Capítulo

# 4

# Redes Neurais de Grafos no Contexto das Cidades Inteligentes

Cláudio Gustavo Santos Capanema, Fabrício Aguiar Silva, Antonio Alfredo Ferreira Loureiro

***Abstract***

*Graph neural networks, also known as GNNs, have been applied to solve problems in different domains, such as biology, chemistry, physics, natural language processing, computer vision, economics, among others. In particular, this class of neural network has been shown to be effective in modeling problems related to smart cities, such as traffic prediction; urban function classification of elements such as points of interest, roads and regions; forecasting the spread of diseases; autonomous agents; anomaly detection, among other activities. The objective of this chapter is to present the theoretical and practical foundations of graph neural networks in the context of smart cities. For this, the following topics are presented: types of tasks performed, taxonomy of GNNs layers, concepts, commonly used architectures, problems related to smart cities, modeling for the transformation of raw data to the graph structure and GNN structure, in addition to the implementation a model in practice.*

***Resumo***

*Redes neurais de grafos, também conhecidas como Graph Neural Networks (GNNs), têm sido aplicadas para resolver problemas em diferentes domínios, como biologia, química, física, processamento de linguagem natural, visão computacional, economia, dentre outros. Em particular, essa classe de rede neural tem-se mostrado eficaz na modelagem de problemas relacionados às cidades inteligentes, como predição de tráfego; classificação de função urbana de elementos como pontos de interesse, estradas e regiões; previsão de disseminação de doenças; agentes autônomos; detecção de anomalia, dentre outras atividades. O objetivo deste capítulo é apresentar os fundamentos teóricos e práticos das redes neurais de grafos no contexto das cidades inteligentes. Para isto, são apresentados os seguintes tópicos: tipos de tarefas realizadas, taxonomia das camadas GNNs, conceitos, arquiteturas comumente utilizadas, problemas relacionados*

*às cidades inteligentes, modelagens para a transformação de dados brutos para a estrutura de grafo e rede GNN, além da implementação de um modelo na prática.*

## 4.1. Introdução

Grafo é um objeto matemático cuja representação computacional pode modelar elementos individuais, seus relacionamentos e estrutura. O seu potencial de utilização em diferentes contextos tem atraído a atenção de muitos pesquisadores que, por sua vez, têm explorado grafos no contexto de redes neurais para a solução de diversos problemas que envolvem grandes volumes de dados. As redes neurais de grafos (do inglês *Graph Neural Networks*, ou simplesmente GNNs) têm demonstrado maior capacidade do que métodos tradicionais (e.g., algoritmos de aprendizado de máquina) em capturar relacionamentos diretos (e.g., usuário-item) e colaborativos (e.g., estrutura topológica entre elementos que se relacionam) (S. Wu, Sun, et al., 2020). Os principais problemas nos quais as *GNNs* são empregadas podem ser organizados nos seguintes tópicos:

- **Cidades inteligentes**: O estudo sobre cidades inteligentes é crescente e evolui constantemente uma vez que novos elementos e soluções são criados. Apesar da grande variedade de definições, o conceito de cidades inteligentes está comumente relacionado ao uso de tecnologia da informação para a resolução de problemas e desafios enfrentados por governos, empresas, ou indivíduos (Yin et al., 2015). Como resultado, as soluções desenvolvidas têm como objetivo trazer benefícios como eficiência e sustentabilidade. No presente capítulo, os problemas que envolvem cidades inteligentes são categorizados da seguinte forma: sistemas de recomendação, previsão de tráfego, classificação de função urbana, previsão de disseminação de doenças, veículos autônomos, detecção de anomalia, reidentificação de veículos e reconhecimento de atividades.

- **Biologia e química**: No contexto de biologia e química, os principais tópicos são: predição de reação química, predição de interface de proteína e engenharia biomédica (Zhou et al., 2020). Em (Do et al., 2019), moléculas interagem com outras a partir da presença ou não de moléculas reagentes. Assim, são geradas "moléculas produto" a partir da adição ou quebra de conexões. Cada vértice representa um átomo e cada aresta representa o tipo de ligação entre átomos. Por fim, são gerados grafos intermediários para a predição de pares de vértices. Rhee et al. (2017) utilizam uma rede *GNN* com aprendizagem por reforço para classificar subtipos de câncer a partir da interação de proteínas.

- **Física**: Neste contexto, a rede neural deve apreender as leis que governam o comportamento e a interação de partículas ou objetos para simular o seu próximo estado (Zhou et al., 2020). Problemas de rastreamento e reconstrução de partículas podem ser divididos em: rastreamento de partículas carregadas, reconstrução de vértice secundário, mitigação de acumulação, reconstrução de calorímetro e reconstrução de fluxo de partícula (Duarte & Vlimant, 2022). Em (T. Kipf et al., 2018), cada grafo representa uma trajetória de objetos (vértices) que são caracterizados pela sua posição no espaço e velocidade, além de interagirem entre si

(arestas). O modelo prevê, simultaneamente, a trajetória futura dos objetos e os tipos das arestas que os conectam.

- **Processamento de linguagem natural**: Redes neurais de grafos são utilizadas para os seguintes tópicos de processamento de linguagem natural: classificação de texto, rotulagem de sequência, tradução de máquina, extração de relação, verificação de fatos, dentre outros (Zhou et al., 2020). Neste contexto, os vértices frequentemente representam palavras e documentos de texto e as arestas associam palavras dispostas em uma mesma frase ou documentos relacionados (Malekzadeh et al., 2021). Yao et al. (2019) apresentaram um modelo que classifica os tipos de documentos em termos dos seus assuntos, onde cada grafo representa um documento e as suas palavras e arestas estabelecem relações entre palavra-palavra e documento-documento.

- **Visão computacional**: Em visão computacional, os principais tópicos são: classificação de imagem, detecção de objeto, detecção de interação, classificação de região e segmentação semântica (Zhou et al., 2020). Um problema particular de grande relevância é o *MAIC* (*Multi Label Aerial Image Classification*). Por exemplo, dada uma imagem de satélite, é possível identificar diferentes elementos como árvore, água, areia, grama, carro, barco, dentre outros. Lin et al. (2021) combinam redes *CNN* (*Convolutional Neural Network*) e *GNN* para gerar representações de imagens aéreas ao mesmo tempo que gera novas representações semânticas de cada rótulo (objeto do mundo real) com base nas suas inter-relações presentes em um grafo de conhecimento.

- **Outras áreas**: Redes neurais de grafos também podem ser empregadas em outras áreas, como: (1) redes de citação de artigos (T. N. Kipf & Welling, 2016a); (2) mineração de grafo, o que inclui correspondência (Li et al., 2019) e agrupamento (Tsitsulin et al., 2020) de grafo; (3) economia, onde preços de ações são previstos (Cheng et al., 2022).

Em particular, as Redes Neurais de Grafos (*GNNs*) têm demonstrado grande potencial na modelagem de problemas relacionados ao contexto de cidades inteligentes, um domínio específico de sistemas distribuídos e sistemas de computadores. A computação de borda (*edge computing*) tem se desenvolvido com o avanço das redes neurais de grafos, onde dispositivos de borda se tornaram capazes, por exemplo, de lidar com ruídos inerentes aos dados relacionados ao contexto de reconhecimento de atividade humana (Sanchez et al., 2021). No entanto, para o desenvolvimento de soluções que envolvam *GNNs* é necessário compreender quais são as características específicas desse tipo de rede neural. Dessa forma, este capítulo tem como objetivo apresentar os seguintes aspectos:

- Fundamentos teóricos: tipos de entradas, tarefas, treinamentos e camadas *GNN* (e.g. *message passing, pooling, skip connections* e módulos de amostragem).

- Aplicações de *GNN* no contexto das cidades inteligentes (e.g., sistemas de recomendação, previsão de tráfego, detecção de anomalia, dentre outras).

- Abordagens utilizadas para modelar dados brutos de diferentes tipos para a estrutura de grafo e posterior construção de rede neural de grafo, dentro do contexto de cidades inteligentes.
- Implementação prática de um modelo de rede neural de grafo.

Este capítulo segue a seguinte organização: na Seção 4.2 são apresentados os tipos possíveis de entradas para redes neurais de grafos; na Seção 4.3 são apresentados os tipos de tarefas e treinamentos existentes; a Seção 4.4 contém os fundamentos teóricos das principais camadas *GNN*; a Seção 4.5 descreve os principais tipos de problemas em que as redes neurais de grafos são empregadas no contexto das cidades inteligentes; na Seção 4.6 são sumarizadas as principais abordagens de conversão de dados brutos para a estrutura de grafo e posterior modelagem de rede neural; na Seção 4.7 são descritos os desafios e questões abertas de maior relevância; na Seção 4.8, um problema e a sua respectiva implementação são apresentados; Por último, as considerações finais são apresentadas na Seção 4.9.

## 4.2. Grafo como entrada para *GNN*

Nesta seção, são apresentadas as notações e os conceitos utilizados neste capítulo. A Tabela 4.1 sumariza as principais notações utilizadas ao longo deste capítulo.

Seja um grafo direcionado $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ composto por um conjunto $\mathcal{V}$ de vértices e $\mathcal{E}$ de arestas, onde $v_i \in \mathcal{V}$ denota cada vértice e $(v_i, v_j) \in \mathcal{E}$ denota uma aresta de $v_i$ para $v_j$. Além disso, assuma que $N = |\mathcal{V}|$ representa o número de vértices e $M = |\mathcal{E}|$ o número de arestas de $\mathcal{G}$, e que $\mathcal{N}(v_i) = \{v_j \in \mathcal{V} | (v_i, v_j) \in \mathcal{E}\}$ representa a vizinhança de $v_i$. O grafo não direcionado é um caso particular onde para cada aresta $(v_i, v_j)$ existe uma aresta $(v_j, v_i)$ no sentido contrário (Z. Wu et al., 2020). Ambos os tipos de grafos têm suporte nas *GNNs* dependendo das camadas utilizadas. A partir de $\mathcal{G}$ são geradas três matrizes de entrada para modelos *GNN*:

- **Matriz de adjacência** $A \in \mathbb{R}^{N \times N}$, com $A_{ij} \neq 0$ se $(v_i, v_j) \in \mathcal{E}$ e $A_{ij} = 0$, caso contrário.
- **Matriz de atributos de vértices** $X \in \mathbb{R}^{N \times D}$ onde cada vetor $x_i \in \mathbb{R}^D$ contém os valores dos $D$ atributos do vértice $v_i$.
- **Matriz de atributos de arestas** $E \in \mathbb{R}^{M \times C}$ onde cada vetor $e_{ij} \in \mathbb{R}^C$ contém os valores dos $C$ atributos da aresta $(v_i, v_j) \in \mathcal{E}$. Essa matriz é utilizada em contextos onde os vértices possuem diferentes tipos de relacionamentos (e.g., conhecidos, amigos ou parceiros) (Bianchi et al., 2020).

É importante notar que a maioria das soluções utilizam a própria matriz de adjacência de um grafo ponderado para caracterizar as arestas. Isto é, cada aresta contém um valor que representa o seu peso. Dessa forma, ainda são poucas as abordagens que tiram proveito da matriz de atributos das arestas $E$.

A utilização das matrizes de adjacência, de atributos de vértices e de atributos de arestas, varia de acordo com cada tipo de camada de *GNN*. Por exemplo, as camadas

de *message passing* necessariamente utilizam como entrada a matriz de adjacência. Além da matriz de adjacência, a maioria dessas camadas utilizam em conjunto apenas a matriz de atributos de vértices, sendo menor o número de métodos que aproveitam da matriz de atributos de arestas. Já as camadas de *pooling*, geralmente têm como entrada as matrizes de atributos, podendo ou não utilizar a matriz de adjacência. Ambos os tipos de camadas serão apresentados e discutidos na Seção 4.4.

**Tabela 4.1. Notações utilizadas**

| Notação | Descrição |
|---|---|
| $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ | Um grafo um conjunto $\mathcal{V}$ de vértices e $\mathcal{E}$ de arestas. |
| $N$ | O número de vértices, $N = \lvert\mathcal{V}\rvert$ |
| $M$ | O número de arestas, $M = \lvert\mathcal{E}\rvert$. |
| $D$ | A dimensão do vetor de atributos dos vértices. |
| $C$ | A dimensão do vetor de atributos das arestas. |
| $A \in \mathbb{R}^{N \times N}$ | matriz de adjacência. |
| $X \in \mathbb{R}^{N \times D}$ | Matriz de atributos dos vértices. |
| $E \in \mathbb{R}^{M \times C}$ | Matriz de atributos das arestas. |
| $x \in \mathbb{R}^{N}$ | Sinal de grafo. Corresponde a uma coluna na matriz de atributos. |
| $x_i \in \mathbb{R}^{D}$ | Vetor de atributos do vértice $v_i$. |
| $e_{ij} \in \mathbb{R}^{C}$ | Vetor de atributos da aresta $(v_i, v_j)$. |

Além disso, os grafos podem ser categorizados de acordo com diferentes aspectos, dentre eles: (1) **direcionado/não direcionado** onde a existência de uma aresta $(v_i, v_j)$ implica em uma aresta $(v_i, v_j)$ no caso do grafo não direcionado e onde a existência de uma aresta $(v_i, v_j)$ não implica em uma aresta $(v_i, v_j)$ no caso do grafo direcionado; (2) **ponderado/não ponderado** onde a existência de uma aresta $(v_i, v_j)$ implica em $A_{ij} = 1$ no caso do grafo não ponderado e a existência da mesma implica em $A_{ij} \neq 0$ no caso do grafo ponderado. (3) **homogêneo/heterogêneo** onde vértices e arestas são de tipos iguais (homogêneo) ou tipos diferentes (heterogêneo); (4) **estático/dinâmico** onde o grafo dinâmico tem os seus atributos e a sua topologia variando ao longo do tempo.

Para cada um dos aspectos acima, existe um conjunto mais adequado de camadas e abordagens no contexto das redes neurais para grafos. Por exemplo, a camada *GCN* (*Graph Convolutional Network*) é frequentemente utilizada em grafos não direcionados, ponderados, homogêneos e estáticos. No entanto, de acordo com a implementação, alterações nesses aspectos podem ser realizadas. Este capítulo, no entanto, tem maior foco nos tipos de grafos mais comumente encontrados na literatura como homogêneos e estáticos.

## 4.3. Tarefas e tipos de treinamentos em uma *GNN*

Redes neurais para grafos realizam tarefas específicas sobre os elementos que compõem um grafo. Essas tarefas são divididas nas seguintes classes: nível de vértice, nível de aresta e nível de grafo.

- **Nível de vértice**: os vértices podem ser classificados, agrupados e associados a

valores contínuos (ou seja, regressão). No processo de agrupamento, os vértices são divididos em conjuntos disjuntos onde vértices similares pertencem a um mesmo grupo. Em particular, o agrupamento de vértices está frequentemente associado às camadas de *pooling*, como em (Bianchi et al., 2020).

- **Nível de aresta**: nesta categoria, as tarefas possíveis são classificação e predição de arestas (*link prediction*), onde o objetivo da última é prever a existência ou não de arestas entre os vértices. Considerando as representações ocultas de dois vértices, é possível prever a classe/conexão entre eles considerando uma função de similaridade ou uma rede neural (S. Zhang et al., 2019).
- **Nível de grafo**: inclui classificação de grafo, regressão e correspondência. Normalmente, as tarefas de nível de grafo incluem camadas de *pooling* responsáveis por gerar uma representação de alto nível, já que a estrutura original do grafo não precisa ser preservada.

A escolha do tipo de tarefa a ser realizada é importante para a definição da função de perda a ser utilizada. Por exemplo, em uma regressão de grafo a função de perda *mean squared error* pode ser a mais adequada, enquanto que para a classificação de grafo a *categorical cross entropy* pode ser a melhor opção. Em determinadas soluções, é comum que as predições não sejam diretamente calculadas sobre as representações latentes encontradas pela *GNN*. Ao contrário, a predição é calculada a partir de uma combinação (e.g., produto, concatenação, dentre outras operações) das representações latentes da *GNN* com representações encontradas por outros componentes da rede neural, como ocorre em (Capanema et al., 2021b).

Com relação ao aspecto da configuração do treinamento, as redes neurais podem ser treinadas nas seguintes configurações:

- **Supervisionado**, onde rótulos estão disponíveis. As classificações de vértice e grafo são exemplos de tarefas comumente relacionadas.
- **Semi-supervisionado**, onde se treina o modelo considerando uma pequena quantidade de amostras rotuladas e uma grande quantidade de amostras não rotuladas. A solução de (T. N. Kipf & Welling, 2016a) emprega a camada *GCN* no contexto semi-supervisionado.
- **Não supervisionado**, onde os dados utilizados não são rotulados. Esse tipo de treinamento está frequentemente associado com a tarefa de agrupamento de vértices. Além disso, treinamentos não supervisionados têm como variantes os métodos de *auto-encoder* e aprendizagem contrastiva. Em redes *auto-encoder*, tanto as matrizes de adjacência quanto as de atributos podem ser reconstruídas e comparadas com as originais para se computar o erro. Os autores de *VGAE* (*Variational Auto-Encoder*) (T. N. Kipf & Welling, 2016b) treinaram o modelo com a adição de ruído, e aplicam a seguinte equação:

$$\begin{aligned} Z &= \mathrm{GCN}(X, A), \\ \hat{A} &= \phi\left(ZZ^T\right), \end{aligned} \tag{1}$$

onde $Z$ são os *embeddings* atualizados de vértices e $\hat{A}$ é a matriz de adjacência reconstruída. No contexto de aprendizagem contrastiva não supervisionada, You et al. (2020) apresentaram quatro soluções de *data augmentation* para grafos onde são aplicadas as seguintes variações: vértices e arestas são adicionados ou retirados; alguns atributos de vértices são removidos; amostragem de um subgrafo. Após o processo de *encoder*, as representações latentes do grafo criado e do grafo original tendem a ser semelhantes.

## 4.4. Camadas *GNN*

Nesta seção, as camadas *GNN* são categorizadas de acordo com as tarefas que elas podem desempenhar bem como de acordo com as motivações teóricas por trás de cada classe de camadas.

As camadas *GNN* são divididas em *message passing* e *pooling*, e a utilização delas está intimamente relacionada com o tipo de tarefa desempenhada pela rede neural. Por exemplo, as camadas de *pooling* alteram a topologia original do grafo, fornecendo uma representação de alto nível. Assim, esse tipo de camada está mais associada às tarefas a nível de grafo. Por outro lado, as camadas de *message passing* não alteram a estrutura do grafo em termos dos vértices e das arestas, o que permite que esse tipo de camada seja utilizada, durante alguma etapa, em todos os tipos de tarefas.

Por último, esta seção apresenta uma discussão sobre a complexidade das camadas de *GNN*.

### 4.4.1. Camadas de *message passing*

As camadas de *message passing* ou de convolução, são responsáveis por computar novas representações de cada vértice considerando a informação local (mensagem) dos seus vizinhos (Grattarola & Alippi, 2021). Assim, esse tipo de camada generaliza o conceito de convolução presente nas *CNNs* (*Convolutional Neural Networks*) para o contexto de grafos.

Em arquiteturas *GNN*, é comum que existam camadas convolucionais consecutivas/empilhadas. Neste sentido, uma dada camada de nível $k$ representa uma operação de convolução para cada vértice central sobre os seus vizinhos de ordem $k$. Por exemplo, a primeira camada *GCN* (*Graph Convolutional Network*) (T. N. Kipf & Welling, 2016a) aplica a convolução sobre os vizinhos de primeira ordem de cada nó, a segunda camada aplica a convolução sobre os vizinhos de segunda ordem, e assim por diante. Isto significa que, as matrizes de atributos dos vértices são atualizadas a cada passo de convolução. No entanto, algumas abordagens não se constituem de um conjunto de camadas empilhadas, adotando-se mecanismos específicos para agregar as características dos vértices vizinhos em diferentes níveis (e.g., (Defferrard et al., 2016) e (Atwood & Towsley, 2016)).

Considerando os fundamentos teóricos empregados, uma camada de *message passing* pode ser categorizada em abordagem espectral ou espacial. Na primeira, as representações de cada vértice são mapeadas para o domínio espectral através da transformação de *Fourier*. Na segunda, a convolução é realizada considerando a vizinhança

de cada vértice, sem a necessidade de conversão para o domínio espectral. Essas duas abordagens são detalhadas a seguir.

#### 4.4.1.1. Abordagem espectral

Esse tipo de camada utiliza a teoria espectral de grafos. Os filtros de grafos são introduzidos para reduzir ruídos dos sinais de grafos. Em operações espectrais, o sinal de grafo denotado por $x \in \mathbb{R}^N$ é um vetor de atributo de todos os vértices de um grafo (Z. Wu et al., 2020), ou seja, ele representa uma determinada coluna da matriz $X \in \mathbb{R}^{N \times D}$. Esse vetor é convertido para o domínio espectral pela transformação de *Fourier*, seguida da operação de *message passing* e convertendo o resultado de volta pela transformada inversa de *Fourier* $\mathscr{F}^{-1}$ (Zhou et al., 2020). Esse processo, é apresentado pela Equação 2:

$$\begin{aligned} \mathscr{F} &= U^T x, \\ \mathscr{F}^{-1} &= Ux, \end{aligned} \tag{2}$$

onde $U$ é a matriz de auto-vetores do grafo Laplaciano normalizado $L = I_N - D^{\left(-\frac{1}{2}\right)} A D^{\left(-\frac{1}{2}\right)}$, sendo $I \in \mathbb{R}^{N \times N}$ a matriz identidade, $D$ a matriz de graus e $A$ a matriz de adjacência. Uma vez que o grafo Laplaciano normalizado é real, simétrico, positivo semi-definido, ele pode ser fatorado como $L = U \Lambda U^T$, onde $\Lambda$ é a matriz diagonal dos auto-valores (i.e., $\Lambda_{ii} = \lambda_i$). Dessa forma, a operação de convolução é definida como:

$$g \cdot x = \mathscr{F}^{-1}\left(\mathscr{F}\left(g\right) \odot \mathscr{F}\left(x\right)\right) = U\left(U^T g \odot U^T x\right), \tag{3}$$

onde $\odot$ denota o produto elementar e $U^T g$ representa o filtro no domínio espectral. Ao simplificar o filtro como uma matriz diagonal treinável $g_\theta = diag(U^T g)$, temos

$$g_\theta \cdot x = U g_\theta U^T x, \tag{4}$$

onde o filtro de grafo $g_\theta$ remove ruídos do sinal de grafo $x$. Em geral, as soluções espectrais se diferenciam uma das outras na forma como o filtro de grafo $g_\theta$ é definido.

A camada *Spectral CNN* (Bruna et al., 2013) considera que $g_\theta = \text{diag}(w)$, é uma matriz diagonal com parâmetros treináveis onde $w \in \mathbb{R}^N$ (Zhou et al., 2020). Esta abordagem requer a autodecomposição da matriz Laplaciana, o que resulta em três problemas (Z. Wu et al., 2020): (1) qualquer variação em um grafo resulta na mudança de sua base de auto-valores e auto-vetores; (2) os filtros de grafo computados não podem ser aplicados a grafos com estruturas diferentes porque são baseados no domínio onde foram treinados; (3) a autodecomposição tem complexidade computacional de $O(N^3)$. As abordagens *ChebNet* (Defferrard et al., 2016) e *GCN* (T. N. Kipf & Welling, 2016a) reduzem a complexidade para $O(M)$.

A solução *ChebNet* (Defferrard et al., 2016) utiliza os polinômios de Chebyshev da matriz ortogonal de auto-valores para aproximar o filtro do grafo. Esses polinômios são recursivamente definidos por $T_i(x) = 2xT_{i-1}(x) - T_{i-2}(x)$ com $T_0(x) = 1$ e $T_1(x) = x$, onde $i$ indica a ordem do polinômio. Ao invés de calcular a autodecomposição da matriz Laplaciana, o que tem alto custo, o filtro do grafo é definido em função do polinômio de *Chebyshev* até a ordem $K$, como $g_\theta(\Lambda) = \sum_{i=0}^{K} \theta_i T_i(\tilde{\Lambda})$, onde $\theta_i$ é um parâmetro

treinável, $\tilde{\Lambda} = \frac{2}{\lambda_{max}}\Lambda - I_N$ representa os auto-valores reescalados, $\lambda_{max}$ é o maior auto-valor e cada auto-valor de $\Lambda$ está compreendido no intervalo $[-1, 1]$. A normalização de $\Lambda$ é necessária por causa da base ortonormal dos polinômios de *Chebyshev* (Z. Zhang et al., 2020). Substituindo a nova aproximação de $g_\theta$ na Equação 4, temos

$$g_\theta \cdot x = U\left(\sum_{i=0}^{K} \theta_i T_i\left(\tilde{\Lambda}\right)\right) U^T x. \tag{5}$$

Uma vez que $T_i(\tilde{L}) = UT_i(\tilde{\Lambda})U^T$, onde $\tilde{L} = \frac{2}{\lambda_{max}}L - I_N$, a Equação 5 pode ser reescrita como:

$$g_\theta \cdot x = \sum_{i=0}^{K} \theta_i T_i\left(\tilde{L}\right) x. \tag{6}$$

Assim, a solução *ChebNet* atualiza os atributos de cada vértice com base nos seus vizinhos de ordem $K$.

A camada *GCN* (*Graph Convolutional Network*) (T. N. Kipf & Welling, 2016a) simplifica a Equação 6 assumindo que $K = 1$ para evitar sobreajuste, e considera que o maior auto-valor é $\lambda_{max} = 2$, fazendo

$$g_\theta \cdot x = \theta_0 x + \theta_1 x\left(L - I_N\right) x = \theta_0 x - \theta_1 D^{\left(-\frac{1}{2}\right)} A D^{\left(-\frac{1}{2}\right)} x, \tag{7}$$

com dois parâmetros livres $\theta_0$ e $\theta_1$. Ao se considerar $\theta_0 = -\theta_1$, temos

$$g_\theta \cdot x = \theta\left(I_N + D^{\left(-\frac{1}{2}\right)} A D^{\left(-\frac{1}{2}\right)}\right) x. \tag{8}$$

Em seguida, é aplicado um *truque de renormalização* para evitar a perda de gradiente, com $I_N + D^{\left(-\frac{1}{2}\right)} A D^{\left(-\frac{1}{2}\right)} \rightarrow \tilde{D}^{\left(-\frac{1}{2}\right)} \tilde{A}\, \tilde{D}^{\left(-\frac{1}{2}\right)}$. Ao final do processo de simplificação, cada camada *GCN* é definida como:

$$X^{(k+1)} = \sigma\left(\tilde{D}^{\left(-\frac{1}{2}\right)} \tilde{A}\, \tilde{D}^{\left(-\frac{1}{2}\right)} X^{(k)} W^{(k)}\right), \tag{9}$$

onde $\sigma$ é uma função de ativação, auto-conexões são adicionadas via $\tilde{A} = A + I_N$ e os valores dos atributos são normalizados a nível de vértice através da matriz de graus $\tilde{D}$, onde $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$.

#### 4.4.1.2. Abordagem espacial

As camadas espaciais de *message passing* tendem a ser mais flexíveis e eficientes se comparadas com as espectrais, uma vez que não necessitam de computar auto-vetores ou processar todo o grafo em um mesmo passo. Além disso, dependendo da camada espacial, é possível processar grafos direcionados/não direcionados, diferentes entradas como as matrizes de atributos de arestas, grafos heterogêneos, dentre outros aspectos.

As abordagens espaciais aplicam a operação de convolução diretamente considerando a topologia do grafo. Neste sentido, as operações baseadas no espaço se

assemelham às camadas de convolução das redes CNN. Por exemplo, uma imagem pode ser considerada um caso especial de um grafo onde cada pixel corresponde a um nó, e os seus vértices vizinhos são os *pixels* adjacentes. Na operação de convolução, um filtro $3 \times 3$ é aplicado sobre o *pixel*/vértice e os seus vizinhos obtendo a média dos valores calculados. Da mesma forma, as camadas baseadas no espaço realizam a convolução, isto é, atualizam a representação de cada vértice, ao agregar as representações dos vértices vizinhos de cada nó central (Z. Wu et al., 2020). Como maior desafio, as abordagens espaciais buscam realizar a operação de convolução considerando diferentes tamanhos de vizinhança e mantendo a invariância local (Zhou et al., 2020).

A *DCNN* (*Diffusion Convolutional Neural Network*) (Atwood & Towsley, 2016; Z. Wu et al., 2020) assume que as características de um vértice são agregadas com base em uma probabilidade de transição que equilibra a distribuição de informação a cada passo de convolução. O método *DCNN* é definido como:

$$X^{(k+1)} = \sigma\left(W^{(k+1)} \odot P^{(k+1)} X^{(0)}\right), \tag{10}$$

onde $\sigma$ é uma função de ativação, a matriz $P \in \mathbb{R}^{N \times N}$ é calculada como $P = D^{-1}A$. O resultado final é uma concatenação de $X^{(1)}, X^{(2)}, \ldots, X^{(K+1)}$. Essa camada tem suporte para grafos ponderados e direcionados.

Camadas que utilizam mecanismos de atenção são frequentemente classificadas como um caso particular de métodos de convolução espacial (Zhou et al., 2020). A camada *GAT* (*Graph Attention Network*) (Velickovic et al., 2017; Z. Wu et al., 2020) possui as vantagens de ser paralelizável na computação dos *attention heads*, e pode ser aplicada em contextos onde os vértices que têm diferentes graus ao especificar pesos para os seus vizinhos. A camada *GAT* utiliza o mecanismo de atenção para aprender o peso entre vértices conectados e a saída da $k$-ésima camada é definida como:

$$x_i^{(k+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(k+1)} W x_j^{(k)}\right), \tag{11}$$

onde $x_i^{(0)} = x_i$. O peso do coeficiente de atenção normalizado $\alpha_{ij}^{(s+1)}$ representa o quão conectados estão os vértices $v_i$ e $v_j$ no mecanismo de nível $s+1$, e é definido como:

$$\alpha_{ij}^{(s+1)} = \text{softmax}\left(g\left(a^T\left[Wx_i^{(k)} || Wx_j^{(k)}\right]\right)\right), \tag{12}$$

onde $g(.)$ é a função de ativação *LeakyRelu*, e $a$ é um vetor treinável. Em particular, a Equação 11 tem duas variações ao se utilizar o *multi-head attention* (veja as Equações 13 e 14). Na Equação 13, cada uma das $S$ *heads* são calculadas e concatenadas (Zhou et al., 2020):

$$x_i^{k+1} = \|_{s=0}^{S} \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(s+1)} W^{(s+1)} x_j^{(k)}\right), \tag{13}$$

Por outro lado, a Equação 14 é aplicada quando o mecanismo de atenção é executado na última camada da rede.

$$x_i^{k+1} = \sigma\left(\frac{1}{S}\sum_{s=0}^{S}\sum_{j\in\mathcal{N}(i)}\alpha_{ij}^{(s+1)}W^{(s+1)}x_j^{(k)}\right), \tag{14}$$

onde $\|$ é o operador de concatenação e $\alpha_{ij}^{(s+1)}$ é o coeficiente de atenção normalizado do *attention head* de nível $s+1$.

A maioria das camadas de *message passing* atualizam a matriz de atributos dos vértices $X$. No entanto, a camada *XENet* (Maguire et al., 2021) atualiza tanto as matrizes de atributos dos vértices $X$ quanto as matrizes de atributos das arestas $E$.

$$S_{ij} = \phi^{(s)}\left((x_i^k \| x_j^k \| e_{ij}^k \| e_{ji}^k)\right) \tag{15}$$

$$S_i^{(\text{out})} = \sum_{j\in\mathcal{N}(i)} a^{(out)}\left(S_{ij}\right) S_{ij} \tag{16}$$

$$S_i^{(\text{in})} = \sum_{j\in\mathcal{N}(i)} a^{(in)}\left(S_{ij}\right) S_{ij} \tag{17}$$

$$x_i^{k+1} = \phi^{(n)}\left((X_i^k \| S_i^{(\text{out})} \| S_i^{(\text{in})})\right) \tag{18}$$

$$e_{ij}^{k+1} = \phi^{(e)}\left(S_{ij}\right), \tag{19}$$

onde $\phi^{(s)}$, $\phi^{(n)}$ e $\phi^{(e)}$ são *multi-layer perceptrons* com a função de ativação *PRELU*, $a^{(out)}$ e $a^{(in)}$ são camadas *Dense* que utilizam a função de ativação *softmax* e têm apenas um escalar como saída. Na Equação 15, para cada par de vértices são concatenados os respectivos atributos de vértices e de arestas. Nas Equações 16 e 17 as mensagens em direção de saída $S_i^{(out)}$ e de entrada $S_i^{(in)}$ são calculadas para cada vértice $v_i$. O vetor de atributos de cada vértice $x_i^{k+1}$ do nível $k+1$ é atualizado com base na sua representação de nível anterior $x_i^k$ concatenado com as mensagens de direção de entrada e saída, que carregam dados de vértices e arestas (veja a Equação 18). Os atributos de cada aresta são atualizados de acordo com a Equação 19, onde a nova representação $e_{ij}^{k+1}$ de cada aresta depende das mensagens trocadas entre seus respectivos vértices. De acordo com a implementação desta camada na biblioteca *Spektral* (Bianchi et al., 2020), as mensagens de saída e entrada podem ser calculadas utilizando o mecanismo de *self-attention*. Além disso, na versão para processamento em *batch*, a matriz de adjacência $A$ é utilizada, através de multiplicação, como mascaramento para cada $S_{ij}$.

#### 4.4.1.3. *Skip connections*

As *skip connections* são operações comumente adicionadas às camadas de *message passing* com o objetivo de permitir que o modelo tenha maior profundidade em termos de camadas *GNN*. Para isso, as *skip connections* tentam preservar a representação histórica do grafo e, assim, reduzir as chances de sobre-suavização (Zhou et al., 2020).

A camada *ARMA* (Bianchi et al., 2021, 2020) (veja a Figura 4.1) é composta de atualizações recursivas da camada *Graph Convolutional Skip (GCS)*, que é definida como:

$$X^{(t+1)} = \sigma\left(\tilde{L}X^{(t)}W + X^{(0)}V\right), \tag{20}$$

onde $X^{(t+1)}$ são atualizações dos atributos dos vértices na iteração $t+1$, $\sigma$ é uma função de ativação, $X^{(t)}$ é a saída da última camada *GCS*, $X^{(0)}$ é matriz inicial de atributos dos vértices e representa a *skip connection* desta camada, e $W \in \mathbb{R}^{out \times out}$ e $V \in \mathbb{R}^{out \times out}$ são parâmetros treináveis. *out* é o hiperparâmetro do tamanho da saída. $\tilde{L}$ é a matriz Laplaciana modificada $\tilde{L} = D^{(-\frac{1}{2})}AD^{(-\frac{1}{2})}$, onde $D$ é a matriz de graus. Por fim, a saída da camada *ARMA*, $\tilde{X}$, é a média das saídas empilhadas das camadas *GCS*, o que é computado como:

$$\tilde{X} = \sigma\left(\frac{1}{K}\sum_{k=1}^{K} X_k^{(T)}\right), \tag{21}$$

onde $K$ é o número de camadas *GCSs*, $X_k^{(T)}$ é a última saída da k-ésima camada *GCS* e $\sigma$ é uma função de ativação que pode ser *softmax* ou *sigmoid*, por exemplo, caso essa seja a camada que gera a predição final.

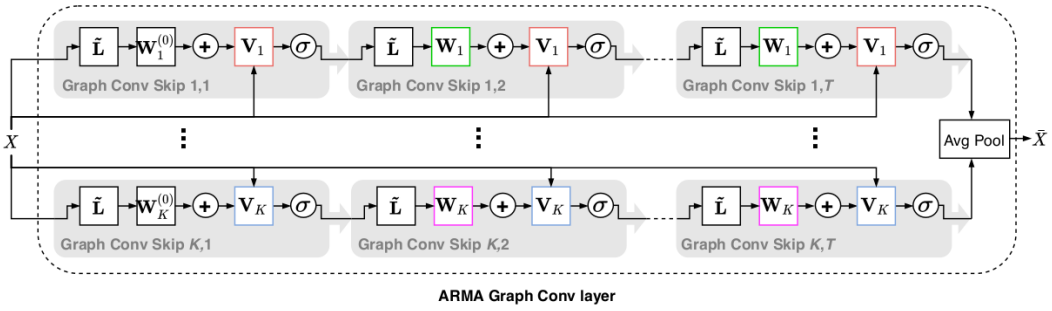



**Figura 4.1. Diagrama da camada *ARMA* de (Bianchi et al., 2021)**

#### 4.4.1.4. Módulos de amostragem

Os módulos de amostragem (do inglês *sampling modules*) são principalmente empregados em camadas de *message passing* e são úteis para grafos grandes e para reduzir o problema de explosão de vizinhança. Existem três tipos de módulos de amostragem: por vértice, camada e por subgrafo (veja a Figura 4.2).

Na amostragem por vértice, um subconjunto de vértices vizinhos é selecionado em cada passo de propagação de mensagem. Hamilton et al. (2017) selecionam um subconjunto de tamanho fixo de vizinhos de cada vértice. R. Ying et al. (2018) aplicam caminhos aleatórios partindo de cada vértice e selecionam aqueles com o maior número de visitas normalizadas.

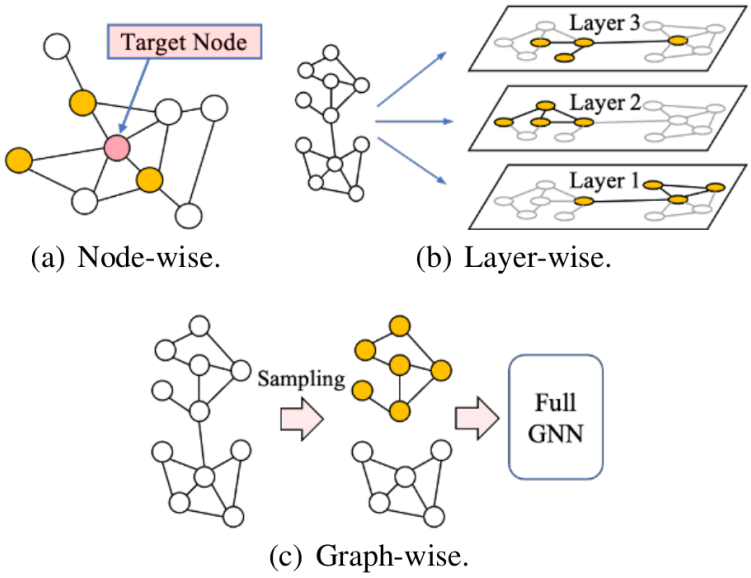



**Figura 4.2. Módulos de amostragem de (L. Wu et al., 2022). a) indica a amostragem por vértice b) indica a amostragem por camada c) indica a amostragem por subgrafo**

Em métodos de amostragem por camada, ao invés de se selecionar um subconjunto de vizinhos para cada vértice, apenas um subconjunto de vértices de todo o grafo será utilizado para a agregação de informação no processo de propagação de mensagem (Zhou et al., 2020). Esse tipo de método sofre com conexões esparsas entre vértices (Zou et al., 2019). J. Chen et al. (2018) selecionam os vértices com maior grau para construir uma matriz de adjacência menor que será utilizada na propagação de mensagem e representará menor custo computacional. No entanto, de uma camada para outra, o conjunto de vértices selecionados pode não estar suficientemente conectado (e.g., um dos vértices selecionados pode não estar conectado com os outros vértices de maior grau), podendo gerar uma matriz esparsa e até mesmo com valores zero em todas as posições (Zou et al., 2019). Zou et al. (2019) aliviam o problema de conexões esparsas ao selecionar amostras a partir dos vizinhos dos vértices selecionados na camada anterior.

Os métodos de amostragem por subgrafo aplicam a propagação de mensagem apenas dentro dos múltiplos subgrafos selecionados. Chiang et al. (2019) obtêm esses subgrafos são obtidos a partir de algoritmos de agrupamento, onde cada subgrafo representa um *cluster*.

### 4.4.2. Camadas de *pooling*

Analogamente ao contexto de visão computacional, as camadas de *pooling* são operações de redução de amostragem utilizadas para gerar uma representação simplificada de um dado elemento. Na literatura, elas também são comumente referidas como operações *readout*.

Nas arquiteturas de *GNNs*, as camadas de *pooling* estão, em geral, associadas às tarefas a nível de grafo, de modo que vértices são agregados para gerar uma repre-

sentação simplificada do grafo. Como caso particular, esse tipo de camada também é utilizado em operações de agrupamento de vértices. Essas camadas estão posicionadas após as operações de convolução. Em termos de implementação, esta classe de operações gera como saídas novas matrizes de adjacência e de atributos, onde ambas representam uma quantidade reduzida de vértices. Esta operação de redução de amostragem/vértices tem os seguintes benefícios (Z. Wu et al., 2020):

- Redução do custo computacional das operações, das chances de sobreajuste e de invariância de permutação.
- Melhoria no desempenho de tarefas a nível de grafo.

Como desafio, as camadas *pooling* devem manter a *invariância de ordem* dos vértices (Z. Zhang et al., 2020), i.e., ao se trocar os índices dos vértices e arestas usando uma função bijetora entre um par de vértices, a representação simplificada do grafo após a operação de *pooling* deve ser a mesma.

Existem dois tipos de camadas *pooling* para grafos: *pooling* direto e *pooling* hierárquico (Zhou et al., 2020).

#### 4.4.2.1. *Pooling* direto

Esta categoria de camadas gera representações simplificadas de grafos ao diretamente aplicar estratégias de seleção de vértices (Zhou et al., 2020).

No *pooling* global, todo o grafo é reduzido a um vértice representado por um vetor de atributos. Neste caso, operações de soma, média, máximo e *gated attention* (*GAP*), dentre outras, podem ser utilizadas (Grattarola & Alippi, 2021). As representações calculadas para cada vértice são resumidas/simplificadas pelos operadores de *pooling*.

#### 4.4.2.2. *Pooling* hierárquico

Esta classe de operações explora a hierarquia ou topologia de um grafo para gerar uma nova representação. A camada *DiffPool* (Z. Ying et al., 2018) realiza um agrupamento hierárquico, e é definida por:

$$\begin{aligned} S^k &= \text{softmax}\left(GNN_{k,pool}\left(A^k, X^k\right)\right), \\ A^{k+1} &= \left(S^k\right)^T A^k S^k, \\ X^{k+1} &= \left(S^k\right)^T \text{ConvGNN}_{k,embed}\left(A^{(k)} X^k\right), \end{aligned} \tag{22}$$

onde $X^k$ é a matriz de atributos dos vértices, $A^k$ é a matriz de adjacência do nível $k$, $S^k \in \mathbb{R}^n$ contém as probabilidades para que cada vértice no nível $k$, e $A^{k+1}$ e $X^{k+1}$ são as matrizes de adjacência e de atributos de vértices simplificadas, ou seja, com tamanho reduzido no nível $k+1$.

### 4.4.3. Complexidade

A complexidade de tempo de cada camada pode ser dominada pelo custo do pré-processamento, se existir, ou da operação de *message passing* em si. Camadas espectrais que calculam a decomposição de auto-valores sem que haja nenhuma simplificação no pré-processamento, como *Spectral CNN*, requerem $O(N^3)$. O modelo de (Tran et al., 2018) tem custo de $O(N^3)$ devido ao cálculo de menor caminho entre pares de diversos vértices. Quanto a operação em si de *message passing*, a sua complexidade na maioria das camadas é de $O(N^2)$ quando a matriz de adjacência é densa e $O(M)$ quando a matriz é esparsa (Z. Wu et al., 2020).

## 4.5. *GNN* para cidades inteligentes

O termo cidades inteligentes é definido pelo uso inteligente da tecnologia para melhorar a qualidade de vida das pessoas, o que é frequentemente associado com a melhoria da saúde de indivíduos, da eficiência de produtos e serviços, da sustentabilidade, do planejamento urbano, da mobilidade urbana, dentre outros aspectos.

As redes neurais para grafos se inserem nesse contexto a partir da modelagem em grafo de elementos presentes no dia-a-dia de ambientes urbanos como ruas/estradas, estabelecimentos, fluxo de veículos em rodovias, movimentação de massas no ambiente urbano, dentre outros. Nesta seção, serão apresentadas as principais áreas dentro de cidades inteligentes onde pesquisadores têm obtido relevantes avanços a partir do uso das *GNNs*. Dentre estes tópicos estão: sistemas de recomendação, previsão de tráfego, classificação de função urbana, previsão de disseminação de doenças, agentes autônomos, reidentificação de indivíduos e reconhecimento de atividades. Para cada um dos tópicos, são apresentadas aplicações reais e como os dados originais são mapeados para o contexto de grafos.

### 4.5.1. Sistemas de recomendação

Sistemas de recomendação usualmente inferem a preferência de indivíduos por itens. Neste sentido, as interações Usuário-Item podem ser modeladas a partir de um grafo bipartido heterogêneo (i.e., Usuário-item) ou a partir de grafos homogêneos (i.e., Usuário-Usuário e Item-Item) para se encontrar representações latentes que sejam úteis para estimar a preferência de um usuário por um item (S. Wu, Sun, et al., 2020), o que frequentemente se traduz em tarefas a nível de aresta de predição de *link*.

Com relação ao processo de recomendação, existem dois tipos principais: recomendação geral e recomendação sequencial. Na recomendação geral, assume-se que os usuários têm preferências estáticas, isto é, que não mudam ao longo do tempo. A maior parte dos modelos constroem as representações de usuários e itens a partir dos dados históricos das interações entre ambos. Essa relação pode ser descrita por:

$$y_{u,i} = f\left(h_u^*, h_i^*\right), \tag{23}$$

onde $y_{u,i}$ é a preferência do usuário $u$ pelo item $i$, $f(.)$ é uma função que utiliza as representações $h_u^*$ de usuário e $h_i^*$ de item para gerar a saída $y_{u,i}$.

Por outro lado, as recomendações sequenciais assumem que as preferências de cada usuário são dinâmicas e evoluem ao longo do tempo. Dada uma sequência de $n$

interações Usuário-Item, o objetivo é prever qual é o item mais provável na interação $n+1$, como descrito por:

$$i^*_{s,n+1} = \text{argmax}_{i \in I} \text{P}\left(i_{s,n+1} = i | S^u\right), \tag{24}$$

onde $S^u = [i_{s,1}, i_{s,2}, \ldots, i_{s,n}]$ é a sequência de itens $i_{s,t}$ que o usuário $u$ interagiu em cada tempo $t \leq n$.

Com relação às recomendações sequenciais, S. Wu, Zhang, et al. (2020) e Capanema et al. (2021b) combinam redes *RNN* e *GNN* para estimar preferências de usuários por *PoIs* (*Points of Interest*) e categorias de *PoIs*, respectivamente. S. Wu, Zhang, et al. (2020) utilizam um bloco com duas camadas consecutivas *GCN* para modelar as características geográficas de cada *PoI*, onde cada vértice indica a influência geográfica entre cada par de *PoIs* (modelagem Item-Item), de modo que quanto maior a proximidade maior a influência. Na matriz de atributos dos vértices, são utilizadas as representações latentes encontradas de cada *PoI* através da camada de *Embedding* da componente recorrente. Na camada recorrente, são processadas sequências de locais visitados por um usuário. As componentes de *RNN* e *GNN* são combinadas a partir do produto entre as representações latentes de cada componente, o que se traduz na preferência de um usuário por cada *PoI*.

O modelo apresentado em (Capanema et al., 2021b) prevê a categoria do próximo local a ser visitado, o que é importante para que provedores de serviços móveis realizem ações de *marketing* mais assertivas. Neste trabalho, cada vértice do grafo corresponde a uma categoria de *PoI* (modelagem Item-Item) e diferentes matrizes de atributos de vértices (e.g., matrizes de distância e duração de tempo entre visitas aos estabelecimentos de cada categoria) são utilizadas em blocos separados de convolução. É importante ressaltar que, um dos *datasets* utilizados contém *check-ins* de usuários de uma rede social, de modo que cada estabelecimento visitado é considerado como um *PoI*. Por outro lado, a segunda base de dados utilizada contém trajetórias de *GPS* de usuários móveis e, dessa forma, foi necessário utilizar o algoritmo de identificação de *PoIs* descrito em (Capanema et al., 2021a, 2019; Capanema & Silva, 2021). A componente de camada *RNN* indica o comportamento recente do usuário, enquanto a componente *GNN* indica o comportamento global/geral do usuário em termos da sua mobilidade entre *PoIs* e as suas respectivas categorias. A predição da categoria do próximo local se dá pela combinação dos resultados de cada componente, o que caracteriza a solução como uma abordagem híbrida.

No contexto de abordagem não sequencial, Xiao et al. (2020) utilizam os grafos Usuário-Usuário e Usuário-Item para prever relacionamentos de amizade entre indivíduos (*link prediction*) e a probabilidade de aquisição de produtos/itens no contexto de redes sociais. Para isso, os autores consideram que usuários que são amigos em uma rede social têm propensão a consumir os mesmos produtos, ao mesmo tempo que indivíduos que têm hábitos de consumo similares tendem a se tornarem amigos. Neste sentido, vértices são associados aos usuários e produtos, e representações latentes são calculadas para cada um, tendo um papel similar à matriz de atributo de vértices.

Liang et al. (2021) apresentam uma rede neural para a recomendação de *apps* do *Google Play* considerando o contexto de grafos heterogêneos onde cada vértice

pode representar um usuário, um aplicativo, ou uma categoria de aplicativo. Além disso, as arestas são divididas em dois tipos: avaliação de usuários a aplicativos e associação entre aplicativo e a sua categoria. Um exemplo de meta caminho é: considerando a preferência de um usuário por um aplicativo, percorre-se o grafo considerando apenas aplicativos da mesma categoria. Neste meta caminho, será alcançado um conjunto específico de usuários que têm preferência pela mesma categoria de aplicativo. O processo de *message passing* é realizado através de meta caminhos onde se agrega os atributos de vértices de arestas de diferentes tipos.

### 4.5.2. Previsão de tráfego

A previsão de tráfego é um problema importante para o desenvolvimento de sistemas de transportes inteligentes. Neste sentido, as redes neurais de grafos têm contribuído para o desenvolvimento de soluções no estado da arte. Essas soluções partem do princípio de que o estado de tráfego (e.g., fluxo de tráfego e velocidade de tráfego em um dado local) influenciará, em certo grau, o estado de tráfego futuro de outros locais. Para isso, as soluções processam, em geral, sequências históricas de grafos que representam a variação do estado de tráfego ao longo do tempo.

Os sistemas de previsão de tráfego são inicialmente divididos de acordo com o tema (Jiang & Luo, 2021): fluxo de tráfego, velocidade, demanda por recursos, dentre outros. Cada um desses problemas podem ser analisados em diferentes níveis: estrada, estação e região. Além disso, cada nível exige uma abstração diferente dos dados. A nível de estrada, cada vértice pode representar uma interseção de ruas e cada aresta pode representar uma rua. Em outro cenário, cada vértice pode representar um sensor em um segmento de estrada ao mesmo tempo em que o peso de cada aresta indica a distância entre cada par de sensores. A nível de estação, cada estação de ônibus ou metrô é representada por um vértice e as linhas/trajetórias realizadas são correlacionadas com as arestas. A nível de região, cada vértice é associado a uma região e cada aresta indica o fluxo de elementos entre duas regiões.

Com relação à geração do grafo a partir de dados de fluxo de tráfego, sejam eles obtidos através de sensores ou dados de GPS de usuários móveis, existem diversos tipos de matrizes de adjacência. Os principais tipos são: (1) matrizes baseadas em conectividade de transporte, onde dois locais $v_i$ e $v_j$ podem estar associados, isto é $A_{ij} = 1$, se uma viagem pode ser realizada de modo conveniente ou dentro de um tempo máximo, utilizando os meios de transporte disponíveis; (2) matrizes baseadas em distância, onde a aresta entre dois vértices $v_i$ e $v_j$ é ponderada pela distância (e.g., distância geográfica) entre os elementos representados pelos vértices; (3) matrizes de similaridade, onde a posição $A_{ij}$ indica o nível de similaridade entre dois locais considerando o histórico de seus estados de tráfego.

#### 4.5.2.1. Fluxo de tráfego

O fluxo de tráfego é definido como o número de veículos que passam por um local em um dado intervalo de tempo. A predição eficaz do fluxo de tráfego é importante para o controle de tráfego ou o controle de semáforos, por exemplo. Nesse último, o tempo

do semáforo pode ser otimizado de modo que os veículos permaneçam parados por um período menor. Em problemas a nível de estrada, são previstos os seguintes tipos de fluxos: estrada, origem-destino (OD) e vazão de tráfego em uma interseção. Em problemas a nível de região, cada área pode ser dividida regularmente (i.e., grids de tamanhos iguais) ou irregularmente (e.g., com base no CEP ou limites de um bairro). Por outro lado, nos problemas a nível de estação se prevê o fluxo em uma estação de metrô ou de ônibus.

Um aspecto desafiador é a coleta de dados de tráfego. A utilização de sensores em rodovias, por exemplo, tem um alto custo de implantação e manutenção. Como alternativa, sensores de GPS em dispositivos móveis podem fornecer dados de movimentação de indivíduos a um baixo custo. Ao mesmo tempo, essa abordagem apresenta desafios com relação à qualidade dos dados, por exemplo, dados faltantes.

Dai et al. (2020) utilizam dados de navegação de veículos para caracterizar o fluxo de tráfego em segmentos de estradas (vértices) e, a partir disso, estimar o tempo de viagem em uma região. Cada grafo representa o tráfego de uma região, e é construído para cada intervalo de tempo de 30 minutos. As arestas da matriz de adjacência conectam estradas do grafo não direcionado, e o peso é obtido por uma operação composta que considera a proximidade entre as estradas (i.e., menor caminho) e a correlação do tempo de viagem entre elas. Esse trabalho utiliza a camada *STGCN* (*Spatio-Temporal Graph Convolutional Network*) (Yu et al., 2017) para realizar o treinamento sobre sequências de grafos que representam os estados de tráfego anteriores. Essa solução se enquadra na tarefa de regressão de grafo.

No contexto de transporte ferroviário, vértices podem estar associados a estações de metrô, como em (Ye et al., 2020). Nesse trabalho, as arestas do grafo direcionado são ponderadas pelo menor tempo de viagem entre cada par de estações (Origem-Destino). Em particular, esses valores de tempo são obtidos a partir do algoritmo de *Dijkstra*. A partir disto, são gerados diferentes tipos de matrizes de adjacência, onde duas estações estão conectadas apenas se a aresta tem um peso dentro de um intervalo específico de valores. A matriz de atributos dos vértices contém os fluxos de saída e chegada de passageiros em cada estação. No modelo proposto, existem duas componentes de redes neurais. Na primeira, camadas *RNN* processam sequências de matrizes de atributos para diferentes intervalos de tempos. Na segunda, camadas *GNN* processam sequências de grafos. A solução se enquadra na tarefa de regressão de vértice, onde se pretende prever o fluxo de passageiros em cada estação.

#### 4.5.2.2. Velocidade de tráfego

A velocidade de tráfego é definida como a velocidade média de veículos que passam por um local. A predição da velocidade de tráfego é útil para estimativas de melhores rotas, tempo de chegada no destino e duração de viagem. Yu et al. (2017) apresentam um modelo para prever a velocidade do tráfego em cada estação de sensor (vértice) ao longo da estrada, configurando-se como uma tarefa de regressão de vértice. Cada aresta, indica a conectividade entre esses sensores e é ponderada pela proximidade geográfica entre eles.

#### 4.5.2.3. Demanda de tráfego

Esta classe de problemas se refere à previsão da demanda de sistemas de transporte considerando o potencial de viagens a serem realizadas. Isto pode se traduzir em demandas por táxis, veículos compartilhados, bicicletas, dentre outros. Em (Ke et al., 2021), cada vértice representa um par de regiões Origem-Destino (OD) e cada aresta indica a conexão entre ODs. A partir do conceito de grafo OD, são construídos quatro grafos onde as matrizes de adjacência têm os pesos das arestas ponderados pelos seguintes aspectos: vizinhança de regiões, similaridade de função urbana de regiões, distância entre regiões, correlação de padrões de mobilidade entre regiões. Os vértices são caracterizados pelas demandas de viagem para cada par de regiões OD considerando diferentes períodos. Assim, é possível estimar a demanda futura por viagens de táxi em cada par de regiões OD.

#### 4.5.2.4. Outros problemas

Outros tópicos estão relacionados com as seguintes predições (Jiang & Luo, 2021): disponibilidade de vagas de estacionamento, atraso de linhas de trem, emissão de poluentes por veículos, acidentes, dentre outras.

### 4.5.3. Classificação de função urbana

Nesta categoria de problemas, as redes neurais de grafos são utilizadas para prever a função ou o tipo de cada elemento urbano. Essa tarefa é importante para o planejamento urbano e governança, uma vez que com o passar do tempo, os elementos urbanos podem ter a sua semântica alterada. A classificação de função urbana pode ser realizada a nível de local específico, rua/estrada, área e região. Como esse tópico não está relacionado com a previsão de fluxos de tráfego, a utilização de grafos direcionados não é frequentemente necessária.

(S. Hu et al., 2021) classificam segmentos de estradas em: (1) tráfego, que são caracterizadas pela alta velocidade, faixas mais largas e pelo baixo volume de pedestres; (2) comercial, onde há predominância de shoppings e comércio em geral; (3) pública, onde a velocidade de tráfego tende a ser menor, as faixas são levemente mais estreitas e as ruas são cercadas por estabelecimentos que oferecem serviços gerais, como estacionamentos e edifícios públicos. Além disso, cada vértice é associado a um segmento de estrada, e as arestas indicam segmentos de estradas adjacentes. Considerando essa modelagem, o trabalho realiza a tarefa de classificação de vértices. São utilizadas três fontes de dados: rede de estradas urbanas, dados de GPS de trajetórias de táxis e dados de *PoIs* que contêm as categorias de estabelecimentos. A partir desses dados, são criadas matrizes de atributos que caracterizam cada rua com base nos *PoIs* ao redor e nos padrões das viagens de táxis que passam por elas.

Yang et al. (2022) classificam regiões em industrial, comercial, residencial e outro. Cada vértice representa um edifício de uma dada região e cada aresta conecta vértices de edifícios vizinhos sendo, portanto, um grafo não direcionado. Como a conexão entre os edifícios é construída utilizando a Triangulação *Delaunay*, as arestas

são ponderadas de acordo com o tamanho da aresta que liga um centroide a outro na triangulação. Cada vértice é caracterizado pelos seguintes atributos geométricos referentes ao diagrama de *Delaunay* obtido: raio, área, perímetro, média de orientação da aresta, compacidade, alongamento e concavidade. Este trabalho se enquadra na categoria de tarefas de classificação de vértice.

#### 4.5.4. Previsão de disseminação de doenças

Como uma tendência crescente, grafos têm sido utilizados para representar a movimentação de massas entre diferentes regiões geográficas para modelar a disseminação de doenças. A previsão de disseminação de doenças pode ser a nível de indivíduo (i.e., de pessoa para pessoa) até a nível de região (i.e., de região para região). Essa última, é um caso particular da predição de fluxos de massas. Além disso, nesse tipo de modelagem, a matriz de atributos dos vértices normalmente carrega o estado de saúde de cada indivíduo ou de grupos de pessoas (e.g., pessoas que vivem em uma dada região).

Tomy et al. (2022) utilizam dados de redes sociais para estimar a disseminação de Covid-19. Cada grafo não direcionado representa uma rede de usuários, onde cada indivíduo é associado a um vértice e o contato entre duas pessoas é representado por uma aresta. A matriz de atributos dos vértices indica estado de saúde de cada indivíduo. Ao final, a solução proposta prevê a condição de saúde de cada pessoa, podendo ser: recuperado, saudável ou contaminado. Dessa forma, essa solução se enquadra na tarefa de classificação de vértice.

La Gatta et al. (2020) apresentam um modelo que processa sequências de grafos direcionados a partir da combinação de redes *GNN* e *RNN*, onde cada $G^{(t)}$ com $t \in [1, T]$ representa um *snapshot* de grafo no instante $t$, onde $T$ é o tamanho da sequência. Cada vértice representa uma região, e é caracterizado por atributos estáticos (e.g., população e densidade demográfica) e dinâmicos (e.g., fração de indivíduos não viajantes, fluxo de entrada de indivíduos e raio de giro), onde os valores variam ao longo do tempo. Além disso, as arestas são ponderadas com base nos fluxos de usuários móveis entre as regiões analisadas. Essas informações são obtidas através de dados governamentais e de operadores de serviços móveis. Por fim, para cada região/vértice são previstos os seus estados futuros, como números de novos casos e de indivíduos recuperados, o que se caracteriza como uma tarefa de regressão de vértice.

#### 4.5.5. Agentes autônomos

A modelagem de redes *CAV* (*Connected Autonomous Vehicle*) em grafos e a subsequente utilização de *GNNs* é importante para o auxílio de tomadas de decisões de veículos autônomos.

S. Chen et al. (2021) modelam *CAVs* para assegurar a comunicação e cooperação entre veículos na tarefa de mudança de faixa, isto é, mover-se para a esquerda, permanecer na faixa ou mover-se para a direita (veja a Figura 4.3). No processo de modelagem da matriz de adjacência do grafo não direcionado, cada veículo é associado a um vértice e seus vizinhos estão conectados por arestas. A matriz de atributos dos vértices contém quatro características: velocidade, posição, localização e intenção de movimento. Posição e intenção são variáveis categóricas, e portanto são representa-

das por *one hot encoding*. Por fim, cada veículo/vértice é classificado de acordo com as ações de mudar para a faixa da esquerda, direita, ou manter a posição.

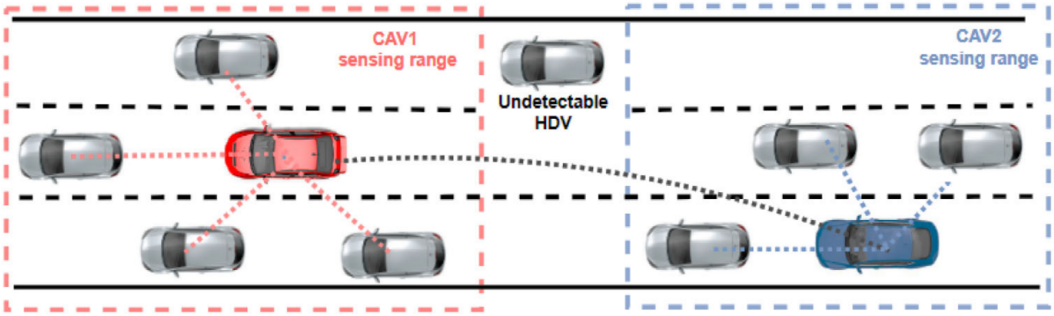



**Figura 4.3. Exemplo de rede *CAV* de (S. Chen et al., 2021)**

### 4.5.6. Detecção de anomalia

A modelagem em grafos de sistemas inteligentes de transporte, energia e de empresas (e.g., indústrias e prestadoras de serviços) tem se mostrado promissora para o desenvolvimento de soluções que empregam *GNNs* (Y. Wu et al., 2021). Neste sentido, a Internet das Coisas (*IoT*) e o advento das redes 5G tem um papel fundamental na coleta dos dados utilizados na modelagem de grafos para a detecção de anomalias.

No contexto de sistemas inteligentes de transporte, anomalias no trânsito estão frequentemente associadas a acidentes, eventos públicos, crimes, dentre outros. (Y. Hu et al., 2020) apresentam uma arquitetura *auto-encoder* de grafo, onde cada vértice corresponde a uma região e as arestas correspondem às viagens realizadas entre cada região. A matriz de adjacência é ponderada pelo quão rápidas são as viagens entre cada par de regiões. Cada vértice é caracterizado pelos valores das latitudes e longitudes mínimas e máximas da região correspondente. O erro de reconstrução do *auto-encoder* é utilizado para indicar o grau de anomalia de cada grafo. Nesse processo é realizada a predição de peso de arestas, o que enquadra a solução no contexto da tarefa de predição de *link*.

Deng et al. (2022) utilizam uma arquitetura *GAN* (*Generative Adversarial Network*) para sequências de grafos direcionados, onde o gerador aprende a criar grafos falsos e os adiciona na última posição de cada sequência, e o discriminador aprende a distingui-las. Neste trabalho, cada vértice representa uma região ou faixa de trânsito, de acordo com o contexto avaliado. Cada aresta conecta vértices de localidades adjacentes. O peso de cada aresta é determinado pela distância entre as regiões, de modo que regiões mais próximas têm maior valor associado. Com relação à matriz de atributos de vértices, cada atributo é caracterizado de acordo com os fluxos de entrada e saída de viagem em cada região ou o volume de tráfico e a velocidade média em cada faixa, dependendo do *dataset* avaliado. Esse estudo se encaixa na tarefa de regressão de vértice, de modo que o cálculo da função de erro envolve a matriz $X$ e a gerada $X'$.

Os sistemas de energia inteligentes estão relacionados pelo advento de fontes renováveis (e.g., eólica e fotovoltaica), bem como com as suas redes de geração distribuída, transmissão e consumo. Anomalias que causem a interrupção ou a sobrecarga

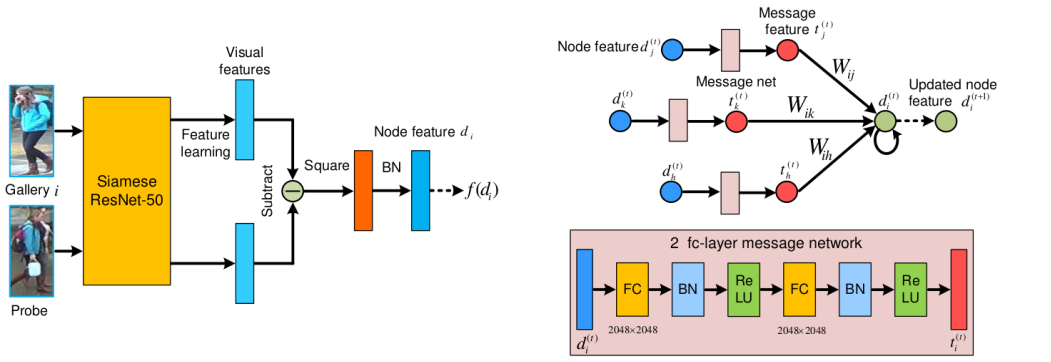
desses sistemas podem afetar sistemas de produção, em especial de empresas altamente tecnológicas como na Indústria 4.0. Neste sentido, a detecção e predição de anomalias é importante para que organizações mantenham a segurança e a previsibilidade. (Owerko et al., 2018) desenvolveram um modelo baseado em redes neurais de grafos para prever a interrupção de transmissão elétrica tendo como base as condições meteorológicas. Em particular, cada vértice corresponde a uma estação meteorológica e cada aresta indica a correlação entre elas, onde estações próximas têm um alto valor associado na matriz de adjacência. Com relação à matriz de atributos dos vértices, cada coluna representa uma medição meteorológica, como pressão, temperatura, velocidade do vento, umidade, nível de precipitação dentre outras.

No contexto de empresas inteligentes, anomalias podem ser detectadas ou previstas na linha de produção, nos produtos em si, ou nos serviços prestados (Y. Wu et al., 2021). Com isso, atividades de reparos podem ser antecipadas ou mais bem planejadas e prejuízos são eventualmente reduzidos. (D. Chen et al., 2021) detectam falhas em processos industriais a partir da seguinte modelagem: os vértices representam sensores e as arestas representam a relação entres os sensores. Cada posição da matriz de adjacência tem um peso treinável. A matriz de atributos dos vértices contém as características coletadas por cada sensor. A matriz de atributos das arestas indica o tipo de cada aresta. Na matriz de adjacência, o peso de cada aresta (i.e., relação entre cada sensor) não é obtido diretamente, como ocorre na maioria dos métodos. Ao contrário, esses pesos são calculados utilizando o mecanismo de atenção sobre os vetores de atributos de cada vértice, indicando a importância que cada sensor tem para o outro em ocorrências de falhas no sistema.

### 4.5.7. Reidentificação de indivíduos

Em sistemas de vigilância por vídeo, a identificação de um mesmo indivíduo em imagens de diferentes câmeras tem sido tratada utilizando-se *GNNs* juntamente com redes *CNNs* para extrair as similaridades entre imagens (Shen et al., 2018). A partir da identificação automática de indivíduos que representem perigo potencial a uma população, eleva-se a sensação de segurança e de bem-estar de uma sociedade. A utilização de recursos tecnológicos para este fim é um importante aspecto para as cidades inteligentes.

Em (Shen et al., 2018), cada vértice representa um par de uma imagem de referência (e.g., imagem do indivíduo alvo) e uma imagem de galeria (i.e., amostra de imagem de câmera), onde cada aresta é ponderada pela similaridade entre cada par de amostras de galeria. Cada vértice possui uma representação latente gerada por uma componente baseada em camadas *CNN* que codificam a similaridade entre a imagem de referência e a imagem da galeria (veja a Figura 4.4). O processo de *message passing* é guiado pela similaridade (i.e., peso das arestas) de imagens de galeria, de modo que imagens similares têm as suas representações agregadas com maior peso, resultando em um melhor desempenho na tarefa de reidentificação de indivíduo, onde cada vértice é classificado como pertencente ou não à imagem de um indivíduo de referência.




(a) Node input feature generating. (b) Deep message passing of SGGNN.

**Figura 4.4. Modelo de reidentificação de indivíduos de (Shen et al., 2018)**

### 4.5.8. Reconhecimento de atividade humana

O reconhecimento de atividade humana, também conhecido como *HAR* (*Human Activity Recognition*), tem aplicações na saúde humana, esportes, investigação criminal dentre outros (Mondal et al., 2020). Caminhar, correr e descansar são exemplos de atividades de pessoas em uma cidade que podem ser detectadas a partir de uma modelagem em grafo capaz de agregar informações de diversos sensores. O tratamento desses dados de forma isolada e/ou combinada (fusão) aparece frequentemente em sistemas distribuídos.

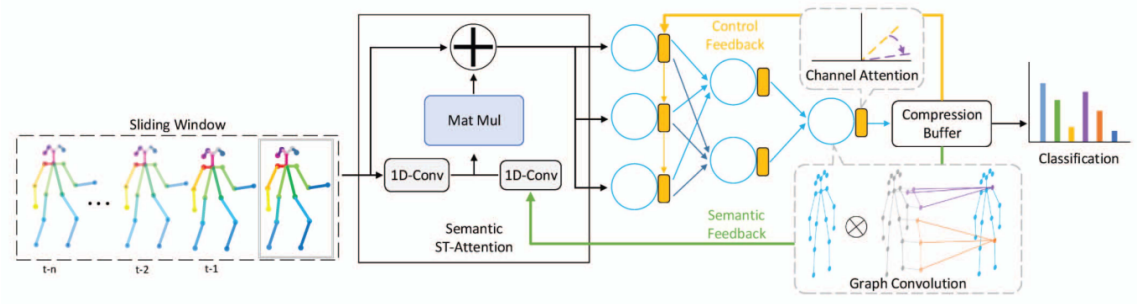



**Figura 4.5. Modelo de reconhecimento de atividade humana de (Mondal et al., 2020)**

Mondal et al. (2020) utilizam dados coletados de sensores de *smartphones* (e.g., acelerômetro, giroscópio, dentre outros) para prever o tipo de atividade desempenhada por um usuário em um dado momento (veja a Figura 4.5). Como esses dados são coletados continuamente, eles representam uma série temporal. Na modelagem do grafo, cada vértice representa uma atividade realizada em uma janela de tempo na série temporal. Essa atividade é caracterizada por dados agregados dos sensores embutidos (atributos dos vértices). Cada vértice é conectado com o vértice da janela de tempo anterior e posterior, considerando a série temporal. Em outras palavras, essa modelagem assume que cada atividade realizada está associada com os comportamentos anteriores e posteriores do indivíduo. A classificação de atividades (e.g., correr, caminhar, permanecer parado, andar de bicicleta, dentre outras) é realizada tanto a nível de vértice quanto a nível de grafo.

O reconhecimento de atividade humana também pode ser utilizado para sistemas de vigilância inteligente por vídeo (*smart video surveillance*). (Sanchez et al., 2021) consideram que uma atividade pode ser identificada com base em uma sequência de poses/ações contidos em um conjunto de *frames*. Nesse trabalho, é proposta uma rede espaço-temporal para o reconhecimento de atividade humana onde cada *frame* representa um corpo humano, que por sua vez é mapeado para um grafo, onde cada vértice corresponde a uma junção (*keypoint*) do corpo e as arestas conectam partes adjacentes. A matriz de atributos de vértices contém atributos que caracterizam as possíveis poses ou movimentos de cada indivíduo em um *frame*, e o processo de convolução agrega os movimentos realizados para se classificar o tipo de atividade realizada. Essa solução processa sequências de grafos utilizando uma arquitetura *ST-GCN* e se enquadra no contexto de tarefas de classificação de grafo.

#### 4.5.9. Discussão

As redes neurais para grafos têm se mostrado flexíveis para a inserção en novos contextos. Porém, um ponto a se notar é a pouca utilização da matriz de atributos das arestas $E$. Isso se deve a três motivos:

1. **Camadas**: a quantidade de camadas capazes de processar a matriz $E$ juntamente com as matrizes $A$ e $X$ é pequena.
2. **Modelagem**: na maior parte dos trabalhos, os autores utilizam a matriz de adjacência ponderada, de modo que os pesos são utilizados para caracterizar as arestas, em detrimento da utilização da matriz $E$.
3. **Contexto**: nem todos os contextos são abrangentes o suficiente para exigir a utilização da matriz $E$, onde cada aresta indica um tipo de relacionamento entre os vértices.

### 4.6. Dos dados brutos para o modelo *GNN*

Uma das principais razões para o advento das redes neurais de grafo é a sua capacidade de modelar problemas de diferentes contextos. Neste sentido, um dos aspectos relevantes que surge é o processo de transformação dos dados originais para a estrutura de grafo. Dependendo da modelagem utilizada, dados brutos de diferentes formas podem ser transformados em grafo, como: imagens, dados de *check-ins* de localização, trajetórias coletadas por sensores de *GPS*, sensores espalhados por estradas, dentre outros. Para isso, é necessária uma sequência de passos, desde o pré-processamento dos dados brutos até a implementação da rede neural. Além disso, séries temporais correspondem a um tipo de dado muito importante no contexto de *GNNs*, uma vez que elas podem ser utilizadas tanto para gerar grafos estáticos quanto para grafos dinâmicos, dependendo do contexto.

#### 4.6.1. Representatividade dos vértices

Vértices podem representar usuários, pontos de interesse, estações de metrô, segmento de estrada, regiões, veículos, estações meteorológicas, sensores, imagens, junções do corpo humano, dentre outros elementos. A Tabela 4.2 sumariza os principais

elementos representados pelos vértices em cada tópico onde as redes neurais de grafos são empregadas no contexto das cidades inteligentes.

### 4.6.2. Tipos de matrizes de adjacência

Além de modelar a representatividade dos vértices, é importante definir como eles interagem entre si. As matrizes de adjacência podem indicar diferentes tipos de interações entre vértices, como: distância, duração, conectividade de transporte, relacionamento composto, relacionamento/contato, frequência, similaridade e vizinhança. A Tabela 4.3 apresenta os tipos de matrizes de adjacência e os estudos relacionados no contexto das cidades inteligentes.

### 4.6.3. Série temporal

Uma série temporal é uma sequência ordenada de observações comumente realizadas entre períodos iguais ou dentro de um intervalo máximo de tempo. As séries temporais podem ser utilizadas para se gerar grafos estáticos ou dinâmicos, dependendo do contexto e da abordagem utilizada. Elas são essenciais para as redes neurais de grafos e, em especial, para problemas relacionados às cidades inteligentes. Por exemplo, a partir de séries temporais de dados de navegação, são construídas sequências de grafos dinâmicos que caracterizam a movimentação de veículos em um conjunto de ruas no contexto de previsão de tráfego (Dai et al., 2020).

No contexto das cidades inteligentes, grafos podem ser gerados a partir de séries temporais de diferentes tipos: trajetórias de *GPS* de usuários móveis (S. Wu, Zhang, et al., 2020; Capanema et al., 2021b); trajetórias de *GPS* de veículos (Dai et al., 2020), sequências de dados coletados por sensores de tráfego (Yu et al., 2017; Deng et al., 2022), embutidos em dispositivos móveis (e.g., acelerômetro e giroscópio (Mondal et al., 2020)); dados coletados periodicamente sobre a movimentação de massas (La Gatta et al., 2020); imagens pré-processadas (e.g., algoritmos que detectam junções do corpo humano para posterior construção de grafo no problema de reconhecimento de atividade humana) (Sanchez et al., 2021).

A partir de séries temporais pode ser gerado um grafo ou uma sequência de grafos. Capanema and Silva (2021) utilizam a trajetória histórica de um usuário para gerar um grafo, que representa a sua mobilidade em todo o período. Por outro lado, sequências de grafos também podem ser criadas, como em (Ye et al., 2020), o que é comumente associado aos grafos dinâmicos, onde as características do grafo variam ao longo do tempo. Considerando que os grafos dinâmicos adicionam a dimensão temporal em sua definição, eles são divididos em dois tipos (L. Wu et al., 2022): grafos dinâmicos contínuos temporalmente e grafos dinâmicos discretos temporalmente. O primeiro não é criado a partir da agregação temporal dos dados, sendo mais rico em informação mas, ao mesmo tempo, mais complexo. O segundo, agrega dados temporalmente em sua criação, sendo mais simples e amplamente utilizado no contexto das cidades inteligentes. Dessa forma, os grafos dinâmicos discretos são descritos nos parágrafos seguintes.

Um grafo dinâmico discreto temporalmente (do inglês *discrete-time dynamic graph* ou simplesmente *DTDG*) é composto por uma sequência de *snapshots* de grafos

**Tabela 4.2. Representatividade dos vértices para cada tópico.**

| Tópico | Vértice | Atributos |
|---|---|---|
| **Sistemas de recomendação** | Usuário | Representação latente de cada usuário (Xiao et al., 2020) |
| | Usuário-PoI (Usuário-Item) | Representação latente de usuário e *PoI* (Xiao et al., 2020) |
| | PoI-PoI (Item-Item) | Representação latente de cada *PoI* (S. Wu, Zhang, et al., 2020) e Distância e duração entre visitas (Capanema et al., 2021b) |
| **Previsão de tráfego** | Estação de sensor | Velocidade (Yu et al., 2017) |
| | Estação de metrô | Fluxos de passageiros (Ye et al., 2020) |
| | Segmento de estrada | Volume de tráfego e tempo de viagem (Dai et al., 2020) |
| | Região | Fluxo de viagens (Ke et al., 2021) |
| **Classificação de função urbana** | Segmento de estrada | (S. Hu et al., 2021) |
| | Região | (Yang et al., 2022) |
| **Disseminação de doenças** | Usuário | Estado de saúde (Tomy et al., 2022) |
| | Região | População, densidade demográfica e fluxo de entrada (La Gatta et al., 2020) |
| **Agentes autônomos** | Veículo | Velocidade, posição, localização e intenção de movimento (S. Chen et al., 2021) |
| **Detecção de anomalia** | Região | Fluxo de entrada e saída (Deng et al., 2022) e dados da fronteira geográfica da região (Y. Hu et al., 2020) |
| | Estação meteorológica | Dados de pressão atmosférica, temperatura, velocidade do vento, humidade, nível de precipitação, dentre outros (Owerko et al., 2018) |
| | Sensor | Dados coletados pelo sensor (D. Chen et al., 2021), (Deng et al., 2022) |
| **Reidentificação de indivíduo** | Imagem de referência - Imagem de galeria | Representação codificada da similaridade entre imagens (Shen et al., 2018) |
| **Reconhecimento de atividade humana** | Amostra de atividade | Dados agregados de acelerômetro, giroscópio, dentre outros (Mondal et al., 2020). esses dados são coletados em uma janela de tempo da atividade |
| | Junção do corpo humano (*keypoint*) | Atributos categóricos que indicam o tipo da pose/movimento em cada *frame* (Sanchez et al., 2021) |

**Tabela 4.3. Principais tipos de matrizes de adjacência em termos dos significados das arestas.**

| Tipo de matriz de adjacência | Definição | Estudo(s) |
|---|---|---|
| **Distância** | $A_{ij} = dis_{ij}$ onde $dis_{ij}$ é a distância entre $v_i$ e $v_j$. Um caso particular é a matriz de **proximidade de distância**, $f(dis_{ij})$ onde $f(.)$ é uma função aplicada na distância que atribui maior peso às distâncias menores | (S. Wu, Zhang, et al., 2020), (Yu et al., 2017), (Ke et al., 2021), (Yang et al., 2022), (Deng et al., 2022) e (Owerko et al., 2018) |
| **Duração** | $A_{ij} = dur_{ij}$ onde $dur_{ij}$ é a duração da viagem entre $v_i$ e $v_j$. Um caso particular é a matriz de **proximidade temporal**, $f(dur_{ij})$ onde $f(.)$ é uma função aplicada na distância que atribui maior peso às distâncias menores | (Y. Hu et al., 2020) |
| **Conectividade de transporte** | $A_{ij} \neq 0$ (ponderado por alguma métrica) se é possível viajar de $v_i$ para $v_j$ e $A_{ij} = 0$ caso contrário | (Ye et al., 2020) |
| **Composta** | $A_{ij} = f^c(ij)$ onde $f^c(.)$ é uma função que agrega dados de proximidade, tempo de viagem, dentre outros | (Dai et al., 2020) e (D. Chen et al., 2021) |
| **Relacionamento/ contato** | $A_{ij} = 1$ se $v_i$ e $v_j$ estabelecem algum relacionamento ou contato e $A_{ij} = 0$ caso contrário | (Xiao et al., 2020), (Tomy et al., 2022), (Mondal et al., 2020), (Sanchez et al., 2021) e (Sanchez et al., 2021) |
| **Frequência** | $A_{ij}$ indica a frequência de transições/viagens entre $v_i$ e $v_j$ | (Capanema et al., 2021b) e (La Gatta et al., 2020) |
| **Similaridade** | $A_{ij}$ representa o grau de similaridade entre as representações de $v_i$ e $v_j$ | (Shen et al., 2018) |
| **Vizinhança** | $A_{ij} = 1$ se $v_i$ e $v_j$ forem vizinhos e $A_{ij} = 0$ caso contrário. Esse é um caso particular de relacionamento/contato para entidades geográficas | (S. Hu et al., 2021) e (S. Chen et al., 2021) |

$[G^{(1)}, G^{(2)}, \ldots, G^{(t)}]$ coletados em intervalos de tempo regulares. Cada grafo é representado como $G^{(t)} = (V^{(t)}, A^{(t)}, X^{(t)})$, onde o conjunto de vértices $V^{(t)}$, a matriz de adjacência $A^{(t)}$ e a matriz de atributos dos vértices $X^{(t)}$ podem variar ao longo de intervalos discretos de tempo. Um tipo específico de grafo dinâmico discreto temporalmente é o grafo espaço-temporal. Nesta categoria de grafo, a topologia se mantém a mesma enquanto que os valores dos atributos podem mudar ao longo do tempo (Skardinga et al., 2021), como em (Yu et al., 2017; Deng et al., 2022; Mondal et al., 2020; La Gatta et al., 2020; Sanchez et al., 2021). No contexto das cidades inteligentes, é frequente a utilização de redes neurais para grafos dinâmicos espaço-temporais na área de previsão de tráfego onde os vértices representam sensores em estradas, o que significa que não ocorrerá nenhuma alteração na estrutura do grafo (Dai et al., 2020).

Um aspecto importante é o intervalo de tempo utilizado para coletar os dados de cada grafo na sequência (i.e., *snapshot*). Caso esse período de tempo seja muito curto, o grafo eventualmente possuirá poucas arestas e as matrizes de atributos terão, eventualmente, pouca informação. Por outro lado, períodos longos para a coleta de *snapshots* podem resultar na perda de informação precisa entre as mudanças de um grafo para o outro.

#### 4.6.4. Principais tarefas para cada tipo de problema

Compreender as tarefas mais comuns realizadas em cada área é importante para que pesquisadores possam direcionar melhor os seus esforços. A Tabela 4.4 apresenta as principais tarefas realizadas em cada tipo de tópico de *GNN* em cidades inteligentes, como descrito a seguir:

- Em **sistemas de recomendação**, as principais tarefas são: (1) predição de *link*, onde relacionamento entre usuários ou usuário-item são previstos; (2) classificação de vértice, onde se recomenda o item (e.g., *PoI*) de maior associação.

- Os problemas de **previsão de tráfego** se concentram nas seguintes tarefas: (1) regressão de grafo, onde o tempo de viagem em uma região ou o fluxo de tráfego de uma rede são previstos, por exemplo; (2) regressão de vértice, onde a velocidade, fluxos de passageiros, dentre outros aspectos, são previstos.

- No contexto de **classificação de função urbana**, a principal tarefa é a classificação de vértice, onde segmentos de estradas e regiões, por exemplo, têm os seus tipos classificados.

- Os problemas de **disseminação de doenças** estão frequentemente associados às tarefas de: (1) classificação de vértice, onde o estado de um indivíduo (vértice) é previsto; (2) em regressão de vértice, onde valores epidemiológicos (e.g., quantidade de indivíduos contaminados) podem ser previstos para cada localidade/região.

- Os problemas de **agentes autônomos** se concentram em tarefas de classificação de vértice onde, por exemplo, as possíveis ações (e.g., mudança de faixa de trânsito) de veículos são previstas.

- Em problemas de **detecção de anomalia**, as principais tarefas são: (1) predição de *link* em uma arquitetura *auto-encoder* ao se reconstruir a matriz de adjacência; (2) regressão de vértice em uma arquitetura *GAN*, onde os valores previstos dos atributos da matriz $X$ são utilizados na detecção de anomalia.
- Os problemas de **reidentificação de indivíduos** se concentram em tarefas de classificação de vértice. Nesse caso, cada vértice pode conter um conjunto de atributos que representam a similaridade entre duas imagens. Por fim, cada vértice é classificado como pertencente a um dado indivíduo ou não.
- Em sistemas de **reconhecimento de atividade humana**, as principais tarefas são: (1) classificação de vértice para a predição do tipo de atividade realizada; (2) classificação de grafo para a predição da atividade global composta por um conjunto de ações (vértices).

**Tabela 4.4. Principais tópicos e tarefas de *GNN* associadas**

| Tópico | Tarefa | Trabalho(s) |
|---|---|---|
| **Sistemas de recomendação** | Predição de *link*, classificação de vértice | (Xiao et al., 2020) |
| **Previsão de tráfego** | Regressão de grafo<br>Regressão de vértice | (Dai et al., 2020)<br>(Yu et al., 2017), (Ye et al., 2020) |
| **Classificação de função urbana** | Classificação de vértice | (S. Hu et al., 2021), (Yang et al., 2022), (Capanema et al., 2021b) |
| **Disseminação de doenças** | Classificação de vértice<br>Regressão de vértice | (Tomy et al., 2022)<br>(La Gatta et al., 2020) |
| **Agentes autônomos** | Classificação de vértice | (S. Chen et al., 2021) |
| **Detecção de anomalia** | Predição de *link*<br>Regressão de vértice | (Y. Hu et al., 2020)<br>(Deng et al., 2022) |
| **Reidentificação de indivíduo** | Classificação de vértice | (Shen et al., 2018) |
| **Reconhecimento de atividade humana** | Classificação de vértice<br><br>Classificação de grafo | (Mondal et al., 2020)<br><br>(Mondal et al., 2020), (Sanchez et al., 2021) |

### 4.6.5. Passo a passo

Considerando a capacidade dos grafos em modelar diversos problemas, é necessário estruturar o processo de conversão de dados brutos para a estrutura de grafo. Essa modelagem é composta por uma sequência de passos, onde são definidos os seguintes aspectos:

1. **Estrutura de grafo**: o grafo por ser direcionado/não direcionado, homogêneo/heterogêneo e estático/dinâmico.

2. **Tarefa**: a tarefa pode ser a nível de vértice, aresta ou de grafo.

3. **Função de perda**: com base na tarefa selecionada, se define a função de perda. Por exemplo, se a tarefa é uma classificação de vértice, então uma função de perda apropriada pode ser a *categorical cross entropy*.

4. **Tipo de supervisão**: com base nos dados utilizados e a disponibilidade de rótulos, deve-se definir o tipo de supervisão no treinamento: supervisionado, semi-supervisionado e não supervisionado.

5. **Camadas**: a escolha das camadas utilizadas depende do tipo do grafo, das entradas disponíveis (i.e., as matrizes de entrada $A$, $X$ e $E$) e da tarefa escolhida.

## 4.7. Desafios e questões abertas

As redes neurais de grafos apresentam tanto desafios teóricos quanto práticos. Além disso, alguns tópicos e técnicas têm se sobressaído e demonstrado serem promissoras para futuros trabalhos.

### 4.7.1. Desafios

O processo de aprendizagem em redes neurais de grafos é caracterizado pela agregação recursiva das informações de vértices vizinhos, e à medida que o modelo se torna cada vez mais profundo, dois problemas emergem (Zhou et al., 2020):

- **Explosão de vizinhança**: ao se utilizar consecutivamente múltiplas camadas de *message passing*, o número de vértices vizinhos cresce exponencialmente. Como alternativa, ao invés de agregar as informações de todos os vértices vizinhos a cada iteração, as camadas de amostragem agregam apenas subconjuntos de vértices a cada iteração, o que reduz a quantidade de tempo e memória necessários para a operação (D. Chen et al., 2020).

- **Sobre-suavização**: a intuição por trás do processo de *message passing* é que os vértices mais relacionados entre si assumem representações similares, o que é comumente chamado de suavização, um processo natural das redes *GNN*. No entanto, à medida que mais camadas de *message passing* são adicionadas, o número de vértices vizinhos cresce exponencialmente (explosão de vizinhança) e tanto informações úteis quanto ruídos são adicionados nas novas representações. Por exemplo, interações entre vértices de uma mesma classe são benéficos, enquanto que interações entre vértices de classes diferentes trazem ruídos. Neste sentido, as representações dos vértices tendem a se tornarem sobre-suavizadas (i.e., todos os vértices tendem a assumir representações semelhantes), o que torna difícil para que os modelos possam distinguir corretamente vértices de diferentes classes. A sobre-suavização é especialmente negativa para tarefas a nível de vértice (L. Wu et al., 2022). Para mitigar esse problema, operações de *skip connection* no contexto de grafos têm sido utilizadas para propagar representações históricas de vértices (D. Chen et al., 2020).

Além desses problemas, outro aspecto importante é a **invariância**: as camadas de redes neurais de grafos devem ser equivariantes em tarefas a nível de vértice e invariantes em tarefas a nível de grafo (Z. Wu et al., 2020). Isto significa que, quando a ordem dos vértices é alterada, apenas a ordem (equivariância) em que eles estão dispostos é alterada na nova representação gerada por uma camada de *message passing*. Por outro lado, a variação na ordem dos vértices do grafo não deve alterar a representação final gerada em uma tarefa a nível de grafo, em especial quando se utiliza camadas de *pooling* para a redução no tamanho do grafo.

### 4.7.2. Questões abertas

As principais questões abertas e os temas mais promissores de possíveis trabalhos futuros estão apresentados a seguir:

- **Enriquecimento de Dados**: diferentemente do domínio da imagem, onde rotações, adição de ruído e outras técnicas são suficientes para gerar novos dados, os grafos devem ser enriquecidos de acordo com a estrutura de dados complexa (e.g., correlação entre vértices e arestas). Quanto ao contexto das cidades inteligentes, a necessidade de enriquecimento é alta uma vez que, é comum que os dados utilizados sejam esparsos (e.g., dados de *check-in*), levando à construção de poucas instâncias de grafos ou grafos com poucas arestas. Zhao et al. (2021) melhoraram a classificação semi-supervisionada de vértices através de uma nova abordagem para *data augmentation* para grafos, onde arestas são adicionadas entre vértices que supostamente pertencem a uma mesma classe, ao mesmo tempo em que arestas de vértices de classes distintas são removidas.

- **Uso de múltiplas técnicas**: A utilização das *GNNs* juntamente com as *RNNs* (Capanema et al., 2021b) e *CNNs* (Shen et al., 2018; Lin et al., 2021) tem se mostrado promissora em diversas áreas. Com exceção de (Capanema et al., 2021b), a combinação de *GNNs* com *RNNs* ou *CNNs* está frequentemente associada ao processamento de sequências de grafos dinâmicos, onde é necessário capturar a evolução dos vértices e das arestas ao longo do tempo. Grafos espaço-temporais têm sido frequentemente utilizados para modelar fluxos de tráfego, uma vez que a topologia do grafo permanece a mesma. No entanto, grafos dinâmicos a nível estrutural, com adição e remoção de vértices, ainda foram pouco explorados nos problemas de cidades inteligentes apresentados neste capítulo.

- **Aprendizado multi-tarefa**: a estrutura de grafo pode ser utilizada para integrar o conhecimento sobre diferentes fontes de dados no aprendizado multi-tarefa, como: imagens, texto e bases de conhecimento (L. Wu et al., 2022).

- **Escalabilidade**: os módulos de amostragem têm sido extensivamente estudados como alternativa para se processar grafos grandes. No entanto, é importante que sejam desenvolvidos métodos adequados para grafos heterogêneos e dinâmicos (L. Wu et al., 2022).

## 4.8. Prática

Nesta seção, é apresentado um experimento prático no contexto das cidades inteligentes. Para isto, são descritos os principais conceitos da biblioteca *Spektral* que pertencente à linguagem *Python*. Além disso, é apresentada a modelagem da solução considerando a geração do grafo e o processo de desenvolvimento da rede neral, além da apresentação de um código de exemplo seguido de sugestões de melhorias.

### 4.8.1. Biblioteca *Spektral*

A biblioteca *Spektral* (Bianchi et al., 2020) na sua versão 1.1.0 é utilizada neste capítulo para o treinamento de redes neurais de grafos. O pacote é baseado no *Tensorflow*, se beneficiando dos seus recursos para ambiente de produção.

O treinamento de redes de grafos não é trivial como em outros contextos. Por exemplo, com relação às redes *CNN*, imagens podem ser cortadas ou preenchidas, através do *padding* para assumirem um mesmo tamanho. Da mesma forma, em redes *RNN* as sequências podem ser preenchidas. O domínio de grafo não permite que operações “diretas"como essas sejam realizadas, pois elas assumem, por exemplo, a proximidade espacial dos *pixels*, o que não existe em grafos. A retirada de um vértice pode, por exemplo, desconectar partes de um grafo.

Dessa forma, a biblioteca possui o conceito de *modo de dados*, que organiza os modos como as entradas são processadas pela rede neural. Os quatro modos de dados são apresentados a seguir:

1. *Single*: apenas um grafo é utilizado.

2. *Disjoint*: um *batch* de grafos é representado pelas suas uniões disjuntas. Caso não se deseje utilizar matrizes com representação densa, nem realizar o preenchimento com zero, esse é o modo mais adequado.

3. *Batch*: os grafos devem ter o mesmo número de vértices, o que é comumente alcançado com o preenchimento de zeros (i.e., *zero-padding*). esse modo tem maior comunalidade com os recursos do *Tensorflow*.

4. *Mixed*: todos os grafos compartilham de uma mesma matriz de adjacência, variando apenas as matrizes de atributos. Para evitar replicações e, portanto, melhorar o desempenho, a matriz de adjacência é processada no modo *single*, enquanto que a matriz de atributos é processada no modo *batch*.

### 4.8.2. Experimento

O experimento realizado neste capítulo consiste da modelagem e teste de uma rede neural de grafos para o problema de predição de categoria de pontos de interesse (*PoIs*). Um ponto de interesse é um local/estabelecimento frequentemente visitado por um indivíduo. Esse problema se insere no contexto de classificação de função urbana, onde cada *PoI* corresponde a um vértice e as arestas representam as viagens realizadas por um usuário móvel entre pares de *PoIs*.

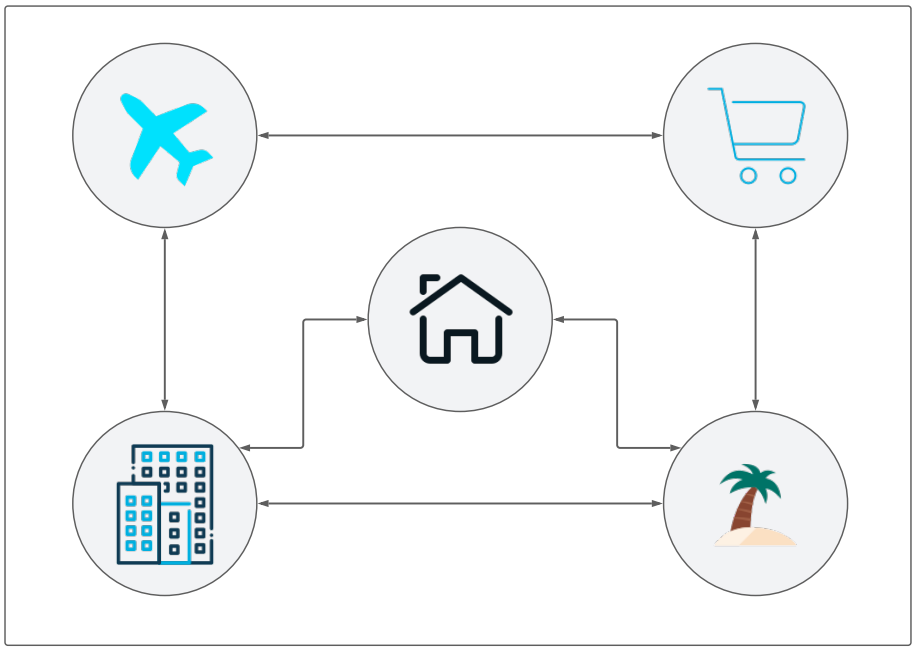


**Figura 4.6. Exemplo de grafo não direcionado *PoIxPoI*, onde cada *PoI* é representado por um vértice e as viagens realizas por um indivíduo entre cada par de *PoIs* são representadas por arestas**

#### 4.8.2.1. Motivação

Em alguns serviços baseados em geolocalização em que a mobilidade dos usuários é frequentemente registrada, é comum que nenhuma informação adicional seja fornecida além dos dados brutos do *GPS*. Serviços de mapas como *OSM* (*Open Street Map*) e *Google Maps* podem ser usados para anotar a semântica dos *PoIs*; no entanto, essa abordagem direta nem sempre é suficiente, pois é comum que muitos lugares ainda não estejam anotados no sistema de mapas, ou haja um custo para usar esse tipo de serviço. Desse forma, é necessário fornecer um modelo *offline* e sem custo, capaz de extrair os padrões de visitas a cada tipo de categoria de *PoI* para realizar, dessa forma, o enriquecimento semântico de outras bases de dados.

#### 4.8.2.2. Modelagem

A base de dados utilizada contém *check-ins* de localização de usuários da rede social *Gowalla* (Liu et al., 2014). Cada registro contém as seguintes informações: identificador do usuário e local visitado (assumimos que esse local é um *PoI*), data e horário, latitude e longitude, além da categoria do local (i.e., rótulo do problema de classificação).

Com base nos *check-ins* de localização de um usuário, é construído um grafo, onde cada vértice corresponde a um *PoI* e cada aresta conecta *PoIs* visitados consecutivamente, como é mostrado na Figura 4.6.

Com relação às entradas para o modelo, a matriz de adjacência $A$ é ponderada

pela quantidade de viagens consecutivas entre cada par de *PoIs*. A matriz de atributos dos vértices $X \in \mathbb{R}^{N \times 48}$ representa a quantidade de visitas feitas à cada *PoI* para cada hora do dia (i.e., 24 horas para dias de semana e 24 horas para finais de semana). No processo de *message passing* cada *PoI*, representado por um vértice, agrega os padrões de visita (i.e., horários de visitação) dos estabelecimentos visitados antes e depois (i.e., vértices vizinhos). Como o grafo é ponderado, os pares de *PoIs* que um dado indivíduo fez mais visitas consecutivas possui um maior peso na etapa de agregação dos valores de seus atributos. Ao final desse processo, a representação de cada *PoI* contêm as características de visitação dos estabelecimentos vizinhos considerando o histórico de mobilidade de um usuário móvel.

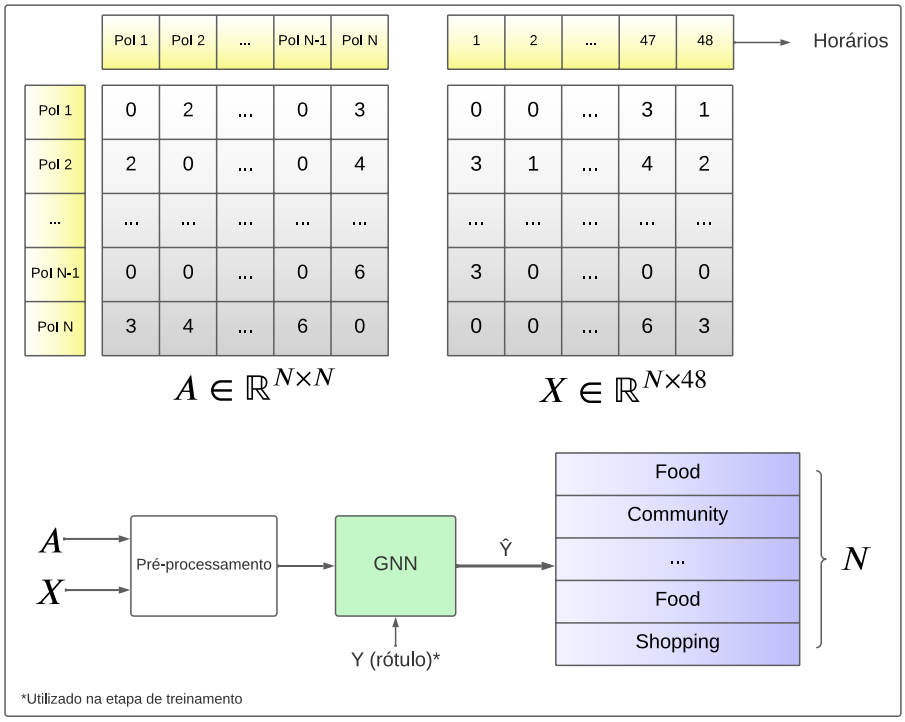



**Figura 4.7. Diagrama de modelagem da solução**

A Figura 4.7 exibe o diagrama de modelagem da solução. As matrizes de entrada $A$ e $X$ estão, inicialmente, não normalizadas. Na etapa de pré-processamento, a matriz de adjacência é modificada de acordo com a camada de *message passing* utilizada que, no presente exemplo, corresponde à camada *ARMA*. A biblioteca *Spektral* fornece essas operações de pré-processamento. Adicionalmente, a matriz de atributos dos vértices $X$ pode ser normalizada. Por fim, as entradas são enviadas para o modelo *GNN*, onde são previstas as categorias dos *PoIs* representados por cada vértice do grafo. A base de dados *Gowalla* contém sete tipos de categorias de estabelecimentos: *Community*, *Entertainment*, *Food*, *Nightlife*, *Outdoors*, *Shopping* e *Travel*.

#### 4.8.2.3. Exemplo de implementação

A Figura 4.8 apresenta um trecho de código de rede neural de grafo para a classificação de vértice e que pode ser utilizado no contexto de classificação de *PoIs*. São utilizadas duas camadas *ARMA* para o processo de *message passing*. Como entrada, são utilizadas as matrizes de adjacência *A_input* e de atributos de vértices *X_input*. Além dos hiperparâmetros comuns de uma camada de rede neural, é importante observar que essa camada requer valores para *order* e *iterations* que indicam a quantidade de blocos de camadas e a quantidade de camadas *GCS* em cada bloco. Além disso, apenas a matriz de atributos de vértices *X* é atualizada a cada iteração, e a matriz de adjacência é mantida com os mesmos valores iniciais. Outro aspecto importante, é que essa camada possui dois parâmetros para funções de ativação, sendo o primeiro utilizado pelas camadas *GCS* internas e o segundo aplicado sobre a saída da camada *ARMA*. Como esse é um problema de classificação com diferentes rótulos, a função de ativação da última camada *ARMA* é a *softmax*.

```
class GNN(Model):
    def __init__(self):
        super().__init__()
        self.mask = GraphMasking()
        self.conv1 = ARMAConv(
        16,
        iterations=1,
        order=2,
        share_weights=True,
        dropout_rate=0.75,
        activation="elu",
        gcn_activation="elu",
        kernel_regularizer=l2(5e-5)
    )
        self.dropout = Dropout(0.6)
        self.conv2 = ARMAConv(
        7,
        iterations=1,
        order=1,
        share_weights=True,
        dropout_rate=0.75,
        activation="softmax",
        gcn_activation=None,
        kernel_regularizer=l2(5e-5),
    )
    def call(self, inputs):
        X_input, A_input = inputs
        X = self.mask(X_input)
        X = self.conv1([X, A_input])
        X = self.dropout(X)
        output = self.conv2([X, A_input])
        return output
```

**Figura 4.8. Definição do modelo *GNN***

A Figura 4.9 apresenta o trecho de código onde o modelo final é criado considerando as matrizes de entrada e a saída. A rede neural é compilada utilizando o otimizador *Adam*, a função de perda *categorical crossentropy* e a métrica acurácia. Após isso, é realizado o treinamento e a validação com os dados de teste utilizando o método *fit*. O código completo juntamente com os resultados estão disponíveis no repositório deste capítulo[1]. O algoritmo do exemplo apresentado alcança cerca de 40% de acurácia.

```
model = GNN()
opt = Adam(lr=0.0001)
model.compile(optimizer=opt,
              loss="categorical_crossentropy",
              metrics=["acc"])

model.fit(
    loader_tr.load(),
    steps_per_epoch=loader_tr.steps_per_epoch,
    epochs=epochs,
    validation_data=loader_va.load(),
    validation_steps=loader_va.steps_per_epoch,
    callbacks=[EarlyStopping(patience=3,
                             restore_best_weights=True)],
)
```

**Figura 4.9. Configuração do treinamento do modelo *GNN***

Considerando o contexto desse experimento e as entradas disponíveis para o modelo, são propostas direções para melhorias no desempenho da rede neural:

- Alteração nos valores dos hiperparâmetros.
- Modificar a quantidade de camadas *GNN* utilizadas.
- Avaliar o emprego de outras camadas de *message passing*. A biblioteca *Spektral* contém diversas camadas que podem ser utilizadas no presente problema, como: *GCN*, *GCS*, *APPNP* (*Approximate Personalized Propagation of Neural Predictions*) (Klicpera et al., 2018), dentre outras. É importante destacar que ao se mudar a camada utilizada, o método de pré-processamento aplicado sobre a matriz de adjacência também deve ser modificado utilizando-se a função apropriada. Outros exemplos de como isto é realizado estão presentes na documentação[2] e no repositório[3] da biblioteca.
- Avaliar o uso de outros tipos de camadas.

[1] https://github.com/claudiocapanema/minicurso_gnn_sbrc2022. Acessado em 13/5/2022.
[2] https://graphneural.network/. Acessado em 3/5/2022.
[3] https://github.com/danielegrattarola/spektral/. Acessado em 3/5/2022.

## 4.9. Considerações finais

Este capítulo discorreu sobre a redes neurais de grafos no contexto das cidades inteligentes, incluindo os fundamentos teóricos e aplicações práticas das *GNNs*. Inicialmente, foram apresentados os principais conceitos, os tipos de entradas possíveis, as tarefas existentes, os tipos de treinamentos e as principais camadas *GNN* juntamente com o fundamento teórico e as motivações para cada classe de camadas.

Em seguida, foram apresentadas as principais aplicações das redes neurais de grafos no contexto das cidades inteligentes, o que inclui os tópicos de sistemas de recomendação, previsão de tráfego, classificação de função urbana, disseminação de doenças, agentes autônomos, detecção de anomalia, reidentificação de indivíduo e reconhecimento de atividade humana. Além disso, foi discutido o processo de conversão de dados brutos para a estrutura de grafo com foco especial em séries temporais. Dois aspectos que foram apresentados e que são importantes para o desenvolvimento de novas soluções são: (1) principais tarefas utilizadas de *GNN* em cada tópico de problema; (2) desafios e questões abertas que são importantes para indicar a direção de novos métodos.

Por último, foi apresentada a modelagem de um experimento prático para o problema de classificação de pontos de interesse envolvendo redes neurais de grafos através da biblioteca *Spektral*.

## 4.10. Agradecimento

Este trabalho contou com o apoio da CAPES.

## Referências


Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. *Advances in neural information processing systems*, *29*.

Bianchi, F. M., Grattarola, D., & Alippi, C. (2020). Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning* (pp. 874–883).

Bianchi, F. M., Grattarola, D., Livi, L., & Alippi, C. (2021). Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence*.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Capanema, C. G. S., & Silva, F. A. (2021). Detecção de pontos de interesse e predição de próximo local de visita de usuários móveis com base em dados esparsos. In *Anais estendidos do xxxix simpósio brasileiro de redes de computadores e sistemas distribuídos* (pp. 129–136).

Capanema, C. G. S., Silva, F. A., & Silva, T. R. M. B. (2019). Identificacão e classificacão de pontos de interesse individuais com base em dados esparsos. In *Anais do xxxvii simpósio brasileiro de redes de computadores e sistemas distribuídos* (pp. 15–28).

Capanema, C. G. S., Silva, F. A., Silva, T. R. M. B., & Loureiro, A. A. F. (2021a). Dcluster: Geospatial analytics with poi identification. *Journal of Information and Data Management*, *12*(2).

Capanema, C. G. S., Silva, F. A., Silva, T. R. M. B., & Loureiro, A. A. F. (2021b). Poirgnn: Using recurrent and graph neural networks to predict the category of the next point of interest. In *Proceedings of the 18th acm symposium on performance evaluation of wireless ad hoc, sensor, & ubiquitous networks* (pp. 49–56).

Chen, D., Lin, Y., Li, W., Li, P., Zhou, J., & Sun, X. (2020). Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 34, pp. 3438–3445).

Chen, D., Liu, R., Hu, Q., & Ding, S. X. (2021). Interaction-aware graph neural networks for fault diagnosis of complex industrial processes. *IEEE Transactions on Neural Networks and Learning Systems*.

Chen, J., Ma, T., & Xiao, C. (2018). Fastgcn: fast learning with graph convolutional networks via importance sampling. *arXiv preprint arXiv:1801.10247*.

Chen, S., Dong, J., Ha, P., Li, Y., & Labi, S. (2021). Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. *Computer-Aided Civil and Infrastructure Engineering*, *36*(7), 838–857.

Cheng, D., Yang, F., Xiang, S., & Liu, J. (2022). Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition*, *121*, 108218.

Chiang, W.-L., Liu, X., Si, S., Li, Y., Bengio, S., & Hsieh, C.-J. (2019). Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (pp. 257–266).

Dai, R., Xu, S., Gu, Q., Ji, C., & Liu, K. (2020). Hybrid spatio-temporal graph convolu-

tional network: Improving traffic prediction with navigation data. In *Proceedings of the 26th acm sigkdd international conference on knowledge discovery & data mining* (pp. 3074–3082).

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems, 29*.

Deng, L., Lian, D., Huang, Z., & Chen, E. (2022). Graph convolutional adversarial networks for spatiotemporal anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems*.

Do, K., Tran, T., & Venkatesh, S. (2019). Graph transformation policy network for chemical reaction prediction. In *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining* (pp. 750–760).

Duarte, J., & Vlimant, J.-R. (2022). Graph neural networks for particle tracking and reconstruction. In *Artificial intelligence for high energy physics* (pp. 387–436). World Scientific.

Grattarola, D., & Alippi, C. (2021). Graph neural networks in tensorflow and keras with spektral [application notes]. *IEEE Computational Intelligence Magazine, 16*(1), 99–106.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems, 30*.

Hu, S., Gao, S., Wu, L., Xu, Y., Zhang, Z., Cui, H., & Gong, X. (2021). Urban function classification at road segment level using taxi trajectory data: A graph convolutional neural network approach. *Computers, Environment and Urban Systems, 87*, 101619.

Hu, Y., Qu, A., & Work, D. (2020). Graph convolutional networks for traffic anomaly. *arXiv preprint arXiv:2012.13637*.

Jiang, W., & Luo, J. (2021). Graph neural network for traffic forecasting: A survey. *arXiv preprint arXiv:2101.11174*.

Ke, J., Qin, X., Yang, H., Zheng, Z., Zhu, Z., & Ye, J. (2021). Predicting origin-destination ride-sourcing demand with a spatio-temporal encoder-decoder residual multi-graph convolutional network. *Transportation Research Part C: Emerging Technologies, 122*, 102858.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., & Zemel, R. (2018). Neural relational inference for interacting systems. In *International conference on machine learning* (pp. 2688–2697).

Kipf, T. N., & Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kipf, T. N., & Welling, M. (2016b). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Klicpera, J., Bojchevski, A., & Günnemann, S. (2018). Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*.

La Gatta, V., Moscato, V., Postiglione, M., & Sperli, G. (2020). An epidemiological neural network exploiting dynamic graph structured data applied to the covid-19 outbreak. *IEEE Transactions on Big Data, 7*(1), 45–55.

Li, Y., Gu, C., Dullien, T., Vinyals, O., & Kohli, P. (2019). Graph matching networks for learning the similarity of graph structured objects. In *International conference on*

*machine learning* (pp. 3835–3845).

Liang, T., Sheng, X., Zhou, L., Li, Y., Gao, H., Yin, Y., & Chen, L. (2021). Mobile app recommendation via heterogeneous graph neural network in edge computing. *Applied Soft Computing*, *103*, 107162.

Lin, D., Lin, J., Zhao, L., Wang, Z. J., & Chen, Z. (2021). Multilabel aerial image classification with a concept attention graph neural network. *IEEE Transactions on Geoscience and Remote Sensing*, *60*, 1–12.

Liu, Y., Wei, W., Sun, A., & Miao, C. (2014). Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management* (pp. 739–748).

Maguire, J. B., Grattarola, D., Mulligan, V. K., Klyshko, E., & Melo, H. (2021). Xenet: Using a new graph convolution to accelerate the timeline for protein design on quantum computers. *PLoS computational biology*, *17*(9), e1009037.

Malekzadeh, M., Hajibabaee, P., Heidari, M., Zad, S., Uzuner, O., & Jones, J. H. (2021). Review of graph neural network in text classification. In *2021 ieee 12th annual ubiquitous computing, electronics & mobile communication conference (uemcon)* (pp. 0084–0091).

Mondal, R., Mukherjee, D., Singh, P. K., Bhateja, V., & Sarkar, R. (2020). A new framework for smartphone sensor-based human activity recognition using graph neural network. *IEEE Sensors Journal*, *21*(10), 11461–11468.

Owerko, D., Gama, F., & Ribeiro, A. (2018). Predicting power outages using graph neural networks. In *2018 ieee global conference on signal and information processing (globalsip)* (pp. 743–747).

Rhee, S., Seo, S., & Kim, S. (2017). Hybrid approach of relation network and localized graph convolutional filtering for breast cancer subtype classification. *arXiv preprint arXiv:1711.05859*.

Sanchez, J., Neff, C., & Tabkhi, H. (2021). Real-world graph convolution networks (rw-gcns) for action recognition in smart video surveillance. In *2021 ieee/acm symposium on edge computing (sec)* (pp. 121–134).

Shen, Y., Li, H., Yi, S., Chen, D., & Wang, X. (2018). Person re-identification with deep similarity-guided graph neural network. In *Proceedings of the european conference on computer vision (eccv)* (pp. 486–504).

Skardinga, J., Gabrys, B., & Musial, K. (2021). Foundations and modelling of dynamic networks using dynamic graph neural networks: A survey. *IEEE Access*.

Tomy, A., Razzanelli, M., Di Lauro, F., Rus, D., & Della Santina, C. (2022). Estimating the state of epidemics spreading with graph neural networks. *Nonlinear Dynamics*, 1–15.

Tran, D. V., Navarin, N., & Sperduti, A. (2018). On filter size in graph convolutional networks. In *2018 ieee symposium series on computational intelligence (ssci)* (pp. 1534–1541).

Tsitsulin, A., Palowitch, J., Perozzi, B., & Müller, E. (2020). Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904*.

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. *stat*, *1050*, 20.

Wu, L., Cui, P., Pei, J., Zhao, L., & Song, L. (2022). Graph neural networks. In *Graph neural networks: Foundations, frontiers, and applications* (pp. 27–37). Springer.

Wu, S., Sun, F., Zhang, W., & Cui, B. (2020). Graph neural networks in recommender systems: a survey. *arXiv preprint arXiv:2011.02260*.

Wu, S., Zhang, Y., Gao, C., Bian, K., & Cui, B. (2020). Garg: anonymous recommendation of point-of-interest in mobile networks by graph convolution network. *Data Science and Engineering*, *5*(4), 433–447.

Wu, Y., Dai, H.-N., & Tang, H. (2021). Graph neural networks for anomaly detection in industrial internet of things. *IEEE Internet of Things Journal*.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, *32*(1), 4–24.

Xiao, Y., Yao, L., Pei, Q., Wang, X., Yang, J., & Sheng, Q. Z. (2020). Mgnn: Mutualistic graph neural network for joint friend and item recommendation. *IEEE Intelligent Systems*, *35*(5), 7–17.

Yang, M., Kong, B., Dang, R., & Yan, X. (2022). Classifying urban functional regions by integrating buildings and points-of-interest using a stacking ensemble method. *International Journal of Applied Earth Observation and Geoinformation*, *108*, 102753.

Yao, L., Mao, C., & Luo, Y. (2019). Graph convolutional networks for text classification. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 7370–7377).

Ye, J., Zhao, J., Ye, K., & Xu, C. (2020). Multi-stgcnet: A graph convolution based spatial-temporal framework for subway passenger flow forecasting. In *2020 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Yin, C., Xiong, Z., Chen, H., Wang, J., Cooper, D., & David, B. (2015). A literature survey on smart cities. *Science China Information Sciences*, *58*(10), 1–18.

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining* (pp. 974–983).

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, *31*.

You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., & Shen, Y. (2020). Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, *33*, 5812–5823.

Yu, B., Yin, H., & Zhu, Z. (2017). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*.

Zhang, S., Tong, H., Xu, J., & Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, *6*(1), 1–23.

Zhang, Z., Cui, P., & Zhu, W. (2020). Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering*.

Zhao, T., Liu, Y., Neves, L., Woodford, O., Jiang, M., & Shah, N. (2021). Data augmentation for graph neural networks. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 35, pp. 11015–11023).

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., … Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open, 1*, 57–81.

Zou, D., Hu, Z., Wang, Y., Jiang, S., Sun, Y., & Gu, Q. (2019). Layer-dependent importance sampling for training deep and large graph convolutional networks. *Advances in neural information processing systems, 32*.