Usando Representações Não Supervisionadas Para Extração de Características Em Grafos em Dados de Redes Sociais

Erick S. Nascimento[1], **Ivo A. Pimenta**[1], **Marcelo H. Lee**[1], **Janaína R. Santos**[1], **Thelmo P. Araújo**[1], **Rafael L. Gomes**[1]

[1]Universidade Estadual do Ceará (UECE)

{erick.santos, aguiar.pimenta, marcello.lee,

janaina.ribeiro}@aluno.uece.br,

{rafa.lopes, thelmo.araujo}@uece.br

Abstract. *While Graph Neural Networks (GNNs) are essential for social networks, high computational overhead and intensive tuning often hinder adoption. We propose a decoupled learning framework for node classification, replacing deep architectures with low-dimensional structural embeddings combined with classical machine learning. By separating the embedding and classification phases, our method significantly reduces memory usage and training time. Results across multiple datasets show competitive performance, offering a scalable, interpretable alternative to standard GNNs.*

Resumo. *Embora Redes Neurais em Grafos (Graph Neural Networks - GNNs) sejam essenciais para redes sociais, a alta sobrecarga computacional e o intenso ajuste de parâmetros frequentemente dificultam sua adoção. Propomos um framework de aprendizado desacoplado para classificação de nós, substituindo arquiteturas profundas por embeddings estruturais de baixa dimensionalidade combinados com aprendizado de máquina tradicional. Ao separar as fases de representação e classificação, nosso método reduz significativamente o uso de memória e o tempo de treinamento. Resultados em diversos datasets demonstram desempenho competitivo, oferecendo uma alternativa escalável e interpretável às GNNs convencionais.*

1. Introdução

Grafos são estruturas de dados poderosas e versáteis, capazes de modelar entidades e relações entre elas em uma ampla variedade de domínios, incluindo redes sociais, sistemas biológicos, redes de citações e transações financeiras [Khoshraftar and An 2024]. A habilidade de capturar relações complexas entre nós torna os grafos fundamentais para tarefas como classificação de nós [Wang et al. 2017, Nobre et al. 2025], predição de *links* [Wei et al. 2017, Ferreira et al. 2024], identificação de ameaças [Souza et al. 2024] e detecção de comunidades [Fortunato 2010]. Contudo, essas tarefas exigem técnicas de aprendizado de máquina capazes de lidar com alta dimensionalidade e complexidade topológica de dados relacionais.

O avanço das técnicas de *embedding* de grafos surgiu como uma solução eficiente para esse desafio. O *embedding* converte dados de grafos em espaços vetoriais de baixa dimensionalidade, preservando propriedades estruturais essenciais [Cai et al. 2018].

Essa transformação permite que algoritmos tradicionais de aprendizado supervisionado ou não supervisionado operem sobre vetores compactos, evitando custos computacionais associados ao processamento direto de grandes matrizes de adjacência. No contexto de Computação Urbana e Cidades Inteligentes, grafos tornam-se ainda mais relevantes [Pimenta et al. 2024a, Pimenta et al. 2025]. O rápido crescimento do número de dispositivos conectados, impulsionado por IoT, mobilidade veicular e serviços multimídia, tem produzido redes altamente dinâmicas e distribuídas em ambientes urbanos [Brito et al. 2026, da Silva et al. 2021]. Assim como observado em redes 5G, onde há aumento sem precedentes de tráfego de vídeo, mobilidade e heterogeneidade de dispositivos [Pimenta et al. 2024b], ecossistemas urbanos geram grafos de grande escala com atributos esparsos, interações complexas e fortes restrições de memória em dispositivos de borda. Nesse cenário, nós na rede podem representar sensores, veículos, usuários, estações base ou serviços urbanos, enquanto arestas codificam relações espaciais, temporais ou funcionais. *Embeddings* de grafos surgem, portanto, como um mecanismo-chave para viabilizar análises escaláveis em ambientes urbanos distribuídos e de baixa latência.

Do ponto de vista conceitual, o Aprendizado de Representação de Grafos (*Graph Representation Learning* - GRL) pode ser dividido em duas grandes categorias: métodos de *embedding* tradicionais e métodos baseados em Redes Neurais de Grafos (*Graph Neural Networks* - GNNs) [Khoshraftar and An 2024].Os primeiros utilizam abordagens como fatoração de matrizes [Brochier et al. 2019,Epasto and Perozzi 2019,Zhu and Koniusz 2021], caminhadas aleatórias [Heidari and Papagelis 2020,Xiao et al. 2020] e *autoencoders* [Goyal et al. 2018,Mahdavi et al. 2019], enquanto os segundos adotam agregação hierárquica de vizinhança e operações convolucionais sobre a estrutura do grafo; nesse contexto, vale destacar que *embeddings* permitem representar nós, arestas e subgrafos em espaços vetoriais compactos, preservando propriedades estruturais essenciais [Makarov et al. 2021] e viabilizando o uso de modelos clássicos de aprendizado de máquina, como Regressão Logística, Florestas Aleatórias (*Random Forest* - RF) e Máquinas de Vetores de Suporte (*Support Vector Machines* - SVM), sem necessidade de arquiteturas profundas, reduzindo significativamente o tempo de treinamento e o consumo de memória ao mesmo tempo em que se mantém um desempenho competitivo em tarefas estruturais.

Dentro deste contexto, este trabalho propõe uma nova metodologia para a classificação de nós em redes sociais, baseada em uma estrutura modular e distribuída. O diferencial reside no desacoplamento entre o aprendizado de representação e a classificação final. Esta separação substitui o treinamento tradicional fim a fim por um fluxo de trabalho mais ágil, reduzindo o consumo de recursos e facilitando a implementação em arquiteturas de redes complexas. De maneira geral, a abordagem alternativa às redes neurais em grafos, explorando o uso de *embedding* estrutural para transformar *datasets* de grafos, de variados tamanhos e topologias, em representações em vetores compatíveis com modelos clássicos de aprendizado de máquina. Através desta metodologia, é possível lidar com tarefas de classificação estrutural de nós com uma melhor eficiência computacional, sem a perda significativa de desempenho, oferecendo uma solução prática para aplicações em larga escala ou para dispositivos com recursos limitados.

A contribuição principal deste trabalho é a proposta e avaliação de um *framework* de aprendizado modular e desacoplado que trata a representação de grafos e

a classificação de nós como estágios distintos. Ao substituir o treinamento fim a fim monolítico de GNNs por um *pipeline* composto por um codificador neural não supervisionado baseado em *Autoencoders* de Grafos (*Graph Autoencoders* - GAEs) seguido por classificadores tradicionais leves, demonstramos como superar os gargalos de memória inerentes às arquiteturas de passagem de mensagens (*message-passing*) padrão ($\mathcal{O}(N \times F)$). Este design alinha-se aos princípios fundamentais do Aprendizado de Representação [Bengio et al. 2013], viabilizando a reutilização de um único conjunto de *embeddings* estruturais para dar suporte a múltiplas tarefas ou classificadores subsequentes com custo computacional adicional mínimo.

O restante deste trabalho está organizado da seguinte maneira: a Seção 2 apresenta os principais estudos relacionados sobre aprendizado em representação de grafos e classificação de nós estruturada. A Seção 3 detalha o *framework* proposto, descrevendo o processo de geração de *embeddings* e o *pipeline* de classificação. A Seção 4 descreve os *datasets*, métricas e a configuração experimental empregada para validar nossa metodologia. A Seção 5 discute sobre os resultados de desempenho obtidos em termos de eficácia e eficiência computacional. Por último, a Seção 6 resume as principais contribuições deste trabalho e delineia direções para trabalhos futuros.

2. Trabalhos Relacionados

Esta seção descreve os principais trabalhos relacionados ao gerenciamento de redes e à análise de desempenho publicados recentemente, considerando questões de desempenho e qualidade.

Luo et al. [Luo et al. 2024] apresentam um novo *framework* de tokenização de grafos que gera identificadores semânticos de nós (IDs) sensíveis à estrutura na forma de sequências curtas de códigos discretos. Estes IDs de nós são derivados via quantização vetorial em *embeddings* de nós de múltiplas camadas de uma GNN, compromindo, assim, representações contínuas em códigos simbólicos compactos que ainda refletem a estrutura de vizinhança de múltiplas ordens. Experimentos feitos em 34 *datasets* demonstram que estes IDs não apenas melhoram a eficiência de inferência, mas também alcançam desempenho competitivo comparado a métodos do estado da arte.

Li et al. [Li et al. 2024] introduzem um *framework* para *embedding* de nós em grafos de propriedade por meio de amostragem de vizinhança latente. O método enfatiza como selecionar e avaliar vizinhanças latentes para obter um melhor desempenho na classificação de nós. A contribuição é dupla: primeiro a adaptação de *embeddings* para grafos de propriedade; segundo, a definição de uma métrica quantitativa para a utilidade da vizinhança de um nó em tarefas de classificação. Ademais, ao incorporar atributos dos nós diretamente no processo de *embedding*, o *framework* permite representações mais discriminativas em estruturas de grafos heterogêneas, especialmente quando o contexto relacional por si só é insuficiente.

Dalvi et al. [Dalvi and Honavar 2025] propõem o mapeamento das características dos nós em hipervetores de altíssima dimensão e a agregação de informações da vizinhança por meio de operadores hiperdimensionais. O algoritmo de aprendizado *"one-pass"* dos autores atua como uma alternativa às GNNs tradicionais, apresentando menor custo computacional enquanto mantém uma acurácia competitiva em tarefas de classificação de nós e predição de *links*. Esta abordagem demonstra o potencial de

operações algébricas em espaços vetoriais de alta dimensão para codificar padrões relacionais complexos sem a necessidade de passagem de mensagem iterativa, reduzindo, assim, tanto o tempo de treinamento quanto a complexidade de modelo.

Lexxa et al. [Lecca and Lecca 2023] exploram aplicações de *embedding* de grafos e aprendizado profundo geométrico em domínios como a biologia de redes e a química estrutural. Entre suas descobertas está a observação de que *embeddings* permitem que modelos mais simples (por exemplo, SVMs ou regressão logística) operem de forma eficaz em grandes volumes de dados estruturados em grafos, enquanto modelos profundos podem se tornar computacionalmente inviáveis. A análise dos autores reforça a relevância de soluções escaláveis, particularmente em domínios onde *datasets* crescem rapidamente e recursos computacionais podem ser limitados.

Khan et al. [Khan et al. 2025] introduzem a *Distance-Driven Graph Neural Network* (DDGNN), um modelo projetado para aprimorar a classificação estrutural de nós ao incorporar explicitamente características sensíveis à distância no processo de aprendizado de grafos. Ao contrário das GNNs convencionais, que dependem exclusivamente da agregação de vizinhança local, a DDGNN adota nós marcadores para codificar distâncias (*inter-node*), capturando, assim, dependências estruturais tanto locais quanto globais. Esse mecanismo permite que o modelo diferencie melhor nós com contextos semelhantes, mas com papéis topológicos distintos, mitigando de forma eficaz o problema da suavização excessiva (*over-smoothing*), comumente observado em arquiteturas mais profundas. Resultados experimentais em múltiplos *datasets* de grafos públicos demonstram que a DDGNN alcança melhorias na acurácia de classificação e robustez em diversas estruturas de grafos complexas. Embora a abordagem ainda dependa da passagem de mensagem e de uma parametrização profunda, ela reforça a importância de representações que preservam a estrutura na classificação de nós, uma ideia central também no *framework* de *embedding* neural leve que propomos.

Neste trabalho, ao contrário da maioria dos métodos existentes que dependem de redes neurais em grafos profundas ou de *pipelines* complexos de *embedding*, nós propomos uma abordagem leve e eficaz para classificação estrutural de nós que se afasta das arquiteturas GNNs pesadas. Uma vez obtidos, esses *embedding* são fornecidos como vetores de tamanho fixo a classificadores de aprendizado de máquina padrão da biblioteca *sickit-learn*. Diferentemente de trabalhos anteriores, que frequentemente integram *embedding* e classificação em uma única arquitetura profunda, nenhum deles desacopla explicitamente estes estágios para equilibrar a expressividade estrutural com a eficiência computacional. Nossa metodologia introduz esta separação, preservando a informação topológica por meio de *embeddings* estruturais neurais, ao mesmo tempo que possibilita o uso de classificadores leves e interpretáveis. Esta concepção permite lidar com grafos de tamanhos e topologias variadas sem a sobrecarga de memória e treinamento típica associada com modelos GNN profundos.

3. Proposta

Esta pesquisa propõe uma arquitetura modular para classificação de nós em grafos sociais, concebida sob o paradigma de aprendizado desacoplado e orientada a cenários de redes e sistemas distribuídos. Uma visão geral da proposta é apresentada na Figura 1. Diferentemente de abordagens monolíticas tradicionais, que realizam o treinamento su-

pervisionado de forma fim a fim, a metodologia desenvolvida separa explicitamente o aprendizado de representação estrutural da etapa final de inferência, permitindo maior flexibilidade, eficiência e adequação a ambientes com restrições de recursos.

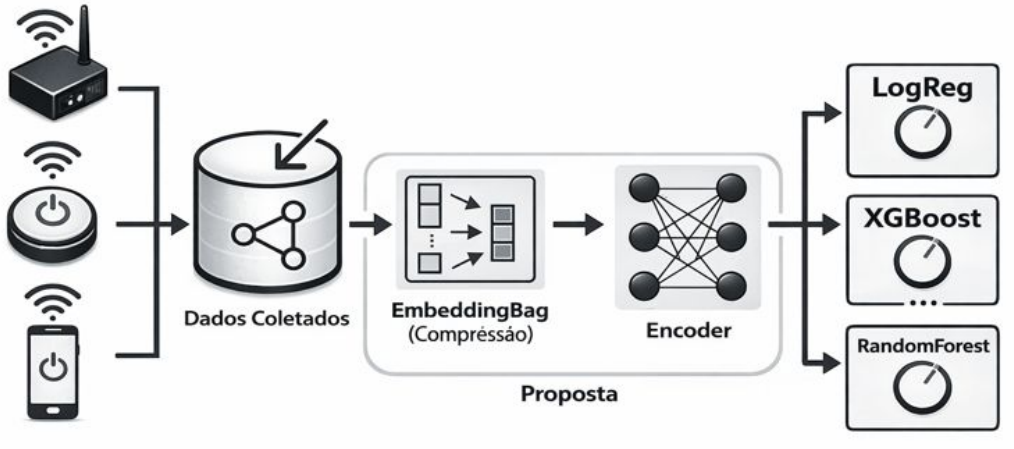

Figura 1. Visão geral da arquitetura proposta: (1) compressão eficiente de atributos esparsos, (2) aprendizado não supervisionado da estrutura do grafo e (3) classificação *downstream* desacoplada.

Sob a perspectiva de redes sociais integradas a ambientes urbanos inteligentes, essa separação torna-se particularmente relevante. Plataformas sociais contemporâneas atuam como infraestruturas digitais urbanas, conectando usuários, serviços e sistemas físicos por meio de interações massivas, dinâmicas e heterogêneas. Nesse contexto, é desejável que a etapa mais custosa de aprendizado, que envolve a estrutura completa do grafo e seus atributos, seja executada em servidores centrais ou na nuvem, enquanto a inferência final possa ser realizada de forma distribuída, a partir de representações compactas, sem a necessidade de manter toda a topologia social e os atributos originais em memória.

O fluxo de processamento é organizado em dois estágios principais: (i) a geração não supervisionada de representações latentes (*embeddings*) robustas, explorando a estrutura do grafo, e (ii) a aplicação de classificadores leves para a tarefa de predição. O objetivo central é otimizar a relação de compromisso entre eficiência computacional, especialmente em termos de consumo de memória, tempo de treinamento e desempenho preditivo, viabilizando o processamento de grafos sociais com atributos de alta cardinalidade em cenários distribuídos e com recursos limitados.

Considera-se um grafo não direcionado $G = (V, E)$, onde $V = \{v_1, \ldots, v_N\}$ representa o conjunto de N nós e E o conjunto de arestas. A estrutura topológica é descrita pela matriz de adjacência $A \in \{0, 1\}^{N \times N}$. Cada nó está associado a um vetor de atributos, organizados na matriz $X \in R^{N \times F}$, em geral esparsa no caso de redes sociais reais, onde F denota a dimensionalidade dos atributos brutos, frequentemente elevada. O objetivo consiste em aprender uma função de mapeamento $f : V \to \mathcal{Y}$ capaz de predizer o rótulo y_v de um nó v, utilizando um *embedding* compacto $Z \in R^{N \times d}$, com $d \ll F$.

A arquitetura proposta é composta por três módulos sequenciais, projetados para mitigar os elevados custos de memória e processamento associados às GNNs tradicionais: (i) Projeção Eficiente e Compressão de Atributos, (ii) *Encoder* Não Supervisionado e

Função Objetivo, e (iii) Classificação *Downstream* Desacoplada.

3.1. Projeção Eficiente e Compressão

O primeiro gargalo em grafos sociais de larga escala está associado à alta dimensionalidade e esparsidade da matriz de atributos X. O processamento direto de vetores multiativos (*multi-hot*), nos quais F pode ultrapassar dezenas de milhares de dimensões, impõe custos significativos de memória e comunicação. Para mitigar esse problema, emprega-se uma camada *EmbeddingBag*, responsável por projetar os atributos esparsos em um espaço denso de menor dimensionalidade antes do processamento estrutural.

Diferentemente de uma camada linear densa que realizaria explicitamente a operação XW, a *EmbeddingBag* computa a soma ou média dos *embeddings* associados aos índices ativos em cada vetor de atributos. Essa estratégia reduz o requisito de memória de entrada de $\mathcal{O}(N \times F)$ para $\mathcal{O}(N \times d_{proj})$, onde d_{proj} representa a dimensão inicial de projeção. Tal redução é particularmente vantajosa em ambientes distribuídos e cenários urbanos, nos quais a eficiência de memória é um fator crítico.

3.2. *Encoder* Não Supervisionado e Função Objetivo

O núcleo da proposta é um *encoder* baseado em GNN, avaliando-se arquiteturas como GraphSAGE, GAT e GIN, responsável por processar a estrutura do grafo e os atributos previamente comprimidos para gerar o *embedding* Z. O treinamento é conduzido de forma estritamente não supervisionada, explorando a própria estrutura do grafo como sinal de supervisão, caracterizando um cenário de *self-supervision*.

A arquitetura do *autoencoder* é definida dinamicamente durante a etapa de otimização de hiperparâmetros, podendo operar sob dois paradigmas distintos: *Graph Autoencoder* (GAE) ou *Variational Graph Autoencoder* (VGAE). No caso do VGAE, o *encoder* parametriza uma distribuição latente gaussiana $q(Z|X, A)$, descrita por vetores de média μ e desvio padrão σ. A função objetivo combina a perda de reconstrução da adjacência com um termo de regularização baseado na Divergência de Kullback–Leibler (KL):

$$\mathcal{L}_{VGAE} = E_{q(Z|X,A)}[\log p(A|Z)] - KL[q(Z|X, A)\|p(Z)], \tag{1}$$

onde $p(A|Z)$ representa o decodificador, implementado por meio do produto interno seguido de uma função sigmoide, e $p(Z)$ corresponde à distribuição a priori gaussiana $\mathcal{N}(0, I)$. Na configuração GAE, o *embedding* é determinística ($Z = \mu$) e o termo de regularização KL é omitido, concentrando-se exclusivamente na reconstrução da estrutura topológica.

3.3. Classificação *Downstream* Desacoplada

Após a convergência do *encoder*, as representações latentes Z (ou a média μ no caso variacional) são extraídas e congeladas. A partir desse ponto, tanto a estrutura topológica A quanto os atributos originais X são descartados da memória, permanecendo apenas a matriz compacta $Z \in R^{N\times d}$. Esses vetores passam então a servir como entrada para classificadores tradicionais, tais como Regressão Logística, SVM, KNN, Random Forest e XGBoost.

Essa estratégia consolida o desacoplamento entre o aprendizado de representação e a inferência, permitindo que múltiplos modelos preditivos sejam avaliados ou atualizados sem a necessidade de reexecutar a dispendiosa etapa de passagem de mensagens no grafo. Tal característica é especialmente relevante em cenários distribuídos e de computação urbana, nos quais a inferência pode ser realizada em dispositivos de borda ou nós com capacidade computacional limitada.

No que se refere ao protocolo de seleção de modelos adotado para a proposta (*Best + Soft-mean*), reconhece-se que, em cenários não supervisionados, a perda de reconstrução nem sempre reflete adequadamente a qualidade da separação entre classes. Para mitigar esse efeito, emprega-se uma métrica *proxy* avaliada em tempo de execução sobre o conjunto de validação. A cada época, sondas leves, como k-NN e Regressão Logística, são utilizadas para avaliar a qualidade do *embedding*. A pontuação final utilizada para seleção do modelo é definida como:

$$Score_{final} = \max(S) + \mu\Big(\{s \in S \mid s \geq 0.75 \cdot \max(S)\}\Big), \tag{2}$$

onde S representa o conjunto de F1-scores obtidos pelas sondas. Essa métrica favorece representações que apresentam alto desempenho para ao menos um classificador, ao mesmo tempo em que preserva consistência global. Para os modelos supervisionados utilizados como *baseline*, a seleção foi realizada diretamente com base no maior F1-score ponderado obtido no conjunto de validação.

Por fim, é válido ressaltar que com o objetivo de garantir a reprodutibilidade científica, o código-fonte da solução proposta, bem como o conjunto de dados utilizado nos experimentos, estão publicamente disponíveis em um repositório online[1].

4. Experimentos

Esta seção descreve os conjuntos de dados, o ambiente computacional e o protocolo experimental adotados para validar a eficácia e a eficiência da abordagem proposta. Inicialmente, define-se o *baseline* comparativo utilizado para mensurar o desempenho relativo da proposta.

4.1. Baseline: Treinamento Supervisionado (fim a fim)

Para estabelecer um ponto de comparação robusto e representar o estado da arte, adotou-se um *baseline* composto por GNNs supervisionadas treinadas fim a fim, nas quais a estrutura do grafo e os atributos brutos dos nós são processados simultaneamente e otimizados diretamente via perda de classificação. Diferentemente da abordagem proposta, esse *pipeline* padrão da literatura utiliza vetores de codificação binária múltipla de alta dimensionalidade sem compressão prévia nem desacoplamento entre representação e classificação, evidenciando o custo computacional real em termos de memória e tempo durante as iterações de passagem de mensagens.

4.2. Conjuntos de Dados

Para avaliar o desempenho em cenários caracterizados por atributos de alta cardinalidade, foram selecionados dois *datasets* reais da coleção MUSAE [Rozemberczki

[1] `https://github.com/LarcesUece/SBRC_2026_embeddings`

et al. 2019], ambos compostos por grafos sociais não direcionados com vetores de atributos esparsos de alta dimensionalidade. O primeiro, **MUSAE-GitHub**, representa uma rede social de desenvolvedores onde as arestas denotam relações de "seguir", definindo uma tarefa de classificação binária (Desenvolvedor Web vs. *Machine Learning*). O segundo, **MUSAE-Facebook**, mapeia páginas verificadas conectadas por curtidas mútuas, configurando um problema multiclasse com quatro categorias (Políticos, Governo, TV e Empresas).

Os dados foram particionados de forma estratificada na proporção de 80% para treino, 10% para validação e 10% para teste. Essa distribuição foi adotada para maximizar a exposição do codificador à variabilidade estrutural dos grafos, garantindo robustez na fase de aprendizado, ao mesmo tempo em que se preserva uma quantidade estatisticamente significativa de amostras para o ajuste de hiperparâmetros e avaliação final, conforme as práticas recomendadas para modelos de aprendizado profundo [Goodfellow et al. 2016]. Adotou-se um protocolo transdutivo, no qual a estrutura topológica completa é acessível durante a propagação de mensagens, mas os rótulos de teste permanecem ocultos durante o treinamento.

4.3. Ambiente Computacional

As especificações de *hardware* foram selecionadas de modo a permitir melhor controle e coleta de métricas, evitando o uso de aceleradores de alto desempenho, como GPUs de data center. O ambiente experimental foi composto por um processador Intel(R) Core(TM) i7-14700T, 32 GB de memória RAM operando a 5600 MT/s e um conjunto de ferramentas de software baseado em Python, incluindo PyTorch, PyTorch Geometric e Scikit-Learn.

4.4. Protocolo de Treinamento e Otimização

Para a etapa de otimização de hiperparâmetros via Optuna, fixou-se a dimensão dos *embeddings* de saída em $d = 32$. Essa escolha fundamenta-se em uma análise preliminar do espaço de busca, situando-se como um ponto de equilíbrio entre representações excessivamente compactas e dimensões mais elevadas (que aumentariam a complexidade computacional sem ganhos proporcionais de capacidade de previsão nesta fase).

Uma vez definida a melhor arquitetura e o conjunto ideal de hiperparâmetros com $d = 32$, essas configurações foram mantidas constantes para a exploração subsequente das demais dimensões latentes $(8, 16, 64, 128)$.

4.5. Critérios de parada e métrica de seleção

Para assegurar a estabilidade e a padronização experimental, o protocolo de treinamento utilizou o otimizador Adam com taxa de aprendizado inicial de 10^{-3} e decaimento de peso (*weight decay*) de 5×10^{-4}. O controle dinâmico da otimização foi realizado pela estratégia *ReduceLROnPlateau*, configurada para reduzir a taxa de aprendizado em um fator de 0,6 após 5 épocas sem melhoria na validação, respeitando o limite inferior de 10^{-8}. Por fim, aplicou-se o critério de parada antecipada com paciência de 32 épocas e tolerância mínima (min_delta) de 10^{-6}, estabelecendo-se um teto de 500 épocas, embora a convergência tenha sido atingida precocemente em todos os cenários avaliados.

4.6. Métricas de Avaliação

Dada a natureza desbalanceada dos *datasets*, adotou-se o F1-score Ponderado (*Weighted*) como métrica primária de eficácia. Para mensurar a eficiência computacional, monitoraram-se o pico de memória (consumo máximo de RAM em MiB) e o tempo de execução. Este último foi mensurado pelo tempo de relógio (*wall-clock time*), que contabiliza o tempo total decorrido do início ao fim da tarefa, englobando latências de sistema e operações de entrada/saída, diferindo, portanto, do tempo de CPU puro e refletindo melhor a latência real em produção.

5. Resultados

Esta seção apresenta e discute os resultados experimentais obtidos na avaliação do *framework* desacoplado proposto, analisando tanto sua eficácia preditiva quanto sua eficiência computacional em comparação com baselines baseados em GNNs treinadas fim a fim. A análise considera múltiplos aspectos complementares, incluindo custo teórico de representação, impacto da dimensionalidade dos *embeddings*, tempo de treinamento, tempo de inferência e pico de consumo de memória, de modo a fornecer uma visão abrangente das relações de compromisso entre desempenho e custo computacional.

5.1. Eficiência de Representação de Dados

Para isolar o impacto da arquitetura proposta no consumo de memória, realizou-se uma análise analítica do custo de representação (*Theoretical Representation Cost*), que considera apenas o volume de bytes necessário para armazenar os tensores fundamentais de entrada, sem o *overhead* de bibliotecas ou alocações temporárias. No Baseline (GNN fim a fim), esse custo é dominado pela matriz de atributos densa ($X \in R^{N \times F}$) e pela estrutura do grafo ($A \in Z^{2 \times E}$), tornando-se particularmente elevado em grafos com alta cardinalidade de atributos ($F > 4000$). Já na Abordagem Proposta, o custo de representação é substancialmente reduzido, pois após a geração dos vetores latentes o grafo original é descartado da memória, permanecendo apenas a matriz compacta de *embeddings* ($Z \in R^{N \times d}$) e os rótulos.

Tabela 1. Estimativa do custo teórico de representação para o baseline e para a abordagem proposta nos *datasets* MUSAE-Facebook e MUSAE-GitHub.

Dataset	Baseline	d=8	d=16	d=32	d=64	d=128
Facebook	409.5	0.9	1.6	3.0	5.7	11.2
GitHub	585.2	1.6	2.7	5.0	9.6	18.8

A Tabela 1 apresenta os resultados obtidos através do cálculo direto do tamanho dos tensores (considerando precisão *float32* para atributos/*embeddings* e *int64* para índices). Desta forma, estes resultados representam a estimativa do custo teórico de representação em memória para os *datasets* MUSAE-Facebook e MUSAE-GitHub, respectivamente, comparando o baseline fim a fim com a abordagem proposta desacoplada. Em ambos os casos, observa-se que o custo do baseline é dominado pela matriz de atributos de alta dimensionalidade, tornando-se significativamente maior que o custo associado apenas aos *embeddings* latentes. À medida que a dimensão dos *embeddings* aumenta,

o consumo de memória da abordagem proposta cresce de forma moderada e linear, enquanto o baseline permanece praticamente inalterado e muito mais elevado. Esse comportamento evidencia que o *pipeline* desacoplado reduz drasticamente a dependência de atributos brutos esparsos, proporcionando ganhos substanciais de eficiência de memória sem comprometer a utilidade das representações para tarefas *downstream*.

As arquiteturas foram selecionadas visando a maximização do F1-score ponderado, com exceção dos modelos não supervisionados, para os quais adotou-se o critério *Best + Soft-mean*. Como resultado deste processo de otimização, as combinações de melhor desempenho identificadas foram: GAE acoplado ao k-NN para o *dataset* MUSAE-Facebook e VGAE combinado com XGBoost para o MUSAE-GitHub.

5.2. Desempenho vs Dimensionalidade

A Figura 2 ilustra a relação entre a dimensionalidade dos *embeddings* e o desempenho preditivo medido pelo F1-score, bem como a relação de compromisso global entre qualidade de representação e custo computacional. Observa-se que, para ambos os *datasets*, o aumento da dimensão latente inicialmente promove ganhos de desempenho, refletindo maior capacidade dos *embeddings* em capturar informações estruturais relevantes do grafo. Entretanto, após certo ponto, esses ganhos tendem a saturar ou apresentar retornos decrescentes, indicando que dimensões muito elevadas não necessariamente resultam em melhorias proporcionais no F1-score. A análise macro evidencia que dimensões intermediárias oferecem um equilíbrio mais favorável entre desempenho e eficiência, sugerindo que é possível alcançar resultados competitivos com *embeddings* relativamente compactos, evitando sobrecarga computacional desnecessária.

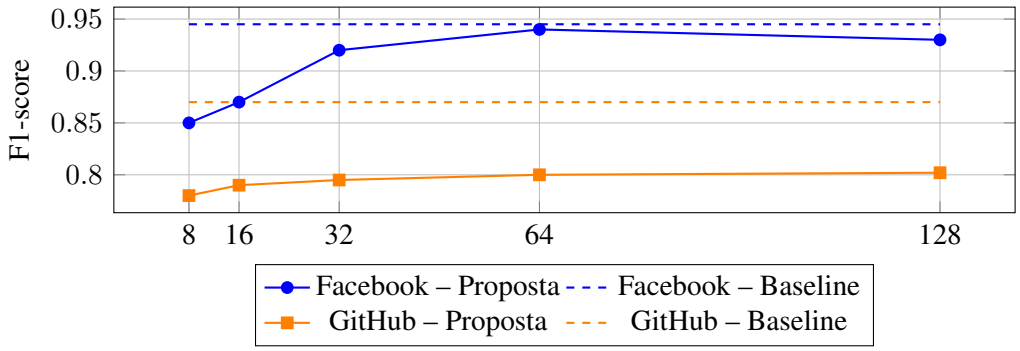

Figura 2. Desempenho em função da dimensionalidade dos *embeddings* (F1-score).

Além disso, os resultados indicam que a escolha da dimensionalidade dos *embeddings* exerce influência direta não apenas no desempenho preditivo, mas também na estabilidade do modelo e na eficiência do *pipeline* como um todo. Dimensões muito reduzidas tendem a limitar a expressividade do *embedding*, resultando em perda de informação relevante para a tarefa de classificação. Por outro lado, dimensões excessivamente elevadas aumentam a complexidade do modelo sem oferecer ganhos proporcionais de desempenho, o que pode amplificar custos computacionais e riscos de sobreajuste. Nesse contexto, a adoção de dimensões intermediárias emerge como uma estratégia pragmática, permitindo capturar adequadamente a estrutura do grafo ao mesmo tempo em que preserva eficiência e escalabilidade, especialmente em cenários com restrições de recursos ou requisitos de resposta em tempo quase real.

5.3. Análise do Tempo de Treino e Inferência

A Tabela 2 compara o tempo de treinamento entre o *baseline* fim a fim e a abordagem proposta para os *datasets* MUSAE-Facebook e MUSAE-GitHub. Observa-se que o treinamento do *baseline* apresenta tempos substancialmente superiores, refletindo o elevado custo computacional associado ao processamento direto da estrutura completa do grafo e dos atributos de alta dimensionalidade durante a passagem de mensagens. Em contraste, o *pipeline* desacoplado reduz significativamente o tempo total de treinamento, uma vez que a etapa mais onerosa, a geração dos *embeddings* é executada apenas uma única vez, enquanto os classificadores *downstream* são leves e rápidos de treinar. Esse comportamento evidencia que a abordagem proposta é mais eficiente e escalável, tornando-se particularmente vantajosa para grafos de grande porte ou para ambientes com recursos computacionais limitados.

Tabela 2. Tempo de treinamento e inferência para diferentes configurações de *embeddings* nos *datasets* MUSAE-Facebook e MUSAE-GitHub.

Dataset	Fase	Baseline	d=8	d=16	d=32	d=64	d=128
Facebook	Treinamento (s)	610	110	200	105	150	180
	Inferência (s)	1.45	0.26	0.38	0.57	0.79	0.97
GitHub	Treinamento (s)	720	320	450	380	420	1000
	Inferência (s)	2.60	0.40	0.50	0.45	0.50	0.55

A Tabela 2 também apresenta o tempo de inferência do baseline fim a fim e da abordagem proposta nos *datasets* MUSAE-Facebook e MUSAE-GitHub. Em ambos os cenários, a abordagem desacoplada alcança latências de inferência significativamente menores quando comparada à GNN treinada de forma fim a fim. Esse ganho decorre do fato de que, no *pipeline* proposto, a etapa de classificação opera exclusivamente sobre a representação compacta dos *embeddings*, evitando a re-computação sobre a estrutura completa do grafo e os atributos originais. Por outro lado, o baseline requer o processamento integral da topologia do grafo e dos vetores Vetor multiativos a cada predição, o que eleva substancialmente o tempo de inferência. Esses resultados indicam que o *pipeline* desacoplado é mais adequado para aplicações que demandam respostas rápidas ou que operam sob restrições rigorosas de latência, como sistemas de borda e cenários urbanos em tempo quase real.

5.4. Análise de Memória no Treinamento e na Inferência

A Tabela 3 apresenta o pico de consumo de memória RAM durante a etapa de treinamento para o *baseline* fim a fim e para a abordagem proposta nos *datasets* MUSAE-Facebook e MUSAE-GitHub. Em ambos os casos, observa-se uma redução substancial do uso de memória quando se adota o *pipeline* desacoplado, evidenciando que a principal fonte de custo do *baseline* está associada à manutenção simultânea da topologia completa do grafo e dos vetores multiativos de alta dimensionalidade durante a *message passing*. Em contraste, na abordagem proposta, após a geração dos *embeddings* latentes, a etapa de treinamento dos classificadores opera exclusivamente sobre representações compactas, o que reduz significativamente a pressão sobre a memória. Esses resultados indicam que a proposta é mais eficiente e viável para cenários de grande escala ou ambientes com recursos computacionais limitados, nos quais o pico de memória constitui um fator crítico.

Tabela 3. Pico de consumo de memória RAM (MiB) durante o treinamento e a inferência para o baseline fim a fim e a abordagem proposta nos *datasets* MUSAE-Facebook e MUSAE-GitHub.

Dataset	Fase	Baseline	d=8	d=16	d=32	d=64	d=128
Facebook	Treinamento (MiB)	8 500	3 000	3 050	3 000	3 050	3 100
	Inferência (MiB)	12 010	2 150	2 175	2 190	2 205	2 550
GitHub	Treinamento (MiB)	11 500	2 800	2 750	2 800	2 900	4 200
	Inferência (MiB)	11 844	2 161	2 161	2 162	2 162	2 528

A Tabela 3 também sumariza o pico de consumo de memória RAM durante a etapa de inferência para o *baseline* fim a fim e para a abordagem proposta nos *datasets* MUSAE-Facebook e MUSAE-GitHub. Em ambos os cenários, observa-se que o *baseline* mantém um consumo elevado de memória mesmo na fase de predição, uma vez que necessita manter simultaneamente em memória a estrutura completa do grafo e os atributos originais para executar a *message passing*. Por outro lado, na abordagem desacoplada, o pico de memória durante a inferência é substancialmente inferior, visto que apenas a matriz compacta de *embeddings* e o classificador leve precisam permanecer carregados. Esse comportamento confirma que o *pipeline* proposto não apenas reduz os custos de memória no treinamento, mas também viabiliza uma inferência mais eficiente e adequada para aplicações que exigem baixa latência, como sistemas de computação de borda e cenários urbanos inteligentes.

6. Conclusão

Este trabalho apresentou um *framework* modular e desacoplado para classificação estrutural de nós em grafos, no qual a etapa de aprendizagem de representações é separada da etapa de classificação downstream. Por meio de *embeddings* estruturais não supervisionados combinados com classificadores leves tradicionais, demonstramos que é possível alcançar desempenho competitivo em relação a GNNs fim a fim, ao mesmo tempo em que se obtêm ganhos expressivos em eficiência de memória e tempo de treinamento e inferência. Os experimentos realizados nos *datasets* MUSAE-Facebook e MUSAE-GitHub evidenciam que a proposta mitiga gargalos típicos de arquiteturas de passagem de mensagens, tornando-se uma alternativa prática e escalável para grafos de grande porte e ambientes com recursos computacionais limitados, incluindo cenários de computação de borda e computação urbana.

Como trabalhos futuros, a expansão da avaliação para cenários indutivos e grafos de escala industrial (milhões de nós), onde a eficiência de memória demonstrada torna-se ainda mais crítica. Planeja-se também investigar a integração de *encoders* robustos à heterofilia, visando superar a suposição de homofilia predominante em GNNs clássicas. Por fim, a validação da capacidade de generalização dos *embeddings* congelados em outras tarefas *downstream*, como predição de links e detecção de comunidades, consolidando o potencial multitarefa e generalista do *framework* proposto.

Agradecimentos

Os autores gostariam de agradecer ao CNPq (N^o 305946/2025-0 e N^o 405940/2022-0) e CAPES (N^o 88887.954253/2024-00 e N^o 88887.972043/2024-00)

pelo apoio financeiro.

Referências

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Brito, M. L. L., Ferreira, M. C. M., Portela, A. L. C., and Gomes, R. L. (2026). Ai-based estimation of bandwidth availability for data offloading in edge-cloud computing. *IEEE Networking Letters*, 8:69–73.

Brochier, R., Guille, A., and Velcin, J. (2019). Global vectors for node representations. In *The World Wide Web Conference*, pages 2587–2593.

Cai, H., Zheng, V. W., and Chang, K. C.-C. (2018). A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, 30(9):1616–1637.

da Silva, M. d. V. D., Rocha, A., Gomes, R. L., and Nogueira, M. (2021). Lightweight data compression for low energy consumption in industrial internet of things. In *2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)*, pages 1–2.

Dalvi, A. and Honavar, V. (2025). Hyperdimensional representation learning for node classification and link prediction. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 88–97.

Epasto, A. and Perozzi, B. (2019). Is a single embedding enough? learning node representations that capture multiple social contexts. In *The world wide web conference*, pages 394–404.

Ferreira, M. C., Ribeiro, S. E., Nobre, F. V., Linhares, M. L., Araújo, T. P., and Gomes, R. L. (2024). Mitigating measurement failures in throughput performance forecasting. In *2024 20th International Conference on Network and Service Management (CNSM)*, pages 1–7.

Fortunato, S. (2010). Community detection in graphs. *Physics reports*, 486(3-5):75–174.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goyal, P., Kamra, N., He, X., and Liu, Y. (2018). Dyngem: Deep embedding method for dynamic graphs. *arXiv preprint arXiv:1805.11273*.

Heidari, F. and Papagelis, M. (2020). Evolving network representation learning based on random walks. *Applied network science*, 5(1):18.

Khan, I., Bokhari, M. U., Afzal, S., and Alam, S. (2025). Distance driven graph neural network for advanced node classification through feature augmentation. *Discover Computing*, 28(1):70.

Khoshraftar, S. and An, A. (2024). A survey on graph representation learning methods. *ACM Transactions on Intelligent Systems and Technology*, 15(1):1–55.

Lecca, P. and Lecca, M. (2023). Graph embedding and geometric deep learning relevance to network biology and structural chemistry. *Frontiers in Artificial Intelligence*, 6:1256352.

Li, S., Zaidi, N. A., Du, M., Zhou, Z., Zhang, H., and Li, G. (2024). Property graph representation learning for node classification. *Knowledge and Information Systems*, 66(1):237–265.

Luo, Y., Liu, Q., Shi, L., and Wu, X.-M. (2024). Structure-aware semantic node identifiers for learning on graphs. *arXiv e-prints*, pages arXiv–2405.

Mahdavi, S., Khoshraftar, S., and An, A. (2019). Dynamic joint variational graph autoencoders. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 385–401. Springer.

Makarov, I., Kiselev, D., Nikitinsky, N., and Subelj, L. (2021). Survey on graph embeddings and their applications to machine learning problems on graphs. *PeerJ Computer Science*, 7:e357.

Nobre, F. V. J., Silva, D. d. S., Ferreira, M. C. M. M., Brito, M. L. M. L., de Araújo, T. P., and Gomes, R. L. (2025). Time-weighted correlation approach to identify high delay links in internet service providers. *Journal of Internet Services and Applications*, 16(1):419–430.

Pimenta, I., Silva, D., Moura, E., Silveira, M., and Gomes, R. L. (2024a). Impact of data anonymization in machine learning models. In *Proceedings of the 13th Latin-American Symposium on Dependable and Secure Computing*, pages 188–191.

Pimenta, I. A., Aquino, C. A., Almeida, Y. O., Lima, V. C., and Gomes, R. L. (2024b). Prediction of multimedia quality over 5g networks in urban environments. In *2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, pages 748–755.

Pimenta, I. A., Lee, M. H., Bittencourt, L. F., and Gomes, R. L. (2025). Adaptive privacy based on mutual information for machine learning in edge-cloud environments. *IEEE Networking Letters*.

Rozemberczki, B., Allen, C., and Sarkar, R. (2019). Multi-scale attributed node embedding.

Souza, M. S., Ribeiro, S. E. S. B., Lima, V. C., Cardoso, F. J., and Gomes, R. L. (2024). Combining regular expressions and machine learning for sql injection detection in urban computing. *Journal of Internet Services and Applications*, 15(1):103–111.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. (2017). Community preserving network embedding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Wei, X., Xu, L., Cao, B., and Yu, P. S. (2017). Cross view link prediction by learning noise-resilient representation consensus. In *Proceedings of the 26th international conference on World Wide Web*, pages 1611–1619.

Xiao, W., Zhao, H., Zheng, V. W., and Song, Y. (2020). Vertex-reinforced random walk for network embedding. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 595–603. SIAM.

Zhu, H. and Koniusz, P. (2021). Refine: Random range finder for network embedding. In *Proceedings of the 30th ACM international conference on information & knowledge management*, pages 3682–3686.