|  |
| --- |
| **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| **ĐỒ ÁN TỐT NGHIỆP** |
| NGÀNH HỆ THỐNG THÔNG TIN |
| **TÊN ĐỀ TÀI: PHÂN TÍCH VÀ TRÍCH RÚT THÔNG TIN TỪ HÓA ĐƠN VỚI GRAPH CONVOLUTION NETWORK** |
|  |
|  |
| **GVHD**  : ThS. Trần Thanh Hùng |
| **Sinh viên thực hiện** : Nguyễn Bá Đông |
| **Mã sinh viên** : 2019605704  **Lớp**  : HTTT2 – K14 |
|  |
|  |
|  |
| Hà Nội – Năm 2024 |

# LỜI CẢM ƠN

Trước tiên, em xin gửi lời cảm ơn đến ThS. Trần Thanh Hùng đã nhiệt tình hướng dẫn và giúp đỡ em trong thời gian thực tập tốt nghiệp hoàn thành đề tài “Phân tích và trích rút thông tin từ hóa đơn với Graph Convolution Network”. Cảm ơn thầy đã hướng dẫn, góp ý cho em về kiến thức phân tích và trích rút đặc trưng, những kiến thức thực tế và ứng dụng cao.

Đặc biệt trong quá trình thực hiện đề tài thực tập tốt nghiệp, thầy đã cho em có cách phân tích và xây dựng model và triển khai hệ thống đáp ứng nhu cầu thiết yếu cho bài toán của đề tài. Với những buổi thảo luận đầy tâm huyết, em vừa được học kiến thức, vừa sửa được những lỗi mắc phải trong khi thực hiện đề tài. Vì vậy em tăng thêm khả năng phân tích và tư duy để xây dựng mô hình chuẩn .

Cuốn đồ án này được hoàn thành theo đúng thời gian quy định của nhà trường cũng như của khoa không chỉ là sự nỗ lực của em mà còn sự giúp đỡ, chỉ bảo của thầy hướng dẫn.

Trong quá trình thực tập, cũng như là trong quá trình làm bài báo cáo đồ án, khó tránh khỏi sai sót, rất mong các thầy, cô bỏ qua. Đồng thời do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên bài báo cáo không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp thầy, cô để em học thêm được nhiều kinh nghiệm và sẽ hoàn thành tốt nhất có thể!

Em xin chân thành cảm ơn!

**MỤC LỤC**

[**LỜI CẢM ƠN 2**](#_Toc156728983)

[**DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT 4**](#_Toc156728984)

[**DANH MỤC CÁC HÌNH VẼ, ĐỒ THỊ 4**](#_Toc156728985)

[**MỞ ĐẦU 6**](#_Toc156728986)

[**1. Lý do chọn đề tài 6**](#_Toc156728987)

[**2. Phương pháp nghiên cứu 6**](#_Toc156728988)

[**3. Mục tiêu nghiên cứu 6**](#_Toc156728989)

[**4. Nội dung nghiên cứu 6**](#_Toc156728990)

[**5. Những kiến thức cần áp dụng 7**](#_Toc156728991)

[**CHƯƠNG 1: GIỚI THIỆU VỀ GCN 7**](#_Toc156728992)

[**1.1. GCN là gì 7**](#_Toc156728993)

[**1.2. Cách hoạt động của một mô hình GCN 13**](#_Toc156728994)

[**1.3. Những kiến thức cần liên quan cần thiết 27**](#_Toc156728995)

[**CHƯƠNG 2 :TRÍCH RÚT THÔNG TIN TỪ HÓA ĐƠN 37**](#_Toc156728996)

[**2.1. Lý thuyết trích rút thông tin 37**](#_Toc156728997)

[**2.2. Huấn luyên mô hình với GCN 49**](#_Toc156728998)

[**KẾT LUẬN 55**](#_Toc156728999)

[**TÀI LIỆU THAM KHẢO 56**](#_Toc156729000)

# DANH MỤC CÁC KÝ HIỆU, CÁC CHỮ VIẾT TẮT

|  |  |  |
| --- | --- | --- |
| **STT** | **Từ viết tắt** | **Ý nghĩa** |
| 1 | GCN | Graph convolution Network |
| 2 | CNN | Convolution Neural network |
| 3 | AI | artificial intelligence |
| 4 | Invoice | Hóa Đơn |
| 5 | ML | Machine learning |
| 6 | DL | Deep learning |

# MỞ ĐẦU

## **Lý do chọn đề tài**

Ngày nay, Trí tuệ nhân tạo ngày càng phát triển để cuộc sống con người thêm nhiều tiện ích , đổi mới . vì vậy ngày hôm nay tôi xin được mang đến một sản phẩm của TTNT đó là GCN bởi rất nhiều lý do quan trọng .

Trong đó :

* GCN mở ra một cánh cửa rộng lớn về kiến thức trong lĩnh vực TTNT , và học máy đặc biệt là trong việc xử lý đồ thị . GCN không chỉ là một phương tiện mạnh mẽ để mô hình hóa mối quan hệ phức tạp giữa các đối tượng , mà còn có tính xử lý ngôn ngữ tự nhiên phân loại hình ảnh và phần tính mạng xã hội .
* GCN thách thức và kích thích sự sáng tạo trong nghiên cứu việc nghiên cứu GCN đưa ra rất nhiều vấn đề về hấp dẫn như tối ưu hóa hiệu suất mô hình đồ thị lớn , giải quyết vấn đề độ phức tạp và tích hợp GCN và các phương pháp khác . Qua quá trình này , tôi có cơ hội không chỉ nắm vững kiến thức chuyên sau về GCn mà còn đóng góp vào sự phát triển lĩnh vực này .
* GCN mang lại giá trị thực tế cao và khả năng ứng dụng trong nhiều lĩnh vực của cuộc sống .Việc áp dụng GCN vào các ứng dụng thực tế như phân loại tin tức , tìm kiếm thông tin , hay phát hiện ra gian lận mang lại giải pháp hiệu quả và có tính ảnh hưởng tích cực đến cộng đồng .
* Tóm lại , việc lựa chọn đề tại GCN không chỉ là 1 sự kết hợp hợp lý giữa quan tâm cá nhân và tiềm năng nghiên cứu , mà còn mở ra một thế giới đầy hứa hẹn và thách thức sáng tạo và ứng dụng sâu sắc .

## **2. Phương pháp nghiên cứu**

Khảo sát trên tình hình AI của Việt Nam nói chung và Hà Nội nói riêng thì tính ứng dụng của GCN chưa quá cao , một số công ty lớn như VIETTEL , VIN , SUN \* ….. cũng đã trong quá trình nghiên cứu và đưa ra sản phẩm của họ .

## **3.** **Mục tiêu nghiên cứu**

Áp dụng mô hình học máy vào đối tượng dữ liệu cụ thể ở đây là invoice .

GCN được thiết kế để hiểu cấu trúc đồ thị và khả năng tương tác các đỉnh (nodes)trong đồ thị

## **4. Nội dung nghiên cứu**

* **Khảo sát hệ thống:**

Khảo sát các tài liệu của nhiều trường đại học tại nước ngoài cũng như tại Việt Nam trên Internet

## **5. Những kiến thức cần áp dụng**

* Để thiết kế được hệ thống em đã sử dụng những kiến thức đã học vào project này:
* Ngôn ngữ lập trình: Python
* Công cụ: Visual studio code, Pycharm , Google Drive , Google Colab .

# CHƯƠNG 1: GIỚI THIỆU VỀ GCN

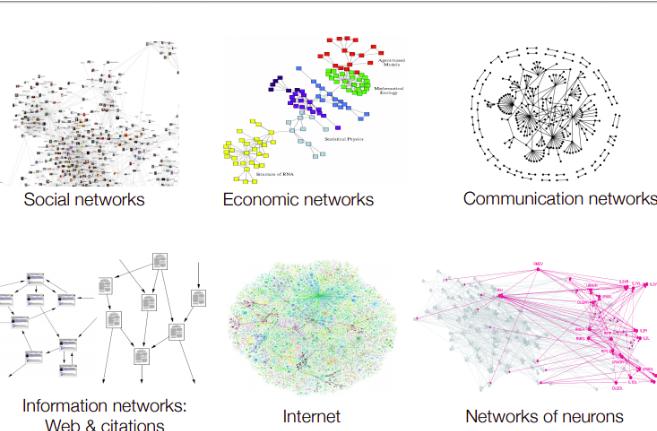
## **GCN là gì**

GCN là một mô hình học máy thuộc MC được thiết kế để ví dụ để xử lý dữ liệu đồ thị .Trong khi các học máy thông thường xử lý dữ liệu đồ thị dưới dạng ma trận , GCN đặc biệt làm việc trên dữ liệu có cấu trúc đồ thị , như mạng xã hội , mạng tương tác hay bất kỳ mối quan hệ thành phần cụ thể trong bài toán này là bài toán trích xuất thông tin từ hóa đơn .

GCN áp dụng cơ chế convolution lên đồ thị mà ban đầu được phát triển cho xử lý ảnh . Tuy nhiên khi được áp dụng lên đồ thị , Convolution truyền thống không tích hợp đo đồ thì đều không có cấu trúc như hình ảnh

Một trong những đặc điểm quan trọng của GCN là khả năng cập nhập thông tin của mỗi đỉnh dưa vào thông tin của các đỉnh kề và các liên kết giữa chúng . Điều này giúp GCN hiểu được cấu trúc và mối quan hệ đồ thị .

Các ứng dụng của GCN bao gồm graph classifier ( phân loại đồ thị ), Link Predict ( Dự đoán liên kết giữa các đỉnh ) , node classifier ( phân loại node) ,….



Hình ảnh 1 : Những ví dụ cụ thể của GCN

Chắc hẳn đối với những bạn đã tiếp xúc với ML , DL không còn xa lạ gì với những bài toán điển hình như :

- Conputer Vision : Với dữ liệu dạng ảnh , video . Các bài toán như Image Classification , Object Detection , Image Segmentation ,….

- NLP : với dữ liệu chính là word , thường được biểu diễn dưới dạng sequence . các bài toán như Neural Machine Translation , Text Classification , Text Summarization , Topic Modeling ,…

- Sound : Dữ liệu chủ yếu nguồn âm thanh , dễ dàng biểu diễn dưới dạng 1D hay 2D . các bài toán như : TTS , STT , Sound Recognition

- Và còn rất nhiều dữ liệu khác nữa

Tuy nhiên , cũng có rất nhiều bài toán và kiểu dữ liệu khác , chúng ta khó có thể biểu diễn dữ liệu dưới dạng 1 D hay 2D như thông thường , hay các dạng dữ liệu Non – Euclidean . Ví dụ như :

- Dữ liệu mạng xã hội : Social network

- Liên kết mạng internet

- Tương tác giữa các phần tử , nguyên tử protein – protein interaction (PPI)

- …..

Với những dữ liệu đặc thù như vậy , cũng yêu cầu những mục tiêu khác nhau của từng yêu cầu của bài toán ví dụ như :

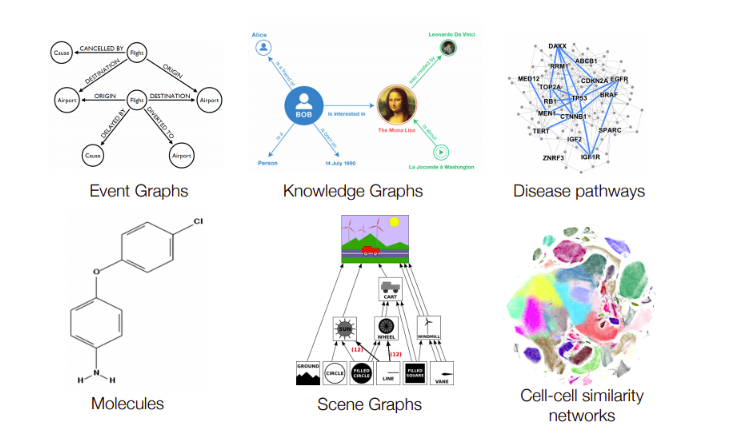
- Phân tích dữ liệu mạng xã hội để nắm bắt được các xu hướng cộng đồng hiện tại , các nhóm đối tượng khách hàng .

- Các gợi ý kết bạn , follow các page trên nền tảng quen thuộc như Facebook , Zalo ,…

- Phân tích sự tương tác ở cấp độ phân tử , nguyên tử nhằm mục đích phục vụ vho các vấn đề về sinh học , ví dụ phân tích tác dụng phụ của thuốc .

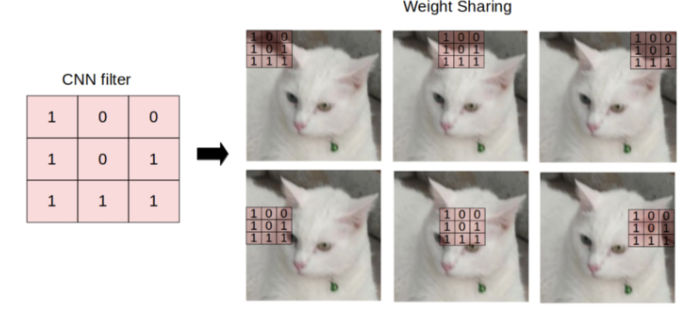
- Xây dựng các hệ thống gợi ý sản phẩm cho các website thương mại điện tử từ dữ liệu của người dùng

- …



Hình ảnh 2 : những ví dụ chi tiết của từng bài toán

Một ví dụ điển hình trong Computer Vision , giả dụ với bài toán phân loại ảnh chó và mèo (Cat and dog classifier )



Hình ảnh 3 : Mô tả quá trình học của 1 kernel cho ảnh mèo

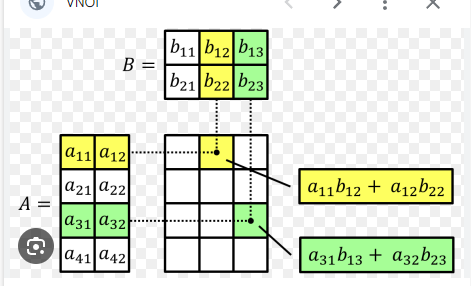
Một mô hình đã rất phổ biến là CNN đã được sử dụng . CNN gồm các kernel (hoặc filter) có tác dụng giống như 1 cửa sổ trượt từ trái sang phải , từ trên xuống dưới nhằm mục đích trích đặc trưng của từng bức ảnh . Các kernel thực chất chính là trọng số của mô hình CNN sẽ được cập nhập dần trong quá trình training với backpropagation(\*) . Mỗi kernel sẽ có một giá trị riêng , học được các feature khác nhau từ ảnh , có kernel thì làm nổi bật các đường biên của đối tượng , có kernel làm nổi bật rõ từng vùng riêng biệt của bức ảnh ….



Hình ảnh 4 : các kernel mô tả trong bài tóan CNN

Cách hoạt động của từng kernel :

* Thường ở dạng ma trận trọng số nhỏ được thiết kế qua từng pixel của hình ảnh (đối với lớp convolution ) hoặc các đầu vào (đối với các Convolution 1d , 3D )thực hiện phép tích chập bằng cách phép nhân ma trận với công thức toán :

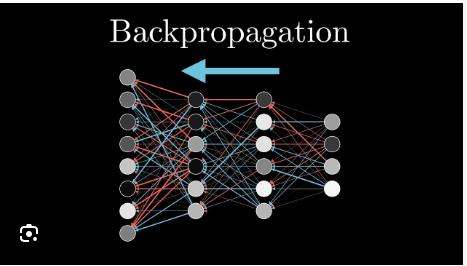


Hình ảnh 5 : mô tả phép nhân ma trận

* Các lớp convolution thường đi kèm với các tham số như bước nhảy (stride ) hay kích thước cửa sổ (window size) . Stride là khoảng cách giữa các vị trí liên tiếp khi kernel di chuyển qua đầu vào . Các tham số này kiểm soát được kích thước của feature map đầu ra .
* Chia sẻ trọng số của : Một trong những ưu điểm quan trọng của convolution là khả nằng chia sẻ trọng số . Điều này có nghĩa là cùng 1 kernel được sử dụng cho tất cả các vùng không gian của đầu vào . Điều này giúp giảm lượng tham số cần học , giảm bớt nguy cơ quá mức fitting dữ liệu .
* Học đặc trưng : kernels trong CNN được học thông qua các quá trình đào tạo mô hình . cấc giá trị trong kernel được điều chỉnh để mô hình học được các đặc trưng quan trọng từ dữ liệu đầu vào .
* Sau phép tích chập một activation function(\*) (hàm kích hoạt ) thường được áp dụng để thêm hàm phi tuyến tính và không gian biểu diễn đa dạng cho mô hình .(\*) Cụ thể hàm kích hoạt hộng như thế nào sẽ được tôi đề cập ở phần chương 3.

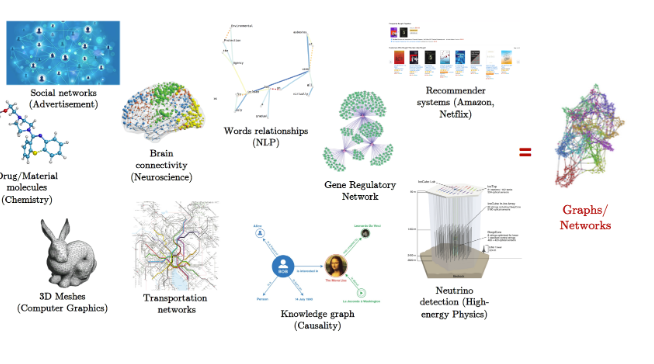
(\*) Backpropagation : lan truyền ngược là quá trình cập nhập lại trọng số của mạng nerural để giảm thiểu sai số . Nó bắt đầu từ lớp đầu ra và di chuyển ngược về lớp đầu vào của mạng .

Dựa trên độ lớn chênh lệch của gradient desent của hàm mất mát của trong số(loss funtion ) bằng cách sử dụng các thuật toán tối ưu hóa như gradient desent hoặc các biến thể khác .Chi tiết tôi xin được nêu trong phần sau .



Hình ảnh 6 : Mô tả hoạt động của backpropagation

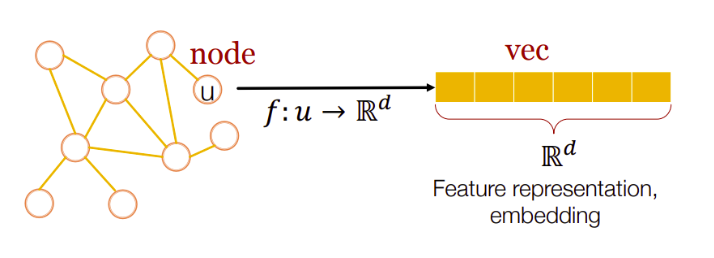
Quay trở lại với GCN , trong thực tế có rất nhiều dạng dữ liệu non-euclidean data . Nếu như trong bài toán sử lý ảnh dùng CCN , các filter hoạt động tổng hợp giá trị của các điểm pixel lận cận nhau thì trong GCN cũng vậy . cũng là tập hợp của nút lân cận nhau .Tuy nhiên các nút trên đồ thị là không có thứ tự và số degree (hay số cạnh liên kết của từng nút ) là không giống nhau .



Hình ảnh 7 : Hình ảnh mô tả GCN trong thực tế

Tương tự như các bài toán khác về ML , DL , cách biểu diễn đơn giản nhất cho các nút và cạnh trên đồ thị là biểu diễn các nút / cạnh đó là vector embedding(\*) , tiếp đó sử dụng các embedding này làm cho các downstream task khác như node classification , graph clustering

(\*)Vector embedding là một phương pháp biểu diễn đối tượng nào đó trong không gian vector .



Hình ảnh 8 : chuyển hóa từ dạng đồ thị sang vector sang embedding vector

Với ví dụ với 1 đồ thị như hình dưới , các nút trên đồ thị được ánh xạ sang 1 không gian vector mới qua 1 hàm f(x) . 2 nút u và v trên đồ thị được coi là tương đồng với nhau nếu khoảng cách giữa 2 vector embedding z (u) và z(v) trên không gian mới là nhỏ .

## **1.2. Cách hoạt động của một mô hình GCN**

- Bước 1 : Chuẩn bị dữ liệu

Việc chuẩn bị dữ liệu được coi là bước tối quan trọng trong quá trình xây dựng mô hình máy học . Trong quá trình này bao gồm nhiều bước chính để đảm bảo dữ liệu vào đầu vào cho mô hình là đủ chất lượng và đại diện cho thực tế .

Đầu tiên , cần thu thập các dữ liệu từ nguồn liên quan đến mục tiêu của mô hình . các nguồn có thể là cơ sở dữ liệu , tệp tin văn bản , hình ảnh , âm thanh hoặc bất kỳ nguồn nào liên quan đến bài toán cụ thể . Việc này đôi khi đòi hỏi sự hiểu biết sâu sắc về lĩnh vực cụ thể và mục tiêu cho mô hình . Sau khi dữ liệu đã được thu thập bước tiếp theo là dữ lý dữ liệu . Điều nàu có thể bao gồm làm sạch dữ liệu để có thể loại bỏ các giá trị còn thiếu , nhiễu , hoặc làm nổi bật những đặc trưng quan trọng . Các biến đổi như chuẩn hóa dữ liệu cũng có thể thực hiện được để đảm bảo rằng các biến có giá trị tương tương và dễ dàng so sánh . Tiếp theo , cần phải chia dữ liệu thành các tập train , test , validation . Tệp huấn luyện nào được sử dụng để đào tạo mô hình , trong khi tập test , validation để đánh giá hiệu suất của mô hình trên dữ liệu mà nó chưa từng thấy . Cuối cùng việc xử lý dữ liệu các vấn đề không cân bằng dữ liệu , như việc lớp thiểu số hoặc sử dụng kỹ thuật tăng cường dữ liệu , cũng là một phần quan trọng trong quá trình xử lý dữ liệu . Tóm lại , quá trình chuẩn bị dữ liệu đóng vai trò quan trọng trong việc đảm bảo mô hình máy học hiệu quả và chính xác trên thực tế .

* Bước 2 : Lan truyền xuôi (Forward Propagation ) :

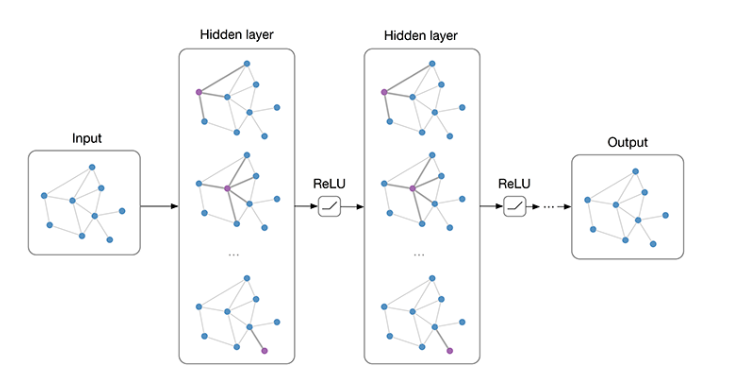
Đầu tiên dữ liệu nào được đưa vào mạng lưới neural network tính toán giá trị đầu vào và đầu ra tương ứng tính toán tại các đơn vị neural và truyền chúng qua hàm kích hoạt .

Cụ thể trong bài toán GCN : quá trình lan truyền xuôi (Forward Propagation ) giống với các học máy khác , nhưng được áp dụng với đồ thị thay vì dữ liệu có mạng dạng dưới hay văn bản thì GCN lại biểu thị dưới dạng đồ thị .

Ví dụ cụ thể : G = (V, E) với :

* G là đồ thị cấu thành
* V là tập hợp các đồ thị (Vertices / node)
* E là tập hợp các cạnh kết nối của các nút đồ thị (edge)
* e­­­­­­­ij = (­­­vi , vj ) thuộc E là biều diễn các ạnh e thuộc E từ nút vi tới nút vj của đồ thị
* N(v) = u thuộc V( v , u ) thuộc E là các node kề u ( node láng giềng có chung) với node v .

Đầu vào của mô hình là 1 biểu đồ , thường là các ma trận kề (adjacency matrix ) và ma trận đặc trưng (feature matrix ) . Ma trận kề mô tả các mối quan hệ liên kết các đỉnh



Hình ảnh 9 : Mô tả hoạt động của mô hình GCN .

Các thuật ngữ này là tương đương nhau khi biểu diễn các bài toán về đồ thị :

* G( V , E ) – System (Object , Interaction )
* System – Network , Graph
* Object – Node , vertical
* Interaction – Link , Edges

Adjacecy matrix (ma trận kề ) A là một ma trận vuông kích thước n xn (với n là tổng số node của đồ thị )

* Aij = 1 nếu eij thuộc E
* Aij = 0 nếu eij  không thuộc E
* Adjacency matrix (A) cũng được gọi là 1 weighted – matrix thể hiện trọng số của các cạnh đồ thị . Với hình minh họa Adjacency matrix bên trên thì các cạnh có trọng số như sau nhưng có thể thay đổi tùy bài toán và dữ liệu .

Degree matrix (ma trận bậc ) D là 1 ma trận đường chéo vuông nxn chưa thông tin bậc của mỗi đỉnh với

- 

- Chú ý rằng với đồ thị có hướng (direct matrix ) thì bậc của từng nút chỉ tính các cạnh nối có chiều tới nút đó .

Indentity matrix (ma trận đơn vị ) I là 1 ma trận có đường chéo nxn , với các đường chéo chính là 1 còn lại bằng 0

- Iij = 1 nếu i==j , ngược lại bằng 0

Laplacian matrix hay ma trận dẫn nạp L với L = D- A

Directed graph và undirect graph

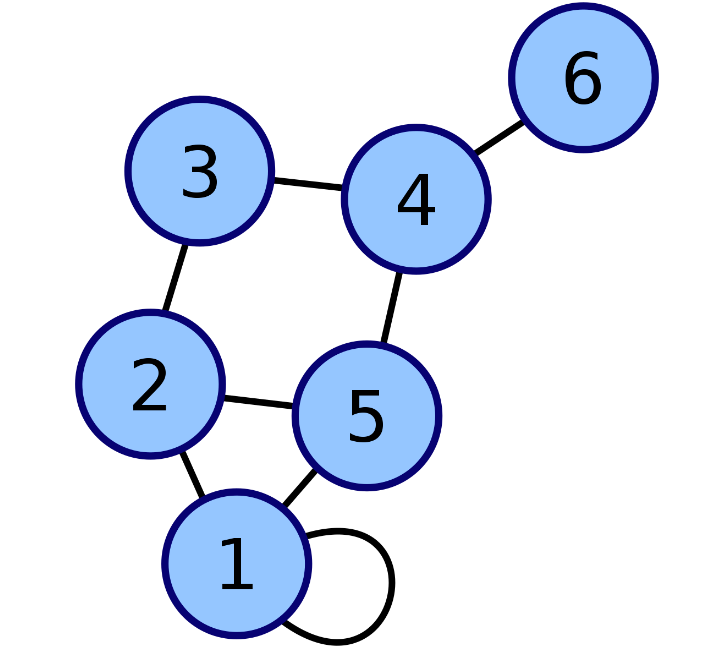
Undirect graph hay ma trận vô hướng , khi cạnh nối giữa 2 đỉnh I và j là như nhau hay

Eij

Direct graph hay ma trận có hướng , có chiều xác định từ đỉnh vi đến vj và tồn tại cạnh

Liên kết eij

Selp-loop : node gần cạnh nối đến chính nó .



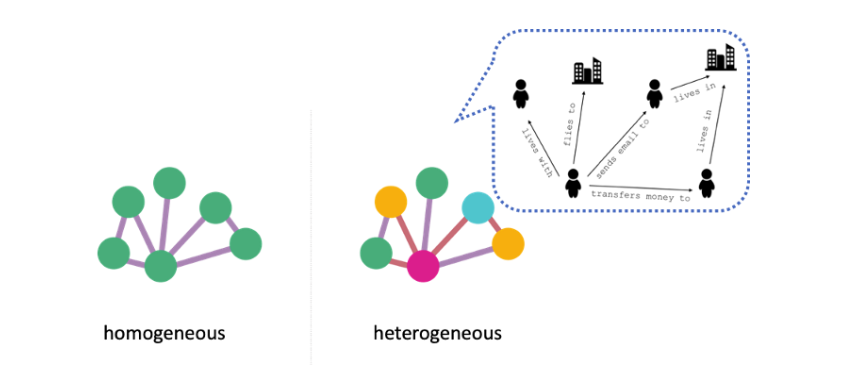
Hình ảnh 10 : Mô tả 1 graph

Multi- Graph : là các đồ thị liên kết các đỉnh kề nhau có nhiều hơn 1 cạnh nối liên kết giữa chúng .

Tồn tại 2 loại graph thông thường đó là Heterogenerous và homogenneous graph

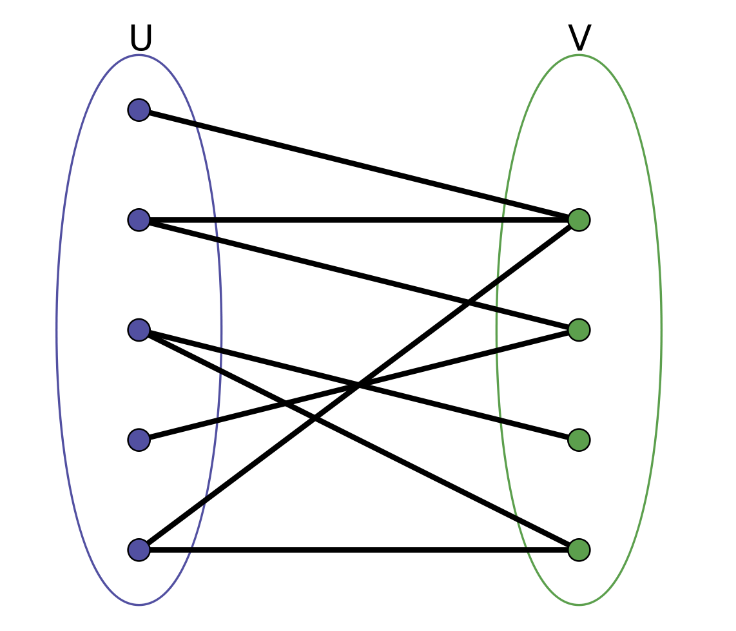
- Homogeneous graph : Có thể hiểu là các đồ thị đơn , khi các nút và các cạnh của đồ thị chỉ biểu diễn 1 duy nhất đối tượng , ví dụ nút biểu diễn con người và cạnh những người bạn bè khác nhau .

- Heterogenerous graph : Hay multi- model graph , là các đồ thị có nút và các cạnh thể hiện nhiều mối quan hệ liên kết giữa các đối tượng với nhau , liên quan mật thiết tới 1 khái niệm rộng hơn là knowledge graph .



Hình ảnh 11 : mô tả 2 loại đồ thị homogenous , heterogeneous

Bi-partite graph (đồ thị 2 phía ) : là 1 đồ thị khá đặc biệt có thể phân các đỉnh thành 1 tập không giao nhau , tức có 2 cạnh nối 2 đỉnh bất kỳ thành 1 tập .



Hình ảnh 12 : Hình ảnh mô tả Bi-partite graph

Mỗi nút trên đồ thị , tùy bài toán sẽ có nhãn lable tương ứng với từng nút .

Ví dụ bạn đang xây dựng 1 bài toàn phân loại thể loại của paper dựa trên các citation giữa các paper với nhau . các nút là các paper , các cạnh thể hiện các paper liên kết (citation )với nhau thì nhãn của nút có thể là : Conputer vision , NLP , Reinforcement Learning (học tăng cường ) …

Node feature : Các nút của đồ thị có thể bao gồm các đặc trưng (feature ) riêng của từng nút (node) . Các đặc tính này hoàn toàn có thể trích rút thông tin của các nút đó . Ví dụ :

* Một bài toán social network , các nút trong mạnh thể hiện 1 người , node feature là ác đặc trưng như tuổi giới tính , công việc , trình độ , học vấn ,…
* Với 1 bài toán về phân loại topic cho document , các nút của đồ thị là các document , node feature có thể biểu diễn đơn giản bằng 1 binary vector với số chiều bằng số lượng từ trong từ điển ( hay vocab ) . Ví dụ là 50000 từ , với giá trị 1 thể hiện từ có xuất hiện trong từ điển và ngược lại . Hoặc có thể biểu diễn đơn giản hơn bằng cách sử dụng language model hoặc word embedding để sinh được 1 feature vector phù hợp ứng với từng đoạn . ví dụ mình có thể sử dụng 1 mô hình doc2vec để ánh xạ 1 document thành 1 vector 300D

Với ma trận adjacency matrix (A) bên trên , 1 số điểm dễ nhận thấy rằng :

* Ma trận A cũng dễ dàng thể hiện trọng số của các cạnh đồ thị
* Đa số các trường hợp là 1 ma trận thưa ( hay sparse matrix ) , hay tỉ lệ 1 /0 trên toàn đồ thị là rất bé .
* Trong nhiều bài toán , để thuận tiện người việc sử dụng adjacency matrix thì người ta cũng sử dụng adjacency list để biểu thị cạnh nối giữa 2 đỉnh cho thuận tiện .

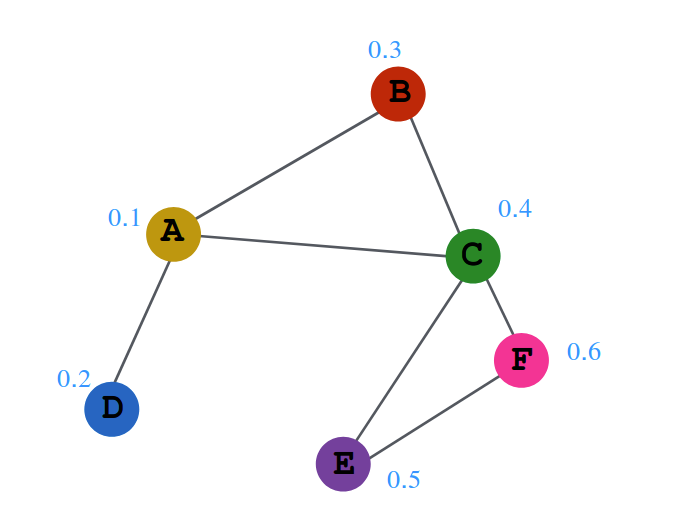
Sau khi đi 1 vòng về những thông tin về những thông tin của từng node và các ứng dụng của nó vào đồi sống thực tế ta quay trở lại với công thức tính Convolution cho mạng lưới này .

Sau khi chúng ta có đầy đủ dữ liệu ma trận đồ thị . ta định nghĩa 1 mô hình GCN như sau

* 1 đồ thị G =(V , E)
* 1 ma trận adjacency matrix A nxn
* X thuộc Rnxd là ma trận feature ứng với các node của đồ thị , với n là tổng số lượng node và d là số chiều của node feature . Node embedding như ta đề cập ở trên , liên quan đến các thông tin của node đó

Để dễ hình dung , chúng ta hãy làm 1 số ví dụ sau :

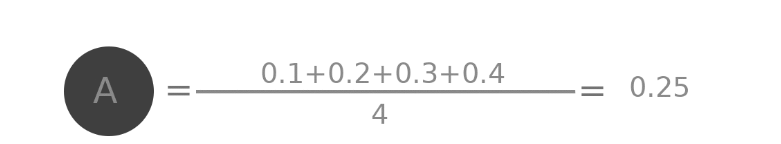
Với 1 đồ thị vô hướng như hình dưới .



Hình 13 : ví dụ về đồ thị vô hướng

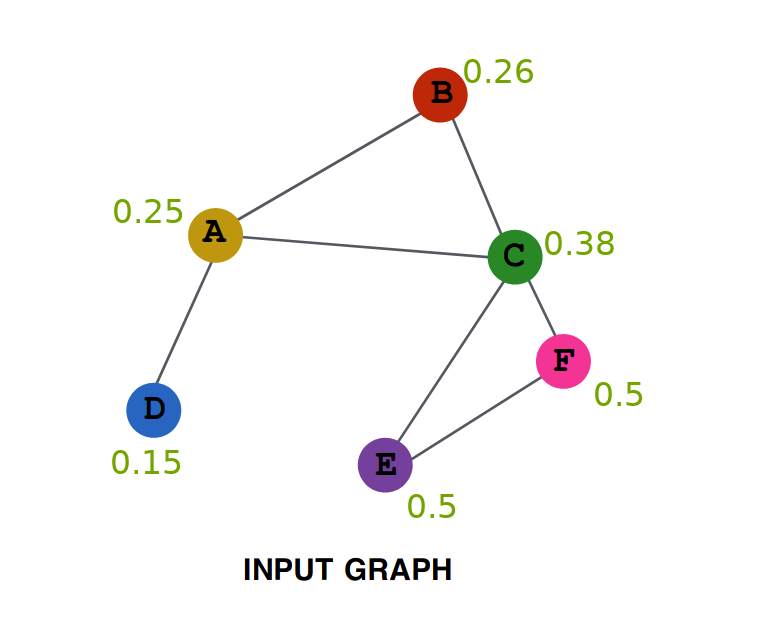
Đồ thị gồm 6 nút từ A -> F . Để đơn giản mỗi node sẽ được biểu diễn bằng 1 trọng số (hay ví dụ node embadding hiện tại của node a= [0,1])

Với giả định từ đầu rằng các nút lân cận nhau là gần giống nhau . Từ đó ta thực hiện 1 phép tính toán cơ bản để tổng hợp lại thông tin lân cận nút vi cụ thể , bằng các lấy giá trị trung bình các giá trị mới p , gắn giá trị của các nút lân cận + giá trị của nút vi  ta thu được giá trị mới của p , gắn giá trị mới p cho nút vi . Đối với các nút khác cũng thực hiện tương tự trong 1 step . Các step kể tiếp cũng tiếp tục thực hiện phép toán vừa nêu với tất cả các nút trong mạng . Sau step 1 , ta thu được các giá trị cho đồ thị sau :



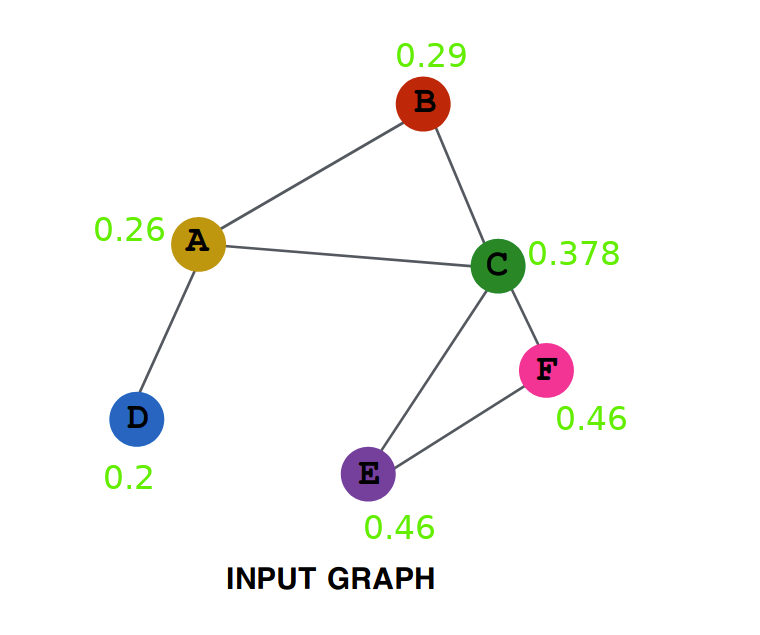
Hình 14 : Cách tính của node A

Sau khi tính toán xong ta cập nhập lại graph :



Hình 15 : Hình ảnh sau khi đã cập nhập trọng số

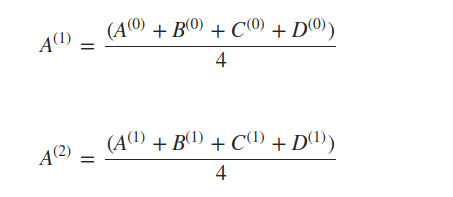
Thực hiện tương tự cho tất cả các node trong step 2



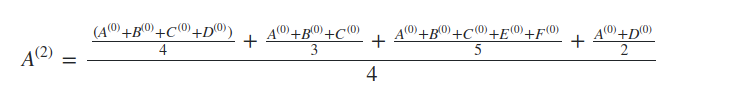
Hình ảnh 16: thực hiện cập nhập lại trọng số

Có thể thấy rằng , giá trị hiện tại của các node trong mạng điều chỉnh lại và san đều giữa các node , những node gần nhau sẽ có giá trị gần nhau hơn , những nút có liên kết giống nhau như nút E và F đều nhận giá trị giống nhau .

Nếu gọi A0 là giá trị nút A tại step 0 tương tự với các nút khác thì ta có công thức sau :

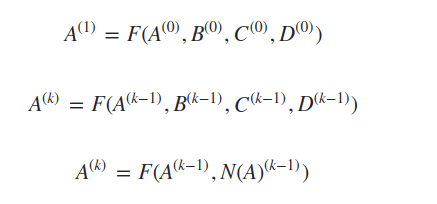


Bằng phép thế , ta có :



Có thể thấy rằng , với 2 step , giá trị hiện tại của nút A đã bị ảnh hưởng bởi các nút nằm trong 2 phạm vi bước nhảy từ nút A , bao gồm cả 5 nút còn lại . Gỉa sử nếu có 1 nút G nối độc lập với nút F chẳng hạn thì giá trị hiện tại của A chưa bị ảnh hưởng bởi nút G (vì 3 bước nhảy)

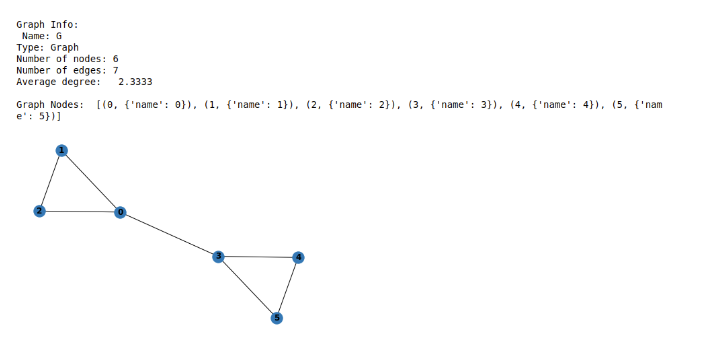
Với phép tính trung bình ( mean ) bên trên , ta hoàn toàn có thể thay thế bằng 1 số phép toán khác , được gọi là aggregation function (sẽ được đề cập kĩ hơn tại mô hình graphSage ) . Tóm lại , giá trị của A1 sẽ được biểu diễn gọn lại như sau :



Với N(v) là tập hợp các nút lân cận của nút v .

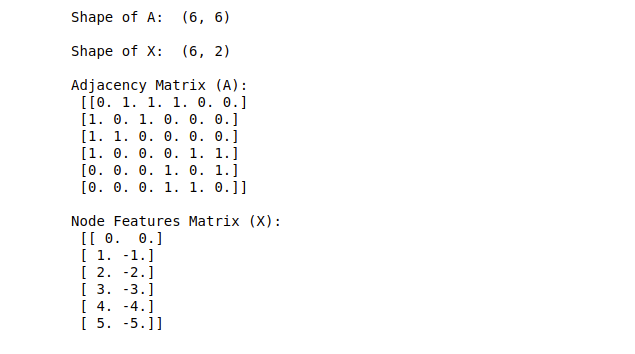
Một ví dụ cụ thể hơn để ta dễ hình dung :

Bài toán cho 1 đồ thị vô hướng gồm 6 nút và ma trận adj A như hình bên dưới



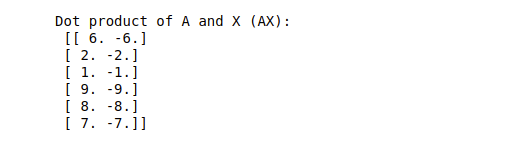
Hình 17: ví dụ về đồ thị vô hướng gồm 6 nút và 1 ma trận

Kèm theo đó là node feature xi với mọi x thuộc X ứng với từng nút trong đồ thị . ở đây , tôi định nghĩa 1 ma trận X như hình bên dưới



Hình ảnh 18 : Hình ảnh 1 ma trận theo định nghĩa và 1 ma trận đặc trưng

Và giờ hãy thực hiện phép nhân ma trận đơn giản AX , ta thu được



Có thể thấy rằng , giá trị ma trận 6x2 thu được hiện tại , với mỗi giá trị tại từng dòng là tổng tất cả các trọng số của các nút lân cận .

Qua 2 ví dụ kể trên ta có thể dễ dàng thấy rằng feature của từng nút ảnh hưởng bởi các giá trị lân cận của đồ thị .

Tiếp sau đó chúng ta thực hiện tích chập đồ thị hay còn gọi là convolution

1 hidden layer của GCN có thể được biểu diễn như sau : Hi = f(Hi-1 , A)

Trong đó :

- Hi là biểu diễn đầu ra của layer thứ i+1 , mỗi layer Hi tương ứng với 1 ma trận có kích thước N x Fi . VỚI Fi thể hiện số feature đầu ra của từng nút tại layer Hi

- H0­ =X có trọng số được khởi tạo luôn là node feature của từng nút

hàm f có thể biểu diễn đơn gian bằng 1 công thức sau :

Hi+1 = f( Hi, A ) = với

- W­I là ma trận trọng số ứng với layer thứ i

- *σ* là 1 hàm kích hoạt phi tuyến tính (activation function), ví dụ hàm ReLU

Nhưng với công thức hiện tại , có 2 điểm hạn chế như sau :

- Dễ thấy rằng , bằng việc cho nhân với ma trận A , với từng nút , giá trị mà các nút lân cận comtribute cho nút vi .ta có thể khắc phục điều đó bằng các cộng thêm identity matrix vào ma trận A tức A = A+I

Công thức ban đầu biến đổi thành



Trong Paper , tác giả sử dụng Symmetric Normalization , đổi công thức thành



Với H0 = X

* Bước 3 :Tính toán sai số (Loss function )

Với bài toán supervised learning , đầu ra bao gồm các feature embedding zv ta có thể thiết kế thêm các layer thêm các layer , đơn giản như 1 mạng lưới fully connected layer ở cuối , với số node đầu ra bằng số class , hàm loss function được sử dụng là cross entropy

Công thức của cross entropy giữa 2 phân bố xác xuất p và q là :



Trong đó :

- p(i) là xác suất thực tế của sự kiện i .

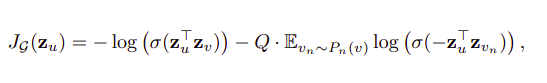
- q(i) là xác xuất dự đoán của sự kiện i bởi mô hình .

- log là hàm logarithm tự nhiên .

Nếu p và q giống nhau thì cross , entropy là 0 , tức mô hình dự đoán chính xác giống với thực tế , ngược lại nếu 2 phân bố khác biệt cross entropy sẽ là 1 giá trị dương .Trong bài toán phân loại nhiều lớp , cross entropy thường được tính cho từng lớp .

Với bài toán un – supervised learing , mặc dù không có label , nhưng với cách thiết kế từ mô hình , các node embedding thu được sau khi traning hoàn toàn có thể sử dụng ở các downstream task .

Từ đó , hàm loss function được định nghĩa cho bài toán un-supervised learning như sau .



Trong đó u và v là 2 nút lân cận nhau , lân cận trong số bước đi ngẫu nhiên cố định đã được xác định từ trước . Q là tập hợp các cặp negative sample , tức 2 nút không lân cận nhau .

Dễ thấy rằng , việc tính loss là dựa vào các nút lân cận , tùy số bước trong bước đi ngẫu nhiên nên giúp mô hình có khả năng tổng quát hóa đơn trên toàn bộ dữ liệu và khi áp dụng cho các unseen node , khác hoàn toàn với việc huấn luyện embedding cố định cho từng node như các mô hình node embedding cố định cho từng node như các mô hình node embedding như Deep walk hoặc Node2Vec . Đó chính là sự khác biệt lớn nhất giữa các mô hình tranductive learning (deepwalk , Node2Vec) với các mô hình inductive như graphSage .

- Bước 4 : Lan truyền ngược :

Lan truyền ngược là quá trình cập nhập trọng số của mạng neural để giảm thiểu sai số.Nó bắt đầu từ lớp đầu ra và di chuyển ngược về lớp đầu vào của mạng Dựa vào độ lớn của độ chệch (gradient desent )của hàm mất mát đã tính ở trêm thuật toán cập nhật trọng số bằng cách sử dụng một thuật toán tối ưu hóa như Gradient desent hoặc các biến thể nâng cao .

Cụ thể bạn có hàm mất mát đã có từ trước , và thực hiện quy tắc chuỗi (chain rule ) là một phần quan trọng của quá trình lan truyền ngược , giúp tính toán gradient desent của hàm mất mát được cập nhập trọng số của mô hình. Các quy tắc chain rule dựa trên các hàm phụ thuộc .

Cho 1 hàm hợp F(x) = f(g(x)) , gradient của F theo x có thể được tính bằng cách nhân gradient của f và g với gradient theo h theo x 

Áp dụng quy tắc chuỗi cho phép tính hợp đặc trưng .

Cho 1 hàm F(x) = AxXxW trong đó

- A là ma trận kề

- X là ma trận đặc trưng

- W là ma trận trọng số ta cần tính gradient desent .của F theo W và X

Sử dụng quy tắc chuỗi , gradient theo W và X có thể được dựa tính gradient của F theo A và X : 

Quy tắc chuỗi cho tính toán trọng số

- Sau khi tính toán đặc trưng mới bằng cách nhân với ma trận trọng số , ta cần tính gradient desent cuiar hàm mất mát theo trọng số để cập nhập chúng .

- Sử dụng quy tắc chuỗi , gradient theo trọng số có thể tính bằng cách nhân gradient của hàm mất mát theo đầu ra với gradient của đầu ra theo trọng số .

Sau đó tổng hợp và cập nhập trọng số .

- Khi đã tính được gradient của hàm mất mát theo các trọng số , tổng hợp chúng và sử dụng lại thuật toán tối ưu để cập nhập trọng số .

Quy tắc chuỗi trong GCN giúp chúng ta tính toán độ chênh của hàm mất mát theo trọng số 1 cách hiệu quả cho phép lan truyền ngược một cách chính xác nhanh chóng , từ đó cập nhập trọng số .

ASP.NET và SQL Server

## **1.3. Những kiến thức cần liên quan cần thiết**

**Những nhược điểm / Hạn chế của của mô hình GCN truyền thống và hướng cải thiện.**

Nhìn chung , Mô hình GCN trong paper [Semi-Supervised Classification with Graph Convolutional Networks - 2016](https://arxiv.org/abs/1609.02907) thiết kế khá đơn giản , tuy nhiên vẫn còn 1 vài hạn chế như sau

* Memory requirement : Trọng số của mô hình vẫn được cập nhập qua từng epoch , những với mỗi epoch được cập nhập theo full-batch gradient desent , không phải mini- batch gradient desent , tức việc cập nhập trên toàn bộ điểm dữ liệu cùng 1 lúc . Điều đó cũng hoàn toàn dễ hiểu vì trong công thức cập nhập ở bên trên , mô hình vừa phải giữ toàn bộ trọng số và ma trận adjacency matrix A . với 1 tập dữ liệu nhỏ như Cora Dataset (2708 paper – node và 5429 citation /edge ) thì không phải vấn đề nhưng với 1 tập dữ liệu lớn hơn với hàng triệu node và dense- adjacency matrix thì cách tiếp cận này hoàn toàn không phù hợp khi memory requirement là rất lớn .
* Directed edges and edge features : mô hình hiện tại về GCN được công bố tại paper đang được sử dụng thêm các yếu tố khác như edge feature ( adj matrix A hiện tại chỉ là binary matrix ) và directed graph (tức ma trận có hướng ) . Hướng xử lý trong paper đang bị giới hạn với undirected graph (ma trận vô hướng ) .
* Limiting assumption : việc công adjacency matrix A với identity matrix I để có thêm feature của chính nút đó . các làm này đang giả định rằng nút vi và các nút lân cận của vi đang contribute với vi là như sau , ta có thê nhân thêm 1 tham số lamda để điều chỉnh lại trọng số của nút vi : thực chất có thể là 1 trainable parameter , hiện trong paper fix cứng =1 và tác giả cũng có đề cập về thông số này .
* Transductive setting : Với những nút mới thêm vào đồ thị (kèm theo các liên kết mới ) , mô hình GCN có khả năng tổng quan hóa (generalization ) rất kém với những nút mới đó và yêu cầu cần retraining để cập nhập mô hình .

**GraphSage (an Inductive learning method )**

Trước tiên chúng ta cần phải hiểu về Inductive learning là gì đã .

Inductive learning (Học chủ động ) là một phương pháp học máy mà mô hình được đào tạo trên 1 tập dữ liệu huấn luyện và sau đó được sử dụng để dự đoán các dữ liệu mà nó chưa được tiếp xúc trước đó . Trong ngữ cảnh này , ‘ inductive ’ chỉ ra khả năng của mô hình tổng quát hóa từ dữ liệu thấy để đối mặt với dữ liệu mới .

Cụ thể trong học máy có 2 loại học theo chủ động (inductive learning ) và học theo kiểu nhớ (memorization ) . Trong học theo kiểu nhớ , mô hìhifnmemory các mẫu trong tệp dữ liệu huấn luyện và không thể tổng quát hóa trên tập dữ liệu mới . Ngược lại , học theo chủ động nhấn mạnh khả năng tổng quát hóa và áp dụng kiến thức học được từ dữ liệu huấn luyện để dự đoạn trên dữ liệu mới .

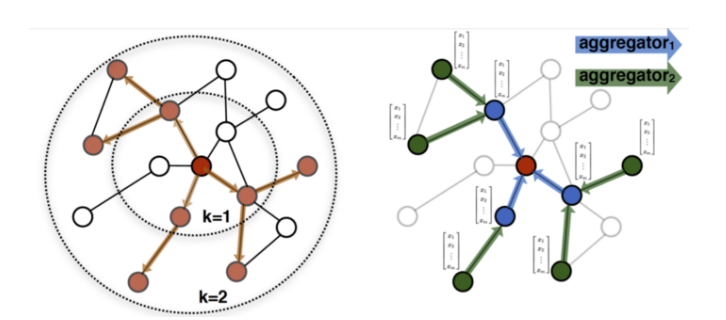
Một ví dụ điển hình của học chủ động là khi đào tạo 1 mô hình phân loại email rác trên

Tập dữ liệu huấn luyện và sau đó sử dụng mô hình để phân loại email mới mà nó chưa

Từng được thấy trước đó . Khả năng của mô hình tổng quát hóa từ những đặc trưng học từ email rác trong tập dữ liệu huấn luyện là một phần quan trọng là chủ động .

Vào năm 2017 , có một bài báo tên gọi là [**Inductive Representation Learning on Large Graphs - 2017**](https://arxiv.org/abs/1706.02216) hay GraphSage được đề xuất , với khá nhiều cải tiến về mặt mô hình sp với các paper [GCN - 2016](https://arxiv.org/abs/1609.02907) . Có thể tóm gọn trong 1 số ý như sau .

* An inductive learning method , tức GraphSage có khả năng tổng quát hóa tốt hơn với các unseen data .
* Vẫn dựa trên ý tưởng sinh các node embedding dựa trên các node lân cận . Trong paper GraphSage , tác giả đề cập với việc thiết kế các hàm aggrerate nhằm tổng hợp lại thông tin từ các nút lân cần đề xuất 3 hàm aggrerate tương ứng
* Mini – batch update gradient descent và là 1 spatical gnn method , khắc phục được hạn chế lớn nhất của GCN là việc cập nhập theo full-batch gradient desent



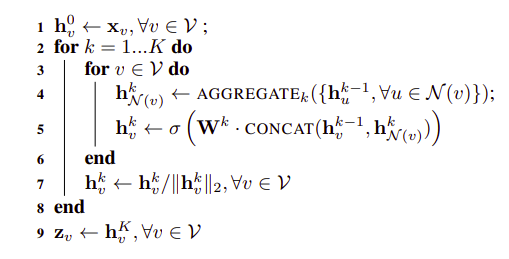
Hình ảnh 18 : Mô tả họa động của từng aggrerate .

Tổng quan , mô hình GraphSage được xây dựng cũng dựa trên ý tưởng là tổng hợp thông tin từ các nút lân cận . Chi tiết giải thuật như sau .

* Input : Đồ thị G(V, E) các note feature xv thuộc X , 1,2,…K,

Là số aggregator function được áp dụng liên tiếp nhau .

* Output : embedding vector 



Mô tả thuật toán theo code

Ta có  với từng aggre function thứ k , tại từng nút v thuộc V , ta tổng hợp thông tin từ các nút lân cận , Với mọi u thuộc N(v) , thu được 1 vector biểu diễn  . các hàm aggre biểu diễn có thể sử dụng 1 số các hàm đươn giản như mean , pooling , … hay thậm chí các mạng nhưu lstm

Với thông tin thu được tổng hợp các nút lân cận , ta tiến hành concat  với thông tin của nút v tại step trước đó (k-1)

Với vector mới vừa được concat , ta được qua 1 fully connected layer đươn giản với 1 hàm kích hoạt phi tuyến ở cuối , ví dụ Relu . đồng thời normalize  , k thộc K

Sau khi thực hiện tính toán qua k lần (K aggre function ) ta thu được feature vector

z­v  .

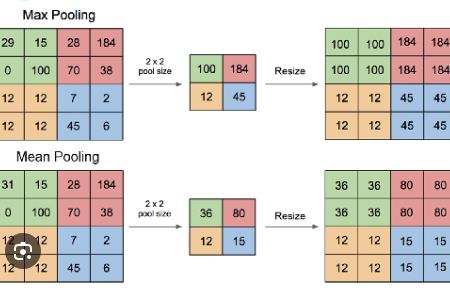
Ta có thể stack nhiều aggre function liên tiếp nhau với mong muốn mô hình sẽ sâu hơn và học được nhiều cấc abstact feature . tuy nhiên , với dữ liệu đồ thị việc stack quá nhiều aggre function cũng không đem lại khác biệt quá nhiều về mặt kết quả , thâm chí ảnh hưởng đến performance của mô hình . Thường ta sử dụng 1, 2 aggre function là đủ .

Bằng việc thiết kế các hàm aggre function như vậy , trọng số của mô hình dần được cập nhật dựa trên thông tin của các nút lân cận u , với mọi u thuộc N(v) của nút v , khi đó ta có khả năng tổng quát hóa tốt hơn (better generalization) với các điểm dữ liệu mới , (unseen node ) và từ chính node feature của nút đó , giải thuật này có tính ứng dụng cao hơn , phù hợp với nhiều bài toán dữ liệu lớn hơn và thay đổi thường xuyên như : dữ liệu mạng xã hội , dữ liệu các trang wikipedia được liên kết , dữ liệu từ các paper mới khi cite . reference tới các paper trước đó , hay dữ liệu người dùng upvote , clip hay follow user .

**Aggregator functions**

Dữ liệu dạng đồ thị không có tính thứ tự , tương đối về vị trí như các dạng dữ liệu như sequence , image , … nên giả định rằng các aggregator function được định nghĩa cũng phải có tính chất symmetric ( tức bị ảnh hưởng bởi hoán vị của nút lân cận ) . Trong paper có đề cập đến việc sử dụng 3 aggregator function là mean , pooling , LSTM

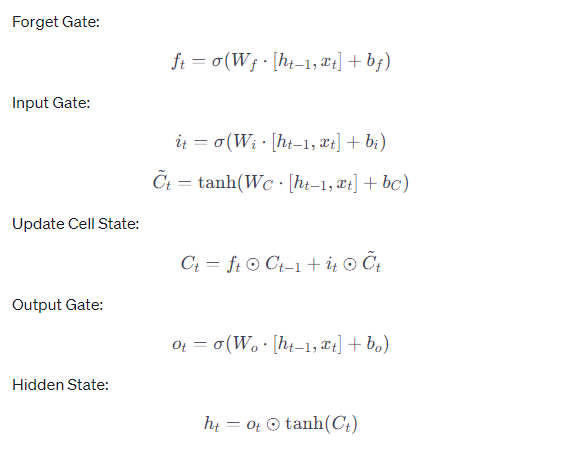
Mean aggregator , là 1 non-parametric function và symmetric , đơn giản là việc của trung bình vector của các nút lân cận tại mỗi vị trí hay element-wisse mean operation . Việc thưc hiện concat 2 vector trên đoạn mô tả bên trên gần như tương tự 1 ‘skip connection ’ trong mạng redidual network .



Hình ảnh 19 : mô tả Mean aggregator .

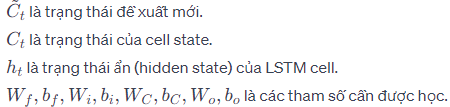
LSTM aggregator , 1 parametric function . LSTM được thiết kế cho bài toán dạng sequence , tức không phải là symmetric . tuy nhiên , trng paper có thể đề cập đến việc sử dụng hoán vị ngẫu nhiên từ input và các nút lân cận . Kết quả thu được khả quan so với các aggregator function khác .

Công thức toán của LSTM aggregator :



Bao gồm :

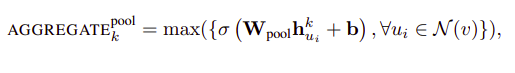
Ft , it , ot là các giá trị của các cổng (forget , input , output ) tương ứng .



Các biểu thức này mô tả quá trình lan truyền xuôi trong 1 LSTM cell , từ đầu vào xt và trạng thái trước đó ht-1 đến trạng thái ẩn ht là trạng thái cell .

Pooling aggregator , vừa là 1 parametric function và symmetric , được thiết kế theo công thức . Với  *σ* là activation function , mã là toán tử element-wise max-pooling operation.

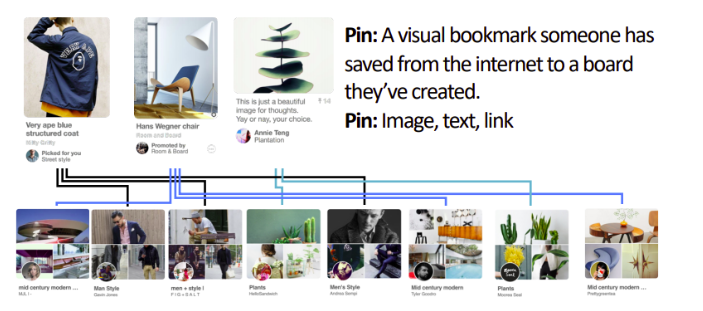
Công thức :



**1 số ứng dụng thực tế :**

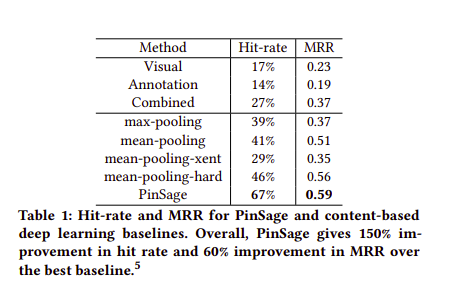
**PinSage – Pinterest Recommender Engine**

* Pinterest là 1 mạng xã hội chia sẻ hình ảnh rất phổ biến. Nguồn thông tin trên Pinterest được xây dựng dựa trên các thông tin về pins và boards. Với pins tương tự như chức năng lưu / yêu thích ảnh từ người dùng, còn boards là tổng hợp các ảnh cùng 1 chủ đề, có liên quan đến nhau, được xây dựng bởi cộng đồng. Bài toán đặt ra với các kĩ sư Pinterest là làm sao để gia tăng lượng tương tác từ người dùng, bằng cách recommend các boards có liên quan khi người dùng thực hiện pin / save 1 bức ảnh nào đó
* Tập dữ liệu mà bên Pinterest lưu trữ tương đối lớn, khoảng 2 tỉ lượt pins và 1 tỉ các boards, với khoảng 18 tỉ connection (edge) giữa các pin và board. Nhiệm vụ của bài toán recommend là xây dựng được bộ node embedding cho các pins để từ đó tiến hành recommend các ảnh tương đồng hoặc các board có liên quan. Và bên Pinterest có kế thừa từ mô hình **GraphSage** ta vừa bàn luận bên trên bằng cách xây dựng 1 bi-partite graph (đồ thị 2 phía) với 1 bên là các pins, 1 bên là các boards và liên kết giữa pins-boards



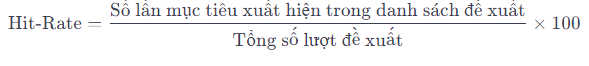
Hình ảnh 20 : Hình ảnh mô tả mạng xã hội Pinterest

* Với Node Embedding được khởi tạo ban đầu, các nhà phát triển bên Pinterest tận dụng các metadata từ nhiều kiểu dữ liệu khác nhau, từ cả ảnh và text để tạo các node embedding *xi*​ ban đầu ứng với từng node. Bên cạnh đó, trong paper PinSage còn đề cập tới các hướng improve khác như: **Producer consumer mini-batch construction**, **Efficient MapReduce Inference**, **Negetive Sampling Mining**, **Curriculum Training**, ...
* Trong quá trình thực hiện A/B test, đội ngũ bên Pinterest cũng thấy rằng hiệu năng của hệ thống được tăng lên đáng kể



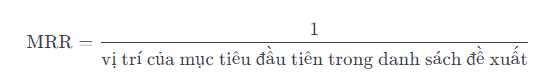
Hình ảnh 21 : Mô tả quá trình và tỉ lệ hit-rate and MRR (\*)

Tỉ lệ Hit-Rate là một bộ đo thường được sử dụng thông lĩnh cực đề xuất hệ thống (recommendation systems ) để đánh giá hiệu xuất mô hình . Nó đo lường tỉ lệ phân trăm các đề xuất mục tiêu thực tế (ground truth ) xuất hiện trong danh sách đề xuất . Công thức tính của Hit-rate



Mức độ đánh giá Mean Reciprocal Rank -MRR :

MRR là bộ đo đánh giá khác thường được sử dụng trong lĩnh vực đề xuất hệ thống . Nó đo lường sự hiệu quả của mô hình bằng cách xem vị trí của mục tiêu đầu tiên trong danh sách đề xuất . Công thức MRR là :

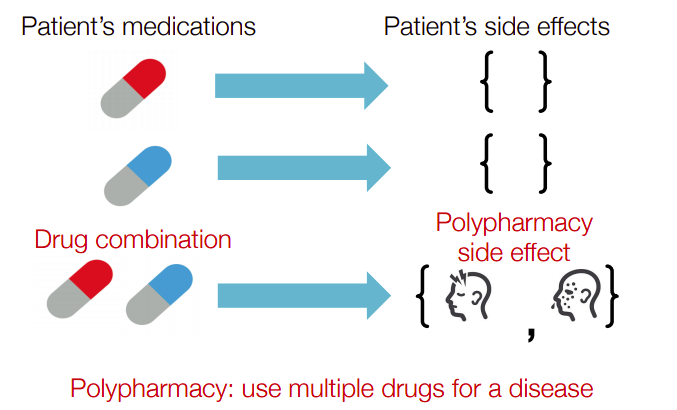


Tính toán MRR bằng cách lấy nghịch đảo của vị trí của mục tiêu đầu tiên trong 1 danh sách đề xuất . Nếu mục tiêu xuất hiện ở vị trí thứ nhất thì MRR sẽ là 1 . nếu mục tiêu xuất hiện ở vị trí thứ 2 MRR sẽ là 0.5

Cả 2 độ đo này đều cung cấp thông tin về khả năng mô hình trong việc đề xuất các mục tiêu mà người dùng có thể quan tâm . Tuy nhiên chúng có nhược điểm là chỉ xem xét một số lượng hữu hạn các lựa chọn đề xuất , và không quan tâm đến sự thứ tự hoặc chất lượng của các lựa chọn đề xuất khác nhau .

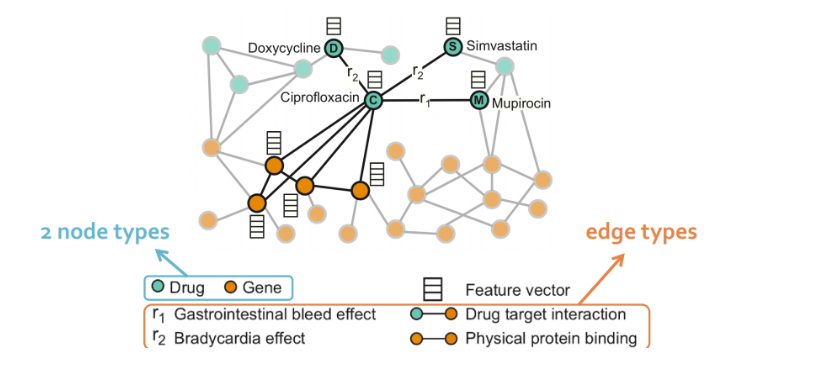
**Decagon - Heterogeneous Drug Side-Effect**

* Drug Side-Effect có thể hiểu là những tác dụng phụ (tác dụng ngược) có ảnh hưởng thêm, không tốt đối với người sử dụng khi sử dụng thuốc hoặc nhiều loại thuốc kết hợp với nhau mà không có chỉ định rõ ràng từ bác sĩ. Vì cấu tạo thành phần các các loại thuốc rất phức tạp về cấu trúc phân tử nên sẽ gây khó khăn trong việc điều chế và sử dụng nhiều loại thuốc khác nhau.



Hình 21 : Mô tả tác dụng phụ của thuốc

* Với dạng dữ liệu kiểu này, ta cũng hoàn toàn có thể sử dụng các mô hình GNN cho từng bài toán nhất định, ví dụ bài toán về chuẩn đoán tác dụng phụ khi sử dụng chung nhiều loại thuốc.

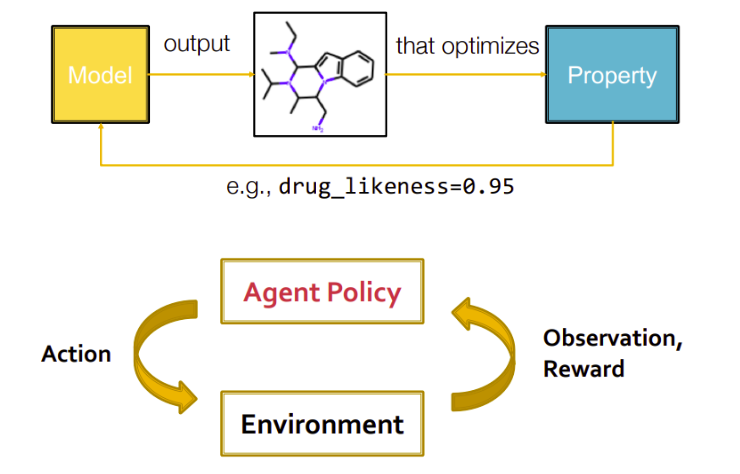


Hình 22 : Mô tả graph tác dụng phụ của thuốc.

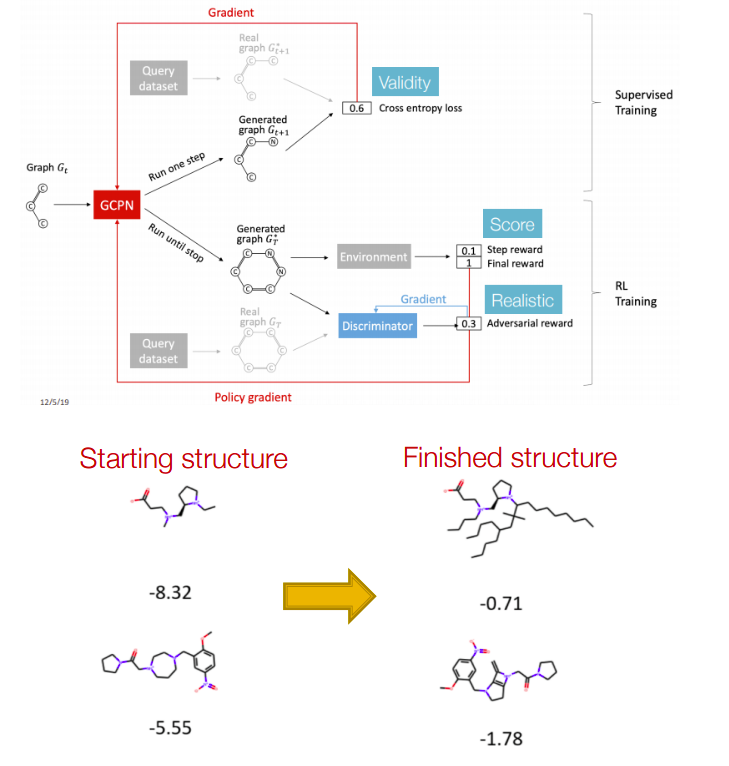
* Về mặt graph cũng được xây dựng phức tạp hơn 1 chút. Với các nút xanh là tên thuốc, các nút cam là thành phần cấu tạo kèm theo node embedding ứng với từng thành phần. Các cạnh bao gồm các tương tác giữa drug-gene, gene-gene và các side-effect giữa các loại thuốc với nhau

**GCPN - Goal-directed generation**

* Là bài toán graph generation hay việc sinh / hình thành các nút và cạnh mới trên đồ thị để tạo ra các kiến trúc mới. 1 bài toán dễ hình dùng là hình thành các cấu trúc phân tử dựa trên 1 số mục tiêu và điều kiện / quy tắc cho trước.



Hình 22 : Mô tả bài toán hình thành các các trúc phân tử

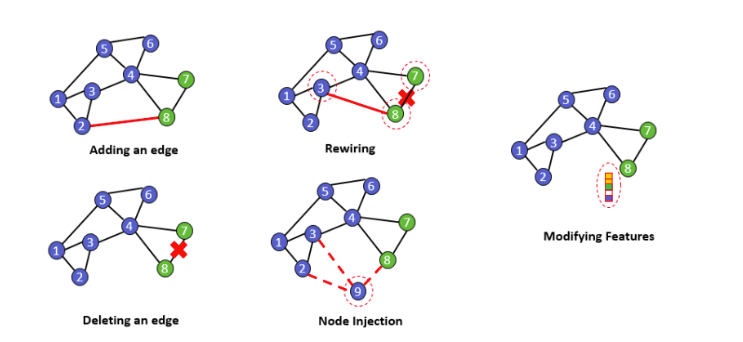


Hình 23 : Mô tả từng hoạt động của bài toán đề ra 4

**Hạn chế và lưu ý**

Non-injective neightbor aggregation & injective neightbor aggregation

* Trong nhiều trường hợp, việc sử dụng các aggre function như mean, max, .. khiến mô hình khó có thể phân biệt được các cấu trúc khác nhau, hay còn gọi là các non-injective aggre function. Trong paper về Graph Isomorphism Network có đề cập tới việc thiết kế mô hình để thu được các injective aggre function, các bạn có thể đọc thêm tại paper sau: <https://cs.stanford.edu/people/jure/pubs/gin-iclr19.pdf>)
* 1 số kiểu tấn công thường gặp như
  + Thêm / xóa / thay đổi các mối liên kết trên đồ thị (modify edge)
  + Thêm / xóa các nút trên đồ thị (node injection)
  + Thay đổi các feature hay node embedding của nút (modify feature)



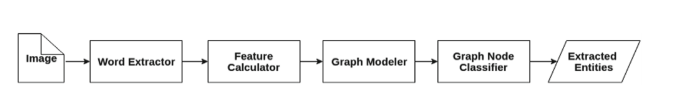
Hình 24 : Hình ảnh mô tả tấn công đối nghịch(Adversarial Attack)

**CHƯƠNG 2 :TRÍCH RÚT THÔNG TIN TỪ HÓA ĐƠN**

**2.1. Lý thuyết trích rút thông tin**

* Bài toán Information Extraction là 1 bài toán không mới, nhưng trong bài hướng dẫn này, dạng dữ liệu mà mình muốn hướng tới là hóa đơn (invoice). Nhiệm vụ đặt ra là làm sao phân loại được các text box vào các trường thông tin tương ứng, bao gồm: company (tên cty, nhà phân phối sản phẩm), address (địa chỉ), date (ngày giao dịch), total (tổng giá tiền) và other (không thuộc 4 trường trên).
* Có 1 chú ý rằng bài toán này được thực hiện với 1 yêu cầu rằng cần thực hiện 2 bài toán con trước đó là Scene Text Detection và Scene Text Recognition. Đầu ra của 2 bài toán này sẽ được sử dụng để xây dựng các feature và đồ thị cho bài toán thứ 3 là Key Information Extraction. Đầu vào của mô hình là ảnh, đầu ra ứng với mỗi text box sẽ được phân loại thuộc 4 trường thông tin tương ứng.
* Thực ra, với bài toán trích rút thông tin từ ảnh này, ta hoàn toàn có thể sử dụng các phương pháp dễ tiếp cận và quen thuộc hơn như: template-based hoặc NLP-based. Tuy nhiên, mỗi phương pháp đều có những hạn chế tương ứng:
  + **Template-based**: đơn giản là việc áp dụng các rule (luật), được định nghĩa từ trước lên các form, văn bản có layout / structure cố định, không thay đổi nhiều. Tiếp đó, sử dụng các phương pháp về text / keyword matching để xác định các trường thông tin tương ứng. Tuy nhiên, nhược điểm lớn nhất của phương pháp này là chúng ta phải định nghĩa từng luật riêng ứng với từng form, không có khả năng adapt sang dạng form mới và bị phụ thuộc hoàn toàn vào domain knowledge của từng người.
  + **NLP-based**: với phương pháp này, các nội dung thu được từ text-box có thể đưa vào 1 mô hình text classification hoặc NER để tiến hành phân loại hoặc xác định các thực thể thuộc từng trường thông tin tương ứng. Ưu điểm của phương pháp này so với Template-based là có khả năng adapt được với dữ liệu mới. Tuy nhiên, 1 số nhược điểm có thể kể tới như: bị phụ thuộc rất nhiều vào layout của form, hạn chế với dữ liệu được biểu diễn dưới dạng bảng / table, hoàn toàn không sử dụng các thông tin / feature về vị trí của text-box, cho dù các thông tin về layout như vậy cũng sẽ giúp ích rất nhiều trong việc xác định các trường tương ứng.
* Việc thay thế và áp dụng graph-based method cho bài toán này đến từ 1 số lý do sau:
  + Local pattern: tương tự như mô hình CNN, nhưng thay vì là các điểm pixel, các node có kết
  + nối với nhau cũng sẽ có mối liên hệ cao hơn với các node xa hơn trong đồ thị.
  + Positional feature: các thông tin về vị trí / tọa độ của nút trên ảnh cũng sẽ giúp mô hình dễ dàng phân biệt các trường thông tin hơn. Ví dụ như thông tin về tên của siêu thị / cửa hàng thực phẩm thường được ghi ngay trên đầu của hóa đơn
  + Textual feature: tương tự như positional feature, các thông tin về text cũng rất quan trọng. Ví dụ như việc phân biệt trường thông tin address với các trường dữ liệu khác
  + Việc stack nhiều các module GCN lên nhau giúp model học được các high level feature tốt hơn

Pipeline của bài toán được mô tả như sau :



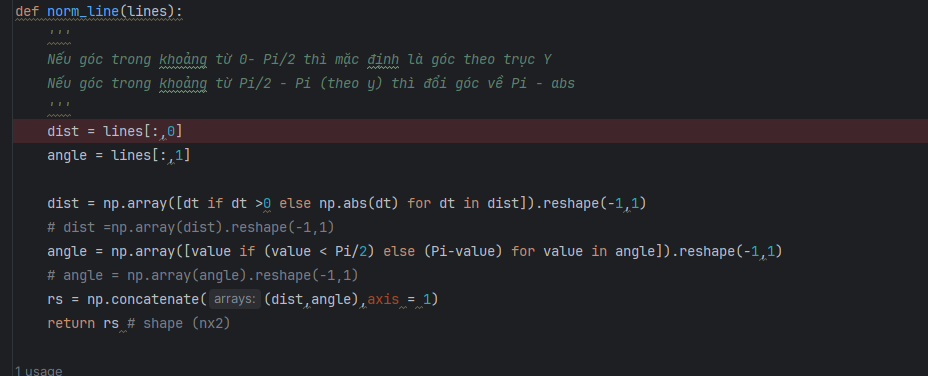
**Tiền xử lý hình ảnh**

Trước khi đi vào xây dựng mô hình để huấn luyện , có một số ngoại lệ được đặt ra về xử lý hình ảnh ví dụ hóa đơn bị gấp không thấy được rõ thông tin , hay bị rách 1 góc vì vậy chúng ta nên có một bước được gọi là tiền xử lý hình ảnh . Chúng ta sẽ sử dụng thuật Mô hình U2net để bỏ nền của hình ảnh , tạo ra một ảnh mà chỉ có đối tương chính mà không có đối tượng nào .

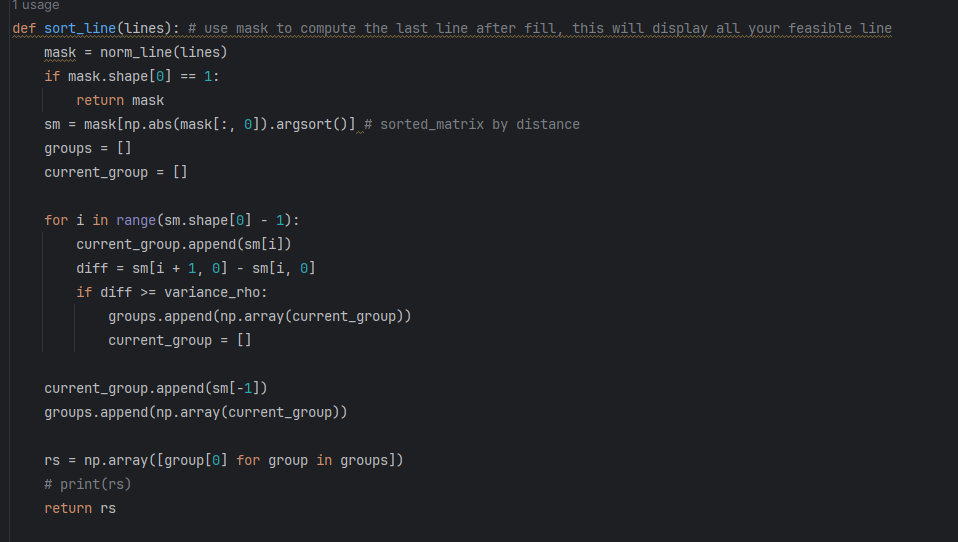
U2Net được phát triển dựa trên mô hình U-net một mạng sử dụng rộng rãi trong lĩnh vực sử lý hình ảnh , phân đoạn hình ảnh . U2net cải tiến U-NET bằng cách áp dụng các kỹ thuật mới như khối cảm biến chú ý và cơ chế tự học . Điều này giúp khả năng phân đoạn và tạo ra các kết quả được chính xác hơn , đặc biêt trong nhiệm vụ loại bỏ nền ảnh .

U2Net được sử dụng trong nhiều ứng dụng , bao gồm xử lý ảnh y khoa , xử lý ảnh trong thị giác máy tính và ứng dụng trong lĩnh vực truyền thông số . Đặc biêt trong lĩnh vực xử lý ảnh việc loại bỏ nền là một việc tối quan trọng trong nhiều ứng dụng như phát hiện vật thể , tạo ảnh ghép , hay phục hồi ảnh .

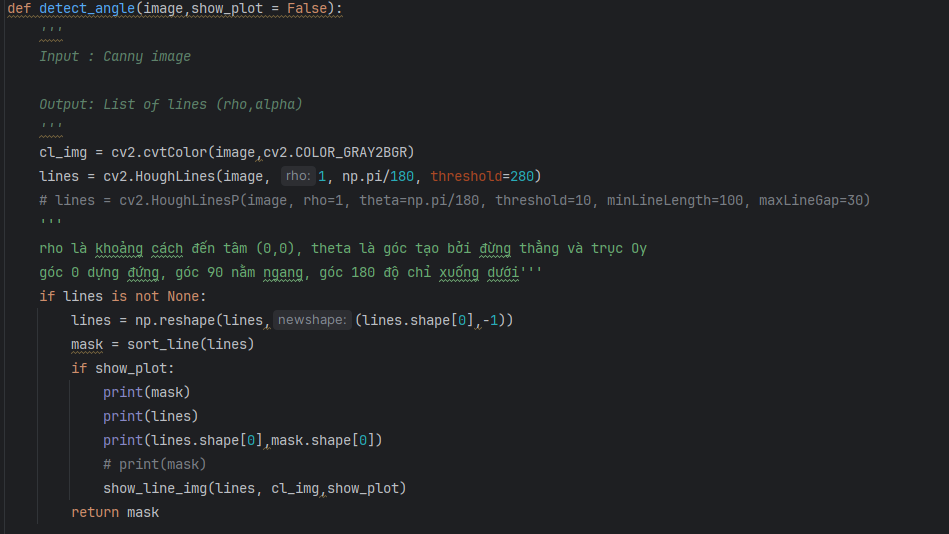
những hóa đơn không có cạnh nào thẳng , không tuân theo giới hạn để duỗi thẳng đồng thời bỏ các hóa đơn có độ tin cậy đánh nhãn không quá cao .



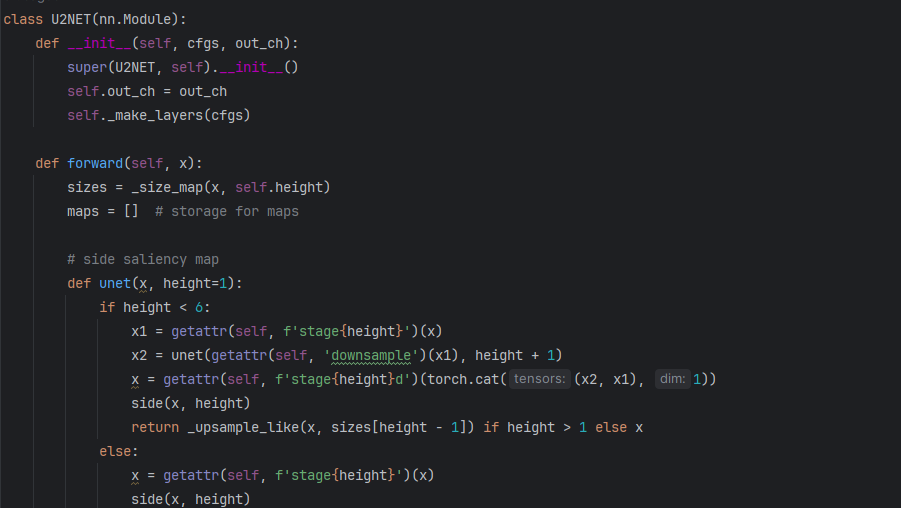
Hình ảnh 25: Mô tả góc của hóa đơn tính theo trục Y

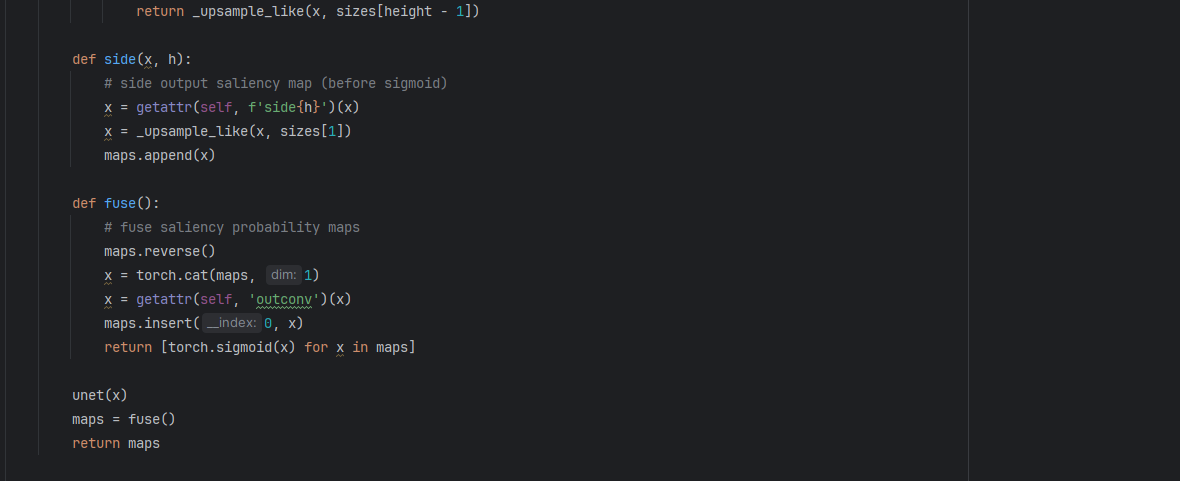


Hình ảnh 26 : Hàm sắp xếp đường kẻ của hóa đơn

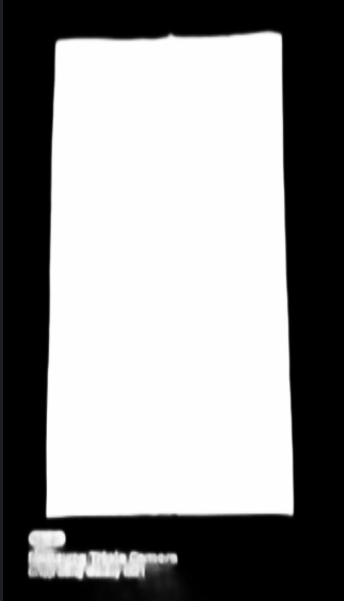


Hình ảnh 27 : hàm tính hệ số góc của hóa đơn





Hình ảnh 28 : hình ảnh mô tả mạng lưới U2Net



Hình ảnh 29 : Mô tả hình ảnh output của mạng lưới U2Net



Hình ảnh 30 : Mô tả hình ảnh sau khi được kẻ của hóa đơn

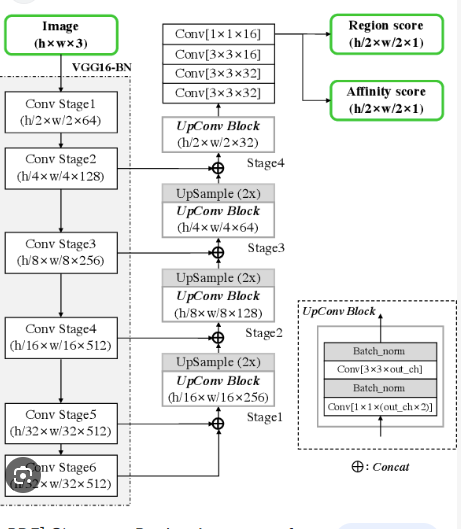
**Phần Word Extractor**

bao gồm 2 phần là Text Detection và OCR như bên trên mình đã có đề cập.

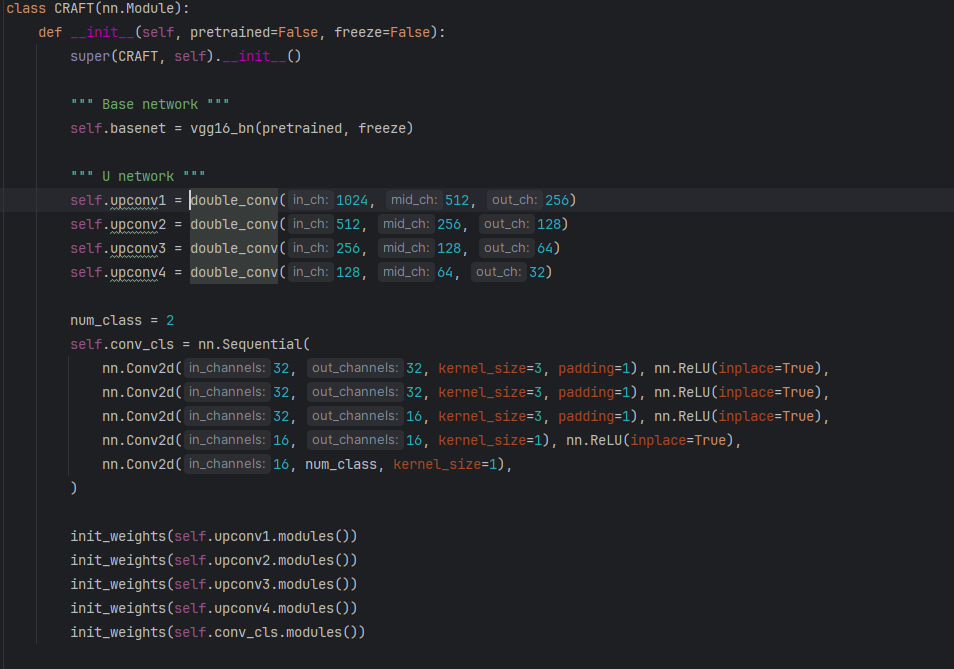
Trong trường hợp của bài toán này chúng ta sẽ sử dụng OCR và Bài toán của Text Detection là Mô hình Craft .

Craft (Character RegionAwareness for Text Detection ) là một mô hình thị giác máy tính được thiết kế để phát hiện và định vị vùng chữ trên ảnh . Mục tiêu chính của CRAFT là nhận biết vùng chữ với hình ảnh không đồng đều , điều này có thể xuất hiện trong các hình ảnh chứa văn bản theo các hình thức không chính quy .

* Mục tiêu mô hình craft là phát hiện vùng chữ trên ảnh , đặc biệt là với văn bản phức tạp và không đồng đều .
* Kiến trúc mô hình , CRAFT sử dụng một mạng neural tích chập học các đặc trưng của hình ảnh , sau đó nó sử dụng một mô hình LSTM (Long Short-Term Memory ) để dự đoán các đỉnh và vùng chữ
* Đặc điểm quan trọng : CRAFT có khả năng làm việc với các vùng chữ có hình dạng phức tạp và không chính quy , điều này là quan trọng trong các ứng dụng thực thế như đọc văn bản trong ảnh chụp từ môi trường không kiểm soát .
* Hiệu suất: CRAFT cho thấy hiệu suất tốt trong việc phát hiện và định vị vùng chữ so với 1 số dự án khác trên tập dữ liệu thử nghiệm
* Ngôn ngữ : Thông thường sẽ được triển khai bằng Python và sử dụng các thư viện tensorflow hoặc Pytorch .



Hình 31 : Cấu trúc mô hình mạng của CRAFT

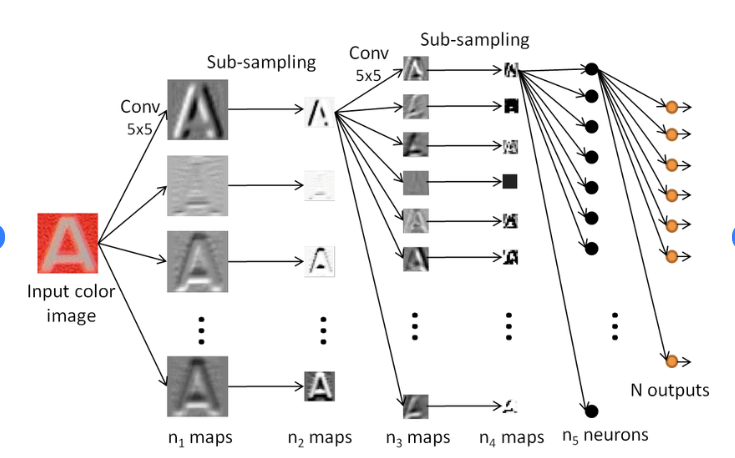




Hình 32 : Code mô tả quá trình pretrain của mạng CRAFT .

OCR : là loại [phần mềm](https://vi.wikipedia.org/wiki/Ph%E1%BA%A7n_m%E1%BB%81m) [máy tính](https://vi.wikipedia.org/wiki/M%C3%A1y_t%C3%ADnh) được tạo ra để chuyển các hình ảnh của chữ viết tay hoặc chữ đánh máy (thường được quét bằng máy [scanner](https://vi.wikipedia.org/w/index.php?title=Scanner&action=edit&redlink=1)) thành các văn bản tài liệu. OCR được hình thành từ một lĩnh vực nghiên cứu về [nhận dạng mẫu](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_d%E1%BA%A1ng_m%E1%BA%ABu), [trí tuệ nhận tạo](https://vi.wikipedia.org/w/index.php?title=Tr%C3%AD_tu%E1%BB%87_nh%E1%BA%ADn_t%E1%BA%A1o&action=edit&redlink=1) và [machine vision](https://vi.wikipedia.org/w/index.php?title=Machine_vision&action=edit&redlink=1). Mặc dù công việc nghiên cứu học thuật vẫn tiếp tục, một phần công việc của OCR đã chuyển sang ứng dụng trong thực tế với các kỹ thuật đã được chứng minh.

Kiến trúc mô hình :



Hình 33 : Kiến trúc mô hình OCR .

Phương pháp học : Mô hình OCR thường được huấn luyện bằng cách sử dụng dữ liệu có nhãn , trong đó mỗi hình ảnh hoặc phần của hình ảnh được gán với 1 nhãn tương ứng với văn bản tương ứng .

Các bước thực hiện : Mô hình OCR thường bao gồm các bước sử lý ảnh , trích xuất đặc trưng , và dự đoán chuỗi ký tự . Các kỹ thuật như attention mechanism cũng thường được sử lý để tập chung vào các phần quan trọng của hình ảnh ,

Thư viện hỗ trợ : Có nhiều thư viện hỗ trợ như Tesseract OCRopus , PyTesseract , Tesorflow , Pytorch ,.. để triển khai và phát triển mô hình OCR .

* Phần Feature Calculator hay feature engineering, là phần tạo feature cho các node (nút, đỉnh) trên đồ thị. Các nút ở đây chính là các bounding box thu được sau bước text detect. Việc định nghĩa các cạnh của graph thuộc phần **Graph Modeler**, sẽ được mình đề cập kĩ hơn ở bên dưới. Giờ việc cần làm tiếp theo là xây dựng feature ban đầu cho các nút của đồ thị. Có nhiều cách nhưng mình sẽ ví dụ cách thực hiện trong paper Invoice-GCN, với việc xây dựng và tổng hợp feature từ nhiều kiểu / thuộc tính khác nhau:
  + Boolean feature: dựa trên đầu ra từ mô hình text recognition, ta xây dựng các thuộc tính như:
    - isDate: có phải là ngày tháng ko (1 / 0)
    - isZipCode: 6 kí tự có thuộc 1 zipcode mã vùng có sẵn ko (1 / 0)
    - isKnownCity, isKnownDept, isKnownCountry: lần lượt kiểm tra xem phần nội dung text có phải là tên của cục, sở, thành phố hay đất nước nào không (1 / 0)
    - nature: gồm 8 phần tử, lần lượt kiểm tra các thuộc tính bao gồm: isAlphabetic, isNumeric, isAlphaNumeric, isNumberwithDecimal, isRealNumber, isCurrency, hasRealandCurrency, mix (except these categories), mixc (mix and currency word). Mình thực ra cũng không hiểu hết các thuộc tính mang ý nghĩa gì, trong paper cũng không đề cập rõ nhưng với nature feature, ta sẽ thu được 1 binary vector 8 chiều ứng với 8 thuộc tính con.
  + Numeric feature: khoảng cách tương đối từ text box hiện tại tới 4 box tương ứng (Top, bottom, left, right). Việc xác định các box tương ứng sẽ được đề cập tại phần **Graph Modeling**.
  + Text feature: dựa trên đầu ra của mô hình text recognize, ta có thể sử dụng các mô hình thông dụng như Word2Vec, Glove,... để lấy vector embedding của từ (vì phần text detect trong paper là dựa theo word-based). Tuy nhiên, các mô hình này có hạn chế là OOV - Out of vocabulary hay các từ không xuất hiện lúc training sẽ không có embedding. Điều này có thể được cải thiện bằng cách sử dụng các phương pháp khác như: FastText hay BPE (InvoiceGCN). Hoặc nếu với 1 text line thì có thể sử dụng các mô hình về Bert-based để lấy embedding cho câu hiện tại!

==> Sau cùng, ta "nối" tất cả các thuộc tính đó lại và thu được 1 feature vector 317 chiều (1 + 1 + 3 + 8 + 4 + 300) làm node feature ban đầu ứng với từng nút (từng text box) trong graph!

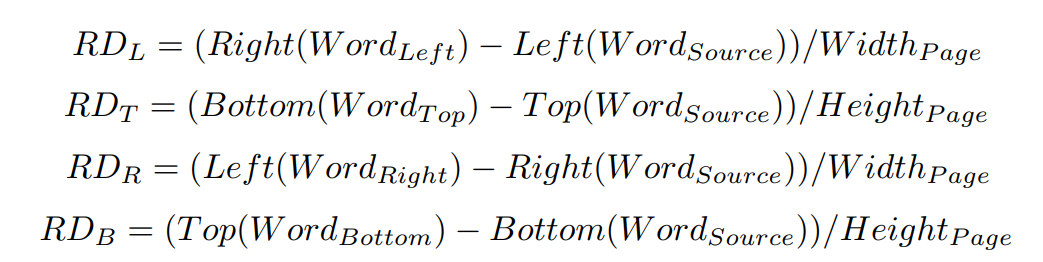
Graph Modeling

* Như bên trên mình có đề cập tới numeric feature, 4 features về tọa độ này được dựa trên vị trí **tương đối** với 4 text box trên, dưới, trái, phải như hình bên dưới

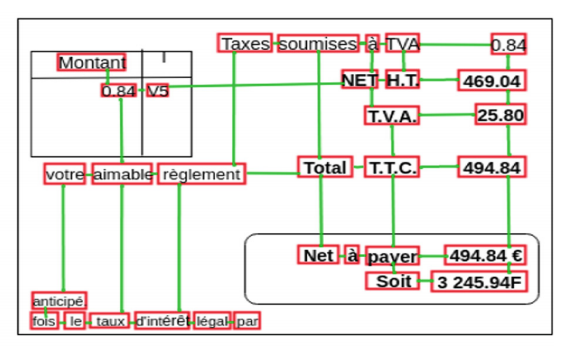


Hình ảnh 34 : mô tả vị trí tương đối của node feature

* Với 4 thông số *RDL*​, *RDR*​, *RDT*​, *RDB*​ được tính toán như sau

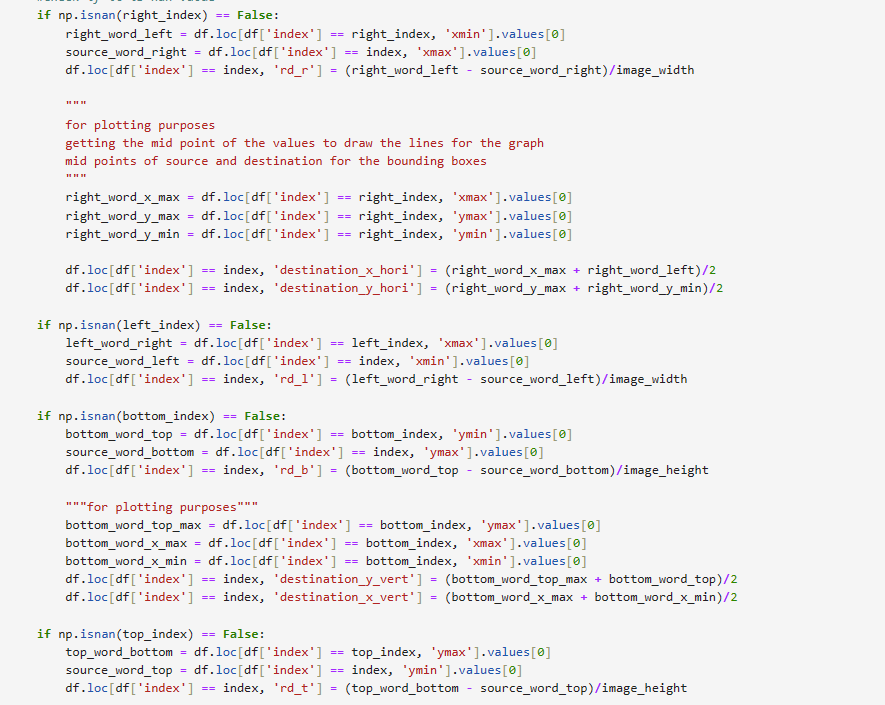


ví dụ với *RDL*​ sẽ được tính toán dựa trên tọa độ của các bounding box (output của model text detection), hay chính bằng khoảng cách từ bounding box Source tới bounding box Left, rồi chia cho độ rộng của ảnh, tương tự với các thông số các cũng như vậy.



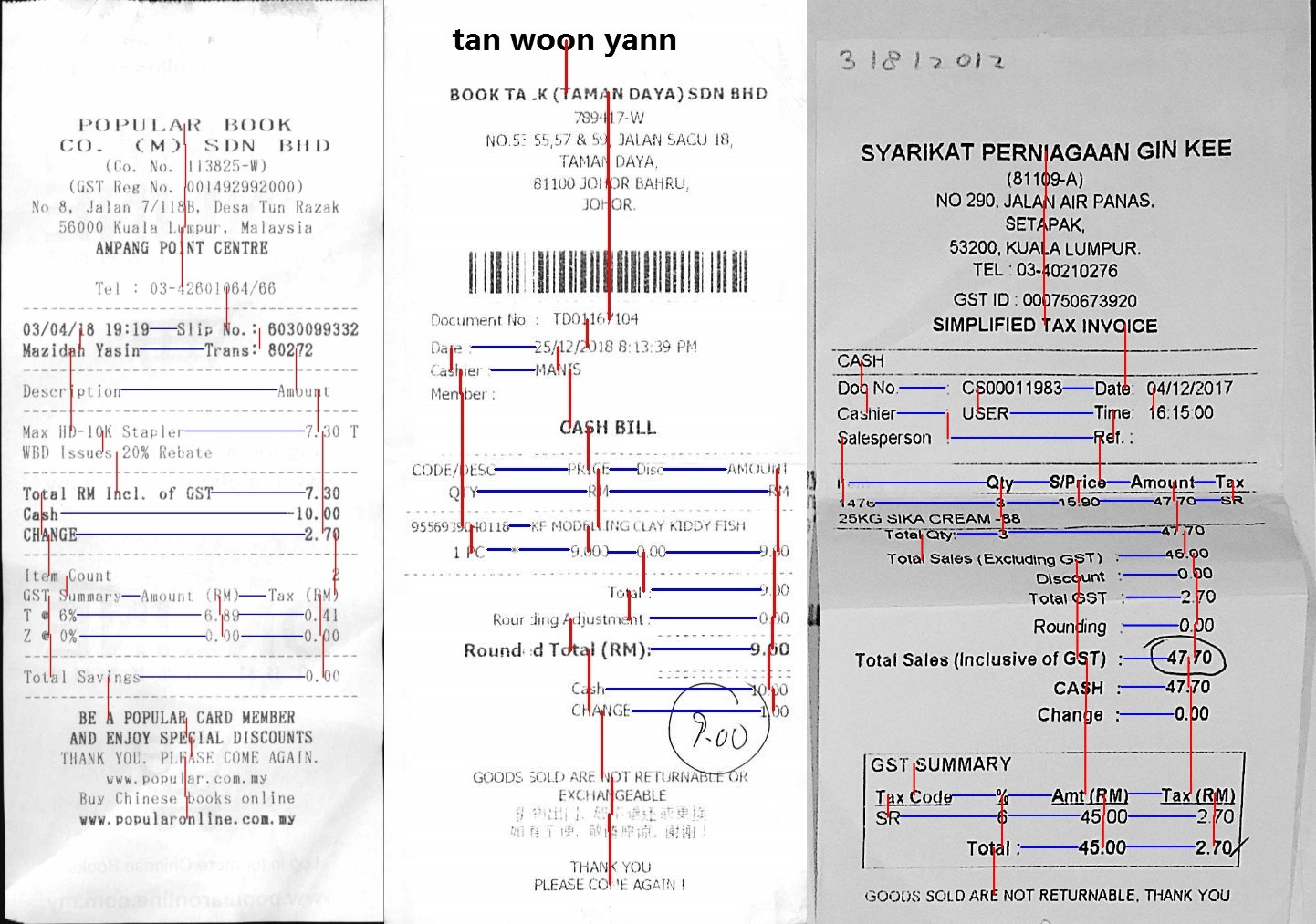
Hình ảnh 35: Mô tả graph của 1 hóa đơn

* Ngoài ra, còn 1 điều chú ý khi xây dựng graph cho từng văn bản (invoice). Ví dụ như hình bên trên, các đường nối giữa các text box đã được thể hiện khá rõ ràng. Nhưng nếu để ý, các bạn có thể thấy rằng không có đường nối giữa 2 text box là anticipé và le. Đơn giản vì việc xác định các text box nào được nối với nhau sẽ theo 1 số luật như sau:
  + Xét theo 4 phía (trên, dưới, trái, phải) và xác định các *RDL*​, *RDR*​, *RDT*​, *RDB*​ tương ứng bằng việc chọn các text box có khoảng cách gần nhất
  + Thứ tự ưu tiên thực hiện sẽ từ trên xuống dưới, từ trái sang phải. Và 1 hướng chỉ có 1 đường nối với 1 text box khác! Như ví dụ bên trên, do text box anticipé và fois đã kết nối với nhau từ trước nên sẽ không có đường nối giữa anticipé và le nữa. Mặc dù 2 text box fois và le đều nằm ngay dưới và có khoảng cách tới text box anticipé là ngang nhau.



Hình ảnh 36 : Đoạn code mô tả cách tính khoảng các giữa các b

* Dưới đây là 1 ảnh minh họa thể hiện các node và edge trên 1 hóa đơn trong tập dữ liệu SROIE

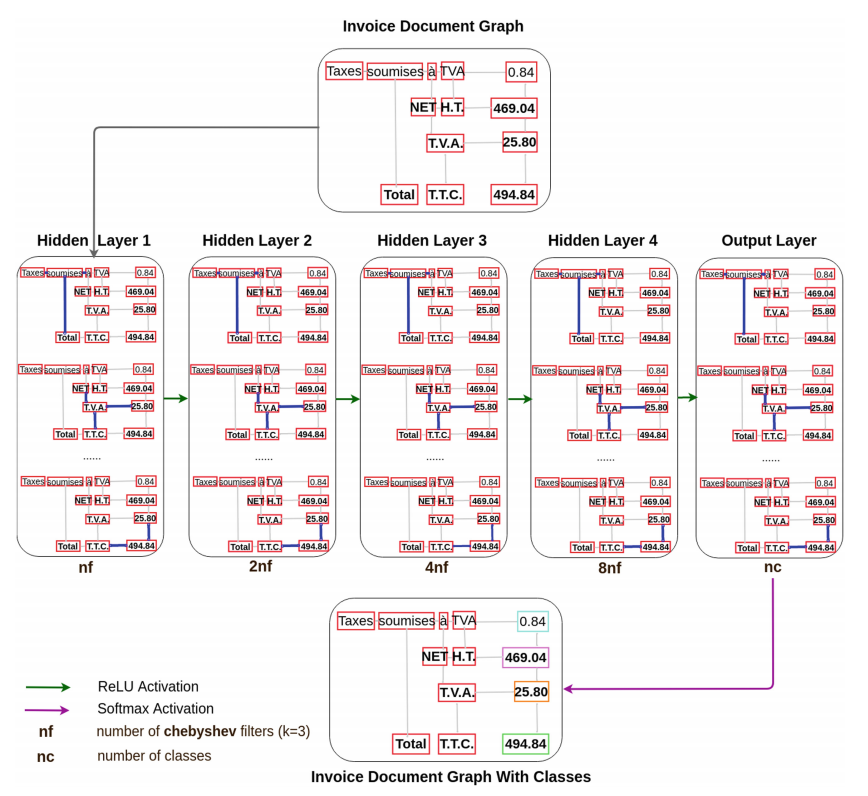


Hình ảnh 37 : Mô tả cụ thể graph 1 hóa đơn

* Tất nhiên, về luật để xây dựng graph cho từng văn bản không hề có hạn chế, hoàn toàn có thể thử nghiệm các cách xây dựng khác nhau, ví dụ việc text box anticipé sẽ có 2 đường nối tới 2 text box fois và le, vì đều nằm ngay bên dưới và có khoảng cách tới text box anticipé là ngang nhau.

**Modeling**

* Trong paper Invoice-GCN có đề cập tới việc sử dụng Chebyshev-GCN model, là 1 spectral graph neural network. Về phần mô hình, có thể tóm gọn như ảnh dưới:

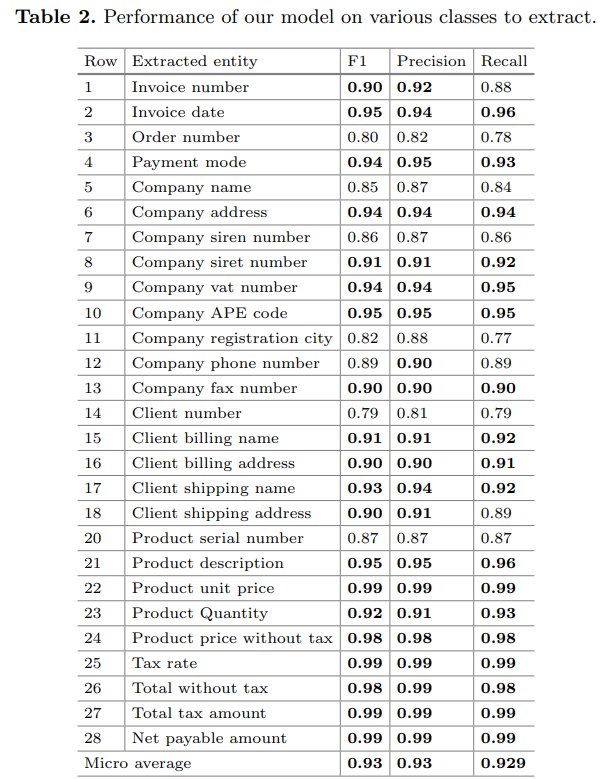


Hình ảnh 37: Mô tả từng hidden layer của 1 graph

* Model Chebyshev-GCN được xây dựng với 5 layers như hình trên, bao gồm 4 hidden layers và 1 output layer. Tham số *K* trong Chebyshev model được chọn mặc định = 3 tại tất cả các hidden layer với số node input được quy định lần lượt là: 16, 32, 64, 128 (với nf = 16) và output layer gồm 28 output ứng với 28 nhãn / thực thể cần phân biệt. Mũi tên xanh là Relu activation function, mũi tên tím là Softmax, loss function là cross-entropy.

Dataset & Experiment

* Vì tập dữ liệu khá đặc thù nên không public, bao gồm khoảng 3100 hóa đơn với 27 trường thông tin cần trích rút (product description, unit price, quantity, total, ...)



Hình ảnh 38 : Mô tả độ chính xác của mô hình

* Trên đây là bảng kết quả ứng với 27 trường thông tin với f1 trung bình = 0.93. Có thể thấy trong paper lựa chọn cách thức extract text theo word-based, không phải theo sentence-based, sử dụng BPE để encode embedding và xử lý với những case OOV. Ví dụ ảnh visualize kết quả đầu ra của mô hình

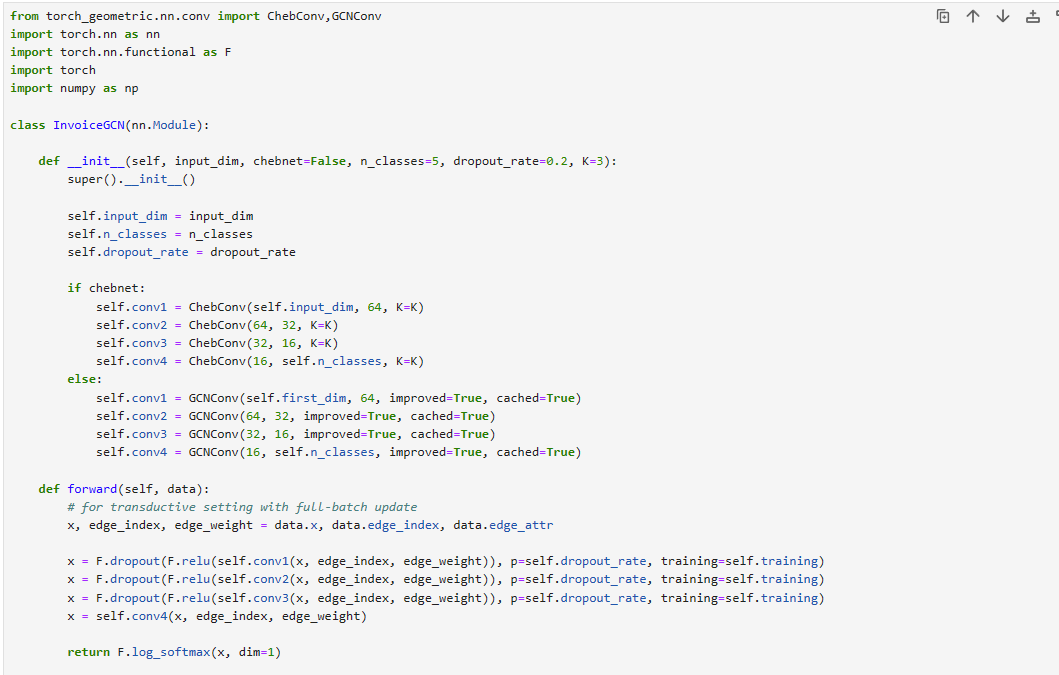


Hình ảnh 39 : Mô tả text detect recogize .

**2.2. Huấn luyên mô hình với GCN**

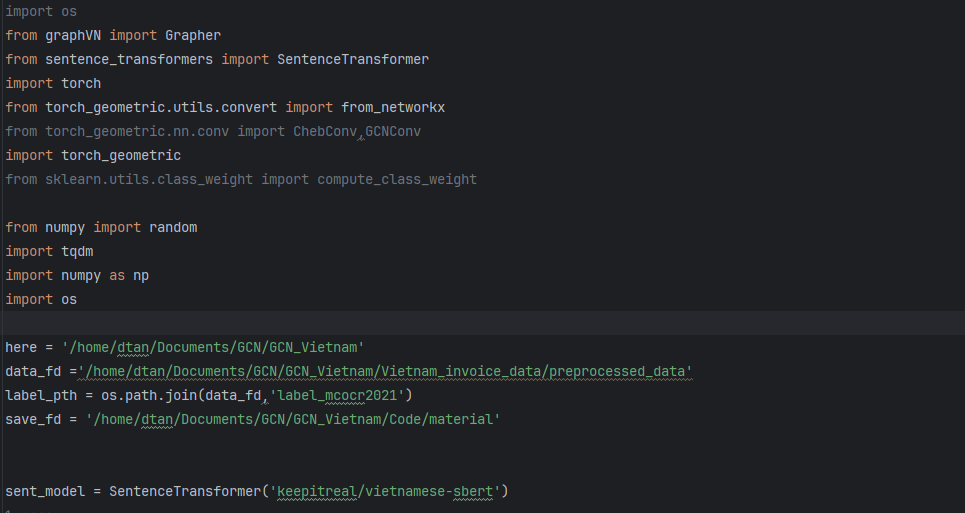
**Tập dữ liệu hóa đơn - SROIE**

* Trong phần này, mình sẽ thực hiện định nghĩa và huấn luyện mô hình trên tập dữ liệu SROIE-2019. SROIE hay Scanned Receipts OCR and Information Extraction là tập dữ liệu được sử dụng trong RRC Competition - ICDAR 2019. Gồm 3 task con: text detection, text recognition và key information extraction. Các bạn có thể download tập dữ liệu tại [trang chủ](https://rrc.cvc.uab.es/?ch=13) hoặc dữ liệu đã được xử lý qua link driver sau: [preprocessed\_SROIE\_2019](https://drive.google.com/drive/folders/1UcW2Fwjd7fSr1wSSOeoCw2i1jCuBEN56?usp=sharing)
* Để nhanh gọn, tại phần đầu này mình sẽ sử dụng các module có sẵn trong thư viện Torch-Geometric để xây dựng 1 mô hình đơn giản cho bài toán Invoice Information Extraction. **Torch-Geometric**, cùng với **DGL**, là 2 trong số rất nhiều thư viện về Graph Network được xây dựng và sử dụng hiện nay, tính đến thời điểm mình viết bài blog này đã có hơn 10k star trên repo. Torch-Geometric được contribute và cập nhật thường xuyên các mô hình về GNN mới, kèm theo các file example, danh sách các mô hình được support có thể xem thêm tại trang chủ: <https://github.com/rusty1s/pytorch_geometric> . Việc định nghĩa các thành phần mới cũng khá dễ dàng.
* Đoạn code bên dưới mô phỏng mô hình GCN được sử dụng trong bài toán này, bao gồm module GCN (trong paper Semi-Supervised Node classification) và Chebyshev-GCN (trong paper Invoice-GCN):



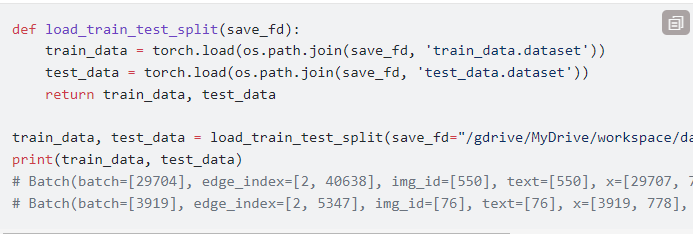
Hình ảnh 40: đoạn code mô tả quá trình pretrain Invoice .

**Định nghĩa Dataset**

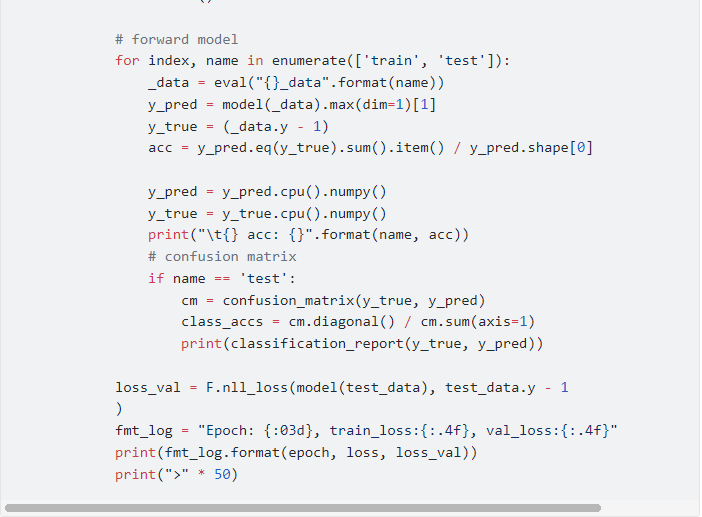
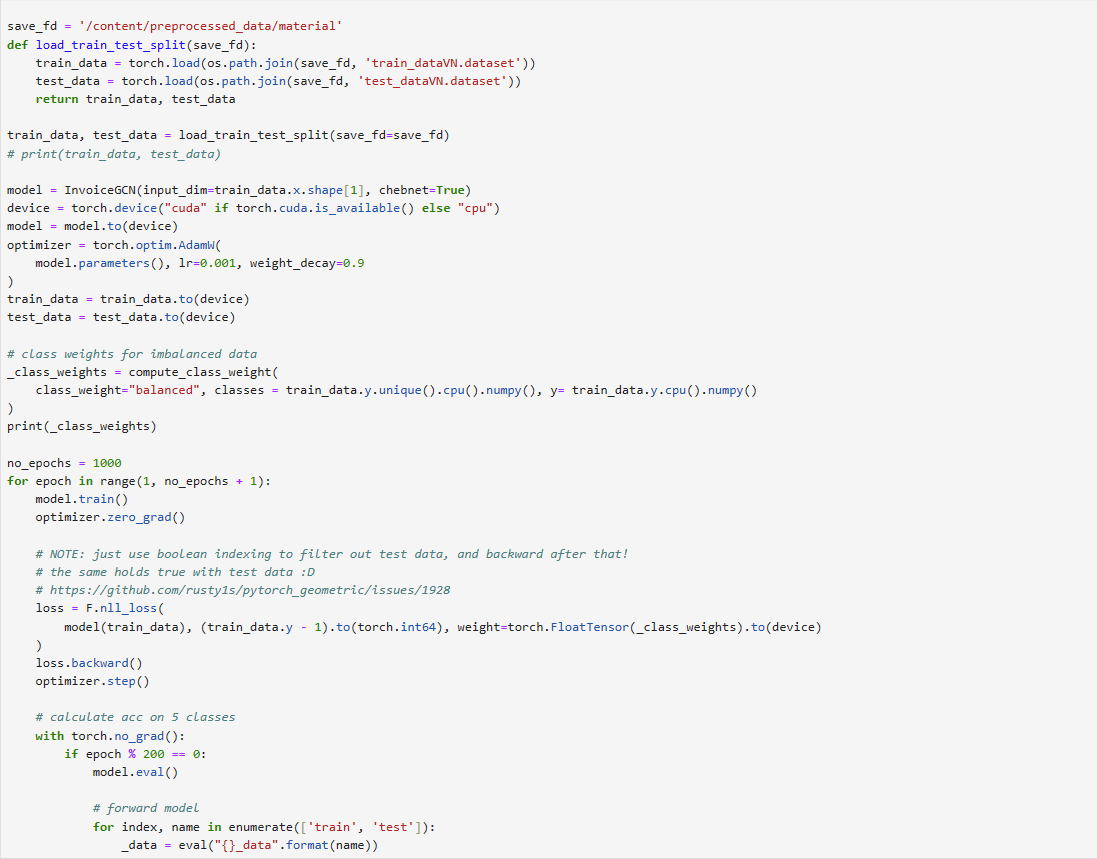
****

****

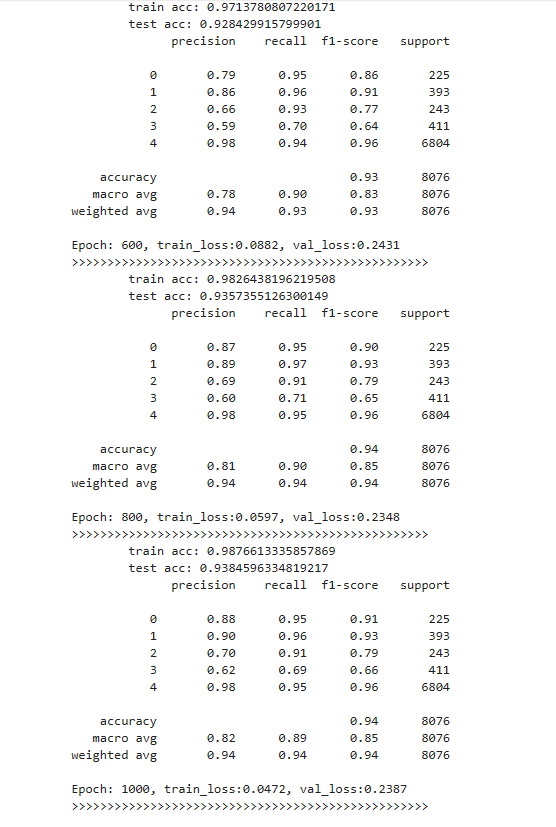
Hình ảnh 41: mô tả hàm lấy dữ liệu



Hình ảnh 42: Mô tả việc chia dữ liệu để thực hiện train và test

**HUẤN LUYỆN MÔ HÌNH ** Hình ảnh 43 : đoạn code mô tả quá trình train .

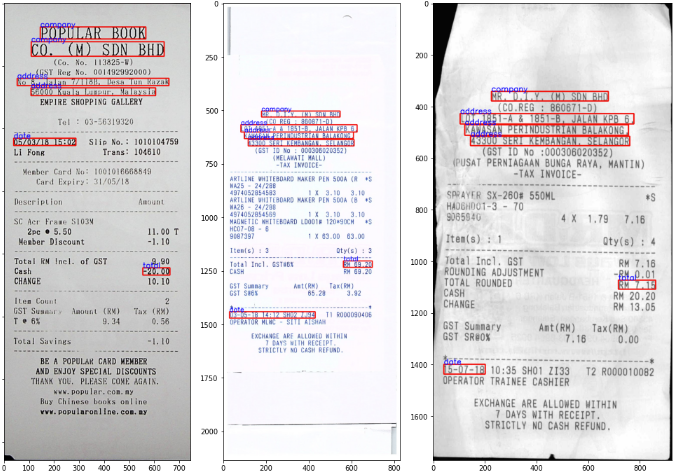
Kết quả của mô hình .

****

Hình ảnh 44 : mô tả kết quả của mô hình .

**KẾT QUẢ TRÊN TẬP TEST .**

* 1 vài kết quả trên tập test, còn 1 số ảnh bị dự đoán nhầm trên từng text box nhưng nhìn chung kết quả thu được khá ổn:

****

Hình ảnh 45 : một số hình ảnh tệp test .

**KẾT LUẬN**

Trên đây là bài chia sẻ về việc áp dụng mô hình Graph Neural Network vào 1 bài toán khá đặc thù là Key Informatrion Extraction .

Sử dụng Graph Convolutional Networks (GCN) trong quá trình đọc hóa đơn đánh dấu một bước quan trọng trong ứng dụng của trí tuệ nhân tạo vào lĩnh vực tài chính. GCN, với khả năng hiểu biểu đồ và quan hệ giữa các yếu tố, giúp tối ưu hóa quá trình trích xuất thông tin từ hóa đơn, từ đó cải thiện độ chính xác và hiệu suất của hệ thống. Việc tích hợp GCN giúp phát hiện các mối quan hệ phức tạp giữa các yếu tố khác nhau trong hóa đơn, bao gồm các thông tin như số lượng sản phẩm, giá cả, và thông tin đối tác kinh doanh.

Hơn nữa, GCN còn đóng vai trò quan trọng trong việc giải quyết vấn đề động và thay đổi trong cấu trúc của hóa đơn. Khả năng này làm cho hệ thống trở nên linh hoạt và có khả năng thích ứng với các định dạng hóa đơn khác nhau từ nhiều nguồn thông tin. Việc tận dụng sức mạnh của GCN giúp tăng cường khả năng tự động hóa trong quá trình xử lý hóa đơn, giảm thiểu sai sót và tăng cường năng suất.

Tổng cộng, việc sử dụng GCN trong đọc hóa đơn không chỉ đem lại những cải tiến đáng kể về độ chính xác và hiệu suất, mà còn giúp tạo ra những ứng dụng mạnh mẽ và linh hoạt trong lĩnh vực quản lý tài chính và kế toán. Đây là một bước quan trọng trong việc áp dụng công nghệ AI để giải quyết các thách thức thực tế và đưa ra những giải pháp hiệu quả cho doanh nghiệp và tổ chức.

**TÀI LIỆU THAM KHẢO**

<https://viblo.asia/p/deep-learning-key-information-extraction-from-document-using-graph-convolution-network-bai-toan-trich-rut-thong-tin-tu-hoa-don-voi-graph-convolution-network-djeZ1yPGZWz>

<https://github.com/huyhoang17/KIE_invoice_minimal>

<https://viblo.asia/p/deep-learning-graph-neural-network-a-literature-review-and-applications-6J3ZgP0qlmB>

<https://viblo.asia/p/deep-learning-graph-neural-network-a-literature-review-and-applications-6J3ZgP0qlmB>

<https://viblo.asia/p/xay-dung-mo-hinh-khong-gian-vector-cho-tieng-viet-GrLZDXr2Zk0>

<https://viblo.asia/p/machine-learning-ung-dung-mo-hinh-word2vec-cho-bai-toan-session-based-recommender-system-Qbq5Q4YJlD8>

<https://viblo.asia/p/lecture-2-simple-word-vector-representations-word2vec-gGJ59bNxKX2>

<https://pytorch-geometric.readthedocs.io/en/latest/notes/batching.html>

<https://stackoverflow.com/questions/42154606/python-numpy-how-to-construct-a-big-diagonal-arraymatrix-from-two-small-array>

<https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.block_diag.html>

<https://link.springer.com/chapter/10.1007/978-3-030-21074-8_12>

https://cinnamonai.medium.com/key-value-extraction-using-graph-key-value-68718e1a4036?\_\_cf\_chl\_rt\_tk=ZjTbUM42nIzGkIB0N1dzRNcp..uf2k8Aq3qvci4Nw40-1705811310-0-gaNycGzNDuU