**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

NGUYỄN THỊ HÀ PHƯƠNG – CNTT

NGUYỄN LÊ TRUNG HIẾU – CNTT

**ĐỀ CƯƠNG**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**XÂY DỰNG HỆ THỐNG TRÍCH XUẤT VÀ TRUY VẤN THÔNG TIN TỪ VĂN BẢN HÀNH CHÍNH**

HÀ NỘI, 4/2025

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

TRẦN THỊ NGỌC KHÁNH – CNTT

**ĐỀ CƯƠNG**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**XÂY DỰNG HỆ THỐNG PHÁT HIỆN SAO CHÉP TÀI LIỆU SỬ DỤNG PHƯƠNG PHÁP XẤP XỈ**

HÀ NỘI, 4/2025

**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**HỌC VIỆN KỸ THUẬT QUÂN SỰ**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

NGUYỄN THỊ HÀ PHƯƠNG – CNTT

NGUYỄN LÊ TRUNG HIẾU – CNTT

**ĐỀ CƯƠNG**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**XÂY DỰNG HỆ THỐNG TRÍCH XUẤT VÀ TRUY VẤN THÔNG TIN TỪ VĂN BẢN HÀNH CHÍNH**

GVHD: TS Vũ Thị Ly- V4

HÀ NỘI, 4/2025

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

NGUYỄN THỊ HÀ PHƯƠNG – CNTT

NGUYỄN LÊ TRUNG HIẾU – CNTT

**ĐỀ CƯƠNG**

**ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**

**XÂY DỰNG HỆ THỐNG TRÍCH XUẤT VÀ TRUY VẤN THÔNG TIN TỪ VĂN BẢN HÀNH CHÍNH**

HÀ NỘI, 4/2025

**MỤC LỤC**

[1. Thông tin đồ án 3](#_Toc196723069)

[1.1. Thông tin đề tài 3](#_Toc196723070)

[1.2. Thông tin sinh viên thực hiện 3](#_Toc196723071)

[1.3. Thông tin giáo viên hướng dẫn 3](#_Toc196723072)

[2. Cơ sở lý thuyết 3](#_Toc196723073)

[2.1. Xử lý hình ảnh và phát hiện văn bản 3](#_Toc196723074)

[2.2. Nhận diện văn bản (OCR) 3](#_Toc196723075)

[2.3. Trích xuất thông tin và phân loại văn bản 4](#_Toc196723076)

[2.4. Truy vấn thông tin với RAG 4](#_Toc196723077)

[2.5. Bài toán 4](#_Toc196723078)

[3. Kế hoạch thực hiện 4](#_Toc196723079)

[4. Cấu trúc báo cáo đồ án tốt nghiệp dự kiến 5](#_Toc196723080)

[5. Tài liệu tham khảo 8](#_Toc196723081)

[6. Cơ sở lý thuyết, kịch bản thử nghiệm và kết quả dự kiến 9](#_Toc196723082)

[6.1. Cơ sở lý thuyết 9](#_Toc196723083)

[6.1.1. Mô hình DBnet 9](#_Toc196723084)

[6.1.2. Mạng hồi quy RNN 12](#_Toc196723085)

[6.1.3. Mạng tích chập CNN 14](#_Toc196723086)

[6.1.4. CRNN (Convolutional Recurrent Neural Network) 19](#_Toc196723087)

[6.1.5. GNN (Graph Neural Network) 24](#_Toc196723088)

[6.1.6. LLM (Large Language Model) 30](#_Toc196723089)

[6.1.7. RAG (Retrieval Augmented Generation) 32](#_Toc196723090)

[6.2. Kịch bản thử nghiệm 37](#_Toc196723091)

[6.2.1. Mục tiêu thử nghiệm 37](#_Toc196723092)

[6.2.2. Chuẩn bị thử nghiệm 37](#_Toc196723093)

[6.2.3. Kịch bản thử nghiệm End-to-End 38](#_Toc196723094)

[6.3. Kết quả dự kiến 44](#_Toc196723095)

[6.3.1. Sản phẩm dự kiến 44](#_Toc196723096)

[6.3.2. Đóng góp khoa học 47](#_Toc196723097)

[6.3.3. Đóng góp thực tiễn 48](#_Toc196723098)

# 1. Thông tin đồ án

## 1.1. Thông tin đề tài

**Tên đề tài**: Xây dựng hệ thống trích xuất và truy vấn thông tin từ văn bản hành chính

**Mã số đề tài**: 25.V4.10

**Lĩnh vực đăng ký thực hiện**: Công nghệ thông tin

**Mục đích đề tài**: Phát triển hệ thống tự động trích xuất thông tin từ văn bản hành chính sử dụng LLM và GNN kết hợp DBNet-CRNN, phân loại thông tin theo các mặt công tác (chính trị, đào tạo, quân sự hành chính, hậu cần-kỹ thuật...) và danh mục (công văn, quyết định, thông báo...). Hệ thống tích hợp chatbot RAG để truy vấn thông tin, hỗ trợ tìm kiếm công văn, quản lý tài liệu, và số hóa, đáp ứng nhu cầu chuyển đổi số trong quản lý văn bản hành chính.

## 1.2. Thông tin sinh viên thực hiện

Nguyễn Lê Trung Hiếu (Lớp CNTT – Đại đội 156 – Tiểu đoàn 1)

Nguyễn Thị Hà Phương (Lớp CNTT – Đại đội 156 – Tiểu đoàn 1)

## 1.3. Thông tin giáo viên hướng dẫn

GV.TS. Vũ Thị Ly - Bộ môn Khoa học máy tính - V4

# 2. Cơ sở lý thuyết

## 2.1. Xử lý hình ảnh và phát hiện văn bản

Xử lý hình ảnh bao gồm các kỹ thuật như làm mịn, chuẩn hóa và tăng cường độ tương phản để chuẩn bị cho việc phát hiện văn bản trên văn bản hành chính. Mô hình học sâu **DBNet** được sử dụng để xác định các vùng văn bản (số văn bản, ngày ban hành, nội dung, v.v.) với kỹ thuật nhị phân hóa có thể vi phân (Differentiable Binarization) và ngưỡng động, phù hợp với các tài liệu có bố cục phức tạp. DBNet đảm bảo phát hiện chính xác các vùng văn bản trong công văn, quyết định, hoặc thông báo tiếng Việt.

## 2.2. Nhận diện văn bản (OCR)

Nhận diện văn bản sử dụng mô hình **CRNN** (Convolutional Recurrent Neural Network) để chuyển đổi hình ảnh văn bản thành chuỗi ký tự. CRNN được tinh chỉnh trên tập dữ liệu văn bản hành chính tiếng Việt, xử lý các đặc điểm như dấu thanh, từ ghép, và số văn bản. Mô hình kết hợp các lớp CNN để trích xuất đặc trưng hình ảnh và RNN để mô hình hóa chuỗi ký tự, phù hợp với cấu trúc phức tạp của văn bản hành chính. Kỹ thuật hậu xử lý được áp dụng để sửa lỗi nhận diện và chuẩn hóa dữ liệu văn bản trước khi chuyển sang bước trích xuất thông tin.

## 2.3. Trích xuất thông tin và phân loại văn bản

Hệ thống sử dụng kết hợp hai phương pháp tiên tiến để trích xuất thông tin từ văn bản. **GNN (RGCN)** được ứng dụng để trích xuất thông tin có cấu trúc (số văn bản, ngày ban hành, người ký, đơn vị ban hành) thông qua việc mô hình hóa mối quan hệ giữa các thành phần văn bản. **LLM** (như BERT hoặc Grok) được sử dụng để phân tích ngữ nghĩa và trích xuất thông tin phi cấu trúc (nội dung chính, mục đích, ý nghĩa văn bản). Dựa trên kết quả trích xuất, hệ thống thực hiện phân loại văn bản theo mặt công tác (chính trị, đào tạo, quân sự hành chính, hậu cần-kỹ thuật) và danh mục (công văn, quyết định, thông báo).

## 2.4. Truy vấn thông tin với RAG

**Retrieval-Augmented Generation (RAG)** kết hợp tìm kiếm thông tin (retrieval) và sinh văn bản (generation) để trả lời truy vấn người dùng. Tài liệu và thông tin trích xuất được vector hóa bằng **Sentence-BERT** và lưu trữ trong **FAISS** hoặc Elasticsearch. LLM sinh câu trả lời tự nhiên dựa trên tài liệu truy xuất, hỗ trợ truy vấn phức tạp như “Tìm công văn về tài chính năm 2025” hoặc “Tóm tắt nội dung công văn số 123”. RAG đảm bảo câu trả lời chính xác và ngữ cảnh phù hợp.

## 2.5. Bài toán

Bài toán đặt ra là phát triển hệ thống tự động trích xuất thông tin từ văn bản hành chính tiếng Việt (số văn bản, ngày ban hành, nội dung, người ký, v.v.), phân loại theo mặt công tác và danh mục, đồng thời hỗ trợ truy vấn thông tin qua chatbot RAG. Hệ thống cần vượt qua hạn chế của các giải pháp OCR truyền thống, đảm bảo độ chính xác cao, xử lý tài liệu phức tạp, và cung cấp giao diện thân thiện để quản lý tài liệu, tìm kiếm công văn, và trả lời truy vấn trong quản lý hành chính.

# 3. Kế hoạch thực hiện

|  |  |  |  |
| --- | --- | --- | --- |
| **Tuần** | **Thành viên 1 (OCR & Trích xuất thông tin)** | **Thành viên 2 (RAG & Backend/UI)** | **Tích hợp chung** |
| 1-2 | Thu thập & tiền xử lý dữ liệu văn bản hành chính | | Thiết kế kiến trúc hệ thống tổng thể |
| 3-4 | Xây dựng & huấn luyện DBNet | Chuẩn bị môi trường huấn luyện RAG (FAISS, Sentence-BERT) | Thống nhất API giữa các module |
| 5-6 | Huấn luyện & tinh chỉnh CRNN | Xây dựng mô hình vector hóa tài liệu | Kiểm tra tích hợp sơ bộ |
| 7-8 | Fine-tune LLM cho trích xuất thông tin | Xây dựng backend cơ bản (Node.js, Express) | Họp đánh giá tiến độ giữa kỳ |
| 9-10 | Phát triển mô hình GNN (RGCN) | Phát triển giao diện người dùng (React.js) và chatbot | Thử nghiệm tích hợp OCR+Backend |
| 11-12 | Cải tiến module OCR & trích xuất thông tin | Cải tiến module RAG & giao diện chatbot | Thử nghiệm tích hợp RAG+Backend |
| 13-14 | Phát triển module phân loại mặt công tác/danh mục | Hoàn thiện giao diện người dùng và quản lý tài liệu | Tích hợp toàn bộ hệ thống |
| 15-16 | Viết báo cáo kỹ thuật & tài liệu phần OCR, trích xuất | Viết báo cáo kỹ thuật & tài liệu cho RAG, UI | Hoàn thiện báo cáo tổng thể & chuẩn bị demo |

# 4. Cấu trúc báo cáo đồ án tốt nghiệp dự kiến

MỤC LỤC

DANH MỤC HÌNH ẢNH

CHỮ VIẾT TẮT

CHƯƠNG 1: GIỚI THIỆU

1.1. Tổng quan về vấn đề nghiên cứu

1.1.1. Bối cảnh và nhu cầu thực tiễn

1.1.2. Thách thức của các giải pháp hiện tại

1.2. Mục tiêu và phạm vi nghiên cứu

1.2.1. Mục tiêu nghiên cứu

1.2.2. Phạm vi nghiên cứu

1.3. Ý nghĩa của đề tài

1.3.1. Ý nghĩa khoa học

1.3.2. Ý nghĩa thực tiễn

1.4. Cấu trúc báo cáo

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN NGHIÊN CỨU

2.1. Đặc điểm và yêu cầu xử lý văn bản hành chính

2.1.1. Đặc điểm văn bản hành chính tiếng Việt

2.1.2. Yêu cầu kỹ thuật và pháp lý

2.2. Công nghệ nền tảng

2.2.1. Phát hiện và nhận diện văn bản (DBNet, CRNN)

2.2.2. Trích xuất và phân loại thông tin (LLM, GNN)

2.2.3. Truy vấn thông tin thông minh (RAG)

2.2.4. Quản lý và tìm kiếm tài liệu (MongoDB, Elasticsearch)

2.3. Tổng quan các giải pháp hiện có

2.3.1. Giải pháp thương mại

2.3.2. Giải pháp mã nguồn mở

2.4. Phân tích hạn chế và định hướng nghiên cứu

2.4.1. Hạn chế của các giải pháp hiện có

2.4.2. Định hướng phát triển hệ thống đề xuất

CHƯƠNG 3: PHƯƠNG PHÁP NGHIÊN CỨU

3.1. Quy trình nghiên cứu

3.1.1. Thu thập và tiền xử lý dữ liệu

3.1.2. Thiết kế và phát triển hệ thống

3.2. Kiến trúc hệ thống

3.2.1. Sơ đồ kiến trúc tổng thể

3.2.2. Các thành phần chính và luồng dữ liệu

3.3. Pipeline trích xuất thông tin

3.3.1. Phát hiện văn bản (DBNet)

3.3.2. Nhận diện văn bản (CRNN)

3.3.3. Trích xuất thông tin (LLM/ GNN)

3.4. Hệ thống truy vấn thông tin

3.4.1. Kiến trúc RAG

3.4.2. Chatbot truy vấn thông minh

3.5. Hệ thống quản lý tài liệu và giao diện

3.5.1. Quản lý và lưu trữ tài liệu

3.5.2. Giao diện người dùng

3.6. Huấn luyện và đánh giá

3.6.1. Quy trình huấn luyện mô hình

3.6.2. Thước đo và phương pháp đánh giá

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1. Kết quả triển khai hệ thống

4.1.1. Kết quả pipeline trích xuất thông tin

4.1.2. Kết quả phân loại thông tin

4.1.3. Kết quả truy vấn thông tin

4.1.4. Kết quả quản lý tài liệu và giao diện

4.2. Đánh giá hiệu suất

4.2.1. Đánh giá pipeline trích xuất và phân loại

4.2.2. Đánh giá hệ thống truy vấn RAG

4.2.3. Đánh giá trải nghiệm người dùng

4.3. Phân tích trường hợp thử nghiệm

4.3.1. Xử lý công văn hành chính

4.3.2. Truy vấn thông tin qua chatbot

4.3.3. Quản lý và tìm kiếm tài liệu

4.4. So sánh với các giải pháp hiện có

4.4.1. So sánh với giải pháp thương mại

4.4.2. So sánh với giải pháp mã nguồn mở

4.5. Thảo luận

4.5.1. Các phát hiện chính

4.5.2. Thách thức và giải pháp khắc phục

CHƯƠNG 5: KẾT LUẬN VÀ KIẾN NGHỊ

5.1. Tóm tắt kết quả

5.2. Đóng góp của đề tài

5.2.1. Đóng góp khoa học

5.2.2. Đóng góp thực tiễn

5.3. Hạn chế của đề tài

5.4. Kiến nghị

5.4.1. Cải tiến kỹ thuật

5.4.2. Hướng nghiên cứu và triển khai tiếp theo

TÀI LIỆU THAM KHẢO

PHỤ LỤC (NẾU CÓ)

# 5. Tài liệu tham khảo

[1]. Liao, M., Wan, Z., Yao, C., Chen, K., & Bai, X. (2020). Real-Time Scene Text Detection with Differentiable Binarization and Adaptive Scale Fusion. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(07), 11462-11469.

[2]. Shi, B., Bai, X., & Yao, C. (2016). An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(11), 2298-2304.

[3]. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33, 9459-9474.

[4]. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2020). A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1), 4-24.

[5]. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., ... & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

[6]. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 4171-4186.

[7]. Nguyễn, T. H., & Trần, V. C. (2023). Số hóa tài liệu tiếng Việt: Thách thức và Giải pháp. *Tạp chí Nghiên cứu Công nghệ Thông tin*, 15(2), 45-60.

[8]. Izacard, G., & Grave, E. (2021). Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, 874-880.

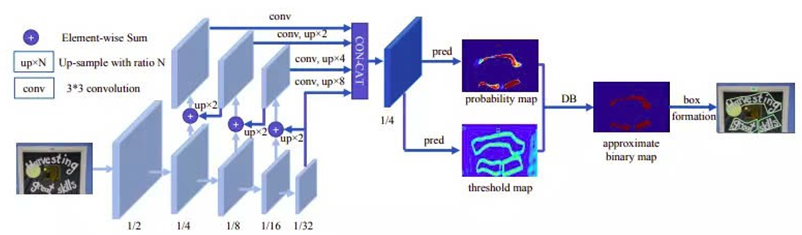
[9]. Asai, A., Wu, X., Wang, Y., Sil, A., Hajishirzi, M., & Yih, W. T. (2023). Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *arXiv preprint arXiv:2310.11511*.

# 6. Cơ sở lý thuyết, kịch bản thử nghiệm và kết quả dự kiến

## 6.1. Cơ sở lý thuyết

### 6.1.1. Mô hình DBnet

Các mô hình trong bài toán phát hiện chữ được nghiên cứu trước đây có quá trình hậu xử lý rất phức tạp để lấy được các thông tin vị trí của đối tượng trong ảnh. Các phương pháp này sẽ đặt một ngưỡng cố định để phân biệt đâu là điểm ảnh nền đâu là điểm ảnh đối tượng từ bản đồ xác suất được mô hình sinh ra. Việc đặt một ngưỡng cố định làm cho mô hình kém linh động và mất nhiều thời gian thử nghiệm để tìm ngưỡng phù hợp. Mô hình DB đã đế xuất một ý tưởng cho mô hình học luôn cách đặt giá trị ngưỡng này thông qua việc huấn luyện.



*Hình 1: Kiến trúc mô hình DBNet*

Kiến trúc mô hình được mô tả ở hình trên. Mô hình sẽ nhận ảnh đầu vào và cho qua một bộ trích xuất đặc trưng sinh ra bản đồ đặc trưng F*F*. Trong quá trình huấn luyện, bản đồ đặc trưng này sau đó được dùng để dự đoán bản đồ xác suất P và bản đồ ngưỡng T. Cuối cùng, sử dụng hai bản đồ này để tính ra bản đồ nhị phân được dùng cho quá trình hậu xử lý để lấy được dự đoán vị trí chữ trong ảnh sau này. Còn trong quá trình suy luận, chúng ta có thể lấy trực tiếp thông tin vị trí qua bản đồ xác xuất hoặc bản đồ nhị nhân.

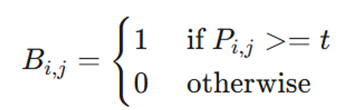
Kiến trúc mô hình DB có hai ưu điểm nổi bật so với các kiến trúc khác cùng dựa trên mô hình phân đoạn:

- Kiến trúc DB có thể đạt độ chính xác cao với những bộ trích xuất đặc trưng nhẹ như ResNet18 như trong bài báo.

- Có thể học được cách đặt ngưỡng động trên phương pháp gọi là Differentiable binarization.

**a) Standard binarization**

Standard binarization hay còn gọi là phương pháp nhị phân hóa chuẩn được rất nhiều mô hình phát hiện chữ sử dụng. Phương pháp này nhận đầu vào là một bản đồ xác suất P∈RHxW*P*∈*RHxW* được sinh ra bởi bộ trích xuất đặc trưng và sinh ra bản đồ nhị phân có kích thước bằng với bản đồ xác suất (H, W lần lượt chỉ chiều cao và chiều rộng của bản đồ). Thuật toán của phương pháp này nhận vào các giá trị điểm ảnh của bản đồ xác suất. Các giá trị điểm ảnh nào lớn hơn một ngưỡng cố định thì giá trị điểm ảnh trên bản đồ nhị phân tương ứng nhận giá trị 1 thể hiện điểm ảnh đó thuộc về đối tượng chữ, còn 0 thì điểm ảnh đó không phải đối tượng chữ.

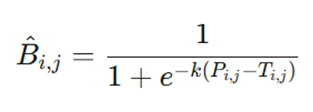


Ở đây t là một ngưỡng cố định, (i, j) là tọa độ vị trí điểm ảnh trên bản đồ.

Phương pháp này có nhược điểm là giá trị ngưỡng bị đặt cố định cho tất cả vị trí trên bản đồ. Do đó về mặt chủ quan sẽ gây nhẫm lần ở các điểm ảnh mà giá trị ngưỡng cần thấp hoặc cao hơn và mất thời gian để tìm ra một ngưỡng phù hợp cho bài toán.

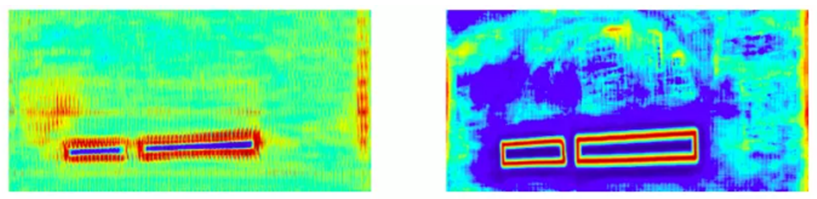
**b) Differentiable binarization**

Để giải quyết vấn đề của phương pháp Standard binarization, bài báo đề xuất phương pháp Differentiable binarization. Do phương pháp mới này có thể thực hiện tính toán giá trị đạo hàm nên có thể tích hợp trực tiếp vào mô hình học sâu.



trong đó là bản đồ nhị phân, T bản đồ ngưỡng động được học từ mô hình, k là một chỉ số khuyếch đại. Công thức differ\_binarization giống với công thức standard\_binarization nhưng có thể đạo hàm do đó có thể tối ưu cùng với mô hình. Phương pháp này kết hợp sử dụng ngưỡng động được trình bày phần tiếp theo không chỉ giúp phát hiện đối tượng so với nền mà còn giúp tách các vùng chữ bị ghép liền vào với nhau.

**c) Ngưỡng động (Adaptive threshold)**

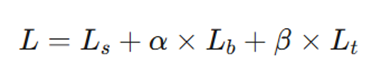


*Hình 2: Bản đồ ngưỡng động*

Bản đồ ngưỡng động được sinh ra từ bản đồ đặc trưng của mô hình phân đoạn. Cùng sinh ra từ bản đồ đặc trưng như bản đồ xác suất nhưng thay vì thể hiện xác suất điểm ảnh này có phải thuộc đối tương hay không thì bản đồ ngưỡng động thể hiện các điểm ảnh đó có thuộc cạnh bao quanh đối tượng đó không? Ở trong bài báo tác giả cũng so sánh bản đồ ngưỡng động được học không giám sát ở ảnh trái và có giám sát. Kết quả việc huấn luyện có giám sát tốt hơn hẳn so với phương pháp không giám sát.

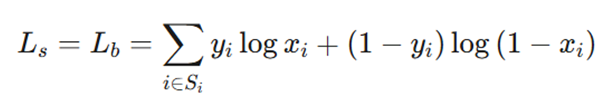
**d) Hàm mất mát**

Hàm mất mát tác giả sử dụng là tổng có trọng số của ba hàm mất mát cho bản đồ xác suất Ls*Ls*​, bản độ nhị phân Lb*Lb*​ và bản đồ ngưỡng Lt*Lt*​:

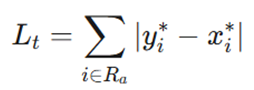
​

trong đó α, β lần lượt là trọng số ưu tiên của hàm mất mát cho bản đồ xác suất và bản đồ ngưỡng.

Do số lượng các điểm ảnh không phải chữ thường chiếm rất nhiều trong một ảnh nên để giảm hiện tượng mất cân bằng giữa các lớp, bài báo đề xuất sử dụng hàm mất mát binary cross-entropy (BCE) đồng thời hạn chế số lượng đối tượng không phải chữ được đưa vào hàm mất mát. Tập hợp các đối tượng không phải chữ được cho vào mất mát được gọi là St*St*​:



Hàm mất mát cho bản đồ ngưỡng được tính bằng tổng khoảng cách L1 giữa nhãn và dự đoán.



Trong đó Rd​ là tập hợp các vị trí bên trong vùng chữ, yi∗*yi*∗​ là nhãn cho bản đồ ngưỡng.

### 6.1.2. Mạng hồi quy RNN

Để có thể hiểu rõ về RNN, trước tiên chúng ta cùng nhìn lại mô hình Neural Network dưới đây:

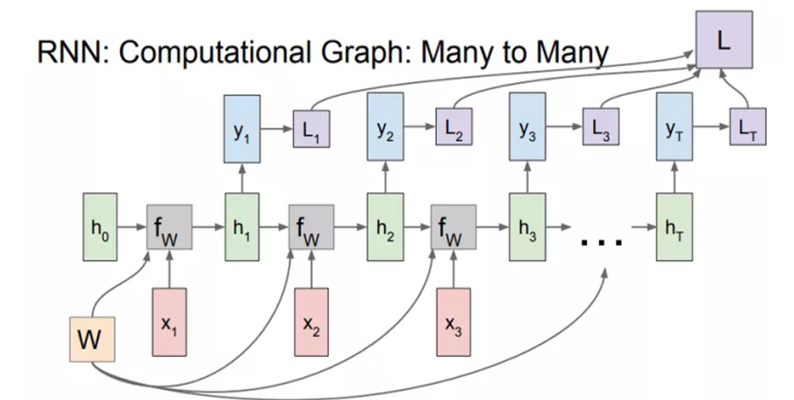
A diagram of a neural network

Description automatically generated

*Hình 3: Sơ đồ cấu trúc của Mạng Nơ-ron Nhân tạo (ANN)*

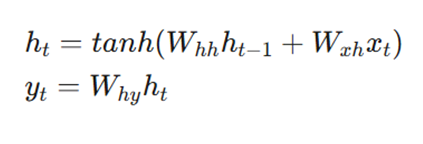
Như đã biết thì Neural Network bao gồm 3 phần chính là Input layer, Hidden layer và Output layer, ta có thể thấy là đầu vào và đầu ra của mạng neuron này là độc lập với nhau. Như vậy mô hình này không phù hợp với những bài toán dạng chuỗi như mô tả, hoàn thành câu, ... vì những dự đoán tiếp theo như từ tiếp theo phụ thuộc vào vị trí của nó trong câu và những từ đằng trước nó.

Và như vậy RNN ra đời với ý tưởng chính là sử dụng một bộ nhớ để lưu lại thông tin từ từ những bước tính toán xử lý trước để dựa vào nó có thể đưa ra dự đoán chính xác nhất cho bước dự đoán hiện tại. Nếu các bạn vẫn chưa hiểu gì thì hãy cùng xem mô hình mạng RNN sau và cùng phân tích để hiểu rõ hơn:

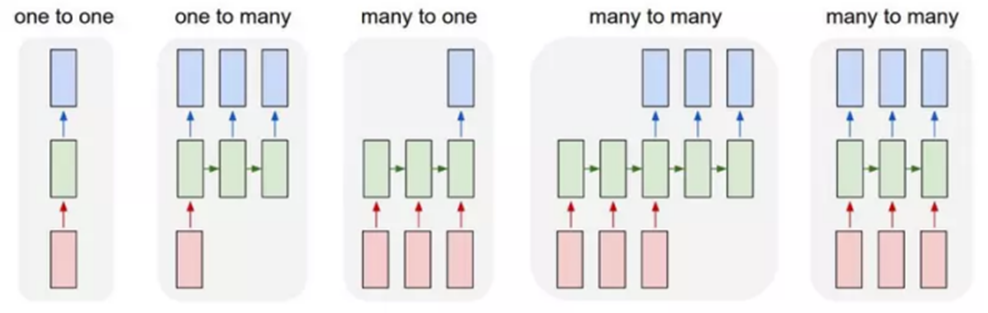


*Hình 4: Sơ đồ Mạng Nơ-ron Hồi tiếp (RNN) với kiến trúc Many-to-Many* Giải thích một chút: Nếu như mạng Neural Network chỉ là input layer x*x* đi qua hidden layer h*h* và cho ra output layer y*y* với full connected giữa các layer thì trong RNN, các input xt*xt*​ sẽ được kết hợp với hidden layer ht−1*ht*−1​ bằng hàm fW*fW*​ để tính toán ra hidden layer ht*ht*​ hiện tại và output yt*yt*​ sẽ được tính ra từ ht*ht*​, W*W* là tập các trọng số và nó được ở tất cả các cụm, các L1,L2,...,Lt*L*1​,*L*2​,...,*Lt*​ là các hàm mất mát sẽ được giải thích sau. Như vậy kết quả từ các quá trình tính toán trước đã được "nhớ" bằng cách kết hợp thêm ht−1*ht*−1​ tính ra ht*ht*​ để tăng độ chính xác cho những dự đoán ở hiện tại. Cụ thể quá trình tính toán được viết dưới dạng toán như sau:



Hàm chúng ta sẽ sử dụng hàm tanh, công thức trên sẽ trở thành :  
 ​  
 Đến đây có 3 thứ mới xuất hiện: Wxh,Whh,Why. Đối với mạng NN chỉ sử dụng một ma trận trọng số W duy nhất thì với RNN nó sử dụng 3 ma trận trọng số cho 2 quá trình tính toán: Whh​ kết hợp với "bộ nhớ trước" ht−1​ và Wxh kết hợp với xt để tính ra "bộ nhớ của bước hiện tại" ht​ từ đó kết hợp với Whyđể tính ra yt

Ngoài mô hình Many to Many như ta thấy ở trên thì RNN còn rất nhiều dạng khác như sau:



*Hình 5: Minh họa các Loại Kiến trúc RNN theo Quan hệ Đầu vào - Đầu ra*

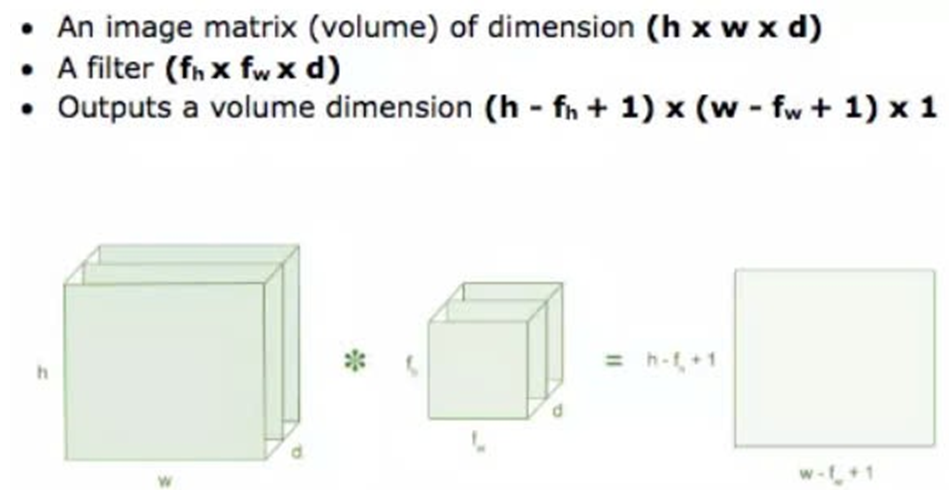
### 6.1.3. Mạng tích chập CNN

Trong mạng neural, mô hình mạng neural tích chập (CNN) là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

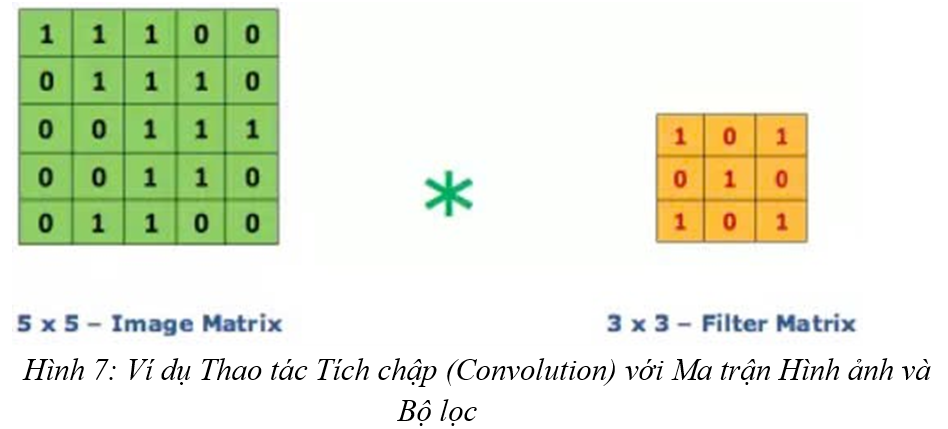
**a) Lớp tích chập - Convolution Layer**

Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vương nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân



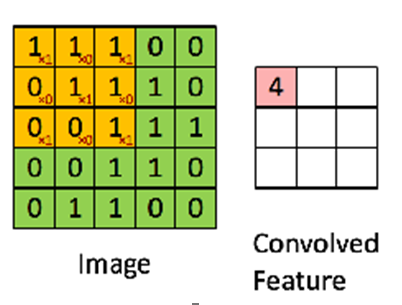
*Hình 6: Sơ đồ Thao tác Tích chập (Convolution) trên Khối Hình ảnh 3D*

Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.



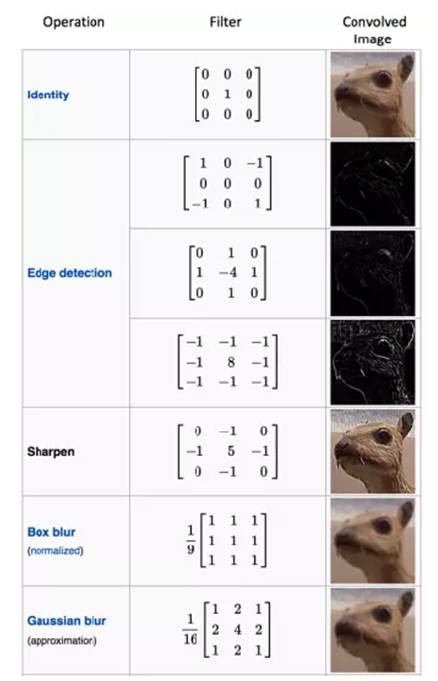
*Hình 7: Ví dụ Thao tác Tích chập (Convolution) với Ma trận Hình ảnh và Bộ lọc*

Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.



*Hình 8: Ma trận Đặc trưng sau Tích chập trong Xử lý Hình ảnh của DBNet*

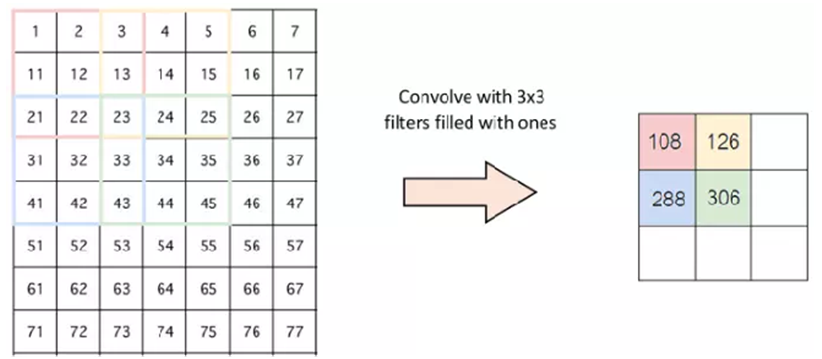
Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau



*Hình 9: Hiệu ứng của Các Bộ lọc Tích chập Khác nhau trên Hình ảnh*

**b) Bước nhảy - Stride**

Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi pixel và tiếp tục như vậy.



*Hình 10: Lớp tích chập hoạt động với stride là 2.*

**c) Đường viền - Padding**

Đôi khi kernel không phù hợp với hình ảnh đầu vào. Ta có 2 lựa chọn:

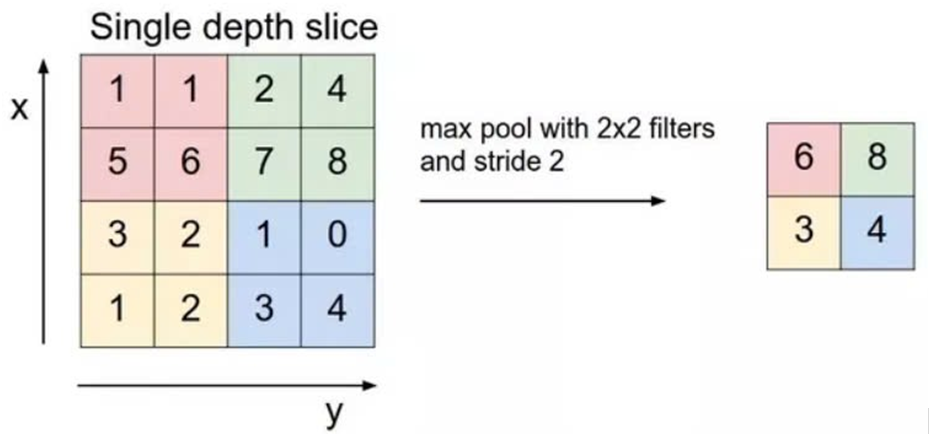
- Chèn thêm các số 0 vào 4 đường biên của hình ảnh (padding).

- Cắt bớt hình ảnh tại những điểm không phù hợp với kernel.

**d) Hàm phi tuyến - ReLU**

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: ƒ (x) = max (0, x).

Tại sao ReLU lại quan trọng: ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm



*Hình 11: Hàm Chuyển Đổi ReLU (ReLU Transfer Function)*

Có 1 số hà phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

**e) Lớp gộp - Pooling Layer**

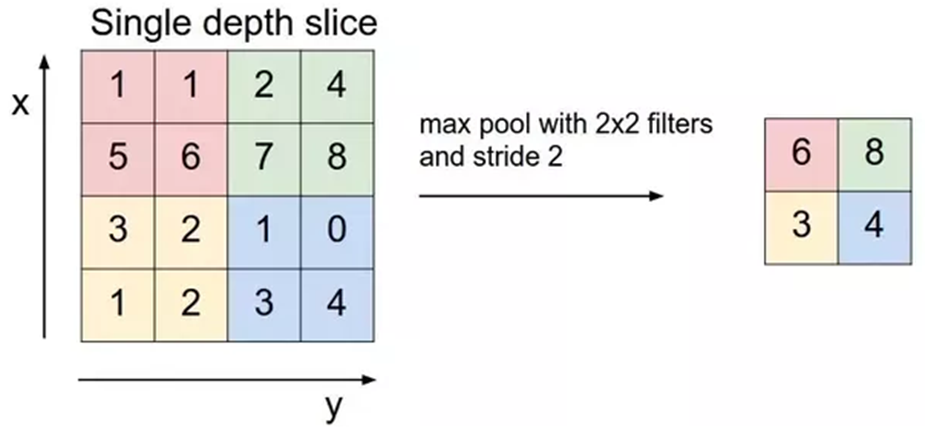
Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau:

- Max Pooling

- Average Pooling

- Sum Pooling

Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling.



*Hình 12: Minh họa Max Pooling 2×2 với bước nhảy 2*

### **6.1.4. CRNN (Convolutional Recurrent Neural** Network)

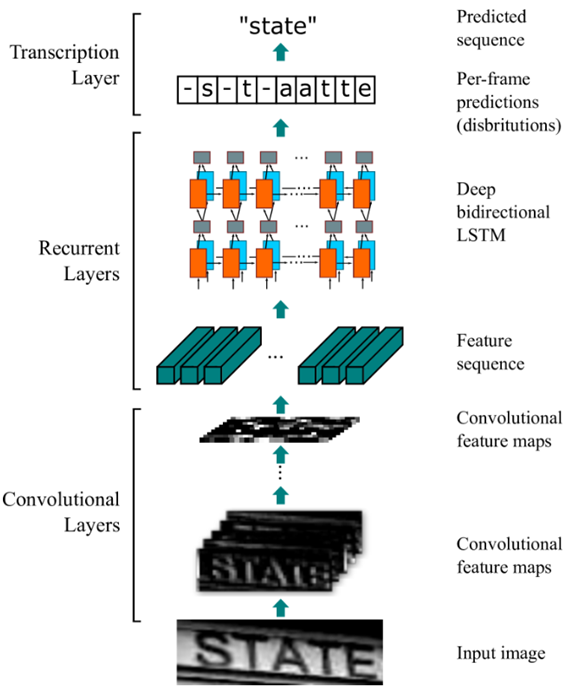
Mô hình CRNN (Convolutional Recurrent Neural Network) được đề xuất như một giải pháp hiệu quả, kết hợp mạng nơ-ron tích chập (CNN) và mạng nơ-ron hồi quy (RNN) để xử lý các chuỗi có độ dài thay đổi. Báo cáo này trình bày chi tiết kiến trúc mạng CRNN, bao gồm ba thành phần chính: lớp tích chập, lớp hồi quy và lớp phiên mã. Đặc biệt, CRNN có thể được huấn luyện đồng bộ với một hàm mất mát duy nhất, giúp đơn giản hóa quá trình tối ưu hóa và nâng cao hiệu quả nhận dạng. Mục tiêu của báo cáo là làm rõ cách CRNN hoạt động trong việc trích xuất đặc trưng, dự đoán chuỗi và phiên mã, từ đó đánh giá tiềm năng ứng dụng của nó trong các bài toán thực tế.

**a) Tổng quan**

Kiến trúc mạng CRNN được thiết kế với ba thành phần chính, được sắp xếp từ dưới lên trên như minh họa trong hình:

1. Lớp tích chập (Convolutional Layers): Trích xuất chuỗi đặc trưng từ hình ảnh đầu vào.
2. Lớp hồi quy (Recurrent Layers): Dự đoán phân phối nhãn cho từng khung trong chuỗi đặc trưng.
3. Lớp phiên mã (Transcription Layer): Chuyển đổi các dự đoán theo khung thành chuỗi nhãn cuối cùng.

CRNN tận dụng ưu điểm của CNN trong việc trích xuất đặc trưng không gian và RNN trong việc xử lý dữ liệu tuần tự, đồng thời tích hợp chúng thành một mạng thống nhất có thể huấn luyện end-to-end.



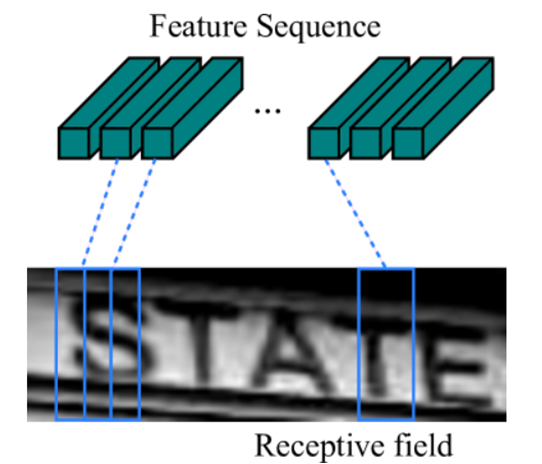
*Hình 13: Kiến trúc mạng CRNN cho nhận dạng văn bản*

**b) Trích xuất chuỗi đặc trưng**

Lớp tích chập trong CRNN được xây dựng dựa trên các lớp tích chập và gộp cực đại (max-pooling) từ một mô hình CNN tiêu chuẩn, nhưng không sử dụng các lớp kết nối đầy đủ. Thành phần này có nhiệm vụ tự động trích xuất chuỗi đặc trưng tuần tự từ hình ảnh đầu vào. Để đảm bảo tính đồng nhất, tất cả hình ảnh được điều chỉnh về cùng chiều cao trước khi đưa vào mạng. Sau đó, các bản đồ đặc trưng (feature maps) được tạo ra từ lớp tích chập, từ đó sinh ra một chuỗi các vectơ đặc trưng làm đầu vào cho lớp hồi quy.

Mỗi vectơ đặc trưng được trích xuất từ trái sang phải trên bản đồ đặc trưng theo cột. Cụ thể, vectơ thứ iii là sự kết hợp của cột thứ iii từ tất cả các bản đồ, với độ rộng mỗi cột được cố định là một pixel. Quá trình này đảm bảo rằng chuỗi đặc trưng phản ánh thứ tự không gian của hình ảnh gốc.

Các lớp tích chập, gộp cực đại và hàm kích hoạt hoạt động trên các vùng cục bộ, mang lại tính bất biến chuyển dịch. Do đó, mỗi cột trong bản đồ đặc trưng tương ứng với một vùng hình chữ nhật (trường tiếp nhận - receptive field) trên hình ảnh gốc, được sắp xếp từ trái sang phải. Như minh họa trong Hình 2, mỗi vectơ trong chuỗi đặc trưng gắn liền với một trường tiếp nhận và đóng vai trò như bộ mô tả hình ảnh cho vùng đó.



*Hình 14: Trường tiếp nhận*

Đặc trưng tích chập sâu có tính mạnh mẽ, phong phú và có thể huấn luyện, đã được áp dụng rộng rãi trong các tác vụ nhận dạng thị giác. Không giống các phương pháp truyền thống sử dụng CNN để trích xuất biểu diễn toàn cục, CRNN chuyển đổi đặc trưng sâu thành biểu diễn tuần tự, giúp xử lý hiệu quả các đối tượng dạng chuỗi với độ dài biến thiên, chẳng hạn như văn bản.

**c) Gắn nhãn chuỗi**

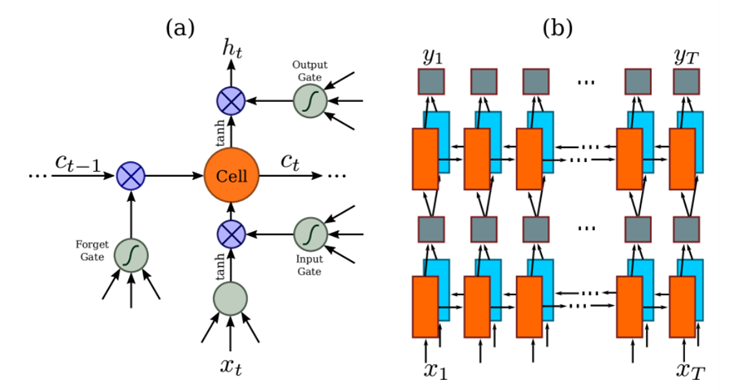
Lớp hồi quy trong CRNN là một mạng nơ-ron hồi quy hai chiều sâu (deep bidirectional RNN), được đặt trên lớp tích chập. Lớp này dự đoán phân phối nhãn yt cho mỗi khung xt  trong chuỗi đặc trưng x= x1, ..., xT​. Lớp hồi quy mang lại ba lợi ích chính:

- Nắm bắt ngữ cảnh: RNN có khả năng ghi nhận thông tin ngữ cảnh trong chuỗi, giúp nhận dạng ổn định hơn so với xử lý từng ký tự độc lập. Ví dụ, các ký tự rộng hoặc mơ hồ (như "il") dễ nhận diện hơn khi xem xét ngữ cảnh xung quanh.

- Lan truyền ngược lỗi: RNN cho phép lan truyền ngược sai số đến lớp tích chập, hỗ trợ huấn luyện đồng bộ toàn mạng.

- Xử lý chuỗi linh hoạt: RNN hoạt động trên các chuỗi có độ dài bất kỳ, phù hợp với các đối tượng dạng chuỗi.

RNN truyền thống gặp vấn đề gradient biến mất, hạn chế khả năng lưu trữ ngữ cảnh dài hạn. Để khắc phục, CRNN sử dụng Long-Short Term Memory (LSTM), với ô nhớ và ba cổng (cổng nhập, cổng xuất, cổng quên), cho phép nắm bắt phụ thuộc dài hạn. Vì ngữ cảnh từ cả hai hướng đều quan trọng trong chuỗi hình ảnh, nghiên cứu kết hợp một LSTM xuôi (forward) và một LSTM ngược (backward) thành bidirectional LSTM. Nhiều lớp bidirectional LSTM được xếp chồng tạo thành deep bidirectional LSTM (Hình 3), nâng cao khả năng trừu tượng hóa.



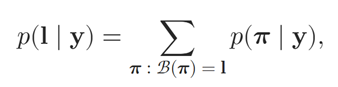
*Hình 15: (a) Cấu trúc đơn vị LSTM cơ bản với ô nhớ và ba cổng. (b) Cấu trúc deep bidirectional LSTM, kết hợp LSTM xuôi và ngược, xếp chồng nhiều lớp.*

Sai số trong lớp hồi quy được lan truyền ngược theo thời gian (Back-Propagation Through Time - BPTT). Tại đáy lớp hồi quy, chuỗi sai số được ghép lại thành bản đồ và truyền ngược về lớp tích chập qua lớp trung gian "Map-to-Sequence", đảm bảo sự liên kết giữa hai thành phần.

**d) Phiên mã**

Lớp phiên mã chuyển đổi các dự đoán theo khung từ RNN thành chuỗi nhãn cuối cùng, tìm chuỗi có xác suất cao nhất dựa trên dự đoán. Có hai chế độ phiên mã: không dùng từ điển (lexicon-free) và dùng từ điển (lexicon-based).

Nghiên cứu sử dụng lớp Connectionist Temporal Classification (CTC) để định nghĩa xác suất có điều kiện p(l∣y) cho chuỗi nhãn l dựa trên dự đoán y=y1​,…,yT . CTC không yêu cầu căn chỉnh vị trí ký tự, chỉ cần hình ảnh và chuỗi nhãn để huấn luyện. Xác suất được tính như sau:





Chuỗi nhãn l∗ được chọn bằng cách lấy nhãn có xác suất cao nhất tại mỗi bước thời gian và ánh xạ:



Với từ điển D, chuỗi nhãn được chọn là:



Để xử lý từ điển lớn, tìm kiếm được giới hạn trong các ứng cử viên lân cận Nδ(l′) dựa trên khoảng cách chỉnh sửa, sử dụng cây BK-tree với độ phức tạp O(log∣D∣).

**e) Huấn luyện mạng**

Tập dữ liệu huấn luyện X={Ii,li}i​ gồm hình ảnh Ii​ và chuỗi nhãn li​. Mục tiêu là tối thiểu hóa:



trong đó yi​ là đầu ra từ lớp tích chập và hồi quy.

Mạng được huấn luyện bằng gradient ngẫu nhiên (SGD) với lan truyền ngược. Lớp phiên mã sử dụng thuật toán tiến-lùi, lớp hồi quy dùng BPTT. Tối ưu hóa bằng ADADELTA tự động điều chỉnh tốc độ học, cho kết quả nhanh hơn phương pháp momentum.

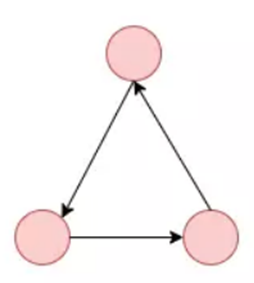
**f) Kết luận**

Kiến trúc CRNN là một giải pháp hiệu quả cho nhận dạng chuỗi dựa trên hình ảnh, tích hợp lớp tích chập để trích xuất đặc trưng, lớp hồi quy để dự đoán chuỗi, và lớp phiên mã để tạo nhãn cuối cùng. Với khả năng xử lý chuỗi độ dài biến thiên và tận dụng ngữ cảnh, CRNN mở ra tiềm năng ứng dụng rộng rãi trong các bài toán thực tế như nhận diện văn bản tiếng Việt. Việc huấn luyện đồng bộ với một hàm mất mát duy nhất càng tăng cường tính thực tiễn của mô hình.

### 6.1.5. GNN (Graph Neural Network)

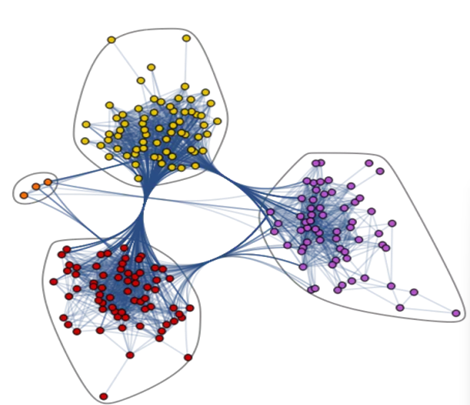
**a) Giới thiệu**

Đồ thị (Graph) là một cấu trúc dữ liệu quan trọng trong khoa học máy tính, được sử dụng để biểu diễn các mối quan hệ và tương tác giữa các thực thể. Với sự phát triển của trí tuệ nhân tạo, đồ thị đã trở thành nền tảng cho nhiều ứng dụng thực tế như phân tích mạng xã hội, hệ thống gợi ý, và nhận dạng mẫu. Đặc biệt, Mạng Nơ-ron Đồ thị (Graph Neural Networks - GNNs) là một loại mạng nơ-ron tiên tiến, được thiết kế để xử lý dữ liệu đồ thị, mở ra hướng tiếp cận mới cho các bài toán phức tạp liên quan đến mối quan hệ và cấu trúc. Trong ví dụ dưới đây, các hình tròn màu đỏ là các nút và các mũi tên là các cạnh. Hướng của các cạnh xác định các phụ thuộc giữa hai nút.



*Hình 16: Nút và cạnh*

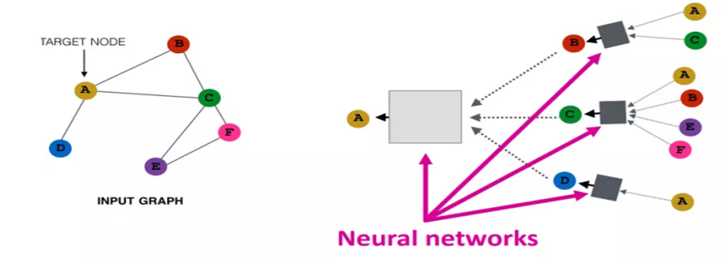
Một ví dụ thực tế hơn: Mạng lưới Nhạc Jazz - Jazz Musicians Network. Nó chứa 198 nút và 2742 cạnh. Trong biểu đồ đồ thị cộng đồng dưới đây, các màu sắc khác nhau của các nút đại diện cho các cộng đồng khác nhau của các nhạc sĩ Jazz và các cạnh kết nối giữa chúng. Có một mạng lưới hợp tác, trong đó một nhạc sĩ đơn lẻ có thể có mối quan hệ bên trong và bên ngoài cộng đồng



Hình 17: Biểu đồ đồ thị cộng đồng

**b) Mạng Nơ-ron Đồ thị (GNNs)**

Mạng Nơ-ron Đồ Thị (Graph Neural Network - GNN) là một loại mô hình học máy được thiết kế đặc biệt để làm việc với dữ liệu đồ thị. GNN có khả năng mở rộng và áp dụng trên các đồ thị có cấu trúc phức tạp, như mạng xã hội, mạng lưới giao thông, hay bất kỳ hệ thống nào có mối quan hệ giữa các đối tượng. GNN hoạt động bằng cách truyền thông tin qua các đỉnh và cạnh trong đồ thị. Mô hình học thông qua việc cập nhật và kết hợp thông tin từ các hàng xóm của mỗi đỉnh, cho phép nắm bắt thông tin cấu trúc và tương tác giữa các đối tượng trong đồ thị. Một trong những đặc điểm đáng chú ý của GNN là khả năng tích hợp thông tin từ cả đặc trưng của các đỉnh và cấu trúc đồ thị. Điều này cho phép GNN học mô hình phức tạp và biểu diễn các mối quan hệ phức tạp giữa các đối tượng trong đồ thị. GNN đã chứng tỏ được hiệu quả trong nhiều nhiệm vụ, bao gồm phân loại đồ thị, phân loại nút, dự đoán liên kết và nhúng đồ thị. Các ứng dụng của GNN rất đa dạng, từ phân tích mạng xã hội, gợi ý người dùng, cho đến phát hiện và kiểm soát các hiện tượng trong các hệ thống phức tạp.



*Hình 18: Minh Họa Mạng Nơ-ron Đồ Thị (Graph Neural Network - GNN)*

Đồ thị đầu vào được đi qua một loạt mạng neural. Cấu trúc đồ thị đầu vào được chuyển đổi thành nhúng đồ thị, cho phép chúng ta duy trì thông tin về các nút, cạnh và ngữ cảnh toàn cục. Sau đó, vectơ đặc trưng của các nút A và C được thông qua lớp mạng neural. Nó tổng hợp những đặc trưng này và truyền chúng vào lớp tiếp theo.

Tuy GNN đã mang lại nhiều tiến bộ, nhưng vẫn còn nhiều thách thức trong việc khai thác toàn bộ tiềm năng của dữ liệu đồ thị và tăng tính khả chuyển của mô hình. Các nghiên cứu về GNN đang tiếp tục phát triển để nâng cao hiệu suất và ứng dụng của mô hình trong các lĩnh vực khác nhau.

**c) Các loại GNN phổ biến**

Mạng Nơ-ron Đồ thị (Graph Neural Networks - GNNs) là một nhóm các mô hình học sâu được thiết kế đặc biệt để xử lý dữ liệu dạng đồ thị. Tùy thuộc vào cách tiếp cận và mục tiêu ứng dụng, GNNs có nhiều biến thể khác nhau. Trong phần này, chúng ta sẽ đi sâu vào bốn loại GNN phổ biến nhất: Mạng Tích chập Đồ thị (GCNs), Mạng Mã hóa Tự động Đồ thị (Graph Auto-Encoders), Mạng Hồi quy Đồ thị (RGNNs), và Mạng Đồ thị Có Cổng (GGNNs). Mỗi loại đều có cách tiếp cận riêng để khai thác thông tin từ cấu trúc đồ thị, từ đó giải quyết các bài toán như phân loại nút, dự đoán liên kết, hay tạo đồ thị

*\* Mạng Tích chập Đồ thị (Graph Convolutional Networks - GCNs)*

Thay vì xử lý dữ liệu lưới (grid) như hình ảnh, GCNs hoạt động trên dữ liệu đồ thị phi Euclid, nơi các nút và cạnh không có thứ tự cố định. GCNs học đặc trưng bằng cách tổng hợp thông tin từ các nút lân cận, giúp khai thác mối quan hệ cục bộ trong đồ thị

Quy trình xử lý của GCNs bao gồm các bước chính sau:

- GCNs thu thập thông tin từ các nút lân cận (neighbors) của một nút mục tiêu bằng cách sử dụng ma trận kề (adjacency matrix) và vectơ đặc trưng của các nút. Quá trình này thường được thực hiện thông qua phép nhân ma trận hoặc phép tích chập trên đồ thị.

- Sau khi tổng hợp, vectơ đặc trưng được truyền qua một lớp nơ-ron dày đặc để chuyển đổi thông tin thành dạng biểu diễn cấp cao hơn.

- Một hàm kích hoạt như ReLU hoặc sigmoid được áp dụng để thêm tính phi tuyến, giúp mô hình học được các mẫu phức tạp hơn.

GCNs được chia thành hai biến thể chính dựa trên cách tiếp cận tích chập:

- Spatial GCNs (Tích chập Không gian): Dựa trên đặc trưng không gian của đồ thị, Spatial GCNs trực tiếp tổng hợp thông tin từ các nút lân cận dựa trên cấu trúc vật lý của đồ thị. Phương pháp này tương tự cách CNNs trích xuất đặc trưng từ các vùng lân cận trong hình ảnh, nhưng được điều chỉnh cho dữ liệu đồ thị không đều.

- Spectral GCNs (Tích chập Phổ): Sử dụng phân tích phổ của ma trận Laplacian đồ thị (graph Laplacian matrix) để lan truyền thông tin giữa các nút. Phương pháp này dựa trên lý thuyết tín hiệu, biến đổi đồ thị thành miền tần số để xử lý, sau đó áp dụng tích chập phổ để học đặc trưng.

GCNs được sử dụng rộng rãi trong các bài toán như phân loại nút (node classification), dự đoán liên kết (link prediction), và phân loại đồ thị (graph classification). Ví dụ, trong mạng xã hội, GCNs có thể dự đoán sở thích của một người dựa trên thông tin từ bạn bè của họ.

*\* Mạng Mã hóa Tự động Đồ thị (Graph Auto-Encoders)*

Mạng Mã hóa Tự động Đồ thị (Graph Auto-Encoders) là một loại GNN sử dụng kiến trúc mã hóa-giải mã (encoder-decoder) để học biểu diễn đồ thị trong không gian vectơ. Mục tiêu chính của mô hình này là tái tạo lại cấu trúc đồ thị từ dữ liệu đầu vào, đồng thời tạo ra một biểu diễn nén (embedding) có thể sử dụng cho các tác vụ khác.

Graph Auto-Encoders bao gồm ba thành phần chính:

- Bộ mã hóa (Encoder): Chuyển đổi đồ thị đầu vào thành một biểu diễn nén trong không gian chiều thấp hơn. Encoder thường là một GNN (như GCN) để tổng hợp thông tin từ nút và cạnh.

- Lớp cổ chai (Bottleneck): Là lớp trung gian giữ biểu diễn nén, kết nối encoder và decoder. Biểu diễn này chứa thông tin cô đọng về cấu trúc đồ thị.

- Bộ giải mã (Decoder): Tái tạo lại đồ thị từ biểu diễn nén, thường bằng cách dự đoán ma trận kề hoặc các cạnh của đồ thị.

Quá trình huấn luyện nhằm tối thiểu hóa sai số giữa đồ thị đầu vào và đồ thị tái tạo, giúp encoder học được biểu diễn chất lượng cao.

Mô hình này thường được sử dụng trong dự đoán liên kết (link prediction), chẳng hạn như dự đoán mối quan hệ tiềm ẩn giữa hai người trong mạng xã hội, hoặc trong các hệ thống gợi ý để tìm kiếm kết nối mới dựa trên cấu trúc hiện có.

*\* Mạng Hồi quy Đồ thị (Recurrent Graph Neural Networks - RGNNs)*

RGNNs học mẫu khuếch tán tối ưu, xử lý đồ thị đa quan hệ với hiệu suất cao và ít tài nguyên tính toán. Ứng dụng trong tạo văn bản, dịch máy, nhận diện giọng nói, và tóm tắt văn bản.

RGNNs sử dụng cơ chế hồi quy để xử lý dữ liệu đồ thị theo cách tuần tự:

- Học mẫu khuếch tán: RGNNs lan truyền thông tin qua các nút theo thời gian, cập nhật trạng thái của nút dựa trên thông tin từ các nút lân cận.

- Sử dụng regularizers: Để tránh hiện tượng quá tham số hóa (over-parameterization) và tăng tính mượt mà (smoothness), RGNNs áp dụng các bộ điều hòa (regularizers) trong quá trình học.

RGNNs có hiệu suất cao với mức tiêu thụ tài nguyên tính toán thấp hơn so với các mô hình GNN khác, nhờ cơ chế hồi quy tối ưu hóa việc lan truyền thông tin. Điều này làm cho RGNNs phù hợp với các bài toán yêu cầu xử lý nhanh trên đồ thị lớn.

**\*** *Gated Graph Neural Networks - GGNNs*

GGNNs là một cải tiến của RGNNs, được thiết kế để xử lý các bài toán có phụ thuộc dài hạn (long-term dependencies) trong đồ thị. GGNNs lấy cảm hứng từ Gated Recurrent Units (GRUs), bổ sung các cổng điều khiển để quản lý luồng thông tin

GGNNs tích hợp ba loại cổng chính:

- Cổng nút (Node Gate): Điều khiển thông tin nào từ nút lân cận được sử dụng để cập nhật trạng thái nút hiện tại.

- Cổng cạnh (Edge Gate): Quản lý mức độ ảnh hưởng của các cạnh trong quá trình tổng hợp thông tin.

- Cổng thời gian (Time Gate): Quyết định thông tin nào được giữ lại hoặc quên đi qua các bước thời gian, tương tự cổng quên (forget gate) trong LSTM.

Quá trình hoạt động của GGNNs dựa trên việc cập nhật trạng thái nút qua nhiều bước thời gian, sử dụng các cổng để kiểm soát luồng thông tin, từ đó cải thiện khả năng nắm bắt phụ thuộc dài hạn.

**d) Các tác vụ của GNNs**

- Phân loại đồ thị: nhằm gán nhãn hoặc phân loại toàn bộ một đồ thị vào một danh mục cụ thể dựa trên đặc trưng tổng thể của nó. Không giống như các phương pháp tập trung vào từng nút hay cạnh, phân loại đồ thị xem xét cấu trúc toàn cục, bao gồm cách các nút được kết nối và mối quan hệ giữa chúng. Quy trình thực hiện thường bắt đầu bằng việc sử dụng GNNs để tổng hợp thông tin từ tất cả các nút và cạnh, tạo ra một biểu diễn vectơ duy nhất cho toàn đồ thị (graph embedding). Vectơ này sau đó được truyền qua một lớp phân loại, chẳng hạn như lớp dày đặc với hàm softmax, để dự đoán nhãn.

- Phân loại nút: tập trung vào việc dự đoán nhãn của một nút trong đồ thị dựa trên thông tin từ chính nút đó và các nút lân cận của nó. Đây thường là bài toán bán giám sát, nơi chỉ một phần nhỏ các nút có nhãn sẵn, và GNNs tận dụng tính chất “đồng nhất” trong đồ thị (các nút lân cận thường có đặc điểm tương tự) để suy ra nhãn cho các nút còn lại. GNNs, đặc biệt là Mạng Tích chập Đồ thị (GCNs), thực hiện tác vụ này bằng cách tổng hợp đặc trưng từ các nút lân cận thông qua ma trận kề, cập nhật vectơ đặc trưng của nút mục tiêu, rồi truyền qua một lớp phân loại để đưa ra dự đoán.

- Dự đoán liên kết: xác định xem có tồn tại một cạnh (liên kết) giữa hai nút trong đồ thị hay không, đặc biệt hữu ích khi ma trận kề của đồ thị không đầy đủ. Mục tiêu là phát hiện các mối quan hệ tiềm ẩn chưa được ghi nhận trong dữ liệu. Để thực hiện, GNNs đầu tiên học biểu diễn vectơ đặc trưng cho từng nút dựa trên cấu trúc lân cận, sau đó sử dụng một hàm đo lường, như tích trong (dot product) hoặc khoảng cách cosine, để tính xác suất tồn tại cạnh giữa hai nút.

- Phát hiện cộng đồng: là quá trình mà GNNs phân chia các nút trong đồ thị thành các cụm (clusters) dựa trên cấu trúc cạnh, sao cho các nút trong cùng cụm có mối liên kết chặt chẽ hơn so với các nút ở cụm khác. Tác vụ này nhằm khám phá tổ chức ẩn trong dữ liệu đồ thị. GNNs thực hiện phát hiện cộng đồng bằng cách học biểu diễn vectơ đặc trưng cho các nút, phản ánh mức độ kết nối với các nút khác, sau đó áp dụng thuật toán phân cụm (như k-means) hoặc tối ưu hóa hàm mục tiêu (như modularity) để nhóm các nút.

- Nhúng đồ thị: là quá trình mà GNNs ánh xạ toàn bộ đồ thị hoặc các thành phần của nó (nút, cạnh) thành các vectơ trong không gian chiều thấp, nhằm bảo toàn thông tin về cấu trúc và mối quan hệ. Đây là bước nền tảng để chuyển dữ liệu phi Euclid của đồ thị thành dạng có thể xử lý bằng các thuật toán học máy truyền thống. GNNs, thường kết hợp với Graph Auto-Encoders, thực hiện nhúng đồ thị bằng cách tổng hợp thông tin từ nút, cạnh và ngữ cảnh toàn cục, sau đó tối ưu hóa vectơ nhúng sao cho khoảng cách giữa các nút trong không gian vectơ phản ánh mức độ gần gũi trong đồ thị.

- Tạo đồ thị: GNNs sinh ra một đồ thị mới dựa trên phân phối của các đồ thị mẫu, sao cho đồ thị mới có cấu trúc tương tự nhưng không hoàn toàn giống đồ thị gốc. Đây là bài toán tạo sinh, thường được thực hiện bằng cách kết hợp GNNs với các mô hình như GANs (Generative Adversarial Networks). GNNs học phân phối của các đặc trưng nút và cạnh từ dữ liệu mẫu, sau đó sử dụng thông tin này để tạo ra các nút và cạnh mới, đảm bảo tính tương đồng về cấu trúc.

### 6.1.6. LLM (Large Language Model)

**a) Tổng quan**

Mô hình ngôn ngữ lớn, còn gọi là LLM, là các mô hình học sâu rất lớn, được đào tạo trước dựa trên một lượng dữ liệu khổng lồ. Bộ chuyển hóa cơ bản là tập hợp các mạng nơ-ron có một bộ mã hóa và một bộ giải mã với khả năng tự tập trung. Bộ mã hóa và bộ giải mã trích xuất ý nghĩa từ một chuỗi văn bản và hiểu mối quan hệ giữa các từ và cụm từ trong đó.

Bộ chuyển hóa LLM có khả năng đào tạo không có giám sát, mặc dù lời giải thích chính xác hơn là bộ chuyển hóa thực hiện việc tự học. Thông qua quá trình này, bộ chuyển hóa học cách hiểu ngữ pháp, ngôn ngữ và kiến thức cơ bản.

Khác với các mạng nơ-ron hồi quy (RNN) trước đó thường xử lý tuần tự dữ liệu đầu vào, bộ chuyển hóa xử lý song song toàn bộ trình tự. Điều này cho phép các nhà khoa học dữ liệu sử dụng GPU để đào tạo các LLM dựa trên bộ chuyển hóa, qua đó giảm đáng kể thời gian đào tạo.

Kiến trúc mạng nơ-ron của bộ chuyển hóa cho phép việc sử dụng các mô hình rất lớn, thường có hàng trăm tỷ tham số. Các mô hình quy mô lớn như vậy có thể thu nạp một lượng dữ liệu khổng lồ, thường là từ Internet, nhưng cũng từ các nguồn, ví dụ như Common Crawl với hơn 50 tỷ trang web, và Wikipedia với khoảng 57 triệu trang.

**b) Cách thức đào tạo**

Một yếu tố quan trọng trong cách thức hoạt động của LLM là cách chúng biểu diễn các từ. Các hình thức [máy học](https://aws.amazon.com/what-is/machine-learning/) trước đây sử dụng một bảng số để biểu diễn từng từ. Tuy nhiên, hình thức biểu diễn này không thể nhận ra mối quan hệ giữa các từ, chẳng hạn như các từ có nghĩa tương tự. Hạn chế này đã được khắc phục bằng cách sử dụng các véc-tơ đa chiều, thường được gọi là nhúng từ, để biểu diễn các từ sao cho các từ có nghĩa theo ngữ cảnh tương tự nhau hoặc các mối quan hệ khác sẽ gần nhau trong không gian véc-tơ.

Bằng cách sử dụng nhúng từ, bộ chuyển hóa có thể xử lý trước văn bản dưới dạng phần biểu diễn bằng số thông qua bộ mã hóa và hiểu ngữ cảnh của các từ lẫn cụm từ có nghĩa tương tự, cũng như các mối quan hệ khác giữa các từ, chẳng hạn như các phần của lời nói. Sau đó, LLM có thể áp dụng kiến thức về ngôn ngữ này thông qua bộ giải mã để tạo ra một kết quả đầu ra độc đáo.

**c) Phương pháp huấn luyện**

Mạng nơ-ron dựa trên bộ chuyển hóa rất lớn. Các mạng này chứa nhiều nút và lớp. Mỗi nút trong một lớp có kết nối với tất cả các nút trong lớp tiếp theo, mỗi nút có trọng số và độ lệch. Trọng số và độ lệch cùng với phần nhúng được gọi là tham số mô hình. Mạng nơ-ron dựa trên bộ chuyển hóa lớn có thể có hàng tỷ tham số. Kích thước của mô hình thường được xác định bởi mối quan hệ thực nghiệm giữa kích thước mô hình, số lượng tham số và kích thước của dữ liệu đào tạo.

Quá trình đào tạo được thực hiện bằng cách sử dụng một tập ngữ liệu lớn gồm dữ liệu chất lượng cao. Trong quá trình đào tạo, mô hình điều chỉnh liên tục các giá trị tham số cho đến khi mô hình dự đoán chính xác mã thông báo tiếp theo từ trình tự mã thông báo đầu vào trước đó. Mô hình thực hiện điều này thông qua các kỹ thuật tự học, giúp mô hình học được cách điều chỉnh các tham số để tối đa hóa khả năng của các mã thông báo tiếp theo trong các mẫu đào tạo.

Sau khi được đào tạo, LLM có thể dễ dàng được điều chỉnh để thực hiện nhiều tác vụ bằng cách sử dụng các tập dữ liệu có giám sát tương đối nhỏ, một quy trình được gọi là tinh chỉnh.

Có ba mô hình học phổ biến:

* Học bằng dữ liệu chưa từng gặp; LLM cơ sở có thể phản hồi một loạt các yêu cầu mà không cần đào tạo rõ ràng, thường thông qua câu lệnh văn bản, mặc dù độ chính xác của câu trả lời có thể khác nhau.
* Học với ít dữ liệu đào tạo: Bằng cách cung cấp một số mẫu đào tạo liên quan, hiệu năng của mô hình cơ sở cải thiện đáng kể trong lĩnh vực cụ thể đó.
* Tinh chỉnh: Đây là một phần mở rộng của mô hình học với ít dữ liệu đào tạo, trong đó các nhà khoa học dữ liệu đào tạo một mô hình cơ sở để điều chỉnh các tham số của nó với dữ liệu bổ sung liên quan đến ứng dụng cụ thể.

### 6.1.7. RAG (Retrieval Augmented Generation)

**a) Tổng quan**

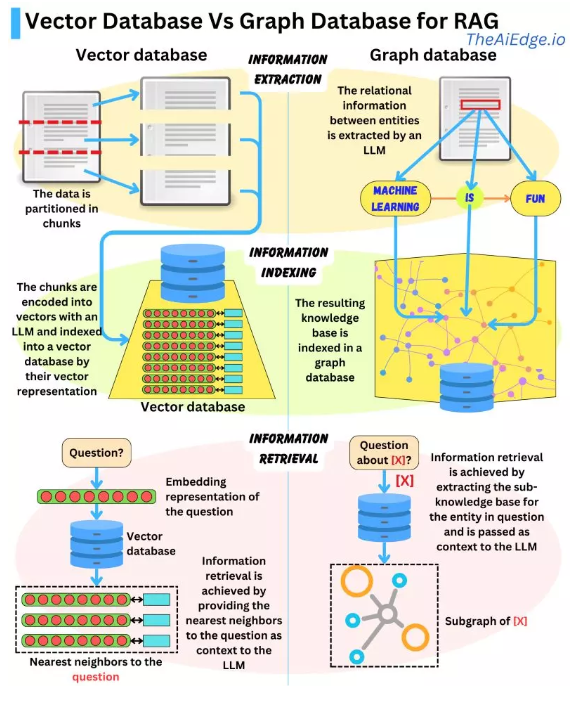
* Retrieval-Augmented Generation (RAG) là một kỹ thuật giúp nâng cao khả năng của mô hình sinh (language model generation) kết hợp với tri thức bên ngoài (external knowledge)
* Phương pháp này thực hiện bằng cách truy xuất thông tin liên quan từ kho tài liệu (tri thức) và sử dụng chúng cho quá trình sinh câu trả lời dựa trên LLMs.

**b) Mô hình Neural Retrieval**

* Mô hình Neural Retrieval là một loại mô hình truy xuất thông tin sử dụng mạng nơ-ron (neural network) để tìm kiếm các đoạn văn (chunks) trong tài liệu (documents) liên quan dựa trên câu truy vấn (query). Chúng mã hóa (encode) câu query và các chunks trong các documents thành các biểu diễn vector, gọi là dense vector và tính toán mức độ tương đồng giữa chúng. Điều này cho phép chúng nắm bắt được sự liên quan về cả ngữ nghĩa thay chỉ vì sự kết hợp từ vựng với nhau.
* Neural Retrieval là sự phát triển đáng kể từ các hệ thống retrieval dựa trên từ khóa (keyword-based) sang hệ thống truy xuất thông tin hiểu được ý nghĩa và mối quan hệ ngữ nghĩa cơ bản trong dữ liệu.

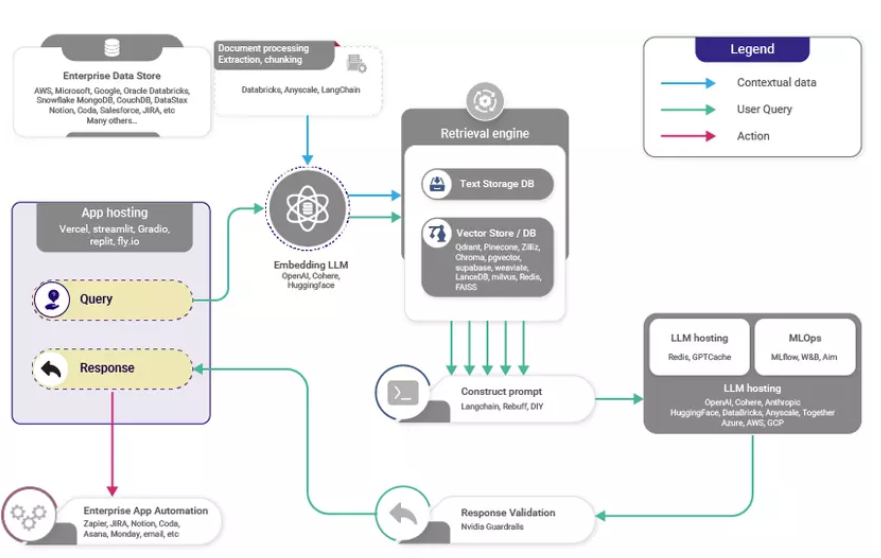
**c) RAG (Retrieval Augmented Generation)**

* Với RAG, LLMs có thể tận dụng tri thức và thông tin mà không nhất thiết phải được lưu trữ trong trọng số mô hình của nó (tức là không chỉ dựa vào những gì nó đã học) bằng cách cung cấp các nguồn tri thức bên ngoài, chẳng hạn như database, sách, báo, website, ...
* Điều này thúc đẩy công cụ truy xuất thông tin (retriever) để tìm kiếm các tri thức liên quan để điều chỉnh LLMs, bằng cách này RAG có thể nâng cao nền tảng tri thức của LLMs với nguồn tri thức bên ngoài.
* Công cụ truy xuất (retriever) ở đây có thể là bất kỳ công cụ nào sau đây tùy thuộc vào nhu cầu về ngữ nghĩa hay không:
  + Vector database: Thông thường, các câu query được embed sử dụng mô hình như BERT (Transformers-based) để tạo ra dense vector embedding. Ngoài ra, các phương pháp truyền thống như TF-IDF có thể được sử dụng như là sparse embeddings. Việc tìm kiếm sau đó được tiến hành dựa trên độ tương đồng về ngữ nghĩa (semantic similarity) hoặc tần suất các thuật ngữ (term frequency).
  + Graph database: Xây dựng cơ sở tri thức (knowledge base) từ mối quan hệ giữa các thực thể (entity) được trích xuất trong văn bản. Cách tiếp cận này đảm bảo tri thức tính xác, nhưng đòi hỏi truy vấn chính xác, điều này cũng sẽ hạn chế trong một số ứng dụng.
  + Regular SQL database: Cung cấp lưu trữ và truy xuất dữ liệu có cấu trúc nhưng có thể thiếu linh hoạt về mặt ngữ nghĩa.
* Hình ảnh dưới nói về sự khác biệt giữa việc sử dụng Graph database và Vector database trong RAG. Graph database được cho là ưu tiên hơn cho RAG khi so với Vector database bởi nó chứa những tri thức chính xác cao. Trong khi Vector database chia và indexing dữ liệu sử dụng vector dựa trên LLM để encode, cho phép có khả năng truy xuất thông tin dựa trên ngữ nghĩa nhưng chúng có thể đi kèm theo những thông tin không liên quan dẫn tới sai lệch có thể sảy ra. Mặc khác, Graph database xây dựng một knowledge base từ các mối quan hệ giữa các thực thể được trích xuất trong văn bản, giúp cho việc truy xuất thông tin trở nên ngắn gọn, tuy nhiên đòi hỏi việc các truy vấn phải chính xác với cách dữ liệu được lữu trữ và liên kết. Tức là nếu chúng ta không biết chính xác cách thông tin được cấu trúc và liên kết sẽ gặp khó khăn trong việc truy xuất thông tin mà chúng ta cần.



*Hình 19: So sánh khác biệt giữa việc sử dụng Graph database và Vector database trong RAG*

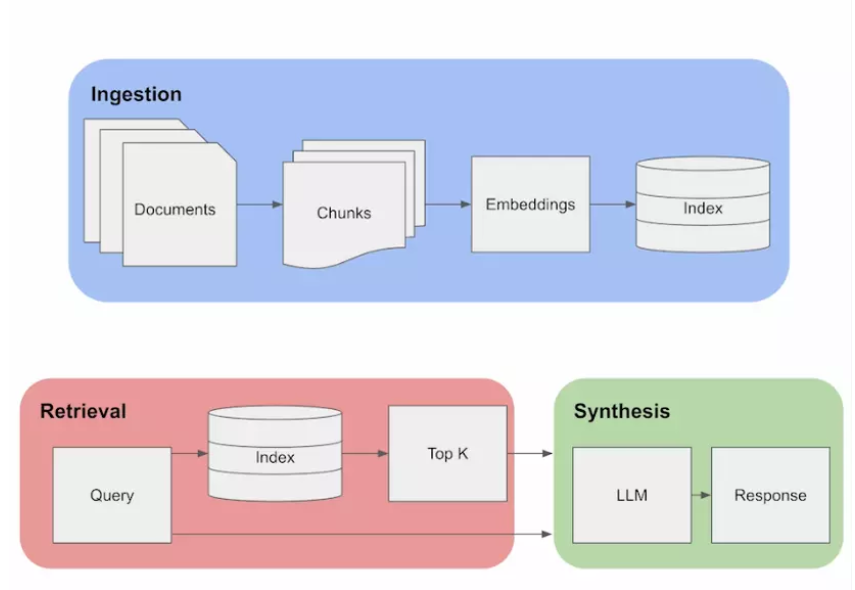
* Sau khi retrieval, chúng ta có thể xem xét lọc tri thức liên quan bằng cách thêm các lớp ranking cho phép lọc ra những tri thức không phù hợp với quy tắc mà chúng ta muốn, hoặc không được cá nhân hóa cho người dùng, bối cảnh hiện tại hoặc giới hạn về độ dài.
* Tóm tắt ngắn gọn quá trình của RAG như sau:
  + Create Vector database: Đầu tiên, convert toàn bộ dữ liệu tri thức thành các vector và lưu trữ chúng vào một vector database.
  + User input: User cung cấp 1 câu truy vấn (query) bằng ngôn ngữ tự nhiên nhằm tìm kiếm câu trả lời hoặc để hoàn thành câu truy vấn đó.
  + Information retrieval: Cơ chế retrieval quét toàn bộ vector trong database để xác định các phân đoạn tri thức (chính là paragraphs) nào có ngữ nghĩa tương đồng với câu truy vấn của người dùng. Các paragraphs này sau đó được vào LLM để làm tăng context cho quá trình sinh ra câu trả lời.
  + Combining data: Các paragraphs được lấy sau quá trình retrieval từ database được kết hợp với câu query ban đầu của user tạo thành 1 câu prompt.
  + Generate text: Câu prompt được bổ sung thêm context sau đó được đưa qua LLM để sinh ra câu phản hồi cuối cùng theo context bổ sung.



*Hình 20:* *Minh họa quá trình hoạt động của RAG*

**d) Xây dựng hệ thống RAG**

* Một góc nhìn tổng quan thì RAG có 3 bước: Ingestion (Biến đổi dữ liệu), Retrieval (Truy xuất thông tin), và Synthesis/ Response Generation (Tổng hợp và sinh câu trả lời)



*Hình 21: Các bước xây dựng RAG*

* **Ingestion**

Ingestion là quá trình có thể hiểu chung là biến đổi dữ liệu, về cơ bản nó gồm các quá trình:

* Thu thập dữ liệu
* Tiền xử lý dữ liệu
* Lập chỉ mục và lưu trữ vào database (Indexing/ Embedding/ Storage DB)
* **Retrieval**
* Trong phần này chúng ta sẽ tìm hiểu 3 kiểu retrieval được phân loại dựa trên cách tiếp cận gồm: Standard, Sentence window và Auto-merging. Mỗi cách tiếp cận này đều có điểm mạnh và điểm yếu cụ thể và mức độ phù hợp của chúng phụ thuộc vào yêu cầu của nhiệm vụ RAG, bao gồm tính chất của tập dữ liệu, độ phức tạp của câu query và sự cân bằng 2 khía cạnh về tính cụ thể (Specificity) và hiểu biết ngữ cảnh (contextual) trong việc xử lý và phản hồi các câu truy vấn đó.
  + Specificity (Độ cụ thể): Đây là khả năng của một hệ thống trong việc cung cấp câu trả lời chính xác và cụ thể cho một truy vấn. Một hệ thống có độ cụ thể cao sẽ cung cấp thông tin rất chính xác, thường là chi tiết và trực tiếp liên quan đến câu hỏi đặt ra. Ví dụ, khi được hỏi về một sự kiện lịch sử cụ thể, hệ thống sẽ cung cấp thông tin chính xác về sự kiện đó mà không lạc đề.
  + Contextual Understanding (Hiểu biết ngữ cảnh): Đây là khả năng của hệ thống trong việc hiểu và tích hợp thông tin ngữ cảnh vào câu trả lời. Một hệ thống có khả năng hiểu biết ngữ cảnh tốt không chỉ trả lời dựa trên thông tin cụ thể được đề cập trong câu hỏi, mà còn xem xét các yếu tố như bối cảnh của câu hỏi, ý định ngầm của người hỏi, và các thông tin liên quan mà không được trực tiếp nêu ra. Điều này giúp cho câu trả lời có độ sâu và liên quan hơn.
* **Response Generation / Synthesis**
* Bước cuối cùng trong hệ thống RAG là quá trình tạo ra phản hồi cho người dùng. Trong bước này, mô hình tổng hợp các thông tin được truy xuất với tri thức có sẵn trong nó (pre-trained knowledge) để tạo ra các phản hồi mạch lạc và phù hợp với ngữ cảnh. Quá trình này bao gồm việc tích hợp những hiểu biết sâu sắc thu thập được từ nhiều nguồn khác nhau, đảm bảo độ chính xác và mức độ liên quan, đồng thời tạo ra phản hồi không chỉ mang tính thông tin mà còn phù hợp với truy vấn ban đầu của người dùng, duy trì giọng điệu tự nhiên và mang tính trò chuyện.
* Trong quá trình tạo prompt (với top-k chunks được retrieval) cho LLM để tạo phản hồi, việc bố trí một cách có chiến lược để những thông tin quan trọng ở đầu hoặc cuối chuỗi đầu vào có thể nâng cao hiệu quả của hệ thống RAG và do đó làm cho hệ thống hoạt động hiệu quả hơn

## 6.2. Kịch bản thử nghiệm

### 6.2.1. Mục tiêu thử nghiệm

* Xác minh quy trình xử lý văn bản từ đầu vào (tải lên) đến đầu ra (truy vấn) hoạt động đúng và hiệu quả.
* Kiểm tra tích hợp giữa các thành phần: trích xuất (DBNet-CRNN-LLM), phân loại (GNN), quản lý tài liệu (MongoDB-Elasticsearch), và chatbot RAG.
* Đánh giá trải nghiệm người dùng (UX) trên giao diện web và chatbot.
* Xác định vấn đề về hiệu suất (thời gian xử lý, tìm kiếm) và độ ổn định trong môi trường thực tế.
* Kiểm tra bảo mật cơ bản và phân quyền người dùng (admin, user, guest).

### 6.2.2. Chuẩn bị thử nghiệm

**a) Dữ liệu thử nghiệm**

* **Tập dữ liệu**: 300 văn bản hành chính đa dạng (180 công văn, 90 quyết định, 30 thông báo) từ nguồn thực tế (trường học, cơ quan nhỏ).
* **Định dạng**:
  + 60% PDF.
  + 30% hình ảnh quét (JPEG/PNG).
  + 10% Word đã scan.
* **Truy vấn mẫu**: 200 truy vấn cho chatbot RAG:
  + 80 truy vấn tìm kiếm.
  + 80 truy vấn tóm tắt.
  + 40 truy vấn phân tích.
* **Nhãn tham chiếu**: Tạo bán tự động (sử dụng regex hoặc công cụ OCR cơ bản) cho trích xuất và phân loại.

**b) Tài khoản thử nghiệm**

* **1 tài khoản quản trị viên (admin)**: Quản lý người dùng, cấu hình hệ thống, truy cập dashboard báo cáo.
* **1 tài khoản người dùng đầy đủ quyền**: Tải lên, chỉnh sửa, truy vấn chatbot, xuất dữ liệu.
* **1 tài khoản khách**: Chỉ xem tài liệu công khai và thực hiện tìm kiếm cơ bản.

**c) Công cụ theo dõi**

* **Prometheus/Grafana**: Giám sát hiệu suất hệ thống (CPU, RAM, thời gian phản hồi).
* **Sentry**: Ghi log lỗi và hành động người dùng.
* **Python scripts**: Đo thời gian xử lý và tính độ chính xác (F1-score cho trích xuất, Accuracy cho phân loại).

### 6.2.3. Kịch bản thử nghiệm End-to-End

**a) Kịch bản 1: Quy trình xử lý văn bản đầy đủ**

* **Mã kịch bản**: E2E-PROCESS-001
* **Mục tiêu**: Kiểm tra toàn bộ quy trình từ tải lên đến truy vấn.
* **Tiền điều kiện**:
  + Hệ thống được cài đặt trên máy chủ (CPU 4-core, RAM 16GB).
  + Cơ sở dữ liệu trống (MongoDB và Elasticsearch).
  + Tài khoản người dùng đầy đủ quyền đã được tạo.
* **Các bước thực hiện**:
  1. **Tải lên văn bản**:
* Đăng nhập bằng tài khoản người dùng đầy đủ quyền.
* Tải lên 5 văn bản (2 công văn, 2 quyết định, 1 thông báo) qua giao diện web.
* Kiểm tra thông báo “Tải lên thành công” trên giao diện.
  1. **Trích xuất thông tin**:
* Quan sát tiến trình trích xuất tự động (sử dụng DBNet-CRNN-LLM/GNN).
* Ghi nhận thời gian từ lúc tải lên đến khi hoàn thành trích xuất.
* Kiểm tra giao diện hiển thị trạng thái xử lý (ví dụ: “Đang trích xuất”, “Hoàn thành”).
  1. **Xem và chỉnh sửa kết quả trích xuất**:
* Xem thông tin trích xuất (số văn bản, ngày ban hành, nội dung chính) của 5 văn bản.
* So sánh với văn bản gốc để đánh giá độ chính xác.
* Chỉnh sửa thủ công 1 văn bản (ví dụ: sửa số văn bản từ “123/2024” thành “124/2024”).
* Nhấn nút “Lưu” và xác nhận chỉnh sửa được lưu vào MongoDB.
  1. **Phân loại tự động**:
* Kiểm tra kết quả phân loại tự động theo mặt công tác (tài chính, nhân sự, giáo dục).
* Thay đổi phân loại thủ công cho 1 văn bản (ví dụ: từ “tài chính” sang “nhân sự”).
* Xác nhận cập nhật lưu vào MongoDB.
  1. **Lưu trữ và đánh chỉ mục**:
* Xác nhận 5 văn bản được lưu vào MongoDB (GridFS cho file gốc, JSON cho metadata).
* Kiểm tra chỉ mục tìm kiếm trong Elasticsearch bằng cách tìm từ khóa mẫu (ví dụ: “quyết định”).
  1. **Tìm kiếm văn bản**:
* Thực hiện 3 truy vấn tìm kiếm:
  + - * 2 truy vấn từ khóa (ví dụ: “công văn tài chính”, “quyết định 2024”).
      * 1 truy vấn kết hợp bộ lọc (ví dụ: công văn trong tháng 3/2024, danh mục tài chính).
* Ghi nhận thời gian phản hồi và kiểm tra độ chính xác của kết quả (so với văn bản gốc).
  1. **Truy vấn thông tin qua chatbot**:
     + Đặt 5 câu hỏi cho chatbot RAG:
       - 2 câu tìm kiếm (ví dụ: “Công văn về thuế tháng 3/2024”).
       - 2 câu tóm tắt (ví dụ: “Tóm tắt quyết định bổ nhiệm 2024”).
       - 1 câu phân tích (ví dụ: “Số công văn tài chính trong quý 1/2024”).
     + Đánh giá độ chính xác của câu trả lời và kiểm tra trích dẫn nguồn (liệu có đúng văn bản gốc).
* **Kết quả mong đợi**:
  + Quy trình hoạt động liền mạch, không có lỗi nghiêm trọng (crash hoặc timeout).
  + Thời gian xử lý mỗi văn bản < 20 giây.
  + Độ chính xác trích xuất ≥ 80% (F1-score).
  + Độ chính xác phân loại ≥ 75% (Accuracy).
  + Chatbot trả lời đúng ≥ 75% câu hỏi (so với nhãn tham chiếu).
  + Thời gian tìm kiếm < 3 giây.

**b) Kịch bản 2: Xử lý đồng thời nhiều văn bản**

* **Mã kịch bản**: E2E-BATCH-002
* **Mục tiêu**: Đánh giá khả năng xử lý đồng thời nhiều văn bản.
* **Tiền điều kiện**:
  + Hệ thống sạch, cơ sở dữ liệu trống.
  + Có sẵn 50 văn bản thử nghiệm (30 PDF, 15 JPEG, 5 Word).
  + Tài khoản người dùng đầy đủ quyền đã được tạo.
* **Các bước thực hiện**:
  1. **Tải lên hàng loạt**:
     + Đăng nhập bằng tài khoản người dùng đầy đủ quyền.
     + Tải đồng thời 50 văn bản qua chức năng “Tải lên hàng loạt” trên giao diện.
     + Kiểm tra tiến trình tải và thông báo lỗi (nếu có) trên giao diện.
  2. **Giám sát xử lý**:
     + Theo dõi hiệu suất hệ thống (CPU, RAM) qua dashboard Prometheus/Grafana.
     + Ghi nhận thời gian xử lý trung bình cho mỗi văn bản (từ tải lên đến lưu trữ).
  3. **Kiểm tra kết quả**:
     + Xác nhận số văn bản xử lý thành công (trích xuất và lưu trữ).
     + Kiểm tra ngẫu nhiên 5 văn bản:
       - So sánh thông tin trích xuất (số văn bản, ngày ban hành) với gốc.
       - Kiểm tra kết quả phân loại (tài chính, nhân sự, giáo dục).
  4. **Tìm kiếm trong dữ liệu**:
     + Thực hiện 3 truy vấn tìm kiếm:
       - 2 truy vấn từ khóa (ví dụ: “thông báo 2024”, “quyết định nhân sự”).
       - 1 truy vấn kết hợp bộ lọc (ví dụ: công văn tài chính quý 1/2024).
     + Ghi nhận thời gian phản hồi và độ chính xác của kết quả.
* **Kết quả mong đợi**:
  + Xử lý thành công ≥ 90% văn bản, không treo hệ thống.
  + Thời gian xử lý trung bình < 25 giây/văn bản.
  + Thời gian tìm kiếm < 3 giây.
  + Độ chính xác trích xuất ≥ 80% (F1-score).
  + Độ chính xác phân loại ≥ 75% (Accuracy).

**c) Kịch bản 3: Truy vấn và phân tích thông tin phức tạp**

* **Mã kịch bản**: E2E-QUERY-003
* **Mục tiêu**: Kiểm tra khả năng truy vấn phức tạp của chatbot RAG.
* **Tiền điều kiện**:
  + Hệ thống chứa 200 văn bản đã xử lý và đánh chỉ mục.
  + Tài khoản người dùng đầy đủ quyền đã được tạo.
* **Các bước thực hiện**:
  1. **Truy vấn phức hợp**:
     + Đăng nhập bằng tài khoản người dùng đầy đủ quyền.
     + Thực hiện 5 truy vấn phức hợp, ví dụ:
       - “Công văn tài chính tháng 3/2024 về thuế”.
       - “Quyết định bổ nhiệm trong năm 2023”.
       - “Thông báo giáo dục quý 1/2024”.
       - “Công văn nhân sự tháng 4/2024”.
       - “Quyết định tài chính trước tháng 2/2024”.
     + Đánh giá độ chính xác và mức độ liên quan của kết quả (so với nhãn tham chiếu).
  2. **Tổng hợp thông tin**:
     + Yêu cầu chatbot tóm tắt: “Tóm tắt tất cả công văn tài chính năm 2024”.
     + Kiểm tra tính logic, tính liên kết và trích dẫn nguồn trong tóm tắt.
  3. **Phân tích xu hướng**:
     + Thực hiện 2 truy vấn phân tích:
       - “Số lượng quyết định bổ nhiệm trong năm 2023”.
       - “Lĩnh vực có nhiều công văn nhất trong quý 1/2024”.
     + Kiểm tra kết quả hiển thị (số liệu hoặc biểu đồ) trên giao diện.
  4. **Truy vấn ngữ nghĩa**:
     + Đặt 2 câu hỏi ngữ nghĩa:
       - “Công văn nào về ngân sách trường học liên quan đến sửa chữa cơ sở vật chất?”.
       - “Công văn tài chính nào nhắc đến thuế trong năm 2024?”.
     + Đánh giá độ chính xác và ngữ cảnh của câu trả lời.
* **Kết quả mong đợi**:
  + Truy vấn phức hợp chính xác ≥ 75%.
  + Tóm tắt nhóm có tính liên kết, trích dẫn đúng nguồn.
  + Phân tích xu hướng trả về số liệu hoặc biểu đồ chính xác ≥ 70%.
  + Truy vấn ngữ nghĩa đúng ≥ 65%.

**d) Kịch bản 4: Thử nghiệm người dùng đa vai trò**

* **Mã kịch bản**: E2E-USER-004
* **Mục tiêu**: Kiểm tra phân quyền và trải nghiệm người dùng với các vai trò khác nhau.
* **Tiền điều kiện**:
  + Hệ thống đã cấu hình 3 loại vai trò: admin, user đầy đủ quyền, guest.
  + Có sẵn 10 văn bản trong hệ thống.
* **Các bước thực hiện**:
  1. **Quản trị viên**:
     + Đăng nhập bằng tài khoản admin.
     + Thêm 1 người dùng mới và xóa 1 người dùng cũ qua giao diện quản lý.
     + Kiểm tra dashboard báo cáo (số văn bản, số truy vấn, số người dùng).
  2. **Người dùng đầy đủ quyền**:
     + Đăng nhập bằng tài khoản người dùng đầy đủ quyền.
     + Tải lên 3 văn bản (1 công văn, 1 quyết định, 1 thông báo).
     + Chỉnh sửa metadata của 1 văn bản (ví dụ: thay đổi ngày ban hành).
     + Thực hiện 1 truy vấn chatbot: “Tóm tắt công văn vừa tải”.
     + Xuất danh sách văn bản dưới dạng CSV.
  3. **Người dùng khách**:
     + Truy cập hệ thống mà không đăng nhập.
     + Thực hiện 1 tìm kiếm cơ bản (ví dụ: “công văn 2024”).
     + Thử truy cập chức năng tải lên hoặc chỉnh sửa, xác minh bị chặn.
* **Kết quả mong đợi**:
  + Phân quyền hoạt động đúng, các hành động trái phép bị từ chối.
  + Giao diện hiển thị phù hợp với từng vai trò (admin thấy dashboard, guest chỉ thấy tìm kiếm cơ bản).
  + Không có lỗi giao diện hoặc quyền truy cập.

**e) Kịch bản 5: Thử nghiệm tính ổn định dài hạn**

* **Mã kịch bản**: E2E-STABILITY-005
* **Mục tiêu**: Đánh giá độ ổn định của hệ thống trong thời gian dài.
* **Tiền điều kiện**:
  + Hệ thống được cài đặt, giám sát bằng Prometheus/Grafana và Sentry.
  + Cơ sở dữ liệu chứa 50 văn bản đã xử lý.
* **Các bước thực hiện**:
  1. **Thiết lập giám sát**:
     + Cấu hình Prometheus/Grafana để theo dõi CPU, RAM, thời gian phản hồi.
     + Bật ghi log lỗi và hành động người dùng qua Sentry.
     + Thiết lập ngưỡng cảnh báo: CPU > 80%, RAM > 90%.
  2. **Tạo tải liên tục**:
     + Mô phỏng 5 người dùng đồng thời trong 12 giờ.
     + Tự động thực hiện:
       - Tải lên 50 văn bản (30 PDF, 15 JPEG, 5 Word).
       - Thực hiện 100 truy vấn (50 tìm kiếm, 30 tóm tắt, 20 phân tích).
  3. **Giám sát tài nguyên**:
     + Theo dõi CPU, RAM, bộ nhớ, thời gian phản hồi qua Prometheus/Grafana.
     + Ghi lại mọi lỗi hoặc sự cố qua Sentry.
  4. **Kiểm tra sau thử nghiệm**:
     + Kiểm tra các chức năng chính (tải lên, tìm kiếm, truy vấn chatbot).
     + Xác minh dữ liệu trong MongoDB (không mất mát hoặc hỏng).
* **Kết quả mong đợi**:
  + Hệ thống hoạt động liên tục, không crash.
  + Không rò rỉ bộ nhớ, CPU và RAM duy trì dưới 85%.
  + Thời gian phản hồi không tăng quá 25% so với ban đầu.
  + Dữ liệu trong MongoDB được bảo toàn 100%.

## 6.3. Kết quả dự kiến

### 6.3.1. Sản phẩm dự kiến

Một ứng dụng được phát triển để hỗ trợ xử lý và truy vấn văn bản hành chính tiếng Việt các chức năng chính: trích xuất thông tin, phân loại thông tin, truy vấn thông tin và quản lý tài liệu.

**a) Trích xuất thông tin**

* **Chức năng**: Tự động trích xuất các trường thông tin (số văn bản, ngày ban hành, đơn vị ban hành, nội dung chính, người ký,...) từ văn bản hành chính.
* **Công nghệ**: Pipeline **DBNet** (phát hiện văn bản), **CRNN** (nhận diện văn bản), và **LLM** / **Graph** để trích xuất ngữ nghĩa.
* **Hiệu suất**:
  + **F1-score ≥ 85%** trên tập dữ liệu 5,000 tài liệu (công văn, quyết định, thông báo, thông tư,...).
  + Cao hơn trên tài liệu chuẩn (**F1-score ≥ 90%**) so với tài liệu phức tạp/chất lượng thấp (**F1-score ≥ 80%**).

**b) Phân loại thông tin**

* **Chức năng**: Phân loại tài liệu theo mặt công tác (quản lý nhân sự, tài chính, hậu cần, giáo dục) và danh mục (công văn, quyết định, thông báo).
* **Công nghệ**: **GNN (RGCN)** để mô hình hóa mối quan hệ giữa các trường thông tin.
* **Hiệu suất**: **Accuracy ≥ 80%**, **F1-score ≥ 0.80** trên tập dữ liệu 5,000 tài liệu.

**c) Truy vấn thông tin**

* **Chức năng**: Chatbot **RAG** trả lời các câu hỏi theo một hoặc nhiều tiêu chí kết hợp, tóm tắt chi tiết, phân tích, thống kê và truy vấn ngữ nghĩa phức
* **Công nghệ**: **Sentence-BERT** (vector hóa tài liệu), **FAISS/Elasticsearch** (tìm kiếm), và **LLM** (sinh câu trả lời).
* **Hiệu suất**: **BLEU/ROUGE ≥ 0.80**, thời gian phản hồi **< 3 giây** trên 500 truy vấn mẫu.

6.3.1.4. Hệ thống quản lý tài liệu

Hệ thống quản lý tài liệu là thành phần cốt lõi, cho phép lưu trữ, tổ chức, và truy xuất tài liệu hành chính một cách hiệu quả, phù hợp với nhu cầu của các cơ quan, doanh nghiệp nhỏ.

Chức năng chính

* **Tải lên tài liệu**:
  + Hỗ trợ định dạng PDF, JPEG, PNG.
  + Cho phép tải lên hàng loạt (batch upload) tối đa 10 tài liệu/lần.
  + Tự động kiểm tra định dạng và chất lượng tài liệu, cảnh báo nếu tài liệu không đọc được.
* **Lưu trữ tài liệu**:
  + Lưu trữ tài liệu gốc (file PDF/hình ảnh) và thông tin trích xuất (số văn bản, ngày ban hành, v.v.) dưới dạng JSON trong **MongoDB**.
  + Lưu trữ metadata (mặt công tác, danh mục, ngày tải lên, người tải) để hỗ trợ tìm kiếm.
  + Dung lượng lưu trữ tối đa 5,000 tài liệu (khoảng 10GB, tùy thuộc vào kích thước file).
* **Tìm kiếm tài liệu**:
  + Tìm kiếm nhanh theo từ khóa (số văn bản, ngày ban hành, đơn vị ban hành) hoặc bộ lọc (mặt công tác, danh mục).
  + Hỗ trợ tìm kiếm mờ (fuzzy search) để xử lý lỗi nhập liệu (ví dụ: “CV123” thay vì “Công văn 123”).
  + Kết quả tìm kiếm hiển thị dưới dạng danh sách với thông tin tóm tắt (số văn bản, ngày ban hành, danh mục).
* **Quản lý danh sách tài liệu**:
  + Hiển thị danh sách tài liệu đã xử lý, cho phép sắp xếp theo ngày ban hành, danh mục, hoặc mặt công tác.
  + Hỗ trợ chỉnh sửa metadata (ví dụ: sửa mặt công tác nếu phân loại sai) và xóa tài liệu.
  + Cho phép xuất danh sách tài liệu (định dạng CSV) để báo cáo hoặc lưu trữ ngoài.

- Công nghệ sử dụng

* **Cơ sở dữ liệu**:
  + **MongoDB**: Lưu trữ tài liệu gốc (dưới dạng GridFS) và thông tin trích xuất/metadata (JSON documents).
* **Backend**:
  + **Node.js/Express**: Xử lý yêu cầu tải lên, lưu trữ, và tìm kiếm tài liệu.
  + API RESTful để kết nối giao diện với cơ sở dữ liệu và pipeline xử lý.
* **Tiền xử lý tài liệu**: Sử dụng **PyPDF2** hoặc **pdf2image** để xử lý PDF, **OpenCV** để xử lý hình ảnh (chuẩn hóa kích thước, giảm nhiễu).
* **Bảo mật cơ bản**:
  + Xác thực người dùng (username/password) để giới hạn quyền truy cập.
  + Mã hóa dữ liệu truyền tải (HTTPS) để bảo vệ thông tin nhạy cảm.

- Hiệu suất

* **Thời gian xử lý**:
  + Tải lên và lưu trữ tài liệu: **< 10 giây/tài liệu** (bao gồm kiểm tra định dạng và lưu vào MongoDB).
  + Tìm kiếm tài liệu: **< 1 giây** cho truy vấn từ khóa đơn giản, **< 2 giây** cho truy vấn phức tạp (kết hợp nhiều bộ lọc).
* **Dung lượng lưu trữ**: Hỗ trợ **≥ 5,000 tài liệu** (10GB), đủ cho nhu cầu của một phòng ban hoặc doanh nghiệp nhỏ.
* **Khả năng mở rộng**:
  + Hệ thống xử lý **≥ 10 tài liệu/phút** trên máy chủ thử nghiệm (CPU 4-core, RAM 16GB).
  + Có thể mở rộng bằng cách tăng dung lượng MongoDB hoặc sử dụng Elasticsearch phân tán (nhưng không triển khai trong đồ án).
* **Tính ổn định**:
  + Tỷ lệ lỗi lưu trữ/tìm kiếm **< 2%** trên 5,000 tài liệu thử nghiệm.
  + Hệ thống hoạt động liên tục trong 8 giờ mà không crash (thử nghiệm trên máy chủ cá nhân).

- Giao diện người dùng

* **Chức năng**:
  + Trang tải lên tài liệu với nút chọn file và hiển thị trạng thái xử lý.
  + Trang danh sách tài liệu với bộ lọc (số văn bản, ngày, danh mục) và nút chỉnh sửa/xóa.
  + Trang chatbot để nhập truy vấn và xem câu trả lời.
* **Công nghệ**: **React.js** với thư viện **Material-UI** để tạo giao diện trực quan, responsive.
* **Hiệu suất**:
  + Tỷ lệ hoàn thành tác vụ (tải lên, tìm kiếm, truy vấn) **≥ 90%** trong 5 phút.

### 6.3.2. Đóng góp khoa học

* **Thử nghiệm tích hợp**:
  + Kết hợp **DBNet-CRNN-LLM/GNN-RAG** để xử lý và quản lý văn bản hành chính tiếng Việt, chứng minh tính khả thi trong môi trường học thuật.
  + Phát triển hệ thống quản lý tài liệu tích hợp với pipeline AI, đóng góp vào nghiên cứu xử lý văn bản không cấu trúc.
* **Xử lý tiếng Việt**:
  + Xử lý các đặc điểm ngôn ngữ (dấu thanh, từ ghép) và bố cục phức tạp của tài liệu hành chính, cung cấp dữ liệu thử nghiệm cho nghiên cứu tiếp theo.
* **Tài liệu tham khảo**:
  + Báo cáo so sánh hiệu suất hệ thống với các công cụ cơ bản (Tesseract, PaddleOCR), làm cơ sở cho các nghiên cứu tương lai.

### 6.3.3. Đóng góp thực tiễn

* **Hỗ trợ quản lý hành chính**:
  + Hệ thống quản lý tài liệu giúp tự động hóa việc lưu trữ, tổ chức, và truy xuất công văn, quyết định, thông báo, giảm thời gian xử lý cho các phòng ban nhỏ (ví dụ: phòng hành chính trường học, doanh nghiệp vừa).
* **Chuyển đổi số**:
  + Cung cấp nguyên mẫu để số hóa tài liệu hành chính, hỗ trợ các đơn vị bắt đầu quá trình chuyển đổi số với chi phí thấp.
* **Ứng dụng cụ thể**:
  + **Giáo dục**: Quản lý công văn, thông báo trong trường học.
  + **Doanh nghiệp**: Lưu trữ và tìm kiếm quyết định nội bộ.
  + **Cơ quan hành chính**: Hỗ trợ tra cứu công văn theo số văn bản hoặc lĩnh vực.