Classificação de Imagens de Letras Cursivas Utilizando Redes Neurais em Grafos

Matheus Santos Araújo

[1]Centro de Ciências e Tecnologia – Universidade Estadual do Ceara (UECE)
Avenida Silas Munguba – Fortaleza – CE – Brazil

Resumo. *Recentemente, as Redes Neurais de Grafos (GNNs) tem crescido em popularidade na comunidade de aprendizagem de máquina e tem sido aplicadas em diversos problemas incluindo problemas de processamento de imagem, visão computacional e em particular análise e reconhecimento de imagens de documentos, generalizando o conceito de Rede Neural Convolucional. Diferentes objetos gráficos são bem representados por vértices e arestas, neste artigo, utilizamos uma GNN para aprender a reconhecer letras manuscritas e símbolos gráficos de três conjuntos de dados de diferentes níveis de dificuldade. Os resultados são promissores e apresentam a capacidade de GNNs para a classificação de símbolos ou letras do alfabeto a partir do aprendizado da topologia e atributos dos vértices da reprensentação da imagem em grafo.*

1. Introdução

Recentemente, as GNNs tem crescido em popularidade na comunidade de aprendizagem de máquina e tem sido aplicadas em diversos problemas incluindo problemas de processamento de imagem, visão computacional e em particular análise e reconhecimento de imagens de documentos, generalizando o conceito de Rede Neural Convolucional. No campo da análise de documentos, a formalização em grafos se tornou muito adequada e eficaz em representar a informação e executar reconhecimentos de padrões utilizando bases de dados de vértices e arestas.

As técnicas baseadas em grafos e Redes Neurais em grafos, apesar de serem muitas vezes menos eficientes que abordagens tradicionais estatíticas para reconhecimento de padrões, são hoje comumente utilizadas na análise de documentos como uma alternativa para capturar o aspecto estrutural informações de caracteres, símbolos e outros conteúdos em imagens [Conte et al. 2004].

Diferentes objetos gráficos são bem representados por vértices e arestas [Foggia et al. 2014], neste artigo, utilizamos uma GNN para aprender a reconhecer letras manuscritas e símbolos gráficos de três conjuntos de dados de diferentes níveis de dificuldade, o *letters-high*, *letters-mid* e *letters-low*. Os resultados são promissores e apresentam a capacidade de GNNs para a classificação de símbolos ou letras do alfabeto a partir do aprendizado da topologia e atributos dos vértices da reprensentação da imagem em grafo em que essas letras manuscritas são convertidas em grafos tal que as linhas são representadas por arestas e os pontos finais destas linhas são representados por nós os quais, por sua vez, possuem atributos. Portanto, é possível fazer uma abordagem de classificação de imagens agora com GNN e grafos e não apenas a tradicional CNN. Os resultados, utilizando três arquiteturas de GNNs: GCN, GAT e GIN, são promissores e apresentam a capacidade de GNNs para a classificação de símbolos ou letras do alfabeto

a partir do aprendizado da topologia e atributos dos vértices da representação da imagem em grafo.

2. Fundamentação Teórica e Revisão da Literatura

2.1. Base de Dados *Letters* (*letter-high*, *letter-med* e *letter-low*)

O conjunto de dados *Letters* consiste em 15 classes. Cada classe representa uma letra maiúscula. O conjunto de dados consiste nas letras A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z desenhadas manualmente. Essas letras manuscritas são convertidas em grafos nos quais as linhas são representadas por arestas e os pontos finais dessas linhas são apresentados por vértices. Cada vértice é rotulado com um atributo bidimensional sendo sua posição X e Y em relação a um sistema de coordenadas de referência. As arestas não possuem atributos. Um total de 6750 representações de grafos são divididas em três categorias, cada uma com 2250 gráficos no total variando em nível de dificuldade para o reconhecimento. Esses 2250 gráficos são então divididos em teste, treinamento e validação e cada conjunto contém 750 gráficos. Os resultados do estado da arte neste conjunto de dados são ligeiramente inferiores a 100%, o que mostra que o conjunto de dados é bastante apropriado para o aprendizado. Em suma, conjunto de dados consiste em três categorias *low*, *med* e *high* com base no número de vértices e arestas e seu nível de distorção. A figura 3 apresenta exemplos das três categorias.

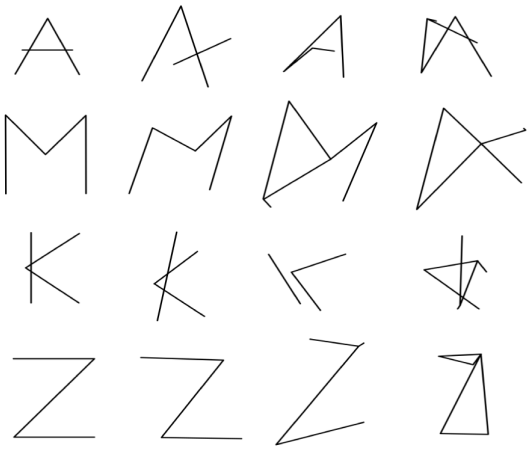

Figure 1. Alguns exemplos de letras com diferentes níveis de distorção (*low*, *mid* e *high* da esquerda para a direita) do conjunto de dados de *Letters* [Kajla et al. 2021].

2.2. *Redes Neurais em Grafos (GNNs)*

2.3. GCNs

Com o êxito e os avanços recentes das Redes Neurais Convolucionais (CNNs) em diferentes áreas de reconhecimento de padrões e aprendizagem de máquina, como processamento de linguagem natural, processamento de imagens e visão computacional, o interesse dos projetistas em estender essas estruturas para as estruturas mais generalistas e não euclidianas aumentou, ou seja, variedades, grafos e demais representações topológicas. Dentre os avanços nessas arquiteturas, [Wu et al. 2016] apresentou um algoritmo de tradução automática baseado em GNNs para Processamento de Linguagem Natural (PNL).

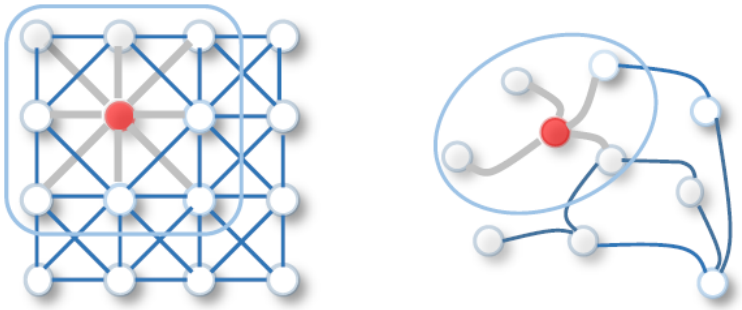

Figure 2. A esquerda uma convolução 2D. De forma análoga a um grafo, cada pixel em uma imagem é considerado um nó onde os vizinhos são determinados pelo tamanho do filtro. A convolução 2D obtém a média ponderada dos valores de pixel do nó vermelho junto com seus vizinhos. Os vizinhos de um nó são ordenados e têm um tamanho fixo. A direita uma convolução de grafo. Para obter uma representação oculta do nó vermelho, uma solução simples da operação convolucional do grafo é pegar o valor médio das características do nó do nó vermelho junto com seus vizinhos. Diferente dos dados de imagem, os vizinhos de um nó são desordenados e de tamanho variável [Wu et al. 2020].

Em [Defferrard et al. 2016] é proposto um método em que CNNs são generalizadas para um baixo nível dimensional regular de *grids*, onde fala, vídeo e imagem são representados, para domínios irregulares de alta dimensão, como redes sociais ou projeção de palavras representadas por grafos. Enquanto isso, [Li et al. 2015] propôs uma Rede Neural de Passagem de Mensagens (MPNN). Eles usaram amarração de peso para cada etapa de tempo. Eles também usaram uma função de atualização para cada etapa de tempo.

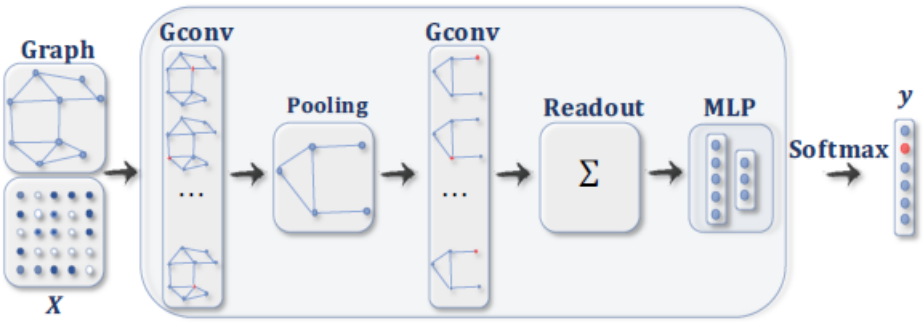

Figure 3. Uma GCN com várias camadas convolucionais de grafo. Uma camada convolucional de grafo encapsula a representação oculta de cada vértice agregando informações de recursos de seus vizinhos. Após a agregação de recursos, uma transformação não linear ou camada de*pooling* é aplicada às saídas resultantes. Ao empilhar várias camadas, a representação oculta final de cada nó recebe mensagens de uma vizinhança posterior. Uma camada posterior de leitura resume a representação final do grafo tomando a soma/média das representações ocultas dos subgrafos [Wu et al. 2020].

GNNs generalizam a operação de convolução de dados, bem como de *grid* para dados de grafo e usam essa estrutura e recursos de vértice X_v para aprender automáticamente um vetor de representação de um vértice, h_v, ou o grafo completo, h_G.

GNNs modernos seguem uma estratégia de agregação de vizinhança, onde atualizamos iterativamente a representação de um nó agregando representações de seus vizinhos. Após k iterações de agregação, a representação de um nó captura as informações estruturais dentro de sua vizinhança de rede k-hop. Essas redes são conhecidas como GCNs ((*Graph Convolutional Network*)).

Além das GCNs de Grafos Espaciais, existem as Redes Convolucionais de Grafos Espectrais. A ideia original por trás da *Spectral* GCN foi inspirada na propagação de sinais. Podemos pensar na propagação da informação na GCN como a propagação do sinal ao longo dos vértices. As GCNs espectrais fazem uso da decomposição Eigen da matriz Laplaciana de grafo para implementar um método de propagação de informação. Simplificando, a decomposição Eigen nos ajuda a entender a estrutura do grafo, portanto, classificar os nós dos grafos. Isso é um pouco semelhante ao conceito básico de Análise de Componentes Principais (PCA) e Análise Discriminante Linear (LDA), onde usamos a decomposição Eigen para reduzir a dimensionalidade e realizar agrupamento.

2.4. GAT

As Redes de Atenção em Grafos ou *Graph Attention Network* (GATs) descritas em [Veličković et al. 2018] operam em dados estruturados em grafos, aproveitando camadas de ”autoatenção” mascaradas para abordar as deficiências dos métodos anteriores baseados em convoluções de grafos ou suas aproximações. Ao empilhar camadas nas quais os vértices são capazes de atender às características de suas vizinhanças e possibilitam (implicitamente) especificar diferentes pesos para diferentes vértices em uma vizinhança, sem exigir qualquer tipo de operação de matriz custosa (como inversão) ou depender de conhecer a estrutura inicial do grafo.

A principal diferença entre a GAT e a GCN é como as informações da vizinhança de um salto são agregadas. Para a GCN, uma operação de convolução de grafo produz a soma normalizada das características dos vértices dos vizinhos.

$$h_i^{(l+1)} = \sigma \sum_{j \in \mathcal{N}} 1/C_{ij} * W_j^{(l)*h^{(l)}} \tag{1}$$

Onde $\mathcal{N}(i)$ é o conjunto de seus vizinhos de um salto (para incluir v_i no conjunto, basta adicionar um auto-loop a cada vértice), $c_{ij} = \sqrt{|\mathcal{N}(\rangle)|} * \sqrt{|\mathcal{N}(|)|}$ é uma constante de normalização baseada na estrutura do grafo, σ é uma função de ativação (GCN usa ReLU), e $W^{(l)}$ é uma matriz de peso compartilhada para a transformação de recursos do nó inteligente. Outro modelo proposto no GraphSAGE emprega a mesma regra de atualização, exceto que eles definem $c_{ij} = |\mathcal{N}(\rangle)|$.

GAT apresenta o mecanismo de atenção como um substituto para a operação de convolução estaticamente normalizada. Abaixo estão as equações para calcular a incorporação do vértice $h^i_{(l+1)}$ da camada $l + 1$ das projeções da camada l.

$$z_i^{(l)} = W^{(l)} h_i^{(l)}, \tag{1}$$

$$e_{ij}^{(l)} = \text{LeakyReLU}(\vec{a}^{(l)^T}(z_i^{(l)} || z_j^{(l)})), \tag{2}$$

$$\alpha_{ij}^{(l)} = \frac{\exp(e_{ij}^{(l)})}{\sum_{k \in \mathcal{N}(i)} \exp(e_{ik}^{(l)})}, \tag{3}$$

$$h_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} z_j^{(l)}\right), \tag{4}$$

A equação (1) é uma transformação linear da projeção das camadas inferior $h_i^{(l)}$ e $= W^{(l)}$ é sua matriz de peso aprendível. A equação (2) calcula uma pontuação de "atenção" não normalizada de pares entre dois vizinhos. Aqui, ele primeiro concatena as projeções dos dois vértices, onde $||$ denota concatenação, então pega um produto escalar dela e um vetor de peso aprendível $\vec{a}^{(l)}$, e aplica a ativção $LeakyReLU$ no final. A equação (3) aplica um softmax para normalizar as pontuações de atenção nas bordas de entrada de cada vértice. A equação (4) é semelhante ao GCN. As incorporações de vizinhos são agregados, dimensionados pelas pontuações de atenção.

2.5. GIN

Um dos desafios é que grafos podem ser combinações de estruturas contínuas e discretas. Uma questão relevante é se é possível para as GNNs distinguir entre diferentes tipos de estruturas de grafos. Esta é uma questão clássica em teoria de grafos conhecida como problema de isomorfismo de grafos, com o objetivo de determinar se dois grafos são topologicamente equivalentes. Dois grafos isomórficos têm a mesma conectividade e diferem apenas por uma permutação de seus nós. Muitas vezes acontece de observar em experimentos que as GNNs se destacam em alguns conjuntos de dados, mas ao mesmo tempo as mesmas redes possuem um desempenho degradado em outros.

A Rede de Isomorfismo de Grafos (GIN) generaliza o teste de Weisfeiler-Lehman (WL) e, portanto, atinge um poder discriminativo máximo entre GNNs. [Xu et al. 2018] notaram que o teste WL tem uma semelhança notável com as GNNs de passagem de mensagens, ou seja, uma maneira de fazer convolução como operações em grafos. Em uma camada de passagem de mensagens, os recursos de cada vértices são atualizados agregando os recursos dos vizinhos. A escolha das operações de agregação e atualização é crucial: apenas as funções injetivas são equivalentes ao algoritmo WL. Algumas escolhas populares para agregadores usados na literatura, como máx ou média, são estritamente menos poderosos do que WL e não conseguem distinguir entre estruturas de grafo muito simples. [Xu et al. 2018] propôs a escolha das funções de agregação e atualização que tornam as redes neurais de passagem de mensagens equivalentes ao algoritmo WL, chamando-as de Redes de Isomorfismo de Grafos (GIN). Isso é tão poderoso quanto uma GNN de passagem de mensagens padrão pode obter. Porém, mais do que uma nova arquitetura, o principal impacto foi formular a questão da expressividade em um ambiente simples que pudesse ser relacionado a um problema clássico da teoria dos grafos.

3. Metodologia e Resultados

3.1. Conjunto de Dados

A linguagem utlizada para a implementação dos experimentos foi a linguagem de programação Python [Van Rossum and Drake 2009], pois oferece as ferramentas necessárias de apoio ao carregamento do conjunto de dados e a implementação tanto dos algoritmos em grafos quanto de aprendizagem de máquina. Primeiro, carregamos as três bases de dados contendo 2250 grafos de letras cada, com 15 rótulos possíveis de classificações. Especificamente, no conjunto de dados *Letters* utilizamos a métrica de acurácia e matriz de confusão para a avaliação. Os dados precisam utilizar a Precisão Média calculada como métrica de avaliação. Para a validação cruzada a seguinte divisão:

Table 1. Detalhes dos Experimentos.

Conjunto	Porcentagem
Dados para Treino	80%
Dados para validação	10%
Dados para Teste	10%

3.2. Grafos

A primeira etapa da implementação consistiu na conversão do conjunto de dados para um formato mais conveniente para visualização, no caso o formato da biblioteca *NetworkX*, o grafo da primeira letra do conjunto de treino pode ser visto na figura 1. onde pode ser observada a presença de 4 vértices representando os pontos finais das linhas da letra e 4 arestas representando, por sua vez, as linhas entre os pontos finais.

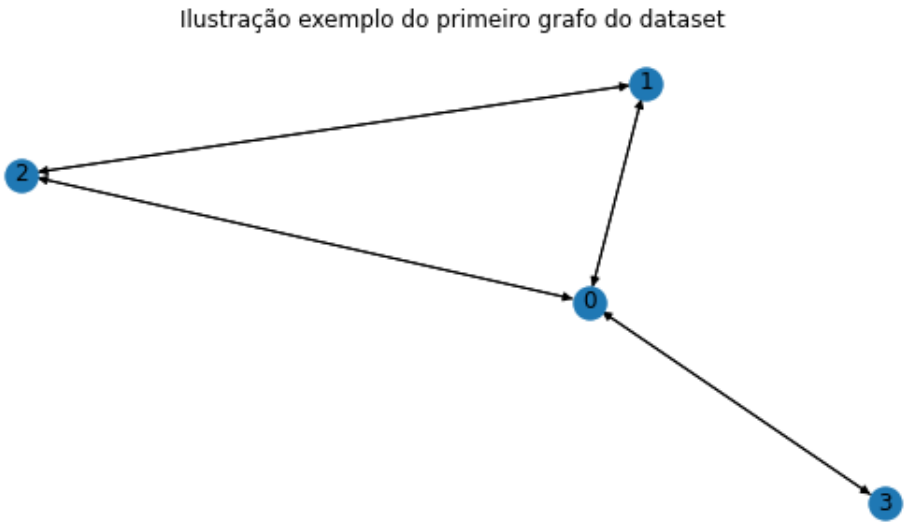

Figure 4. Exemplo de grafo do conjunto de dados letter-high (Representação da letra A).

3.3. GNNs

Para a implementação da GNN, foi utilizada a biblioteca *DeepSNAP*. A *DeepSNAP* é uma biblioteca Python para auxiliar no aprendizado profundo e eficiente de grafos. *DeepSNAP* oferece suporte para manipulação flexível de grafos, *pipeline* padrão, grafos heterogêneos e uma API (*Application Programming Interface* - Interface de Programação de Aplicações) simples. As arquiteturas escolhidas suportadas pela *DeepSNAP* foram a GCN, pois extende a CNN, a rede GAT por ser uma variação que possui camadas de autoatenção que aprimoram a GCN e, por fim, a rede GIN a qual se baseia no algoritmo WL e trás uma melhor diferenciação dos grafos.

3.4. Treinamento

O modelo de classificação é treinado para o conjunto de dados *Letters* nos três níveis de dificuldade, onde possuimos em cada um vetor de 2250 rótulos indicando se qual a letra representada pelo grafo. No caso da entrada X dos conjuntos de dados, foram utilizados além da topologia dos grafos os dois atributos espaciais de seus vértices, as *features* de não arestas não são consideradas. A divisão conforme detalhado anteriormente na tabela 1 consistiu em 80% dos grafos para o conjunto de treino, 10% para o conjunto de validação e 10% para o conjunto de testes. O classificador utilizado variou entre quatro tipos de GNNs, são elas a GCN, GAT e Redes de Isomorfismo em grafos (GIN), a sua configuração de hiperparâmetros utilizado é descrita na tabela 4.

Table 2. Parâmetros das GNNs.

Época	500
Tipo de Modelo	(GCN, GAT, GIN)
Batch Size	16
Dimensão Camadas Ocultas	30
Quantidade de Camadas Ocultas	2
Ativação de Camadas Ocultas	ReLU
Ativação da Camada de Saída	Softmax
Weight Decay	3e-4
Otimizador	Adam
Dropout	0.1
Taxa de Aprendizado	0.001

Os modelos variam em três tipos, todos utilizando a implementação da *Deep Snap*. A quantidade de épocas escolhida foi de 500, a partir das experimentações é o valor apropriado para convergência das redes. O tamanho de lote ou *Batch Size* foi definido como 16 a partir da busca pelo equilibrio entre desempenho de treino e acurácia. A taxa de aprendizado foi definida como 0.001. O número de camadas ocultas ficou fixado em 2 sendo 30 neurônios em cada camada. A ativação das camadas ocultas são ReLU e da camada de saída é a softmax. O otimizador utilizado foi o Adam [Kingma and Ba 2015] com o *Weight Decay* de valor 5e-4. O valor de *Dropout*, o qual é o efeito de algumas saídas de camadas serem ignoradas aleatoriamente é definido para GCN e GAT na proporção de 0.1. O *Dropout* é utilizado apenas para as redes GCN e GAT.

3.5. Resultados

Para execução dos testes do problema utilizamos a linguagem python em um ambiente Linux controlado com a distribuição Ubuntu versão 16.04 LST em um computador com 4GB de RAM e processador i5 onde foram implementados utilizando os algoritmos das bibliotecas *Torch Geometric*, *Networkx*, *Deep Snap* e *Scikit-learn*. Para os experimentos descritos no trabalho, utilizamos o servidor e ambiente *LASIDHUB* disponível pelo Laboratório de Sistemas Digitais (LASID) da Univesidade Estadual do Ceará em Fortaleza.

3.6. Conjunto de dados *leter-low*

O conjunto de dados *leter-low*, possui o nível mais fácil de classificação e mais baixo de distorção, ou seja, com letras bem desenhadas e definidas. Portanto, obviamente, consistiu no conjunto de dados com melhor acurácia nas três arquiteturas GNN utilizadas. Cada arquitetura de GNN foi testada três vezes para cada conjunto de dados e obtida a média de acurácia e *loss*. As tabelas 3, 4 e 5 apresentam os resultados detalhados.

Table 3. Resultados dos Experimentos para GCN.

Conjunto	**Acurácia**
Dados para Treino	97.55%
Dados para validação	93.30%
Dados para Teste	95.56%
Loss	0.1977

O resultado para GCN, descrito na tabela 3, possuiu a menor acurácia de teste para o conjunto de dados *leter-low*, pontuando 95.56%, resultado que reflete as limitações da Rede Convolucional em Grafos comparada as redes GAT e GIN. No entanto, a acurácia obtida ainda é considerada alta para o conjunto de dados.

Table 4. Resultados dos Experimentos para GAT.

Conjunto	**Acurácia**
Dados para Treino	97.66%
Dados para validação	94.64%
Dados para Teste	96.89%
Loss	0.1727

A GAT obteve resultado bastante similares a GCN, descritos na tabela 4, porém conseguiu superar a anterior em acurácia de dados para treino, teste e validação além de uma *loss* menor no treinamento. A rede neural atingiu uma acurácia de teste de 96.89% em classificar corretamente as 15 diferentes letras cursivas em nível fácil. Esses resultados também refletem a superação da GAT em relação a alguma limitações da GCN.

A arquitetura de GNN baseada em isomorfismo em grafos e WL obteve o segundo melhor resultado do conjunto de dados *leter-low*, descrito na tabela 5. A GIN atingiu 96% de acurácia no conjunto de teste. Apesar disso, a rede neural obteve a melhor *loss* e a melhor acurácia de treino o que indica a possibilidade de ter havido um *overfitting* em algum nível em seu treinamento, todavia ainda atingiu uma acurácia considerável para o problema e superou a GCN.

Table 5. Resultados dos Experimentos para GIN.

Conjunto	Acurácia
Dados para Treino	99.94%
Dados para validação	95.98%
Dados para Teste	96.00%
Loss	0.0047

3.7. Conjunto de dados *letter-med*

O conjunto de dados *leter-med*, possui um nível médio de classificação e mais alto que a *letter-low* de distorção, ou seja, com letras razoávelmente bem desenhadas e definidas. Portanto, consistiu no conjunto de dados com acurácia intermediária nas três arquiteturas GNN utilizadas. Cada arquitetura de GNN foi testada três vezes para cada conjunto de dados e obtida a média de acurácia e *loss*. As tabelas 6, 7 e 8 apresentam os resultados detalhados.

Table 6. Resultados dos Experimentos para GCN.

Conjunto	Acurácia
Dados para Treino	86.94%
Dados para validação	81.25%
Dados para Teste	82.67%
Loss	0.5967

O resultado para GCN, descrito na tabela 6, surpreendentemente possuiu a segunda melhor acurácia de teste para o conjunto de dados *leter-med*, pontuando 82.67%, mesmo resultado que a GIN. A acurácia obtida ainda é considerada razoável para o conjunto de dados e a *loss* obtida foi 0.5967.

Table 7. Resultados dos Experimentos para GAT.

Conjunto	Acurácia
Dados para Treino	85.38%
Dados para validação	74.11%
Dados para Teste	82.22%
Loss	0.6191

A GAT obteve resultado bastante similares a GCN, descritos na tabela 7, porém desta vez não conseguiu superar a anterior em acurácia de dados para treino, teste e validação além da *loss* no treinamento. A rede neural atingiu uma acurácia de teste de apenas 82.22% em classificar corretamente as 15 diferentes letras cursivas em nível médio.

A GIN atingiu 82.67% de acurácia, no conjunto de teste, como mostrado na tabela 8. Apesar disso, a rede neural obteve mais uma vez a melhor *loss* e a melhor acurácia de treino o que reforça a possibilidade de ter havido um *overfitting* em algum nível em seu treinamento, todavia ainda atingiu uma acurácia considerável para o problema e superou a GAT e obteve o mesmo desempenho da GCN.

Table 8. Resultados dos Experimentos para GIN.

Conjunto	Acurácia
Dados para Treino	99.78%
Dados para validação	82.59%
Dados para Teste	82.67%
Loss	0.0255

3.8. Conjunto de dados *letter-high*

O conjunto de dados *leter-high*, possui um nível difícil de classificação e mais alto que a *letter-low* e *letter-med* de distorção, ou seja, com letras mal desenhadas e definidas. Portanto, consistiu no conjunto de dados com acurácia mais baixa nas três arquiteturas GNN utilizadas. Cada arquitetura de GNN foi testada três vezes para cada conjunto de dados e obtida a média de acurácia e *loss*. As tabelas 9, 10 e 11 apresentam os resultados detalhados.

Table 9. Resultados dos Experimentos para GCN.

Conjunto	Acurácia
Dados para Treino	68.81%
Dados para validação	63.84%
Dados para Teste	62.22%
Loss	1.0979

O resultado para GCN, descrito na tabela 9, assim como na *letter-low* a acurácia de teste para o conjunto de dados *leter-high* foi a pior, pontuando 62.22%, resultado que mias uma vez reflete as limitações da Rede Convolucional em Grafos comparada as redes GAT e GIN. A acurácia obtida ainda é considerada razoável para o conjunto de dados e a *loss* obtida foi 1.0979 também a pior dentre as três.

Table 10. Resultados dos Experimentos para GAT.

Conjunto	Acurácia
Dados para Treino	72.04%
Dados para validação	66.52%
Dados para Teste	68.89%
Loss	1.0037

A GAT obteve resultado bastante similares a GCN mais uma vez, descritos na tabela 10, porém conseguiu superar a anterior em acurácia de dados para treino, teste e validação além de uma *loss* menor no treinamento. A rede neural atingiu uma acurácia de teste de 68.89% em classificar corretamente as 15 diferentes letras cursivas em nível difícil. Esses resultados também refletem a superação da GAT em relação a alguma limitações da GCN.

Por fim, a GIN atingiu 72.04% de acurácia, no conjunto de teste, como mostrado na tabela 11. Apesar disso, a rede neural obteve mais uma vez a melhor *loss* e a melhor acurácia de treino e também atingiu uma acurácia razoável para o problema superando a GAT e a GCN em todas as métricas mais uma vez.

Table 11. Resultados dos Experimentos para GIN.

Conjunto	Acurácia
Dados para Treino	98.16%
Dados para validação	69.20%
Dados para Teste	72.00%
Loss	0.1166

4. Conclusão

Redes Neurais de Grafos (GNNs) tem crescido em popularidade na comunidade de aprendizagem de máquina e tem sido aplicadas em diversos problemas incluíndo problemas de processamento de imagem, visão computacional e em particular análise e reconhecimento de imagens de documentos, generalizando o conceito de Rede Neural Convolucional. Este trabalho apresentou uma solução para a realização da classificação de grafos do conjunto de dados *Letters*. O conjunto de dados consiste em 15 classes. Cada classe representa uma letra maiúscula. O conjunto de dados consiste nas letras A, E, F, H, I, K, L, M, N, T, V, W, X, Y, Z desenhadas manualmente. Cada vértice é rotulado com um atributo bidimensional sendo sua posição X e Y em relação a um sistema de coordenadas de referência. Foram utilizadas três diferentes arquiteturas de GNNs e testadas nos três níveis de dificuldade e distorção das letras do conjunto de dados, *letter-low*, *letter-med* e *letter-high*.

Os experimentos mostraram que, para o conjunto de dados de treino as acurácias foram próximas de 100% no nível mais fácil, enquanto para os conjuntos de nível médio e difícil obtivemos resultados razoáveis. Em resumo, os resultados foram promissores, apresentando um estudo empírico acerca da utilização de diferentes arquiteturas GNN para classificação de grafos variando em diferentes níveis de dificultade da tarefa. Demonstrou-se que, apesar de utilizar classificadores simples e apenas utilizando a estrutura do grafo e os atributos de vértices, os algoritmos obtiveram desempenho relevante ao classificar as letras cursivas e semelhante as CNNs que utilizam as próprias imagens.

References

Conte, D., Foggia, P., Sansone, C., and Vento, M. (2004). Thirty years of graph matching in pattern recognition. *International journal of pattern recognition and artificial intelligence*, 18(03):265–298.

Defferrard, M., Bresson, X., and Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29:3844–3852.

Foggia, P., Percannella, G., and Vento, M. (2014). Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(01):1450001.

Kajla, N. I., Missen, M. M. S., Luqman, M. M., and Coustaty, M. (2021). Graph neural networks using local descriptions in attributed graphs: An application to symbol recognition and hand written character recognition. *IEEE Access*, 9:99103–99111.

Kingma, D. P. and Ba, J. L. (2015). Adam : A method for stochastic optimization.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. (2015). Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

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

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2018). How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.