Unified Framework for Graph and Hypergraph Neural Networks — UniGNN) (HUANG; YANG, 2021). Nos experimentos realizados, utilizamos as versões integradas da *Graph Isomorphism Network* (GIN) (XU *et al.*, 2019) e da *Graph Convolution Network II* (GCNII) (CHEN *et al.*, 2020) ao *framework* UniGNN, denominadas, respectivamente, UniGIN e UniGCNII.

Os hiperparâmetros (incluindo taxa de aprendizado, otimizador, tamanho do lote, largura, profundidade, e outros) utilizados foram os mesmos reportados no TopoBenchmarkX para CWN, CXN e UniGCNII em NCI1, NCI109, MUTAG, Proteins, ZINC (TELYATNIKOV *et al.*, 2024) e estão reportados na Tabela 7. Como o TopoBenchmarkX não reporta os hiperparâmetros ótimos para UniGIN, utilizamos os mesmos hiperparâmetros usados para o UniGCNII.

Tabela 7 – Principais hiperparâmetros presentes no TopoBenchmarkX para CWN, CXN e UniGCNII.

Hiperparâmetro	Valor / Range
Otimizador	Adam.
Taxa de aprendizado	{0.01, 0.001}
Tamanho do lote (batch size)	{128, 256}
Camadas ocultas	{32, 64, 128}
Número de camadas	{1,2,3,4}
Dropout	{0.25, 0.5}
k_max	{3,5,7,9,11}
Weight decay	0
Épocas	200
Early stopping	50

Para a base de dados MOLHIV, que não está presente no TopoBenchmarkX, foram utilizadas redes com 2 camadas ocultas, com taxa de aprendizado selecionada entre os valores $\{10^{-2}, 5 \times 10^{-3} \text{ e } 5 \times 10^{-3}\}$, e a dimensão (número de neurônios ocultos em cada camada) foi fixada em 64.

Para a otimização dos hiperparâmetros do ∂ lift, escolhemos entre duas GNNs distintas — *Graph Transformer* (RAMPÁŠEK *et al.*, 2022) e *Graph Isomorphism Network*—com dimensões de embeddings de {32, 64, 128}, e {2, 3} camadas ocultas. Além disso, buscamos por valores ótimos de k_v considerando $k_{\max} \in \{3, 5, 7, 9, 11\}$. Escolhemos também k em {3, 5, 7, 9} para o *lifting* k -NN. Para a elevação baseada em kernel, escolhemos os valores de temperatura igualmente espaçados entre 0.1 e 9.6, com incrementos de 0.5.

Todos os modelos foram treinados por 200 épocas, com parada antecipada (*early stopping*) após 50 épocas sem melhoria na acurácia de validação. Três testes foram realizados

independentemente para calcular a média e o desvio padrão das métricas de desempenho. Os hiperparâmetros ótimos foram selecionados com base na acurácia de validação.

Reportamos a média e o desvio padrão das métricas de desempenho (MAE ↓ para ZINC, AUC ↑ para MOLHIV, e acurácia ↑ para todas as outras bases) para três execuções independentes.

Tabela 8 – Classificação de grafos: ∂ lift vs liftings estáticos. O melhor modelo para cada conjunto de dados/TNN está destacado em negrito. Para qualquer TNN fixa e conjunto de dados, ∂ lift é melhor que os liftings estáticos em 90% dos casos, oferecendo uma melhoria de desempenho de até 45%.

Domínio	TNN	Elevação	NCI1↑	NCI109↑	MOLHIV↑	MUTAG↑	PROTEINS↑	ZINC↓
Grafo	GCN	-	74.45±1.05	76.46±1.03	74.99±1.09	64.91±4.96	70.18±1.35	0.87±0.01
	GIN	-	76.89±1.75	76.90±0.80	70.76±2.46	80.70±2.48	72.50±2.31	0.62±0.01
Celular	CWN	Ciclo	76.93±1.18	76.71±1.34	70.15±3.98	66.67±12.41	69.05±2.95	0.46±0.01
		∂ lift	79.81 ±0.40	80.55 ±0.50	75.37 ±0.80	85.96 ±4.96	70.54 ±3.34	0.17 ±0.00
	CXN	Ciclo	72.02±1.69	75.01±0.62	69.17±1.20	61.40±2.48	70.83 ±1.52	0.79±0.02
		∂ lift	82.08 ±1.50	82.57 ±0.40	74.83 ±1.96	84.21 ±4.30	69.94±2.10	0.17 ±0.01
Hipergrafo	UniGCN2	<i>k</i> -viz.	72.70±0.52	72.01±1.55	50.72±1.06	61.40±2.48	72.92±1.11	0.66±0.02
		<i>k</i> -NN	71.78±0.20	68.60±0.93	57.73±6.84	64.91±2.48	73.51±0.42	1.10±0.01
		kernel	73.80±0.94	72.64±0.40	57.07±10.32	63.16±8.59	73.21±0.73	0.79±0.02
		∂ lift	77.45 ±1.88	75.30 ±1.10	69.32 ±1.62	89.47 ±4.30	73.51 ±0.84	0.56 ±0.03
	UniGIN	<i>k</i> -viz.	65.50±1.99	66.97±7.25	63.49±9.55	64.91±2.48	71.43±0.73	1.15±0.01
		<i>k</i> -NN	72.83 ±1.09	70.14±1.48	52.34±3.21	59.65±4.96	72.62±1.52	1.10±0.02
		kernel	60.50±1.26	66.59±1.49	49.60±0.07	57.89±4.30	66.67±1.83	1.45±0.02
		∂ lift	64.88±1.09	79.74 ±0.23	72.04 ±0.88	66.67 ±6.56	73.81 ±1.52	1.01 ±0.05

5.2.1.4 Resultados

A Tabela 8 mostra que o ∂ lift é o operador de *lifting* com melhor desempenho em mais de 90% das combinações TNN/conjunto de dados, tanto para complexos de células quanto para hipergrafos. Notavelmente, o ∂ lift resultou em uma melhoria na acurácia média de até $\approx 45\%$ em comparação com os *liftings* estáticos utilizando a mesma TNN. Para CWN e UniGCNII, o ∂ lift supera os *liftings* estáticos em todos os conjuntos de dados. Nos conjuntos de dados NCI109 e ZINC, o ∂ lift é consistentemente melhor que os *liftings* estáticos, estando próximo do SOTA (estado da arte) nesses conjuntos de dados, e supera outros métodos de aprendizado profundo topológicos, como os modelos propostos em Papillon *et al.* (2025) e Immonen, Souza e Garg (2023). Também foi observado que as TNNs têm um desempenho melhor do que as GNNs. No geral, esses resultados validam a eficácia da proposta.

5.2.1.5 Impacto da escolha da GNN no desempenho

Aqui o objetivo é avaliar quão sensível é a abordagem à escolha da GNN. Para isso, os resultados foram reportados usando GIN (XU *et al.*, 2019) e GPS (Graph Transformer) (RAMPÁŠEK *et al.*, 2022).

A Tabela 9 indica que escolher GNNs que são capazes de gerar representações latentes de nós mais ricas e informativas leva a melhores resultados no ∂lift . Em particular, o modelo GPS apresenta desempenho superior ao GIN na maioria dos conjuntos de dados. Uma possível explicação para essa observação é a maior expressividade do GPS, que se beneficia da incorporação de codificações posicionais (*positional encoding*). Notavelmente, em complexos de células e no conjunto de dados ZINC, o GPS permite reduzir o MAE de 0,46 para 0,17.

Tabela 9 – Efeito da GNN escolhida no desempenho do ∂lift . Os resultados sugerem que o poder expressivo da escolha das GNNs tem um impacto direto no desempenho do ∂lift . Exceto para MOLHIV e NCI1, o GPS leva a um desempenho melhor do que o GIN em geral.

TNN	GNN	NCI1↑	NCI109 ↑	MOLHIV ↑	MUTAG↑	Proteins↑	ZINC↓
CWN	GPS	79.81±0.40	80.55 ±0.50	64.31±5.32	87.72 ±2.48	70.54±3.34	0.17 ±0.00
	GIN	81.59 ±0.80	78.69±1.43	75.37 ±0.80	82.46±2.48	71.13 ±2.76	0.46±0.00
CXN	GPS	79.97±0.61	82.57 ±0.40	65.58±4.42	84.21 ±4.30	69.94 ±2.10	0.17 ±0.01
	GIN	81.35 ±2.29	79.98±1.01	72.25 ±3.23	77.19±2.48	67.86±2.19	0.43±0.01
UniGCN2	GPS	78.67 ±1.46	74.50±1.16	68.22±2.38	89.47 ±4.30	73.81 ±0.42	0.56 ±0.03
	GIN	75.38±1.39	74.98 ±1.12	68.73 ±2.05	64.91±6.56	73.51±0.84	0.63±0.01
UniGIN	GPS	66.42 ±1.79	79.74 ±0.23	68.32±3.12	66.67±6.56	72.32 ±0.73	1.01 ±0.05
	GIN	64.40±0.41	78.53±0.69	68.86 ±3.05	70.18 ±6.56	72.02±3.74	1.12±0.26

5.2.2 Classificação de Nós

5.2.2.1 Conjuntos de dados

Para classificação de nós, o ∂lift foi avaliado em quatro conjuntos de dados: Cora, Citeseer (YANG; COHEN; SALAKHUDINOV, 2016), Texas e Wisconsin (ROZEMBERCZKI; ALLEN; SARKAR, 2020). Desses conjuntos de dados, dois são homofílicos (Cora e Citeseer) e dois são heterofílicos (Texas e Wisconsin). As estatísticas dos conjuntos de dados podem ser encontradas na Tabela 10.

Tabela 10 – Estatísticas para os conjuntos de dados utilizados para classificação de nós. São apresentados o número de nós, arestas, características (*features*), classes, e quantidade de nós para treino, validação e teste.

Conjuntos de Dados	#Nós	#Arestas	#Features	#Classes	#Treino	#Val	#Teste
Cora	2708	5429	1,433	7	1,208	500	1,000
Citeseer	3327	4732	3,703	6	1,827	500	1,000
Texas	183	309	1703	5	87	59	37
Wisconsin	251	499	1703	5	120	80	51

5.2.2.2 Baselines

O método proposto (para domínios de células) também foi comparado com a abordagem aprendível de (BATILORO *et al.*, 2024), chamada Differentiable Cell Complex Module (DCM), que foi originalmente avaliada em tarefas de classificação de nós. Para isso, considerou-se o ∂ lift combinado com CWN. Também foram inclusos os resultados de CWNs com *lifting* de ciclos. Além disso, também consideramos as mesmas baselines de TNN para hipergrafos conforme a Subseção 5.2.1.

5.2.2.3 Configuração da avaliação

Para todos os conjuntos de dados realizamos uma divisão aleatória para treino, validação e teste com uma proporção de 60/20/20%. De forma similar aos experimentos de classificação de grafos, os hiperparâmetros para as arquiteturas foram os mesmos utilizados em (TELYATNIKOV *et al.*, 2024) para os conjuntos de dados (Cora e Citeseer) que estavam disponíveis. Para o Wisconsin e Texas, foi utilizada 1 camada oculta da TNN com as mesmas configurações (Taxa de aprendizagem, dimensão das features) definidas na Subseção 5.2.1.3 com decaimento de peso de 5×10^{-6} . A respeito da tarefa de otimização dos hiperparâmetros para as operações de elevação (*lifting*), foi utilizada a mesma descrita na Subseção 5.2.1.3. A acurácia média e o desvio padrão são reportados em três execuções independentes.

Tabela 11 – Comparação do DCM e ∂ lift na classificação de nós. O ∂ lift alcança o melhor desempenho médio em todos os conjuntos de dados (lado direito) e supera significativamente o DCM nos conjuntos de dados heterofílicos.

TNN	Lifting	Cora	Citeseer	Texas	Wisconsin	Avg
DCM	-	80.73 $\pm$ 0.33	77.90 $\pm$ 0.80	56.76 $\pm$ 6.62	73.86 $\pm$ 1.85	72.31
CWN	Cycle	74.80 $\pm$ 0.08	75.83 $\pm$ 0.90	63.06 $\pm$ 7.75	80.39 $\pm$ 4.24	73.52
CWN	∂ Lift	80.17 $\pm$ 1.59	72.83 $\pm$ 2.15	80.18 $\pm$ 3.37	77.78 $\pm$ 3.70	77.74

Tabela 12 – Comparação do ∂ lift e dos *baselines* de *lifting* estáticos para hipergrafos na classificação de nós. O ∂ lift supera todos os *liftings* estáticos em média, nos conjuntos de dados selecionados para classificação de nós.

TNN	Lifting	Cora	Citeseer	Texas	Wisconsin	Avg
UniGCN2	<i>k</i> -hop	86.03 $\pm$ 0.63	78.40 $\pm$ 0.36	66.67 $\pm$ 6.74	69.28 $\pm$ 5.62	75.09
	<i>k</i> -NN	74.00 $\pm$ 0.65	18.17 $\pm$ 0.05	70.27 $\pm$ 3.82	79.74 $\pm$ 2.45	60.54
	kernel	29.93 $\pm$ 1.33	18.07 $\pm$ 0.21	57.66 $\pm$ 7.75	57.52 $\pm$ 10.42	40.79
	∂ Lift	81.93 $\pm$ 1.11	78.03 $\pm$ 0.91	69.37 $\pm$ 2.55	73.20 $\pm$ 5.62	75.63
UniGIN	<i>k</i> -hop	78.73 $\pm$ 0.66	74.47 $\pm$ 1.72	65.77 $\pm$ 9.19	58.82 $\pm$ 5.77	69.44
	<i>k</i> -NN	62.00 $\pm$ 1.08	19.33 $\pm$ 0.48	65.77 $\pm$ 1.27	73.20 $\pm$ 4.03	55.07
	kernel	40.93 $\pm$ 2.52	18.53 $\pm$ 0.61	58.56 $\pm$ 7.09	51.63 $\pm$ 4.03	42.41
	∂ Lift	84.23 $\pm$ 0.53	77.97 $\pm$ 0.45	63.96 $\pm$ 6.37	63.40 $\pm$ 4.03	72.39

5.2.2.4 Resultados

A Tabela 11 compara o ∂ lift com o DCM e o *lifting* de ciclos. O método desenvolvido alcança as maiores acurácias em dois dos quatro conjuntos de dados e o melhor desempenho médio entre os conjuntos de dados. Em particular, o ∂ lift obteve uma média de 77,74, comparado com 73,52 do segundo melhor método (*lifting* de ciclos). Nos conjuntos de dados Texas e Wisconsin, os conjuntos heterofílicos, o ∂ lift supera o DCM por uma grande margem.

A Tabela 12 relata os resultados dos métodos de *lifting* para redes neurais de hipergrafo. Comparado ao *k*-hop, a abordagem proposta é melhor em conjuntos de dados heterofílicos, mas pior em conjuntos homofílicos para o UniGCN2. Além disso, a Tabela 12 mostra que o ∂ lift leva a uma acurácia média melhor do que outros *liftings* estáticos. Esses resultados indicam que o ∂ lift é uma abordagem mais adaptável para a tarefa de classificação de nós, oferecendo uma solução mais eficiente em comparação com os métodos convencionais, em diferentes cenários.

6 CONCLUSÃO

Na primeira metade desta dissertação, foi apresentado o SimpleXCN, que apesar das simplificações, demonstrou-se competitivo em diversos conjuntos de dados, mantendo um desempenho próximo ou equivalente às arquiteturas mais complexas como SCN e SCCN. Os resultados experimentais indicam que, embora não haja uma arquitetura claramente superior, o SimpleXCN demonstrou ser uma alternativa computacionalmente eficiente. As análises estatísticas sugerem que as diferenças entre os métodos não são estatisticamente significativas na maioria dos casos, reforçando a viabilidade do SimpleXCN como uma solução prática e eficiente para a classificação de nós.

É importante considerar que a operação de elevação (*lifting*) tem permanecido amplamente não supervisionado e agnóstico à tarefa, o que pode levar à construção de representações topológicas subótimas para o aprendizado subsequente. Na segunda parte deste trabalho foi apresentado o ∂ lift (*DiffLift*), um método de elevação diferenciável. Em um conjunto abrangente de benchmarks e arquiteturas de TNNs, o ∂ lift consistentemente superou os métodos tradicionais de *lifting* não supervisionado, demonstrando o benefício de tornar o processo de *lifting* aprendível e informado pela tarefa.

6.1 Principais achados

A partir dos experimentos e análises apresentados ao longo desta dissertação, destacam-se os seguintes resultados principais:

- O **SimpleXCN** provou ser uma alternativa computacionalmente eficiente para redes convolucionais em complexos simpliciais: redução significativa no tempo de execução por iteração, menor consumo de memória e número reduzido de parâmetros, mantendo desempenho competitivo em tarefas de classificação de nós.
- O ∂ lift (*DiffLift*) mostrou-se capaz de *aprender* procedimentos de elevação que melhoram o desempenho de TNNs em múltiplos *benchmarks*, frequentemente superando *liftings* estáticos (por exemplo, ganhos reportados de até 45% em determinados cenários de classificação de grafos). Os resultados são robustos em diferentes arquiteturas de TNNs e em tarefas tanto de classificação de grafos quanto de nós.

6.2 Limitações

Para domínios de hipergrafos, o ∂ lift pode criar hiperarestas candidatas e decidir se deve mantê-las de maneira *embarrassingly parallel*, o que torna o ∂ lift especialmente eficiente em termos de computação para este domínio. No entanto, para complexos de células, é preciso calcular uma base de ciclos para gerar células candidatas, o que pode ter um custo cúbico em relação ao número de nós no grafo de entrada. Nesse caso, pode-se reduzir o número de células candidatas ao, por exemplo, regularizar as variáveis k_v ou deslocar sua distribuição para zero. No entanto, desenvolver algoritmos mais eficientes para identificação de candidatos em domínios hierárquicos é uma direção clara de melhoria para trabalhos futuros.

Os ganhos observados em benchmarks podem não se replicar de forma idêntica em domínios muito distintos (p.ex., redes biológicas muito esparsas ou nuvens de pontos 3D com ruído). Avaliações adicionais em conjuntos de dados de aplicação real são necessárias.

O caráter probabilístico do ∂ lift e a grandeza das estruturas geradas dificultam a interpretação direta das razões pelas quais certas células/hiperarestas são selecionadas — trabalho futuro em explicabilidade ajudaria adoção prática.

6.3 Potenciais aplicações

As contribuições propostas abrem caminho para diversas aplicações práticas, podendo ser utilizadas para exploração de grupos e padrões de co-interação (cliques, ciclos) para melhorar recomendações e detectar comunidades. Modelagem de interações multicomponentes (p.ex., complexos proteicos), onde relações de ordem superior são cruciais. Além de que pode ser estendido para outros domínios topológicos, como nuvens de pontos, tornando-o aplicável a tarefas de mapeamento 3D.

6.4 Direções para trabalhos futuros

Os trabalhos futuros devem se concentrar em abordar os desafios computacionais do ∂ lift e escalá-lo para lidar com grafos maiores. Outra direção promissora é explorar o *lifting* diferenciável em grafos dinâmicos ou temporais, onde as estruturas topológicas evoluem ao longo do tempo. Além disso, integrar o ∂ lift com estratégias de pré-treinamento poderia gerar informações a priori topológicas generalizáveis para diferentes tarefas.

REFERÊNCIAS

- AKIBA, T.; SANO, S.; YANASE, T.; OHTA, T.; KOYAMA, M. Optuna: A next-generation hyperparameter optimization framework. In: **The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [S.l.: s.n.], 2019. p. 2623–2631.
- ALARFAJ, F. K.; MALIK, I.; KHAN, H. U.; ALMUSALLAM, N.; RAMZAN, M.; AHMED, M. Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. **IEEE Access**, IEEE, v. 10, p. 39700–39715, 2022.
- ALPAYDIN, E. **Introduction to Machine Learning**. [S.l.]: The MIT Press, 2014. ISBN 0262028182.
- BARABÁSI, A.-L. Network science. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, The Royal Society Publishing, v. 371, n. 1987, p. 20120375, 2013.
- BASTINGS, J.; TITOV, I.; AZIZ, W.; MARCHEGGIANI, D.; SIMA'AN, K. Graph convolutional encoders for syntax-aware neural machine translation. **arXiv preprint arXiv:1704.04675**, 2017.
- BATTILORO, C.; SPINELLI, I.; TELYATNIKOV, L.; BRONSTEIN, M. M.; SCARDAPANE, S.; LORENZO, P. D. From latent graph to latent topology inference: Differentiable cell complex module. In: **The Twelfth International Conference on Learning Representations**. [s.n.], 2024. Disponível em: <<https://openreview.net/forum?id=0JsRZEGZ7L>>.
- BENGIO, Y.; LÉONARD, N.; COURVILLE, A. Estimating or propagating gradients through stochastic neurons for conditional computation. **ArXiv e-prints**, 2013.
- BERNÁRDEZ, G.; TELYATNIKOV, L.; MONTAGNA, M.; BACCINI, F.; PAPILLON, M.; FERRIOL-GALMÉS, M.; HAJIJ, M.; PAPAMARKOU, T.; BUCARELLI, M. S.; ZAGHEN, O. *et al.* Icm1 topological deep learning challenge 2024: Beyond the graph domain. **arXiv preprint arXiv:2409.05211**, 2024.
- BODNAR, C.; FRASCA, F.; OTTER, N.; WANG, Y.; LIÒ, P.; MONTUFAR, G. F.; BRONSTEIN, M. Weisfeiler and lehman go cellular: CW networks. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 2021.
- BODNAR, C.; FRASCA, F.; WANG, Y.; OTTER, N.; MONTUFAR, G. F.; LIO, P.; BRONSTEIN, M. Weisfeiler and lehman go topological: Message passing simplicial networks. In: **International Conference on Machine Learning (ICML)**. [S.l.: s.n.], 2021.
- BOJCHEVSKI, A.; GÜNNEMANN, S. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In: **International Conference on Learning Representations**. [s.n.], 2018. Disponível em: <<https://openreview.net/forum?id=r1ZdKJ-0W>>.
- BOROVITSKIY, V.; AZANGULOV, I.; TEREININ, A.; MOSTOWSKY, P.; DEISENROTH, M.; DURRANDE, N. Matérn gaussian processes on graphs. In: PMLR. **International Conference on Artificial Intelligence and Statistics**. [S.l.], 2021. p. 2593–2601.
- BRONSTEIN, M. M.; BRUNA, J.; COHEN, T.; VELIČKOVIĆ, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. **arXiv preprint arXiv:2104.13478**, 2021.

CHEN, M.; WEI, Z.; HUANG, Z.; DING, B.; LI, Y. Simple and deep graph convolutional networks. In: III, H. D.; SINGH, A. (Ed.). **Proceedings of the 37th International Conference on Machine Learning**. PMLR, 2020. (Proceedings of Machine Learning Research, v. 119), p. 1725–1735. Disponível em: <<https://proceedings.mlr.press/v119/chen20v.html>>.

DUVENAUD, D. **Automatic model construction with Gaussian processes**. Tese (Doutorado), 2014.

DWIVEDI, V. P.; JOSHI, C. K.; LUU, A. T.; LAURENT, T.; BENGIO, Y.; BRESSON, X. Benchmarking graph neural networks. **Journal of Machine Learning Research**, v. 24, n. 43, p. 1–48, 2023.

FEY, M.; LENSSEN, J. E. Fast graph representation learning with PyTorch Geometric. In: **Workshop track of the International Conference on Representation Learning (ICLR)**. [S.l.: s.n.], 2019.

FREEMAN, L. The development of social network analysis. **A Study in the Sociology of Science**, v. 1, n. 687, p. 159–167, 2004.

GAO, C.; WANG, X.; HE, X.; LI, Y. Graph neural networks for recommender system. In: **Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining**. [S.l.: s.n.], 2022. p. 1623–1625.

GOLDBERG, T. E. **Combinatorial Laplacians of Simplicial Complexes**. Tese (Doutorado) — Bard College, 2002.

HAGBERG, A. A.; SCHULT, D. A.; SWART, P. J. Exploring network structure, dynamics, and function using networkx. In: VAROQUAUX, G.; VAUGHT, T.; MILLMAN, J. (Ed.). **Proceedings of the 7th Python in Science Conference**. Pasadena, CA USA: [s.n.], 2008. p. 11 – 15.

HAIJ, M.; ISTVAN, K.; ZAMZMI, G. Cell complex neural networks. **arXiv preprint arXiv:2010.00743**, 2020.

HAIJ, M.; ZAMZMI, G.; PAPAMARKOU, T.; MIOLANE, N.; GUZMÁN-SÁENZ, A.; RAMAMURTHY, K. N. Higher-order attention networks. **arXiv preprint arXiv:2206.00606**, v. 2, n. 3, p. 4, 2022.

HAIJ, M.; ZAMZMI, G.; PAPAMARKOU, T.; MIOLANE, N.; GUZMÁN-SÁENZ, A.; RAMAMURTHY, K. N.; BIRDAL, T.; DEY, T. K.; MUKHERJEE, S.; SAMAGA, S. N. *et al.* Topological deep learning: Going beyond graph data. **arXiv preprint arXiv:2206.00606**, 2022.

HAMILTON, W. Graph representation learning. **Synthesis Lectures on Artificial Intelligence and Machine Learning**, v. 14, p. 1–159, 09 2020.

HAMILTON, W.; YING, Z.; LESKOVEC, J. Inductive representation learning on large graphs. **Advances in Neural Information Processing Systems**, v. 30, 2017.

HATCHER, A. **Algebraic topology**. [S.l.]: Cambridge University Press, 2002.

HENSEL, F.; MOOR, M.; RIECK, B. A survey of topological machine learning methods. **Frontiers in Artificial Intelligence**, Frontiers Media SA, v. 4, p. 681108, 2021.

HU, W.; FEY, M.; ZITNIK, M.; DONG, Y.; REN, H.; LIU, B.; CATASTA, M.; LESKOVEC, J. Open graph benchmark: Datasets for machine learning on graphs. **arXiv preprint arXiv:2005.00687**, 2020.

HUANG, J.; YANG, J. Unignn: a unified framework for graph and hypergraph neural networks. In: ZHOU, Z.-H. (Ed.). **Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21**. International Joint Conferences on Artificial Intelligence Organization, 2021. p. 2563–2569. Main Track. Disponível em: <<https://doi.org/10.24963/ijcai.2021/353>>.

IMMONEN, J.; SOUZA, A.; GARG, V. Going beyond persistent homology using persistent homology. **Advances in Neural Information Processing Systems**, v. 36, p. 63150–63173, 2023.

JHA, K.; SAHA, S.; SINGH, H. Prediction of protein–protein interaction using graph neural networks. **Scientific Reports**, Nature Publishing Group UK London, v. 12, n. 1, p. 8360, 2022.

JIANG, D.; WU, Z.; HSIEH, C.-Y.; CHEN, G.; LIAO, B.; WANG, Z.; SHEN, C.; CAO, D.; WU, J.; HOU, T. Could graph neural networks learn better molecular representation for drug discovery? a comparison study of descriptor-based and graph-based models. **Journal of Cheminformatics**, Springer, v. 13, p. 1–23, 2021.

KAVITHA, M.; GNANESWAR, G.; DINESH, R.; SAI, Y. R.; SURAJ, R. S. Heart disease prediction using hybrid machine learning model. In: IEEE. **2021 6th International Conference on Inventive Computation Technologies (ICICT)**. [S.l.], 2021. p. 1329–1333.

KAZI, A.; COSMO, L.; AHMADI, S.-A.; NAVAB, N.; BRONSTEIN, M. M. Differentiable graph module (dgm) for graph convolutional networks. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, v. 45, n. 2, p. 1606–1617, 2022.

KERSTING, K.; KRIEGE, N. M.; MORRIS, C.; MUTZEL, P.; NEUMANN, M. **Benchmark Data Sets for Graph Kernels**. 2016. Disponível em: <<http://graphkernels.cs.tu-dortmund.de>>.

KIPF, T. N.; WELLING, M. Semi-supervised classification with graph convolutional networks. **arXiv preprint arXiv:1609.02907**, 2016.

LI, P.; WANG, Y.; WANG, H.; LESKOVEC, J. Distance encoding: Design provably more powerful neural networks for graph representation learning. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2020.

LIU, Z.; DOU, Y.; YU, P. S.; DENG, Y.; PENG, H. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In: **Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval**. [S.l.: s.n.], 2020. p. 1569–1572.

MA, Y.; HAN, R.; WANG, W. Portfolio optimization with return prediction using deep learning and machine learning. **Expert Systems with Applications**, Elsevier, v. 165, p. 113973, 2021.

MUNKRES, J. R. **Topology**. Second edition, reissue. New York, NY: Pearson, 2018. ISBN 978-0-13-468951-7.

NIKITIN, A. V.; JOHN, S.; SOLIN, A.; KASKI, S. Non-separable spatio-temporal graph kernels via spdes. In: PMLR. **International Conference on Artificial Intelligence and Statistics**. [S.l.], 2022. p. 10640–10660.

PAIVA, F. D.; CARDOSO, R. T. N.; HANAOKA, G. P.; DUARTE, W. M. Decision-making for financial trading: A fusion approach of machine learning and portfolio selection. **Expert Systems with Applications**, Elsevier, v. 115, p. 635–655, 2019.

PAPAMARKOU, T.; BIRDAL, T.; BRONSTEIN, M.; CARLSSON, G.; CURRY, J.; GAO, Y.; HAJIJ, M.; KWITT, R.; LIÒ, P.; LORENZO, P. D.; MAROULAS, V.; MIOLANE, N.; NASRIN, F.; RAMAMURTHY, K. N.; RIECK, B.; SCARDAPANE, S.; SCHAUB, M. T.; VELIČKOVIĆ, P.; WANG, B.; WANG, Y.; WEI, G.-W.; ZAMZMI, G. Position: Topological deep learning is the new frontier for relational learning. In: **International Conference on Machine Learning (ICML)**. [S.l.: s.n.], 2024.

PAPILLON, M.; BERNÁRDEZ, G.; BATTILORO, C.; MIOLANE, N. **TopoTune : A Framework for Generalized Combinatorial Complex Neural Networks**. 2025. Disponível em: <<https://arxiv.org/abs/2410.06530>>.

PAPILLON, M.; SANBORN, S.; HAJIJ, M.; MIOLANE, N. Architectures of topological deep learning: A survey on topological neural networks. **ArXiv e-prints**, 2023.

PASZKE, A.; GROSS, S.; CHINTALA, S.; CHANAN, G.; YANG, E.; DEVITO, Z.; LIN, Z.; DESMAISON, A.; ANTIGA, L.; LERER, A. Automatic differentiation in pytorch. In: **Advances in Neural Information Processing Systems (NeurIPS - Workshop)**. [S.l.: s.n.], 2017.

PATON, K. An algorithm for finding a fundamental set of cycles of a graph. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 12, n. 9, p. 514–518, set. 1969. ISSN 0001-0782. Disponível em: <<https://doi.org/10.1145/363219.363232>>.

PLATONOV, O.; KUZNEDELEV, D.; DISKIN, M.; BABENKO, A.; PROKHORENKOVA, L. A critical look at evaluation of gnns under heterophily: Are we really making progress? In: **The Eleventh International Conference on Learning Representations**. [S.l.: s.n.], 2023.

PYT-TEAM. **TopoBenchmarkX**. 2024.

RAHMANI, S.; BAGHBANI, A.; BOUGUILA, N.; PATTERSON, Z. Graph neural networks for intelligent transportation systems: A survey. **Trans. Intell. Transport. Sys.**, v. 24, n. 8, p. 8846–8885, 2023.

RAMPÁŠEK, L.; GALKIN, M.; DWIVEDI, V. P.; LUU, A. T.; WOLF, G.; BEAINI, D. Recipe for a general, powerful, scalable graph transformer. **Advances in Neural Information Processing Systems**, v. 35, p. 14501–14515, 2022.

ROZEMBERCZKI, B.; ALLEN, C.; SARKAR, R. **Multi-scale Attributed Node Embedding**. 2020. Disponível em: <<https://openreview.net/forum?id=HJxiMAVtPH>>.

SANCHEZ-GONZALEZ, A.; GODWIN, J.; PFAFF, T.; YING, R.; LESKOVEC, J.; BATTAGLIA, P. Learning to simulate complex physics with graph networks. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2020. p. 8459–8468.

SATO, R.; YAMADA, M.; KASHIMA, H. Random features strengthen graph neural networks. In: **SIAM International Conference on Data Mining (SDM)**. [S.l.: s.n.], 2021.

SCARSELLI, F.; GORI, M.; TSOI, A. C.; HAGENBUCHNER, M.; MONFARDINI, G. The graph neural network model. **IEEE Transactions on Neural Networks**, v. 20, n. 1, p. 61–80, 2009.

SCHÖLKOPF, B.; SMOLA, A. J. **Learning with kernels: support vector machines, regularization, optimization, and beyond**. [S.l.]: MIT press, 2002.

SHALEV-SHWARTZ, S.; BEN-DAVID, S. **Understanding Machine Learning - From Theory to Algorithms**. [S.l.]: Cambridge University Press, 2014.

SHARMA, K.; LEE, Y.-C.; NAMBI, S.; SALIAN, A.; SHAH, S.; KIM, S.-W.; KUMAR, S. A survey of graph neural networks for social recommender systems. **ACM Computing Surveys**, ACM New York, NY, v. 56, n. 10, p. 1–34, 2024.

SUN, F.; SUN, J.; ZHAO, Q. A deep learning method for predicting metabolite–disease associations via graph neural network. **Briefings in Bioinformatics**, Oxford University Press, v. 23, n. 4, p. bbac266, 2022.

TELYATNIKOV, L.; BERNARDEZ, G.; MONTAGNA, M.; VASYLENKO, P.; ZAMZMI, G.; HAJIJ, M.; SCHAUB, M. T.; MIOLANE, N.; SCARDAPANE, S.; PAPAMARKOU, T. Topobenchmarkx: A framework for benchmarking topological deep learning. **arXiv preprint arXiv:2406.06642**, 2024.

VIRK, Ž. **Introduction to Persistent Homology**. Založba UL FRI, 2022. ISBN 9789617059106. Disponível em: <<https://books.google.com.br/books?id=rfUFzwEACAAJ>>.

WEISFEILER, B.; LEMAN, A. The reduction of a graph to canonical form and the algebra which appears therein. **nti, Series**, v. 2, n. 9, p. 12–16, 1968.

WILCOXON, F. Individual comparisons by ranking methods. **Biometrics Bulletin**, [International Biometric Society, Wiley], v. 1, n. 6, p. 80–83, 1945. ISSN 00994987. Disponível em: <<http://www.jstor.org/stable/3001968>>.

WU, F.; SOUZA, A.; ZHANG, T.; FIFTY, C.; YU, T.; WEINBERGER, K. Simplifying graph convolutional networks. In: CHAUDHURI, K.; SALAKHUTDINOV, R. (Ed.). **Proceedings of the 36th International Conference on Machine Learning**. PMLR, 2019. (Proceedings of Machine Learning Research, v. 97), p. 6861–6871. Disponível em: <<https://proceedings.mlr.press/v97/wu19e.html>>.

WU, Z.; PAN, S.; CHEN, F.; LONG, G.; ZHANG, C.; PHILIP, S. Y. A comprehensive survey on graph neural networks. **IEEE transactions on Neural Networks and Learning Systems**, IEEE, v. 32, n. 1, p. 4–24, 2020.

XU, K.; HU, W.; LESKOVEC, J.; JEGELKA, S. How powerful are graph neural networks? In: **International Conference on Learning Representations**. [s.n.], 2019. Disponível em: <<https://openreview.net/forum?id=ryGs6iA5Km>>.

YANG, R.; SALA, F.; BOGDAN, P. Efficient representation learning for higher-order data with simplicial complexes. In: RIECK, B.; PASCANU, R. (Ed.). **Proceedings of the First Learning on Graphs Conference**. PMLR, 2022. (Proceedings of Machine Learning Research, v. 198), p. 13:1–13:21. Disponível em: <<https://proceedings.mlr.press/v198/yang22a.html>>.

YANG, Z.; COHEN, W.; SALAKHUDINOV, R. Revisiting semi-supervised learning with graph embeddings. In: BALCAN, M. F.; WEINBERGER, K. Q. (Ed.). **Proceedings of The 33rd International Conference on Machine Learning**. New York, New York, USA: PMLR, 2016. (Proceedings of Machine Learning Research, v. 48), p. 40–48. Disponível em: <<https://proceedings.mlr.press/v48/yanga16.html>>.

YING, R.; HE, R.; CHEN, K.; EKSOMBATCHAI, P.; HAMILTON, W. L.; LESKOVEC, J. Graph convolutional neural networks for web-scale recommender systems. In: **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [S.l.: s.n.], 2018. p. 974–983.

YOU, J.; YING, R.; LESKOVEC, J. Position-aware graph neural networks. In: PMLR. **International conference on machine learning**. [S.l.], 2019. p. 7134–7143.

ZAHEER, M.; KOTTUR, S.; RAVANBAKHS, S.; POCZOS, B.; SALAKHUTDINOV, R.; SMOLA, A. Deep sets. In: **Advances in Neural Information Processing Systems (NeurIPS)**. [S.l.: s.n.], 2017.

ZHANG, B.; SHI, H.; WANG, H. Machine learning and ai in cancer prognosis, prediction, and treatment selection: a critical approach. **Journal of Multidisciplinary Healthcare**, Taylor & Francis, p. 1779–1791, 2023.