|  |
| --- |
|  |
| **BỘ CÔNG THƯƠNG**  **TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  **---------------------------------------** |
|  |
| **ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC**  **NGÀNH HỆ THỐNG THÔNG TIN** |
| **NGHIÊN CỨU SỬ DỤNG TRÍ TUỆ NHÂN TẠO TỰ ĐỘNG MÔ TẢ HÌNH ẢNH** |
|  |
|  |
|  |
| **CBHD :TS. Phạm Văn Hà** |
| **Sinh viên : Thân Ngọc Thiện** |
| **Mã số sinh viên : 2021602775** |
|  |
| **Hà Nội – 2024** |

# **LỜI MỞ ĐẦU**

Em xin được phép bày tỏ lòng biết ơn đến tất cả các cá nhân và tổ chức đã tạo điều kiện hỗ trợ, giúp đỡ em trong suốt quá trình học tập và nghiên cứu đề tài “*Nghiên cứu sử dụng trí tuệ nhân tạo tự động mô tả hình ảnh*”. Em cũng xin chân thành cảm ơn các thầy cô ở khoa công nghệ thông tin đã truyền đạt vốn kiến thức quý báu cho em trong suốt thời gian học tập tại trường. Nhờ có những lời hướng dẫn, dạy bảo của các thầy, các cô nên đề tài tốt nghiệp: “*Nghiên cứu sử dụng trí tuệ nhân tạo tự động mô tả hình ảnh*” của em mới có thể hoàn thiện. Một lần nữa, em xin chân thành cảm ơn TS. Phạm Văn Hà - người đã trực tiếp giúp đỡ, quan tâm, hướng dẫn em hoàn thành tốt bài báo cáo này trong thời gian qua. Bài báo cáo đồ án của em thực hiện trong khoảng thời gian 9 tuần.

Bài báo cáo của em bao gồm 5 chương:

*Chương 1: Tổng quan về đề tài*

*Chương 2: Cơ sở lý thuyết thực hiện bài toán mô tả ảnh*

*Chương 3: Các hướng tiếp cận của bài toán mô tả ảnh*

*Chương 4: Các mô-đun thực hiện bài toán mô tả ảnh*

*Chương 5: Thực nghiệm*

Bước đầu đi vào thực tế của em còn hạn chế và gặp nhiều bỡ ngỡ nên không khỏi gặp nhiều thiếu sót, em rất mong nhận được những ý kiến đóng góp quý báu của quý thầy cô để kiến thức của em trong lĩnh vực này được hoàn thiện hơn đồng thời có điều kiện bổ sung, nâng cao kỹ năng nghề nghiệp của em.

Em xin chân thành cảm ơn!

Hà Nội, ngày … tháng … năm 2024

Sinh Viên

**Thân Ngọc Thiện**

# **MỤC LỤC**

[LỜI MỞ ĐẦU i](#_Toc168346910)

[MỤC LỤC ii](#_Toc168346911)

[DANH MỤC CÁC CHỮ VIẾT TẮT v](#_Toc168346912)

[DANH MỤC HÌNH ẢNH vi](#_Toc168346913)

[CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI. 1](#_Toc168346914)

[1.1. Giới thiệu đề tài. 1](#_Toc168346915)

[1.2. Lý do chọn đề tài. 1](#_Toc168346916)

[1.3. Đối tượng nghiên cứu của đề tài. 2](#_Toc168346917)

[1.4. Ý nghĩa khoa học và thực tiễn của đề tài. 3](#_Toc168346918)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT THỰC HIỆN BÀI TOÁN TRÍCH XUẤT THÔNG TIN. 4](#_Toc168346919)

[2.1. Định nghĩa. 4](#_Toc168346920)

[2.2. Ứng dụng. 5](#_Toc168346921)

[2.3. Các dạng triển khai chính của bài toán trích xuất thông tin. 7](#_Toc168346922)

[2.4. Các phương pháp tiếp cận. 7](#_Toc168346923)

[CHƯƠNG 3: CÁC HƯỚNG TIẾP CẬN BÀI TOÁN TRÍCH XUẤT THÔNG TIN 12](#_Toc168346924)

[3.1. Kiến trúc mô hình học sâu và các nhiệm vụ xử lý hình ảnh. 12](#_Toc168346925)

[3.1.1. Mạng nơ-ron nhân tạo (Neural Network) 12](#_Toc168346926)

[3.1.2. Mạng tích chập nơ-ron nhân tạo (Convolutional Neural Network) 13](#_Toc168346927)

[3.1.3. Mạng nơ-ron hồi quy (Recurrent Neural Networks RNNs) 14](#_Toc168346928)

[3.1.4. Bộ nhớ dài-ngắn hạn (Long-Short Term Memory - LSTM) 18](#_Toc168346929)

[3.1.5. Mô hình Sequence to Sequence (Seq2seq) 22](#_Toc168346930)

[3.1.5.1. Cấu trúc, cơ chế hoạt động và hạn chế của mô hình seq2seq 22](#_Toc168346931)

[3.1.5.2. Cơ chế Attention trong mô hình seq2seq 25](#_Toc168346932)

[3.1.6. Mô hình Transformer 27](#_Toc168346933)

[3.1.6.1. Cơ chế Attention trong mô hình Transformer. 27](#_Toc168346934)

[3.1.6.2. Cấu trúc của mô hình Transformer. 31](#_Toc168346935)

[3.2. Hàm mất mát (Loss Function). 37](#_Toc168346936)

[3.2.1. Định nghĩa. 37](#_Toc168346937)

[3.2.2. Connectionist Temporal Classification Loss (CTC Loss). 39](#_Toc168346938)

[3.3. Các phương pháp đánh giá. 42](#_Toc168346939)

[3.3.1. Word Error Rate (WER). 43](#_Toc168346940)

[3.3.2. Character Error Rate (CER). 43](#_Toc168346941)

[3.3.3. Beam-Search Output. 44](#_Toc168346942)

[CHƯƠNG 4: CÁC MÔ-ĐUN THỰC HIỆN BÀI TOÁN TRÍCH XUẤT THÔNG TIN. 46](#_Toc168346943)

[4.1. Tiền xử lý dữ liệu hình ảnh (Pre-processing image). 47](#_Toc168346944)

[4.2. Mô-đun phát hiện chữ (Text Detection Module). 49](#_Toc168346945)

[4.2.1. Mô hình Differentiable Binarization 50](#_Toc168346946)

[4.3.2. Mô hình Differentiable Binarization với Adaptive Scale Fusion. 53](#_Toc168346947)

[4.3. Mô-đun nhận diện chữ (Text Recognition Module). 56](#_Toc168346948)

[4.3.1. Convolutional Recurrent Neural Network (CRNN). 56](#_Toc168346949)

[4.3.2. Scene Text Recognition With a Single Visual Model (SVTR). 60](#_Toc168346950)

[4.4. Mô-đun hậu xử lý (Post processing module) 65](#_Toc168346951)

[CHƯƠNG 5: THỰC NGHIỆM 66](#_Toc168346952)

[5.1. Môi trường Anaconda. 66](#_Toc168346953)

[5.2. Ngôn ngữ Python. 67](#_Toc168346954)

[5.3. Các thư viện chính sử dụng trong bài toán. 69](#_Toc168346955)

[5.3.1. Thư viện Pytorch. 69](#_Toc168346956)

[5.3.2. Thư viện Transformer. 71](#_Toc168346957)

[5.3.3. Thư viện PaddlePaddle. 73](#_Toc168346958)

[5.3.4. Thư viện PaddleOCR. 74](#_Toc168346959)

[5.3.5. Thư viện FastAPI. 76](#_Toc168346960)

[5.4. Bộ dữ liệu huấn luyên. 78](#_Toc168346961)

[5.4.1. Bộ dữ liệu ICDAR2015 78](#_Toc168346962)

[5.4.2. Bộ dữ liệu Total-Text Dataset. 80](#_Toc168346963)

[5.4.3. Bộ dữ liệu CTW1500. 82](#_Toc168346964)

[5.5. Xây dựng hệ thống. 84](#_Toc168346965)

[5.5.1. Huấn luyện mô hình. 85](#_Toc168346966)

[5.5.2. Kết quả đạt được của các thuật toán trên bộ dữ liệu công khai. 87](#_Toc168346967)

[5.5.3. Triển khai các module và tạo backend với FastAPI. 88](#_Toc168346968)

[5.5.4. Triển khai hệ thống với giao diện người dùng đơn giản. 92](#_Toc168346969)

[KẾT LUẬN 95](#_Toc168346970)

[1. Kết quả đạt được. 95](#_Toc168346971)

[2. Hướng phát triển. 95](#_Toc168346972)

[TÀI LIỆU THAM KHẢO 97](#_Toc168346973)

# **DANH MỤC CÁC CHỮ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| Từ viết tắt | Tiếng anh | Tiếng Việt |
| API | Application Programming Interface |  |
| AI | Artificial Intelligence |  |
| ML | Machine Learning |  |
| OCR | Optical Character Recignition |  |
| WER | Word Error Rate |  |
| CER | Character Error Rate |  |
| CNN | Convolutional Neural Network |  |
| RNN | Recurrent Neural Network |  |
| LSTM | Long-Short Term Memory |  |
| RMI | Remote Method Invocation |  |
| FFN | Feed Forward Network |  |
| CTC Loss | Connectionist Temporal Classification loss |  |
| FCN | Fully Connected Network |  |
| GRU | Gated Recurrent Unit |  |
| LM | Language Model |  |

# 

# **DANH MỤC HÌNH ẢNH**

[Hình 1.1: Hình ảnh mô tả hệ thống trích xuất thông tin. 2](#_Toc168144018)

[Hình 1.2 : Hình ảnh mô tả ý nghĩa thực tiễn của đề tài 3](#_Toc168144019)

[Hình 2.1: Logo của Google. 5](#_Toc168144020)

[Hình 2.2: Logo của Convertio 6](#_Toc168144021)

[Hình 2.3: Các phương pháp tiếp cận text detection. 8](#_Toc168144022)

[Hình 2.4: Các phương pháp tiếp cận text recognition. 9](#_Toc168144023)

[Hình 3.1: Hình ảnh biểu diễn mạng neural network 12](#_Toc168144024)

[Hình 3.2: Minh họa cấu trúc mạng CNN 14](#_Toc168144025)

[Hình 3.3: Cấu trúc cơ bản của RNN 15](#_Toc168144026)

[Hình 3.4: Biểu diễn các công thức tính toán của một khối trong RNN. 16](#_Toc168144027)

[Hình 3.5: Cấu trúc của RNN với hàm kích hoạt tanh. 19](#_Toc168144028)

[Hình 3.6: Cấu trúc của LSTM với hàm kích hoạt tanh. 19](#_Toc168144029)

[Hình 3.7: Minh họa cấu trúc trong một cell. 19](#_Toc168144030)

[Hình 3.8: Minh họa cấu trúc của GRU. 21](#_Toc168144031)

[Hình 3.9: Hình ảnh cấu trúc của mô hình seq2seq 23](#_Toc168144032)

[Hình 3.10: Cơ chế dựa vào weight global và weight local. 26](#_Toc168144033)

[Hình 3.11: Cấu trúc của mô hình seq2seq có sử dụng attention 26](#_Toc168144034)

[Hình 3.12: Miêu tả hoạt động của self-attention 29](#_Toc168144035)

[Hình 3.13: Mô tả nhiệm vụ và chức năng của các vector query, key, value 30](#_Toc168144036)

[Hình 3.14: Mô tả chức năng của MutiHead Attention. 31](#_Toc168144037)

[Hình 3.15: Hình mô tả cấu trúc của mô hình Transformer. 32](#_Toc168144038)

[Hình 3.16: Hình ảnh mô tả sự liên kết giữa decoder và encoder 33](#_Toc168144039)

[Hình 3.17: Vị trí của positional encoder trong mô hình Transformer 33](#_Toc168144040)

[Hình 3.18: Hình ảnh vị trí của lớp masked multi head attention 36](#_Toc168144041)

[Hình 3.19 Hình ảnh mô tả CTC loss 40](#_Toc168144042)

[Hình 3.20: Hình ảnh minh họa cho CTC loss 40](#_Toc168144043)

[Hình 3.21: Hình ảnh mô tả ma trận xác suất. 41](#_Toc168144044)

[Hình 4.1: Hình ảnh mô tả tiền xử lý dữ liệu. 48](#_Toc168144045)

[Hình 4.2: Kiến trúc của mạng DB. 50](#_Toc168144046)

[Hình 4.3: Bản đồ ngưỡng động. 51](#_Toc168144047)

[Hình 4.4: Kiến trúc tổng quan của mô hình DB++. 54](#_Toc168144048)

[Hình 4.5: Kiến trúc tổng quan của Adaptive Scale Fusion 54](#_Toc168144049)

[Hình 4.6: Mô hình kiến trúc của Spatial Attention 55](#_Toc168144050)

[Hình 4.7: Minh họa cho lớp tích chập CNNs. 57](#_Toc168144051)

[Hình 4.8: Hình mô tả hoạt động của lớp hồi quy RNN. 58](#_Toc168144052)

[Hình 4.9: Hình minh hoạ cho lớp Transcription. 59](#_Toc168144053)

[Hình 4.10: Hình mô tả transcription layers. 59](#_Toc168144054)

[Hình 4.11: Hình mô tả kiến trúc của SVTR NET. 61](#_Toc168144055)

[Hình 4.12: Mô tả kiến trúc của Patch Embedding. (a) là phép chiếu tuyến tính trong mô hình vision transformer. (b) là cấu trúc patch embedding chồng chéo của mô hình SVTR 62](#_Toc168144056)

[Hình 4.13: Thông số khi sử dụng Path Embedding 62](#_Toc168144057)

[Hình 4.14: Mô tả cấu trúc của Mixing Blocks 63](#_Toc168144058)

[Hình 4.15: Hình mô tả Global mixing, Local mixing. 63](#_Toc168144059)

[Hình 5.1: Logo của Anaconda 66](#_Toc168144060)

[Hình 5.2: Logo của Python 67](#_Toc168144061)

[Hình 5.3: Logo của Pytorch 69](#_Toc168144062)

[Hình 5.4: Logo của Transformer 71](#_Toc168144063)

[Hình 5.5: Logo của PaddlePaddle 73](#_Toc168144064)

[Hình 5.6: Logo của PaddleOCR 75](#_Toc168144065)

[Hình 5.7: Logo của FastAPI 77](#_Toc168144066)

[Hình 5.8: Hình ảnh minh họa ICDAR2015 datasets 79](#_Toc168144067)

[Hình 5.9: Hình ảnh minh họa cho Toal Text Datasets 81](#_Toc168144068)

[Hình 5.10: Hình ảnh minh họa cho CTW1500 Datasets 83](#_Toc168144069)

[Hình 5.11: Mô tả kiến trúc và luồng của hệ thống. 84](#_Toc168144070)

[Hình 5.12: Hình ảnh mô tả label cho text detection 86](#_Toc168144071)

[Hình 5.13: Hình ảnh mô tả label cho text recognition. 86](#_Toc168144072)

[Hình 5.14: Hình mô tả độ chính xác của các thuật toán text detection. 87](#_Toc168144073)

[Hình 5.15: Hình mô tả độ chính xác của text recognition. 88](#_Toc168144074)

[Hình 5.16: Hình ảnh mô tả một số API được thiết kế. 88](#_Toc168144075)

[Hình 5.17: Hình mô tả text detection module. 90](#_Toc168144076)

[Hình 5.18: Hình mô tả hàm read images 90](#_Toc168144077)

[Hình 5.19: Hình mô tả text recognition module. 91](#_Toc168144078)

[Hình 5.20: Mô tả cách sử dụng API upload 92](#_Toc168144079)

[Hình 5.21: Mô tả cách sử dụng API general 93](#_Toc168144080)

[Hình 5.22: Mô tả cách sử dụng API parse-with-group-rule 93](#_Toc168144081)

[Hình 5.23: Mô tả giao diện người dùng 94](#_Toc168144082)

# **CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI.**

## **1.1. Giới thiệu đề tài.**

Đề tài trích xuất thông tin từ các dạng ảnh hoặc pdf thành văn bản hay nói cách khác là chuyển đổi số hoặc số hóa tài liệu là một chủ đề rất nổi bật và là xu hướng phát triển của lĩnh vực khoa học máy tính và trí tuệ nhân tạo. Nó liên quan đến việc sử dụng các kỹ năng phân tích và xử lý ảnh phức tạp. Những kết quả của nghiên cứu này hoàn toàn có thể áp dụng trong các quá trình chuyển đổi số của các công ty sử dụng hồ sơ giấy tờ để lưu trữ thông tin.

Việc chuyển đổi và trích xuất thông tin giúp các tổ chức có thể đẩy nhanh quá trình xử lý hồ sơ, chọn lọc. Đề tài này đòi hỏi sự kết hợp giữ việc xử lý ảnh, học máy và trí tuệ nhân tạo. Nó cũng đòi hỏi sự sáng tạo trong việc thiết kế hệ thống và các thuật toán để đạt được kết quả tốt nhất.

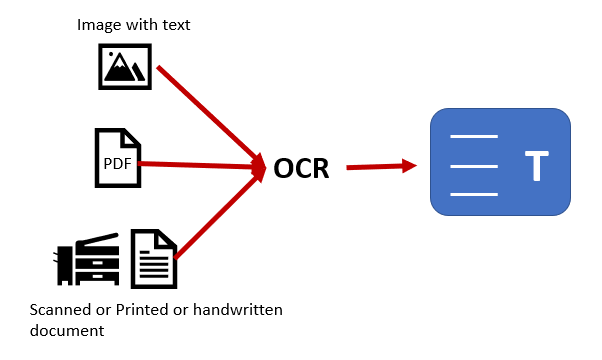
Thông qua đó việc tiếp cận đối với bài toán trên cũng có nhiều hướng khác nhau. Trong bài báo cáo này, chúng ta chỉ tập trung vào việc phân tích hình ảnh, chuyển đổi thành văn bản và trích lọc thông tin từ chứng chỉ tiếng anh quốc tế Ielts hay còn được gọi là bài toán Optical Character Recognition viết tắt là OCR.

## **1.2. Lý do chọn đề tài.**

Mô hình chuyển đổi từ hình ảnh thành sang văn bản với nhiều kỹ thuật khác nhau, tập trung vào các nội dung tìm hiểu chính như sau:

* Machine Learning
* Sequence to Sequence Model
* Attention Mechanism
* Convolutional Recurrent Neural Network
* Deploy a model with FastAPI

Việc ứng dụng chuyển đổi và trích xuất thông tin từ văn bản dạng ảnh hoặc giấy tờ là rất cần thiết, dù không thể hoàn toàn thay thế con người nhưng cũng giúp chúng ta phần nào giảm bớt được việc đọc hiểu, thống kê lại thông tin từ các giấy tờ hồ sơ, văn kiện, … Điều này giúp tăng hiệu quả trong quá trình chuyển đổi số, thống kê hồ sơ, đọc hiểu văn bản, và đặc biệt làm giảm lượng công việc của con người, thay vì phải nhìn, đọc hiểu liên tục các thông tin có tính chất tương đồng, viết lại hoặc soạn thảo lại dạng văn bản từ các hồ sơ giấy tờ thì chỉ cần thống kê và kiểm soát lại để hoàn thiện thông tin trích xuất ra.



Hình .1: Hình ảnh mô tả hệ thống trích xuất thông tin.

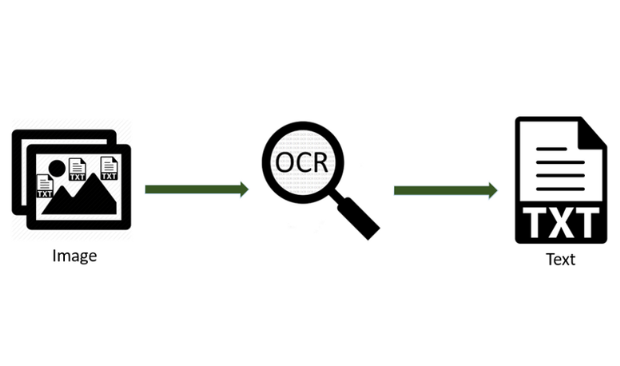
Vì vậy, tính cấp thiết của đề tài “*Nghiên cứu sử dụng trí tuệ nhân tạo trích xuất thông tin tự động từ chứng chỉ IELTs*” là rất cao và đang thu hút nhiều sự quan tâm của nhiều nhà nghiên cứu và doanh nghiệp.

## **1.3. Đối tượng nghiên cứu của đề tài.**

Trong phạm vi đề tài và bài báo cáo, em tiến hành các nội dung sau:

* Định nghĩa và các đặc trưng của bài toán trích xuất thông tin.
* Các hướng tiếp cận chính cho bài toán trích xuất thông tin.
* Kết quả thực nghiệm.

## **1.4. Ý nghĩa khoa học và thực tiễn của đề tài.**

Về mặt thực tiễn, bài toán nhận diện và trích xuất thông tin (OCR) có ý nghĩa rất lớn trong việc chuyển đổi các văn bản dạng ảnh sang các văn bản được lưu dưới dạng file word hoặc file txt, làm giảm thiểu dung lượng lưu trữ, có thể thao tác tìm kiếm, chỉnh sửa hoặc thống kê một cách dễ dàng. Thông qua việc đó làm giảm thiểu lượng công việc của con người, hỗ trợ trực tiếp trong công cuộc cách mạng chuyển đổi số, số hóa tài liệu của đất nước.

Hình 1.2 : Hình ảnh mô tả ý nghĩa thực tiễn của đề tài

Về mặt khoa học, đề tìa đã tổng kết các thuật toán tiêu biểu trong việc phát hiện và nhận diện văn bản, cả truyền thống lẫn hiện đại, thực hiện việc phân tích thuật toán và đánh giá khả năng của từng thuật toán. Bên cạnh đó, đề tài đã thử nghiệm và áp dụng mô hình seq2seq với cơ chế attention vào bài toán cho kết quả khá tốt, thể hiện ra được tính ứng dụng lớn của bài toán và có thể phát triển xa hơn nữa.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT THỰC HIỆN BÀI TOÁN MÔ TẢ HÌNH ẢNH.**

## **2.1. Định nghĩa.**

Bài toán trích xuất thông tin bao gồm 2 bài toán nhỏ là phát hiện chữ và nhận diện chữ, hay còn gọi là text detection và text recognition, là công nghệ cho phép chuyển các hình ảnh sang văn bản viết. Nó liên quan đến việc xử lý đầu vào là hình ảnh từ camera hoặc các nguồn khác, nhận dạng các từ có trong ảnh và chuyển đổi chúng thành văn bản. Công nghệ này có nhiều ứng dụng, đặc biệt là trong chuyển đổi số, thường các cơ quan, công ty lưu trữ dữ liệu ở các dạng giấy tờ cần chuyển sang lưu trữ ở sang máy tính,... Công nghệ chuyển đổi từ ảnh sang văn bản đã được cải thiện đáng kể trong những năm gần đây và tiếp tục phát triển, mang lại kết quả càng ngày càng chính xác và đáng tin cậy.

Theo Xiao-Feng Wang có viết trong một viết “*A survey of text detection and recognition algorithms based on deep learning technology*” : Nhận diện chữ (text recognition) đề cập đến nhiệm vụ nhận dạng chữ viết trong các môi trường khác nhau và chuyển nó thành văn bản. Lĩnh vực nghiên cứu này đã đạt được rất nhiều sự tập trung trong những thập kỷ qua. Đây là một lĩnh vực nghiên cứu quan trọng về chuyển đổi các dạng lưu trữ thông tin từ giấy tờ đến dữ liệu lưu trữ trên máy tính, đám mây, … Các phương pháp ban đầu tập trung vào trích xuất tín năng thủ công và các kỹ thuật xử lý ảnh thông thường. Tuy nhiên gần đây, các mạng thần kinh như mạng thần kinh tái phát (RNN), mạng thần kinh tích chập (CNN) và trong những năm nay Sequence to Sequence model hay gần đây là Transformer, đã được áp dụng trên Text Recognition tạo ra rất nhiều mô hình khác nhau và đạt được hiệu suất tuyệt vời.

## **2.2. Ứng dụng.**

Hiện nay trên thế giới đã phát triển các hệ thống chuyển đổi hình ảnh thành văn bản. Tiêu buổi là một số các phần mềm dưới đây:

* **Camera Translator** : là một sản phẩm được phát hành bởi google, cho phép bạn dịch văn bản trực tiếp từ camera của điện thoại. Tính năng này vô cùng tiện lợi cho những ai thường xuyên du lịch hoặc cần giao tiếp với người nước ngoài. Một số tính năng và tiện ích của Camera Translator là :
  + Dịch văn bản trực tiếp từ camera : chỉ cần hướng camera vào văn bản muốn dịch, Google Dịch sẽ tự động nhận dạng và dịch sang ngôn ngữ bạn chọn
  + Hỗ trợ nhiều ngôn ngữ: Camera Translator hỗ trợ hơn 100 ngôn ngữ, giúp bạn giao tiếp dễ dàng với mọi người trên khắp thế giới.
  + Dịch văn bản trong hình ảnh : Bạn có thể dịch văn bản trong ảnh được lưu trữ trên điện thoại của mình
  + Dịch văn bản trên các đối tượng : Camera Translator có thể dịch văn bản trên các đối tượng thực tế như biển báo, bảng hiệu, thực đơn …
  + Sao chép và chia sẻ bản dịch: Bạn có thể dễ dàng sao chép và chia sẻ bản dịch sang các ứng dụng khác.

Hình 2.1: Logo của Google.

* **Convertio.co** : Đây là một nền tảng chuyển đổi tệp trực tuyến và miễn phía và trả phí, cung cấp cho người dùng nhiều tính năng đa dạng để chuyển đổi hơn 300 định dạng tệp khác nhau một cách nhanh chóng, dễ dàng và bảo mật. Các tính năng và lợi ích chính của Convert io
  + Hỗ trợ đa dạng các định dạng tệp : hơn 300 định dạng tệp phổ biến, tài liệu gồm có word, excel, powerpoint, PDF, txt,... Hình ảnh gồm có JPG, PNG, SVG, WebP, … ngoài ra còn có video, âm thanh, ebook, …
  + Chuyển đổi nhanh chóng và hiệu quả: Convert.io sử dụng công nghệ tiên tiến để tối ưu hóa tốc độ chuyển đổi, đảm bảo người dùng có thể xử lý tệp tin của họ một cách nhanh chóng, hiệu quả. Nhờ đó, bạn có thể tiết kiệm thời gian chờ đợi và tập trung vào công việc quan trọng hơn.
  + Giao diện đơn giản, dễ sử dụng: Convertio được thiết kế với giao diện trực quan và dễ sử dụng, giúp người dùng ở mọi trình độ đểu có thể thao tác dễ dàng.
  + Miễn phí và bảo mật: Convert.io cung cấp gói dịch vụ miễn phí cho phép người dùng ở mọi trình độ đều có thể thao tác dễ dàng. 

Hình 2.2: Logo của Convertio

Những sản phẩm và giải pháp này đang được phát triển và cải tiến liên tục để đáp ứng như cầu của thị trường bán lẻ ngày càng cạnh tranh và phức tạp hơn.

## **2.3. Các dạng triển khai chính của bài toán trích xuất thông tin.**

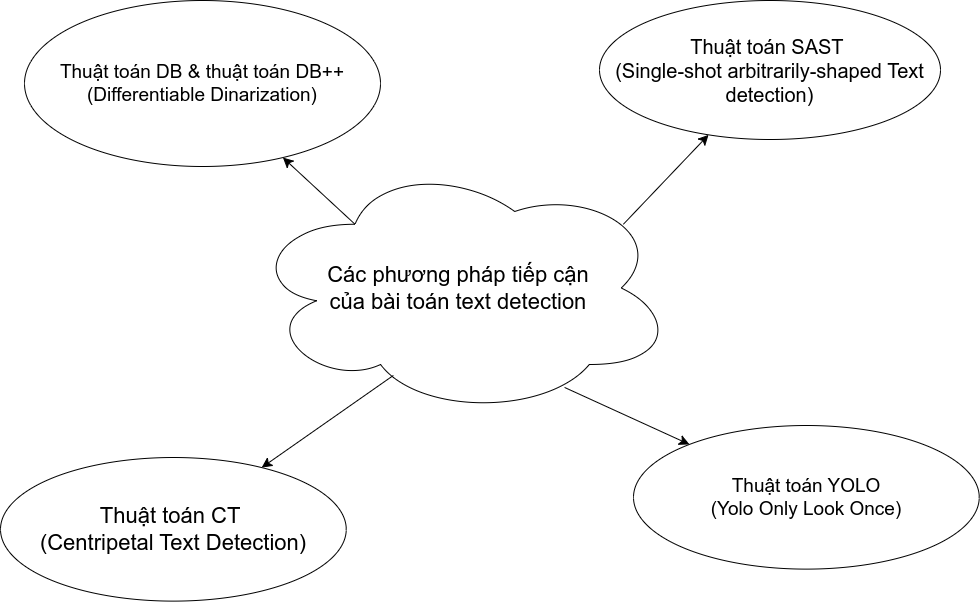
Các mô hình và hướng tiếp cận của bài toán trích xuất thông tin có thể được chia ra nhiều dạng khác nhau:

* Real-time : đây là dạng mà hệ thống phải xử lý hình ảnh và chuyển sang văn bản dạng text khi nó nhận được hình ảnh. Ví dụ như trong google dịch: hình ảnh được chuyển ngay sang văn bản sau khi chụp ảnh
* Batch Inference: đây là dạng mà dữ liệu hình ảnh được chụp và lưu trữ sẵn và sau đó được xử lý bằng các mô hình trích xuất thông tin. Ví dụ khi xử lý số hóa hồ sơ, có một lượng lớn data có sẵn, cần phải tối ưu tốc độ lẫn mức chính xác và không yêu cầu phải xử lý xong luôn thì có thể để batch Inference xử lý, nó sẽ tối ưu hơn Real-time.

Cả hai dạng triển khai đều có những ứng dụng và thách thức riêng. Với real-time thì hệ thống phải xử lý hình ảnh ngay lập tức và đưa ra kết quả trong thời gian ngắn, trong khi với batch inference, thì hệ thống có nhiều thời gian để xử lý dữ liệu âm thanh và đưa ra kết quả chính xác hơn. Tuy nhiên, Batch Inference cũng đòi hỏi phải xử lý một lượng dữ liệu lớn hơn và có thể mất nhiều thời gian để hoàn thành quá trình xử lý.

## **2.4. Các phương pháp tiếp cận.**

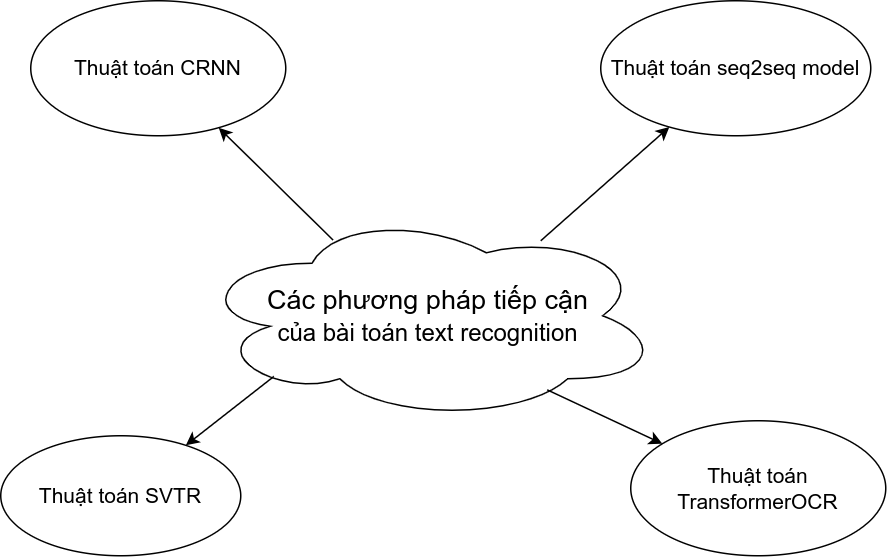
Có nhiều phương pháp tiếp cận khác nhau để giải quyết bài toán trích xuất thông tin, tùy thuộc vào các yêu cầu và điều kiện cụ thể của ứng dụng. Về bài toán phát hiện chữ (text detection):

 Thuật toán DB & DB++ : cả hai thuật toán đều sử dụng một mạng CNN để phát hiện ra các vùng chứa văn bản trong hình ảnh. Thuật toán này tập trung vào việc xác định các hộp chứa văn bản trong hình ảnh, DB có khả năng xử lý văn bản trong điều kiện môi trường đa dạng, yếu sáng, nhiễu, và chữ nghiêng, quay, …

Hình 2.3: Các phương pháp tiếp cận text detection.

* Thuật toán SAST: sử dụng phương pháp single shot, nghĩa là chỉ cần một lần chạy mạng neural để dự đoán các vùng chứa văn bản trực tiếp trên hình ảnh. Điều này giúp SAST có tốc độ xử lý nhanh phù hợp với các ứng dụng phản hồi thời gian thực.
* Thuật toán CT : sử dụng cơ chế của lực ly tâm (centripetal force) để xác định các vùng chứa văn bản trong hình ảnh, cơ chế này giả định rằng các điểm trên biên của vùng chứa văn bản sẽ bị ‘hút’ về một điểm tâm của vùng chứa. CT hoạt động tốt trong các tình huống mà văn bản có thể không có đặc điểm rõ ràng hoặc không tuân theo cấu trúc cố định.
* Thuật toán YOLO: sử dụng phương pháp end-to-end để dự báo các hộp chứa đối tượng trực tiếp trên hình ảnh, trong khi

Về bài toán nhận diện chữ (text recognition):



Hình 2.4: Các phương pháp tiếp cận text recognition.

* Thuật toán CRNN (Convolutional Recurrent Neural Network) là một phương pháp tiên tiến trong việc nhân dạng văn bản trên hình ảnh. Điều đặc biệt về CRNN là nó kết hợp cả kiến trúc mạng neural convolutional (CNN) và mạng neural dựa trên chuỗi (recurrent neural network) để xử lý văn bản trên hình ảnh. Mạng convolutional được sử dụng để trích xuất đặc trưng từ hình ảnh, giúp phát hiện các đặc điểm của văn bản như cạnh và hình dạng của các ký tự. Mạng recurrent sử dụng RNN (Recurrent Neural Network), GRU (Gated Recurrent Unit) hoặc LSTM (Long Sort Term Memory) để xử lý tuần tự thông tin từ các đặc trưng được trích xuất bởi CNN, điều này giúp mô hình có khả năng hiểu và dự đoán các chuỗi ký tự trong văn bản. Ưu điểm là hiệu suất cao trong việc nhận dạng văn bản trên hình ảnh, học dựa trên ngữ cảnh, mạng CRNN có khả năng học dựa trên ngữ cảnh từ các ký tự trước đó trong chuỗi, giúp cải thiện độ chính xác của việc dự đoán ký tự tiếp theo, … Tuy nhiên cũng có những nhược điểm của phương pháp này là : đòi hỏi nhiều tài nguyên tính toán trong quá trình huấn luyện và thử nghiệm, yêu cầu một lượng dữ liệu lớn nếu muốn kết quả tốt, đồng thời mô hình dễ bị overfitting nếu dữ liệu huấn luyện không đủ lớn hoặc không đa dạng.
* Thuật toán seq2seq model với attention mechanism là một kiến trúc mạng neural network nổi tiếng và đạt hiệu quả cao trong việc nhận diện văn bản trên hình ảnh. Mô hình là sự kết hợp của hai thành phần chính: Encoder có nhiệm vụ mã hóa chuỗi đầu vào thành một biểu diễn vector. Biểu diễn này được tạo bằng cách sử dụng một mô hình RNN (Recurrent Neural Network) như LSTM (Long Short Term Memory) hoặc GRU (Gated Recurrent Unit), Decoder có nhiệm vụ giải mã biểu diễn vector của Encoder thành chuỗi đầu ra. Decoder sử dụng một mô hình RNN (hoặc LSTM, …) và kết hợp với cơ chế Attention Mechanism để tập trung vào những phần quan trọng nhất của chuỗi đầu vào khi tạo ra chuỗi đầu ra
* Thuật toán TransformerOCR là thuật toán sử dụng kiến trúc transformer để xử lý nhiệm vụ nhận diện chữ trong ảnh, thay vì sử dụng các model RNN, LSTM hay seq2seq để xử lý thông tin sau khi trích xuất đặc trưng từ hình ảnh thì mô hình sử dụng transformer. Điều này giúp mô hình có khả năng xử lý chuỗi có độ dài biến đổi không gặp vấn đề độ dài cố định; Sự tập trung đồng thời vào toàn bộ ngữ cảnh: các cơ chế attention trong Transformer cho phép mô hình tập trung vào toàn bộ ngữ cảnh của dữ liệu đầu vào, giúp cải thiện hiệu suất nhận dạng, … Tuy nhiên cũng xuất hiện một vài nhược điểm : Khả năng tính toán cao làm cho mô hình yêu cầu nhiều tài nguyên, và lượng lớn dữ liệu huấn luyện; khả năng hiểu ngữ cảnh có hạn, mặc dù mô hình có khả năng tập trung vào ngữ cảnh rộng lớn, nhưng nó vẫn có thể gặp khó khăn khi xử lý các trường hợp phức tạp và ngữ cảnh không rõ ràng.
* Thuật toán SVTR (Scene Text Recognition With a Single Visual Model) là một thuật toán thừa hưởng các ưu điểm của CRNN và Transformer (hoặc seq2seq model). Kiến trúc SVTR nhận đầu vào là ảnh sau đó cho qua một lớp Patch Embedding chia nhỏ ảnh đầu vào thành dạng chuỗi, việc chia nhỏ ảnh thành các phần tử nhỏ hơn giúp dễ dàng xử lý ở dạng chuỗi như các thuật toán trong bài toán thông thường. Mỗi tầng gồm hai module chính là Mixing Block và Merering. Ưu điểm nổi bật của SVTR là sự hiệu quả, SVTR loại bỏ giai đoạn giải mã tuần tự, giúp đơn giản hóa mô hình và cải thiện hiệu quả tính toán; tính linh hoạt, SVTR có thể áp dụng cho nhiều loại văn bản, cả tiếng anh và tiếng trung …

Kết hợp các phương pháp: Nhiều phương pháp tiếp cận có thể được kết hợp để cải thiện độ chính xác và hiệu suất của hệ thống trích xuất thông tin.

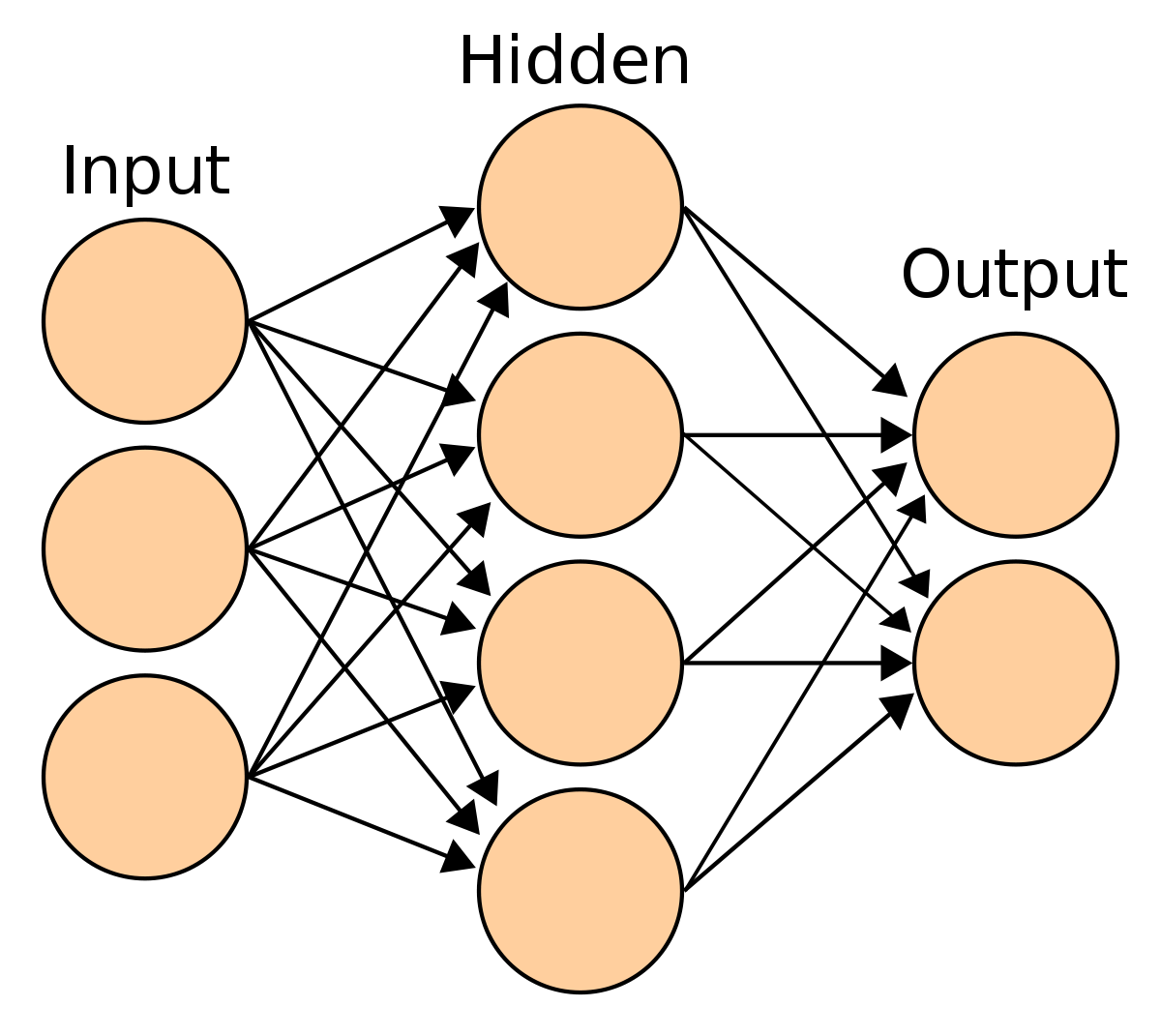
Các phương pháp này có ưu điểm và hạn chế riêng và việc lựa chọn phương pháp phù hợp sẽ phụ thuộc vào nhiều yếu tố, bao gồm độ chính xác, yêu cầu tốc độ, khả năng tính toán của phần cứng và độ phức tạp của ứng dụng.

# **CHƯƠNG 3: CÁC HƯỚNG TIẾP CẬN BÀI TOÁN MÔ TẢ HÌNH ẢNH**

Học sâu (deep learning) đã trở thành một công cụ mạnh mẽ trong việc xử lý hình ảnh vì nó có thể học được cái biểu diễn phức tạp của dữ liệu hình ảnh có được từ hình ảnh thô (raw image). Do đó, những tiến bộ đáng kể đã được thực hiện trong các nhiệm vụ xử lý hình ảnh khác nhau, đặc biệt với hình ảnh bao gồm, nhận diện hình ảnh, phát hiện vật thể trong hình ảnh, phân đoạn hình ảnh (image segmentation), …

## **3.1. Kiến trúc mô hình học sâu và các nhiệm vụ xử lý hình ảnh.**

### **3.1.1. Mạng nơ-ron nhân tạo (Neural Network)**

 Mạng nơ-ron nhân tạo (Neural Network) là một mô hình tính toán được lấy cảm hứng từ cách hoạt động của hệ thần kinh sinh học. Nó là một loại mô hình học máy có khả năng tự động học và áp dụng kiến thức từ dữ liệu.

Hình 3.1: Hình ảnh biểu diễn mạng neural network.

Mạng Neural Network gồm một tập hợp các đơn vị tính toán gọi là nơ-ron được tổ chức thành các lớp. Mỗi nơ-ron trong mạng nhận đầu vào từ các nơ-ron trong lớp trước đó, thực hiện một phép tính và truyền đầu ra đến các nơ-ron trong lớp tiếp theo. Quá trình này được thực hiện thông qua các liên kết trọng số, cho phép mạng học cách điều chỉnh trọng số để tạo ra các đầu ra phù hợp với đầu vào.

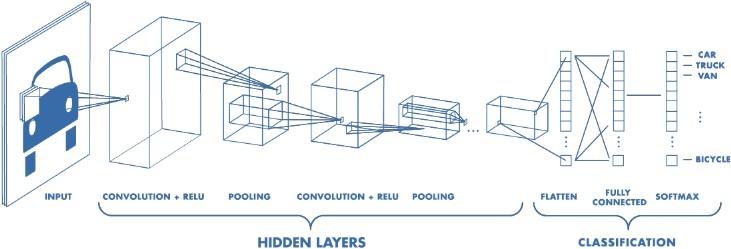
Mạng Neural network có thể có nhiều lớp, gọi là lớp ẩn, giữa lớp đầu vào và lớp đầu ra. Những lớp ẩn này giúp mạng học các mức độ biểu diễn trừu tượng và phức tạp hơn từ dữ liệu đầu vào. Các lớp này cùng với các hàm kích hoạt (activation functions) giúp mạng học các mô hình phi tuyến.

Quá trình huấn luyện mạng Neural network thường bao gồm việc cung cấp dữ liệu huấn luyện đã được gán nhãn và điều chỉnh các trọng số của mạng để giảm thiểu sai số giữa đầu ra dự đoán và đầu ra thực tế. Quá trình này được thực hiện thông qua thuật toán backpropagation, trong đó đạo hàm của hàm mất mát được tính toán và sử dụng để cập nhật trọng số.

Mạng Neural network đã có những ứng dụng rất rộng trong lĩnh vực nhận dạng hình ảnh, xử lý ngôn ngữ tự nhiên, dự báo, điều khiển tự động, và nhiều lĩnh vực khác.

### **3.1.2. Mạng tích chập nơ-ron nhân tạo (Convolutional Neural Network)**

Mạng CNN (Convolutional Neural Network) là một loại mạng nơ-ron nhân tạo được sử dụng rộng rãi trong lĩnh vực xử lý ảnh và thị giác máy tính. Nó được thiết kế để tự động học và phân tích các đặc trưng từ dữ liệu hình ảnh một cách hiệu quả. Cấu trúc của mạng CNN được lấy cảm hứng từ cách mà thị giác của con người hoạt động. Mạng này bao gồm các lớp tích chập (convolutional layers) và các lớp tổng hợp (pooling layers), kết hợp với các lớp kết nối đầy đủ (fully connected layers) ở cuối để thực hiện phân loại hoặc dự đoán.



Hình 3.2: Minh họa cấu trúc mạng CNN.

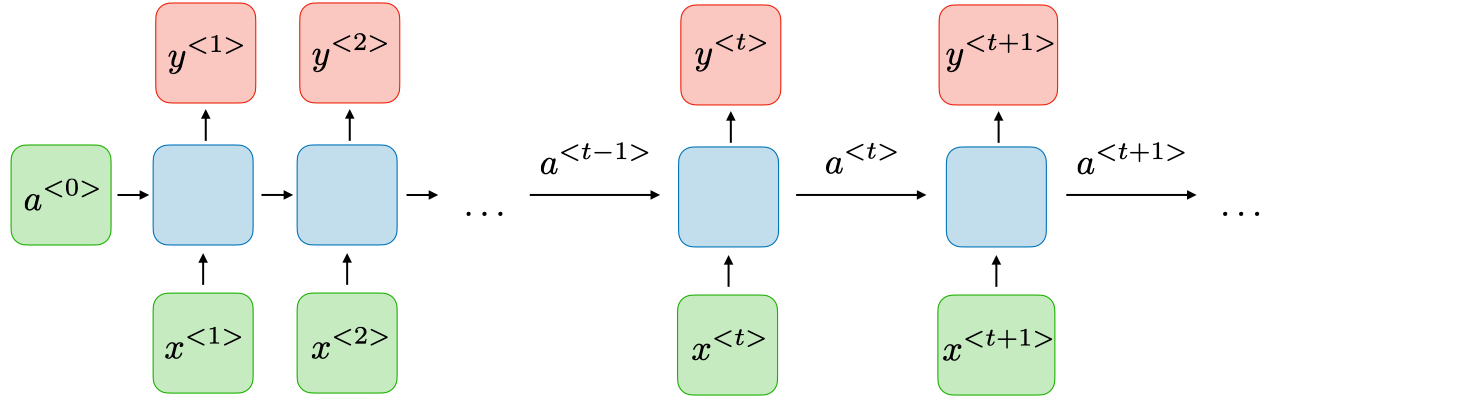
Các lớp tích chập trong mạng CNN thực hiện các phép tích chập giữa các bộ lọc (filters) và các vùng của ảnh đầu vào. Bằng cách áp dụng các bộ lọc này, mạng CNN có khả năng tự động trích xuất các đặc trưng như cạnh, góc, texture, hoặc các đặc trưng phức tạp hơn từ ảnh. Các lớp tổng hợp được sử dụng để giảm kích thước không gian của đặc trưng, đồng thời giữ lại thông tin quan trọng. Điều này giúp giảm số lượng tham số và tính toán của mạng, đồng thời giúp ngăn chặn overfitting.

Sau khi qua các lớp tích chập và lớp tổng hợp, các đặc trưng được đưa vào các lớp kết nối đầy đủ để thực hiện phân loại hoặc dự đoán. Cuối cùng, mạng CNN tạo ra đầu ra là một phân phối xác suất trên các lớp đối tượng mà nó được huấn luyện để nhận dạng. Mạng CNN đã đạt được những thành tựu đáng kể trong các ứng dụng như nhận dạng đối tượng, phân loại ảnh, nhận dạng khuôn mặt, xe tự lái và nhiều lĩnh vực khác.

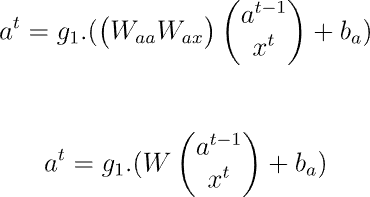
### **3.1.3. Mạng nơ-ron hồi quy (Recurrent Neural Networks RNNs)**

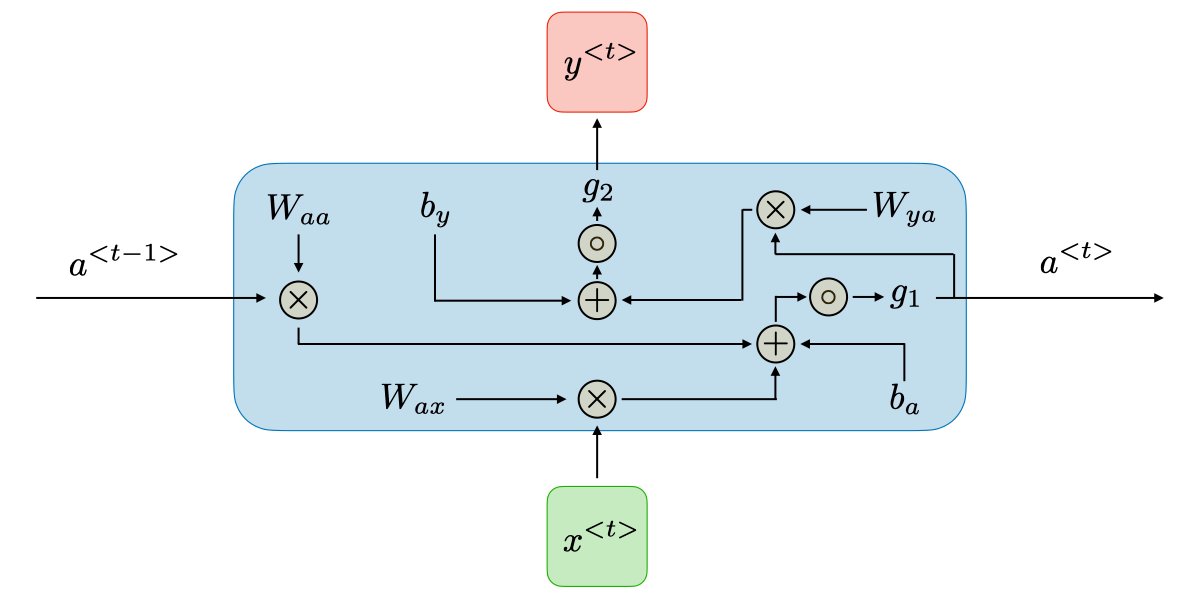
Ý tưởng cốt lõi của RNN: Con người chúng ta không thể bắt đầu suy nghĩ của mình tại tất cả các thời điểm, cũng giống như việc bạn đang đọc bài viết này, bạn hiểu mỗi chữ ở đây dựa vào các chữ mà bạn đã đọc và hiểu trước đó, chứ không phải đọc xong cái là vứt chữ đó đi rồi đến lúc gặp thì lại phải đọc và tiếp thu lại. Nói nôm na ra là kiểu học xong quên luôn xong đến lúc gặp lại là lại học lại. Các mô hình mạng neural truyền thống lại không thể làm được việc trên. vì vậy mạng nơ-ron hồi quy (RNN) được sinh ra để giải quyết việc đó. Mạng này chứa các vòng lặp bên trong cho phép nó được lưu lại các thông tin đã nhận được. RNN là một thuật toán quan trọng trong xử lý thông tin dạng chuỗi hay nói cách khách là dạng xử lý tuần tự. Xử lý tuần tự là mỗi block sẽ lấy thông tin của block trước đó và input hiện tại làm đầu vào.

Hình 3.3: Cấu trúc cơ bản của RNN.



Tại mỗi bước t, giá trị kích hoạt 𝑎𝑡 và đầu ra 𝑦𝑡 được biểu diễn như sau:





Hình3.4: Biểu diễn các công thức tính toán của một khối trong RNN.

Cùng với đó ta có công thức tính đầu ra tương ứng 𝑡:

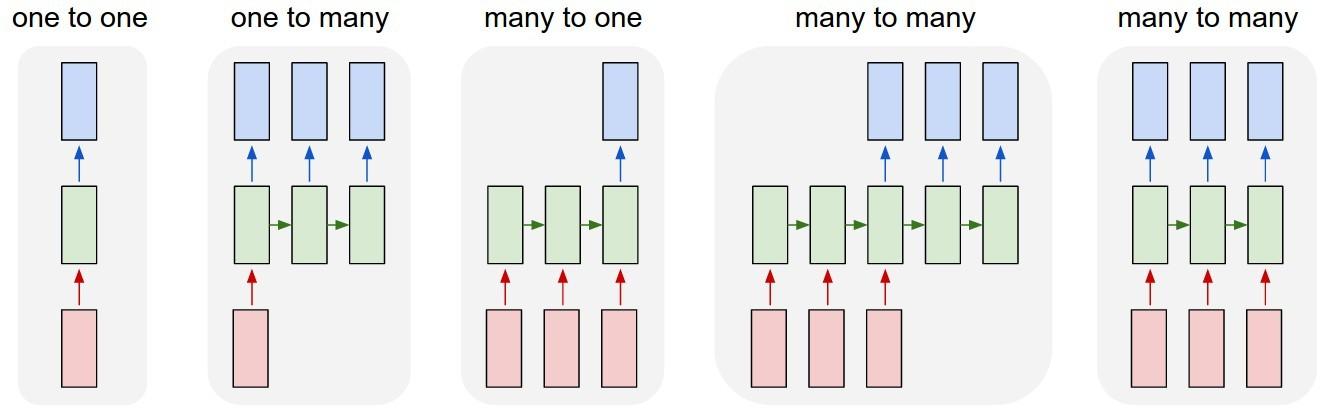
𝑦



Trong đó 𝑊 , 𝑊 , 𝑊 , 𝑏 , 𝑏 là các trọng số chia sẻ tạm thời trong

𝑎𝑎 𝑎𝑥 𝑦𝑎 𝑎 𝑦

từng lớp ẩn (hidden state) tương ứng. Mạng RNN có nhiều cấu trúc khác nhau:

* One to one: một đầu vào, một đầu ra.
* One to many: một đầu vào, nhiều đầu ra.
* Many to one: nhiều đầu vào, một đầu ra.
* Many to many: nhiều đầu vào, nhiều đầu ra.

Ưu điểm và nhược điểm của mạng nơ-ron hồi quy:

|  |  |
| --- | --- |
| Ưu điểm | Nhược điểm |
| Khả năng xử lý các chuỗi đầu vào có độ dài khác nhau | Tính toán khá chậm |
| Kích cỡ mô hình không bị tăng lên theo kích thước đầu vào | Khó truy cập lại thông tin đã đi qua ở một khoảng thời gian dài trước đó - hay còn gọi là bị quên thông tin khi gặp nhiều thông tin mới |
| Quá trình tính toán có sử dụng thông tin trước đó | Phải thực hiện tuần tự nên không tận dụng triệt để được khả năng tính toán song song của GPU |
| Trọng số được chia sẻ trong suốt quá trình học | Vanishing gradient |

Vấn đề của RNN:

Một điểm nổi bật của RNN là nó có thể lấy các thông tin trước đó để dự đoán cho hiện tại. Ví dụ như “Trời hôm nay có mưa nặng” thì RNN có thể dự đoán được chữ tiếp theo sau là “hạt” để hoàn thiện câu hoàn chỉnh “Trời hôm nay có mưa nặng hạt”. Nhưng RNN chỉ hoạt động tốt với chuỗi ngắn, còn đối với chuỗi dài thì khả năng dự đoán của nó bị kém đi. Nguyên nhân là khi chuỗi quá dài thì càng về sau nó sẽ quên các thông tin của các từ trước đó, thậm chí khi càng về cuối thì nó có thể quên luôn thông tin của các từ ở đầu câu.

### **3.1.4. Bộ nhớ dài-ngắn hạn (Long-Short Term Memory - LSTM)**

LSTM là một dạng đặc biệt của RNN, ó có khả năng học các thông tin ở xa. Về cơ bản thì LSTM và RNN không khác nhau là mấy nhưng LSTM có cải tiến một số phép tính trong 1 hidden state và nó đã hiểu quả. Cấu trúc của LSTM không khác gì RNN, những sự cải tiến ở đây nằm ở phần tính toán trong từng hidden state như sau: Thay vì chỉ có một tầng mạng nơ-ron, LSTM thiết kế với 4 tầng mạng nơ-ron tương tác với nhau một các rất đặc biệt.

Với kiến trúc đặc biệt, LSTM có khả năng giữ và quản lý thông tin trong một khoảng thời gian dài, cho phép nó xử lý các chuỗi dữ liệu dài và xử lý các phụ thuộc xa. Điều này khác biệt so với mạng RNN truyền thống, có khả năng mất mát thông tin trong quá trình lan truyền ngược.

Mạng LSTM sử dụng các đơn vị LSTM, trong đó mỗi đơn vị có các thành phần chính sau đây:

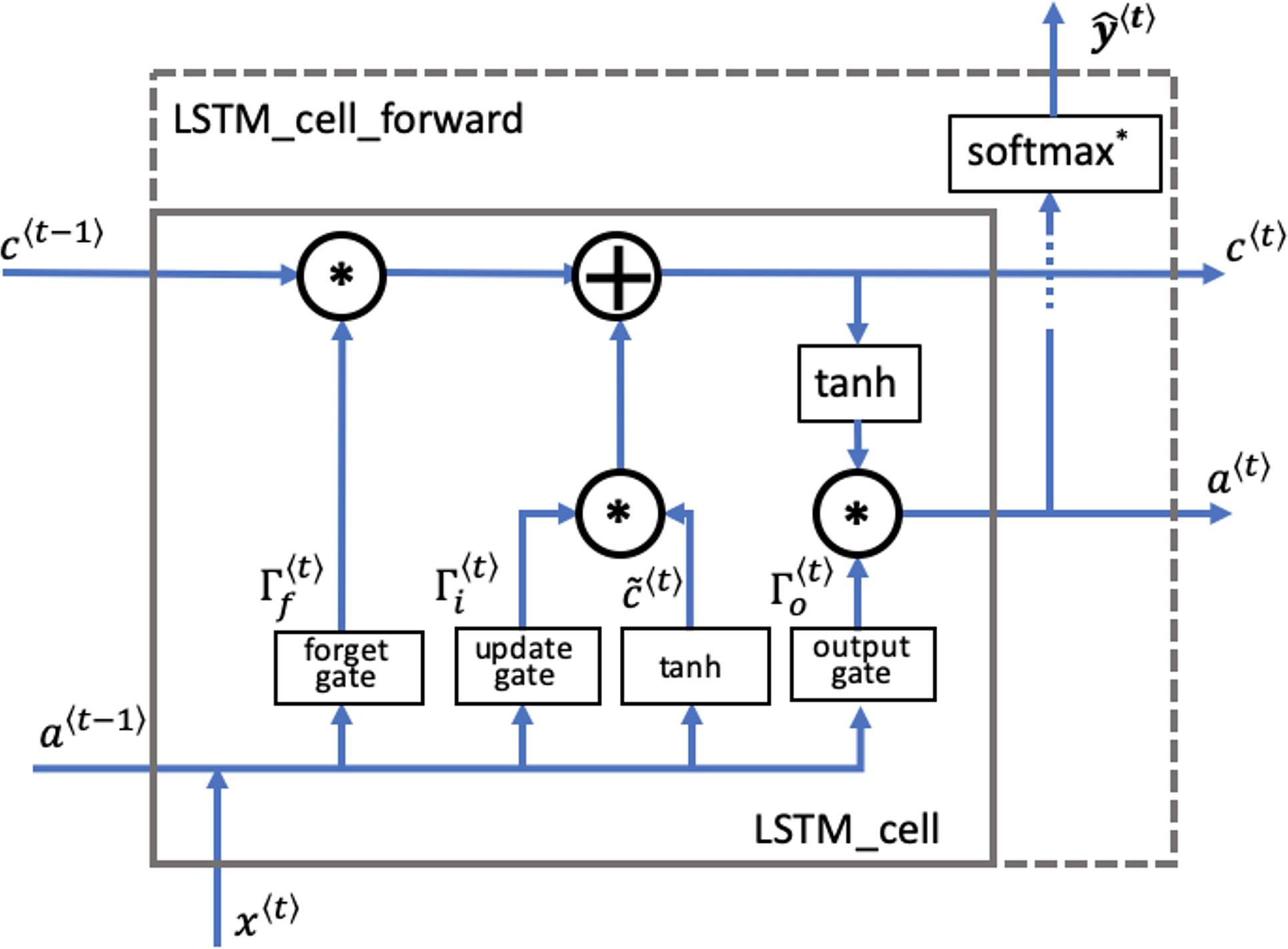
* Cổng quên (Forget gate): Xác định xem thông tin nào nên được giữ lại và thông tin nào nên được bỏ qua từ quá khứ.
* Cổng đầu vào (Input gate): Quyết định thông tin nào nên được cập nhật từ đầu vào mới.
* Cổng ra (Output gate): Quyết định thông tin nào nên được truyền tiếp làm đầu ra của đơn vị LSTM.
* Ô tình trạng (Cell state): Lưu trữ và truyền thông tin dọc theo chuỗi thời gian, giúp duy trì thông tin quan trọng trong quá trình xử lý chuỗi.

Các cổng trong LSTM được điều chỉnh bởi hàm kích hoạt (ví dụ: hàm sigmoid) để quyết định thông tin nào nên được đi qua và thông tin nào nên bị lọc hoặc bỏ qua. Mạng LSTM đã chứng minh hiệu quả của nó trong nhiều ứng dụng như xử lý ngôn ngữ tự nhiên, dịch máy, nhận dạng giọng nói, dự báo chuỗi thời gian và nhiều lĩnh vực khác đòi hỏi xử lý dữ liệu có tính thời gian hoặc phụ thuộc dài hạn. Dưới đây là 2 hình ảnh biểu diễn sự khác nhau giữa RNN và LSTM.

*Hình 3.5: Cấu trúc của RNN với hàm kích hoạt tanh.*

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Hình 3.6: Cấu trúc của LSTM với hàm kích hoạt tanh.

Hình 3.7: Minh họa cấu trúc trong một cell.

Chìa khóa để giúp LSTM có thể truyền tải thông tin giữa các hidden state một cách xuyên suốt chính là cell state.

Trong đó:

* Forget gate **:** Γ𝑓

+ Đầu ra là hàm *sigmoid* chứa các giá trị từ 0 đến 1.

+ Nếu forget gate có giá trị bằng 0, LSTM sẽ "quên" trạng thái được lưu trữ trong đơn vị tương ứng của trạng thái cell trước đó.

+ Nếu cổng quên có giá trị bằng 1, LSTM sẽ chủ yếu ghi nhớ giá trị tương ứng ở trạng thái được lưu trữ.



* Candidate value  : Chứa thông tin có thể lưu trữ từ step hiện tại
* Update gate Γ : quyết định xem phần thông tin nào của và được thêm vào 



* Cell state : Là bộ nhớ trong của LSTM. Cell state như 1 băng tải truyền các thông tin cần thiết xuyết suất cả quá trình, qua các nút mạng và chỉ tương tác tuyển tính 1 chút. Vì vậy thông tin có thể truyền đi thông suốt mà không bị thay đổi.



* Output gate Γ𝑜**:** Cổng điều chỉnh lượng thông tin đầu ra của cell hiện tại và lượng thông tin truyền tới trạng thái tiếp theo.



* Hidden state  : Được sử dụng để xác định ba cổng  của time step tiếp theo.



* Prediction  : Dự đoán trong trường hợp sử dụng này là phân loại, vì vậy bạn sẽ sử dụng softmax.



Nhìn chung LSTM khá giống với RNN hay nói cách khác thì RNN là một dạng đặc biệt của LSTM. LSTM giải quyết phần nào của Vanishing gradient so với RNN, nhưng chỉ một phần.

Cuối cùng, một biến thể khá nổi tiếng của LSTM, đó chính là Gated Recurrent Units hay là GRU. Kiến trúc của GRU kết hợp với cổng quên, cổng đầu vào một cổng cập nhật duy nhất. Ngoài ra, GRU cũng kết hợp các cell state và hidden state để đưa ra các thay đổi khác.



Hình 3.8: Minh họa cấu trúc của GRU.

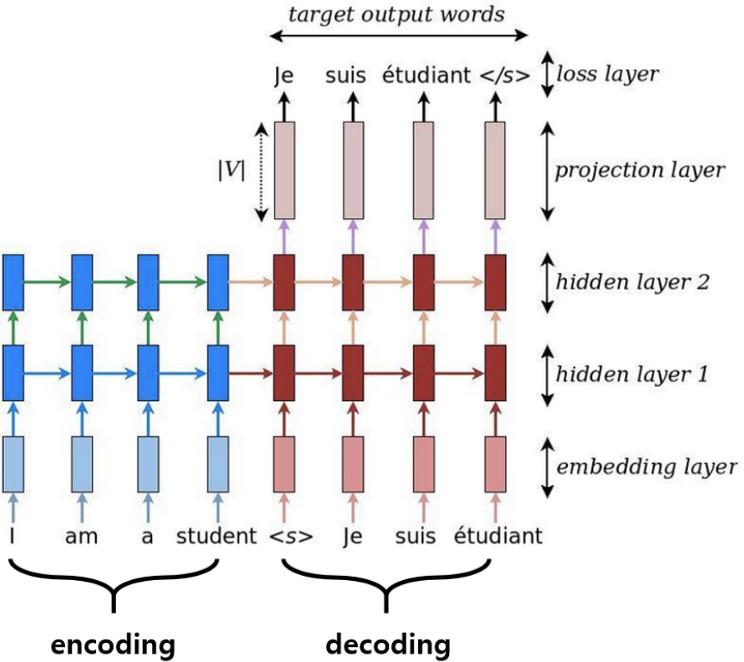
Ngược lại, với lượng tính toán như trên thì RNN đã chậm rồi, LSTM còn chậm hơn nữa:

* RNN và LSTM vẫn bị sẩy ra trường hợp vanishing gradient và thậm chí còn mất khá nhiều thời gian. Bên cạnh đó việc nó không tương thích với dữ liệu có cấu trúc khiến cho việc mô hình bị mất đi cơ chế học được sự liên quan giữa các từ trong câu với nhau.
* Vì vậy, attention đã ra đời, để tăng cường thông tin giữa các hidden state, thể hiện sự tương quan giữa các hidden state với nhau, với việc loại bỏ hoàn toàn tính tuần tự, cải thiện việc song song hóa, tăng tốc độ xử lý. Mục đích cuối cùng là để tính toán mối tương quan giữa input và output.

### **3.1.5. Mô hình Sequence to Sequence (Seq2seq)**

#### **3.1.5.1. Cấu trúc, cơ chế hoạt động và hạn chế của mô hình seq2seq**

Mô hình Seq2Seq (sequence to sequence) là một kiến trúc mạng nơ-ron đặc biệt được sử dụng để xử lý các tác vụ dự đoán, chuyển đổi hoặc tạo ra các chuỗi dữ liệu. Nó được sử dụng rộng rãi trong các ứng dụng như dịch máy, tổng hợp văn bản, trả lời tự động, thậm chí là xử lý ảnh và còn nhiều các tác vụ khác. Giờ chúng ta sẽ phân tích theo bài toán dịch máy để hiểu rõ cấu trúc và cơ chế hoạt động và hạn chế của mô hình Seq2Seq.

Mô hình Seq2Seq bao gồm hai thành phần chính : encoder (bộ mã hóa) và decoder (bộ giải mã). Cả hai thành phần này thường được thể hiện bằng các mạng nơ-ron LSTM hoặc GRU (Gated Recurrent Units).

*Hình 3.9: Hình ảnh cấu trúc của mô hình seq2seq*

* Encoder: Bộ mã hóa nhận đầu vào là một chuỗi dữ liệu và trích xuất các đặc trưng quan trọng của chuỗi đó. Đầu vào có thể là văn bản, âm thanh hoặc bất kỳ dạng chuỗi nào khác.
* Mỗi phần tử trong chuỗi được truyền qua một mạng LSTM/GRU trong encoder để tạo ra một vector đại diện cho toàn bộ chuỗi đầu vào. Vector này thường được gọi là vector ngữ cảnh (context vector) hoặc vector biểu diễn (representation vector).
* Decoder: Bộ giải mã nhận vector ngữ cảnh từ encoder và tạo ra chuỗi đầu ra. Nó cũng sử dụng một mạng LSTM/GRU để dự đoán các phần tử trong chuỗi đầu ra. Trong quá trình huấn luyện, đầu ra thực tế được sử dụng để so sánh với đầu ra dự đoán và điều chỉnh các trọng số của mạng để giảm thiểu sai số.

Ví dụ về mô hình Seq2Seq: dịch máy

* Bước 1: Đầu vào (Input): "Hello, how are you?"
* Bước 2: Bộ mã hóa (Encoder): Mạng LSTM nhận chuỗi đầu vào và tạo ra một vector ngữ cảnh.
* Bước 3: Vector ngữ cảnh được truyền vào bộ giải mã (Decoder).
* Bước 4: Bộ giải mã (Decoder): Mạng LSTM nhận vector ngữ cảnh và bắt đầu tạo ra chuỗi đầu ra.
* Bước 5: Chuỗi đầu ra được dự đoán từ bộ giải mã: "Xin chào, bạn khỏe không?"

Quá trình được lặp lại cho từng phần tử trong chuỗi đầu vào và dự đoán các phần tử tương ứng trong chuỗi đầu ra. Mô hình seq2seq cho phép mô hình học cách ánh xạ từ input đầu vào và output đầu ra.

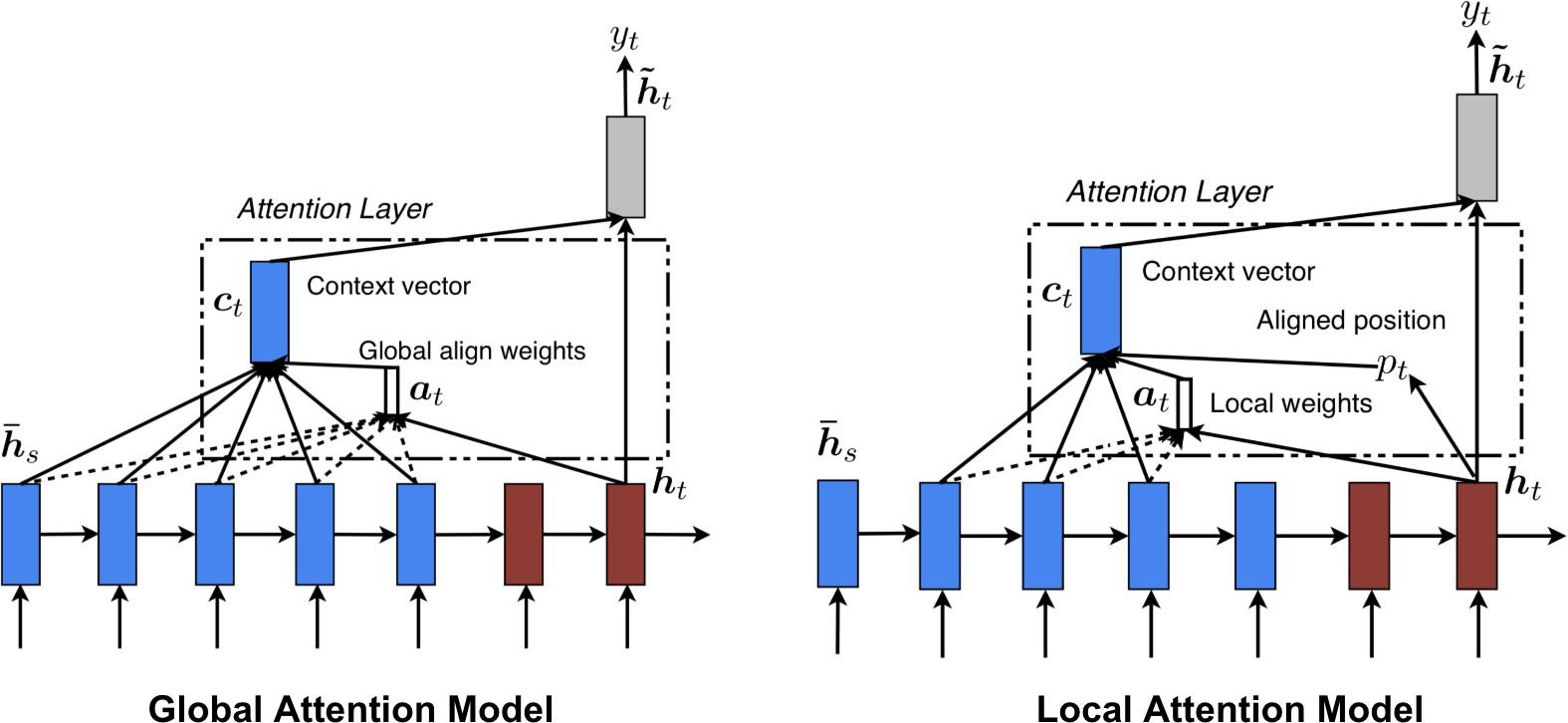
Mục tiêu của chúng ta trong bài toán này là xây dựng một trình dịch văn bản, với đầu vào là một chuỗi dài các từ và đầu ra một chuỗi dài các từ khác ( ở một ngôn ngữ khác). Vì vậy chúng ta có thể thực hiện bài toán này bằng mô hình many-to-many Seq2Seq được thể hiện ở phần RNN.

Vấn đề của mô hình Seq2Seq:

* Vấn đề mất mát thông tin ngữ cảnh dài hạn : mô hình Seq2Seq sử dụng mạng nơ-ron RNN để mã hóa và giải mã chuỗi. RNN có khả năng ghi nhớ thông tin trước đó trong trạng thái ẩn của nó, nhưng nó dễ bị mất thông tin trong quá trình lan truyền ngược. Điều này có nghĩa là các phụ thuộc dài hạn trong quá trình lan truyền ngược có thể không được xử lý tốt.
* Vấn đề độ dài chuỗi biến đổi : Mô hình Seq2Seq không xử lý tốt các chuỗi đầu vào và đầu ra có độ dài khác nhau. Nếu các chuỗi đầu vào và đầu ra có độ dài khác nhau, việc đồng bộ giữa encoder và decoder sẽ trở nên khó khăn và có thể dẫn đến kết quả không chính xác.
* Khả năng mô hình hóa thông tin phi tuyến : Mạng LSTM/ GRU trong mô hình Seq2Seq có khả năng mô hình hóa thông tin phi tuyến, nhưng nó cũng có giới hạn. Đối với các tác vụ phức tạp yêu cầu khả năng mô hình hóa phi tuyến cao, mô hình Seq2Seq có thể gặp khó khăn.
* Độ phức tạp tính toán: Mô hình Seq2Seq có độ phức tạp tính toán cao, đặc biệt là khi xử lý các chuỗi dữ liệu dài. Điều này có thể gây khó khăn trong việc huấn luyện và triển khai mô hình trên các tài nguyên tính toán hạn chế.
* Thiếu khả năng xử lý từ vựng lớn: mô hình seq2seq có thể gặp khó khăn trong việc xử lý các từ vựng lớn. Việc mã hóa và giải mã các từ trong các vector số có thể trở nên tốn kém về mặt tính toán và không hiệu quả đối với từ vựng lớn.
* Thiếu khả năng xử lý thông tin ngoại lệ : Mô hình seq2seq có xu hướng không xử lý tốt các thông tin ngoại lệ hoặc không thường xuyên trong dữ liệu đầu vào, vì nó dựa vào các phép tính trung bình của chuỗi đầu vào.

#### **3.1.5.2. Cơ chế Attention trong mô hình seq2seq**

Trong phần 3.1.5 có nêu lên rõ ràng hạn chế của mô hình Seq2seq, cơ chế Attention sinh ra để khắc phục vấn đề này. Nó nhằm mục đích dự đoán một từ bằng cách chỉ xem xét một vài phần cụ thể của chuỗi, thay vì toàn bộ chuỗi. Xét ví dụ với chuỗi đầu vào là “Which sport do you like most ?” và output là “I love football”. Từ “I” trong câu output có mối liên hệ với từ “you” trong input và tương tự từ “love” trong câu output lại có liên kết với từ “like” trong câu input. Vậy, thay vì nhìn vào tất cả những từ trong đầu vào thì chúng ta có thể tăng tầm quan trọng của một vài từ cụ thể trong đầu vào có ý nghĩa với đầu ra. Đó là ý tưởng cơ bản của cơ chế Attention.

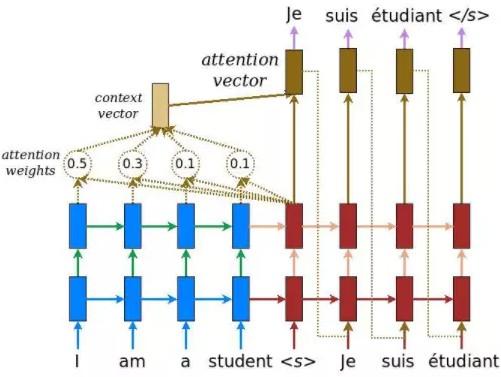
Có hai cơ chế Attention khác nhau phụ thuộc vào vector trung gian: 

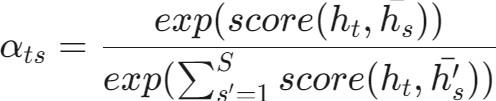
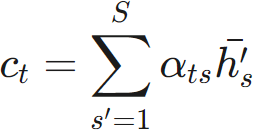
Hình 3.10: Cơ chế dựa vào weight global và weight local.

* Global Attention: tất cả các trạng thái ẩn của bộ mã hóa được xem xét để tạo vector trung gian.
* Local Attention: Chỉ một vài trạng thái ẩn của bộ mã hóa được xem xét và lấy ra để tạo vector trung gian.

Mô hình seq2seq khi áp dụng cơ chế attention vào sẽ có cấu trúc như sau (Các khối màu xanh dương là encoder, màu đỏ là decoder):

Hình 3.11: Cấu trúc của mô hình seq2seq có sử dụng attention

Chi tiết các bước, tại mỗi time step ở phía decoder:

* Bước 1: Nhận vector trạng thái ẩn của decoder và tất cả các vector trạng thái ẩn của encoder .
* Bước 2: Tính trọng số attention. Với mỗi vector trạng thái ẩn (hidden state) của encoder thì ta cần tính toán sự liên quan với vector trạng thái ẩn của decoder. Cụ thể, ta sẽ áp dụng một phương trình tính score của “attention” với đầu vào là vector trạng thái ẩn decoder và một vector trạng thái ẩn của decoder trả về một giá trị vô hướng được gọi là *score(*, *).*
* Bước 3: Tính trọng số Attention. Áp dụng hàm softmax với đầu vào là score attention.
* Bước 4: Tính toán vector bối cảnh (context vector) là tổng của các trọng số attention nhân với vector trạng thái ẩn của decoder tại time-step tương ứng

Như vậy cơ chế attention được ra đời để giải quyết các vấn đề của mô hình seq2seq (Transformer và cơ chế attention ra đời để thay thế cho seq2seq không cần đến mạng nơ-ron hồi quy), với ý tưởng sử dụng một vector bối cảnh có thể tương tác với toàn bộ vector trạng thái ẩn của encoder thay vì chỉ sử dụng vector trạng thái ẩn cuối cùng ddeer tạo ra vector biểu diễn cho decoder.

### **3.1.6. Mô hình Transformer**

#### **3.1.6.1. Cơ chế Attention trong mô hình Transformer.**

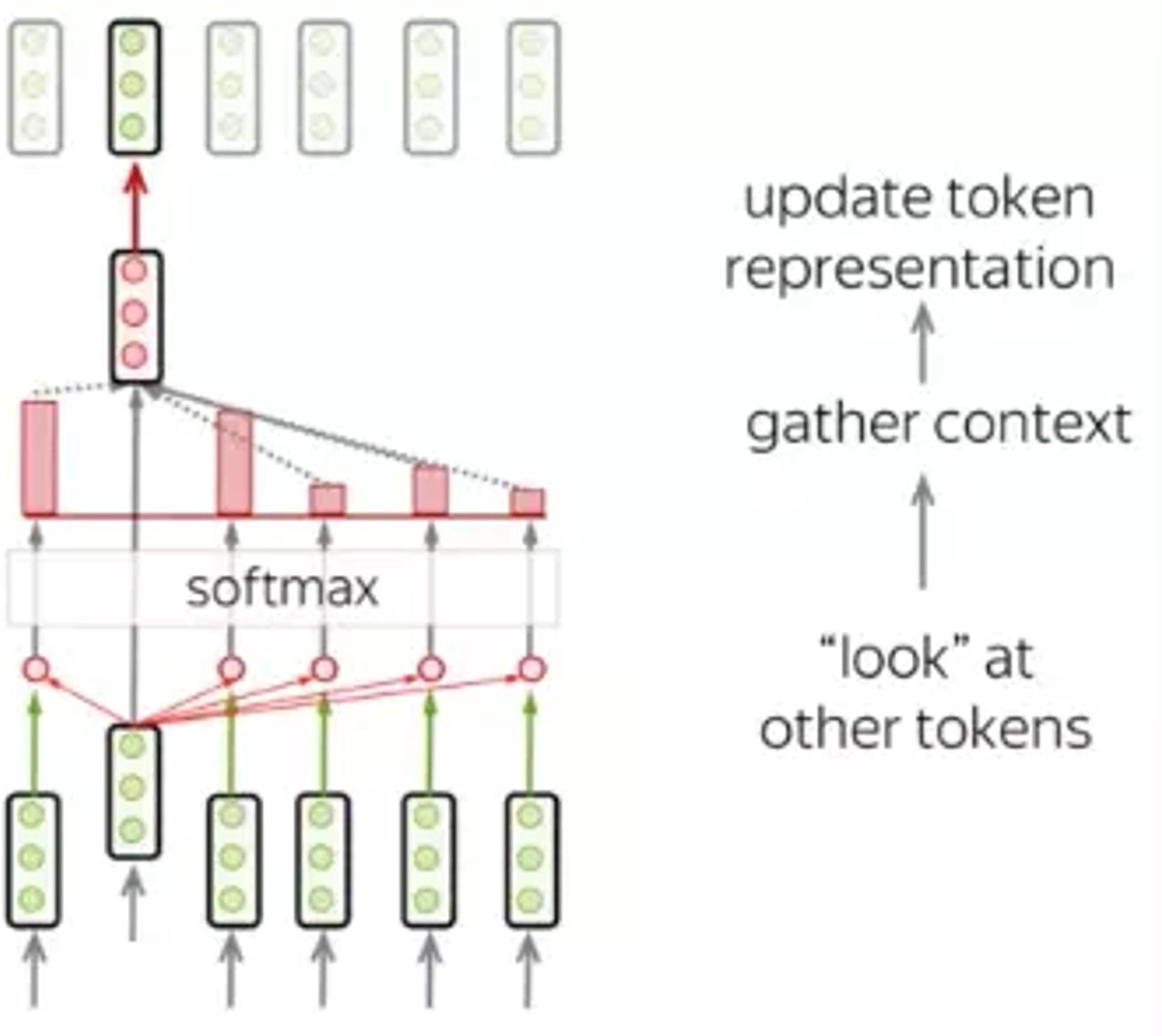
Với cơ chế Attention, Transformer sẽ không cần tới các phép toán convolutional và recurrence, với cơ chế xử lý song song rõ ràng cho thấy được cơ chế này cung cấp một hiệu suất vượt trội và thời gian huấn luyện được tối ưu đang kể. Việc ở các mạng trước đó, khi sử dụng với chuỗi đầu vào là một chuỗi dài, sẽ khiến cho việc tìm hiểu mối tương quan giữa các từ trở nên khó khăn hơn rất nhiều.

Trong Transformer vấn đề này đã được giải quyết nhưng sẽ phải giảm effective resolution do lấy trung bình các attention-weighted position, những vấn để này cũng sẽ được giải quyết ở phần sau với Multi-head attention.

Trong bài toán NLP, đối với trường hợp cụ thể như ta cần mã hóa (encode) một chuỗi đầu vào thành một vector ngữ cảnh rồi sau đó giải mã (decode) dự đoán các từ trước đó. Nhưng chúng ta sẽ gặp một số vấn đề như sau :

* Vanishing gradient: mất mát đạo hàm ở những hidden state cuối khi đầu vào là một chuỗi dài như đoạn văn
* Exploding gradient: Hiện tượng gradient tăng mất kiểm soát do dồn ở những lớp cuối và đặc biệt xảy ra với những câu dài như đoạn văn.
* Nén bộ nhớ (Memory compression): Do việc embed chuỗi đầu vào thành vector cố định kích thước, qua nhiều lần thử nghiệm cho thấy được mô hình này học với các câu dài rất kém trong khi đó lại rất lãng phí bộ nhớ với những câu ngắn. Vấn đề này vẫn còn xuất hiện trong LSTM và GRU

Không phù hợp với dạng dữ liệu có mối quan hệ liên kết ngữ nghĩa: Ví dụ với câu: “She is doing homework”. Nhận thấy “homework” có mối quan hệ nhiều hơn với “doing”. Đối với RNN - học tuần tự từ trái sang phải nhưng không có cơ chế để mô hình học được sự liên quan giữa các từ. Attention sẽ giúp chúng ta thực hiện được việc đó.



Hình 3.12: Miêu tả hoạt động của self-attention

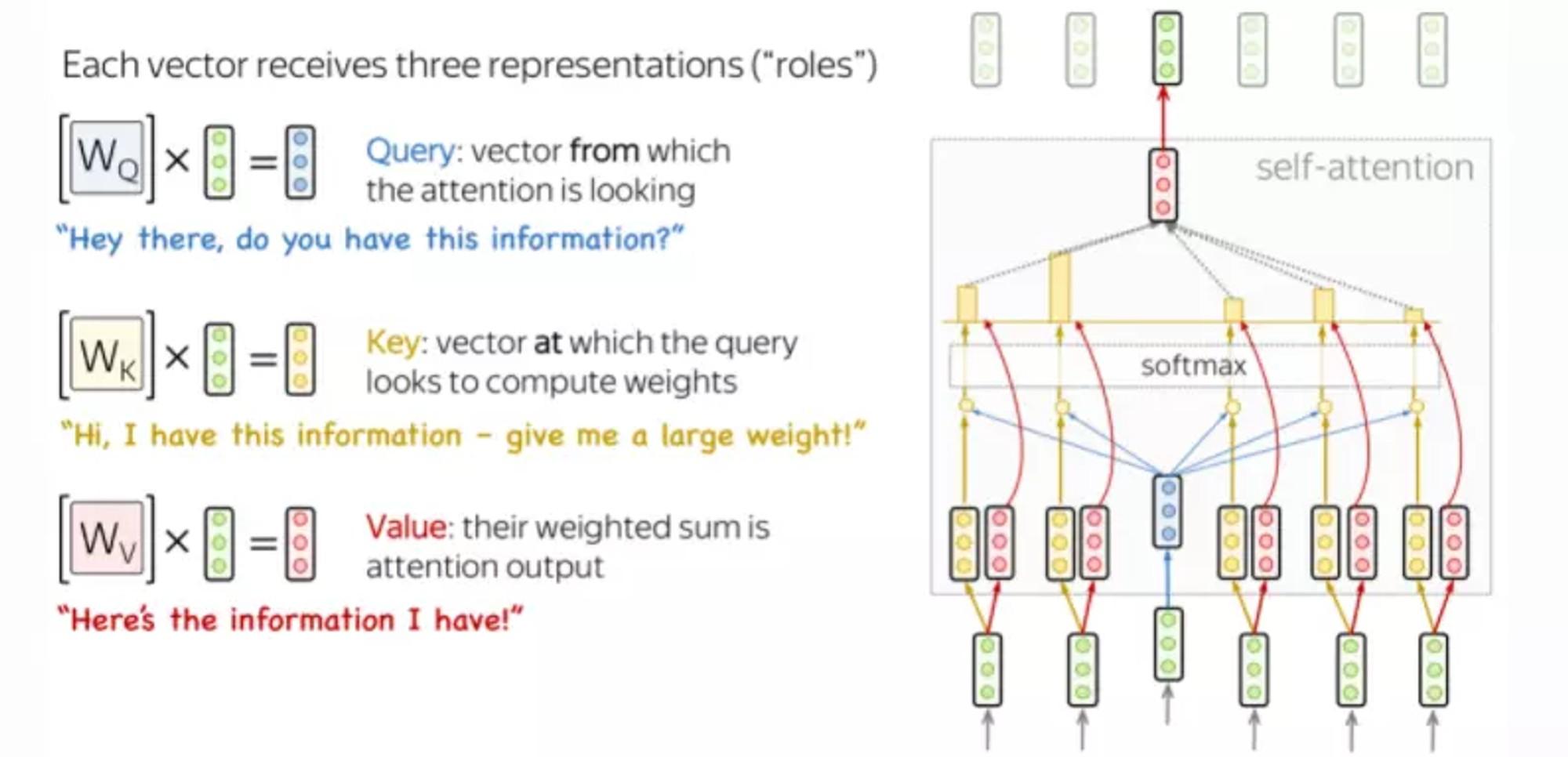
Còn self-attention có thể hiểu là attention trong 1 câu, khi từng thành phần trong câu sẽ tương tác với nhau. Từng token sẽ quan sát các tokens còn lại, thu thập nhữ cảnh của câu và cật nhập vào vector biểu diễn. Để xây dựng cơ chế self-attention ta cần chú ý đến hoạt động của vector biểu diễn cho mỗi từ lần lượt là:

* Query: được sử dụng để đặt câu hỏi từ này ở trong câu có ý nghĩa gì.
* Key: trả lời từ hiện tại có liên quan thế nào đến từ mà query đang hỏi.
* Value: chứa thông tin của từ đang hỏi.



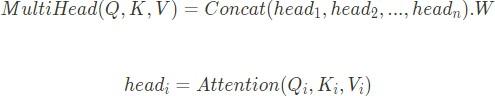
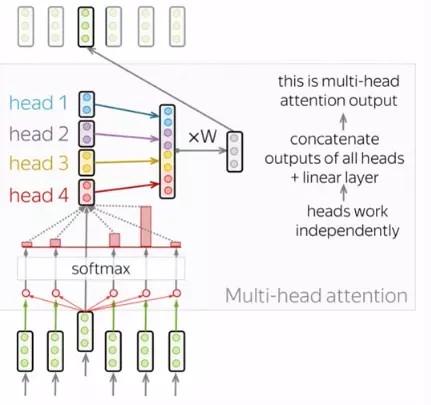
Dựa vào ý nghĩa của *q, k, v* có thể giải thích được công thức của Transformers Attention:

* *q.k* qua hàm softmax để đưa ra xác suất của từ liên quan nhất với từng từ để hỏi tương ứng.
* Sau đó nhân với *v* để đưa ra giá trị dựa vào sự tương quan đó.

Query được sử dụng khi một token quan sát những tokens còn lại, nó sẽ tìm kiếm thông tin xung quanh để hiểu được ngữ cảnh và mối quan hệ của nó với tokens còn lại. Key sẽ phản hồi yêu cầu của Query và được sử dụng để tính trọng số attention. Cuối cùng, Value được sử dụng trong số attention vừa rồi để tính ra vector đại diện (attention vector).

*Hình 3.13: Mô tả nhiệm vụ và chức năng của các vector query, key, value*

Mỗi quá trình như vậy được gọi là 1 head của attention. Khi lặp lại quá trình này nhiều lần (trong bài báo là 3 heads) ta sẽ thu được quá trình Multi-head Attention như biến đổi bên dưới:



*Hình 3.14: Mô tả chức năng của MutiHead Attention.*

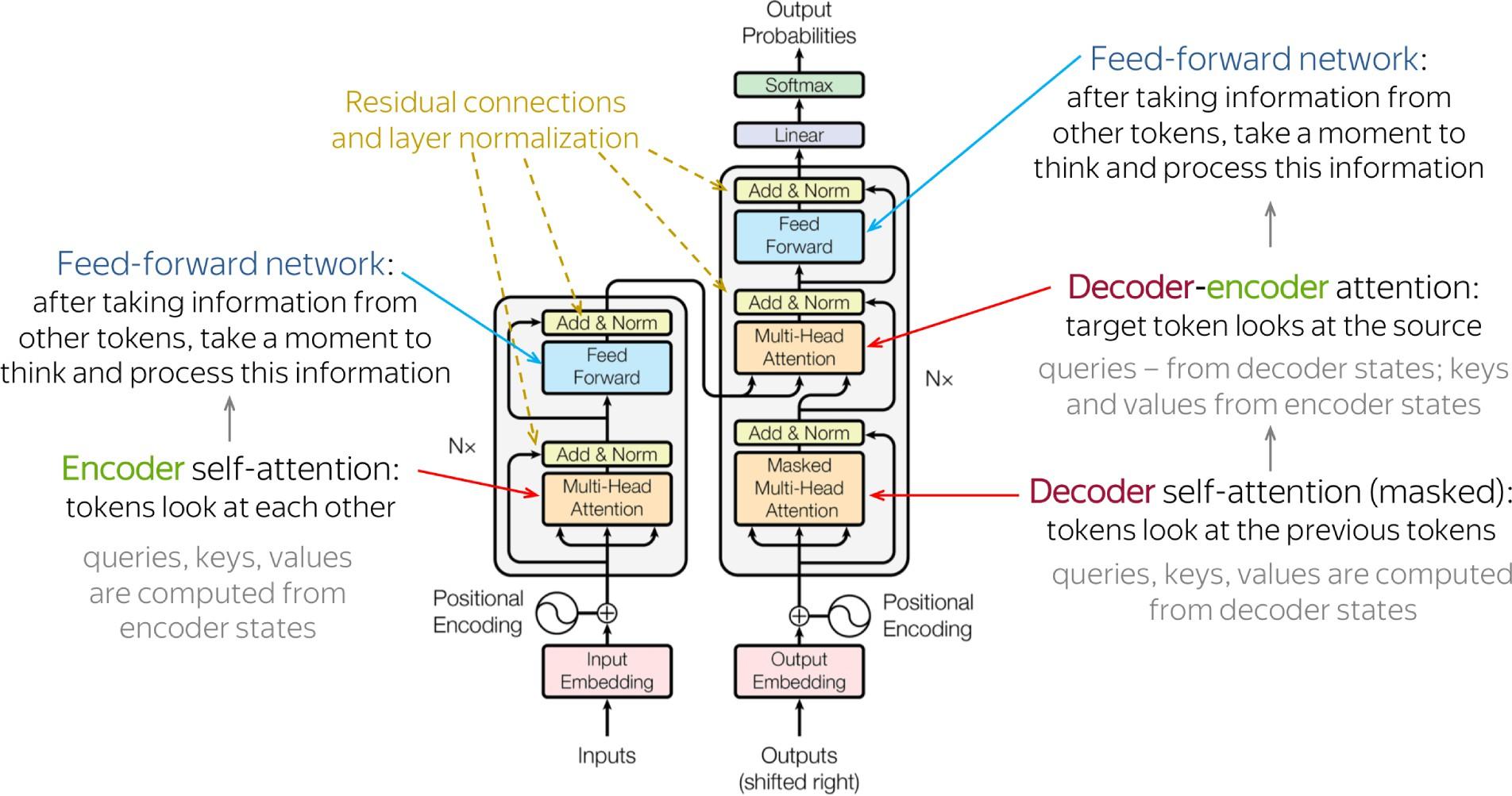
Để trả về output có cùng kích thước với ma trận input chúng ta chỉ cần nhân với ma trận W để chiều rộng bằng với chiều rộng của ma trận input.

#### **3.1.6.2. Cấu trúc của mô hình Transformer.**

Tiếp theo ta sẽ đi tìm hiểu về các thành phần chính cấu tạo thành transformer. Đây là cấu trúc của transformer được giới thiệu trong bài báo “Attention is all you need”. Mô hình thực hiện chính xác những gì đã được giới thiệu ở trên.

Ở bên trái là encoder, thông thường có Nx = 6 layers chồng lên nhau. Mỗi layer sẽ có multi-head attention như đã tìm hiểu và khối feed-forward. Ngoài ra còn các các kết nối residual giống như trong mạng Resnet. Ở bên phải là decoder, tương tự cũng có Nx = 6 layers chồng lên nhau.

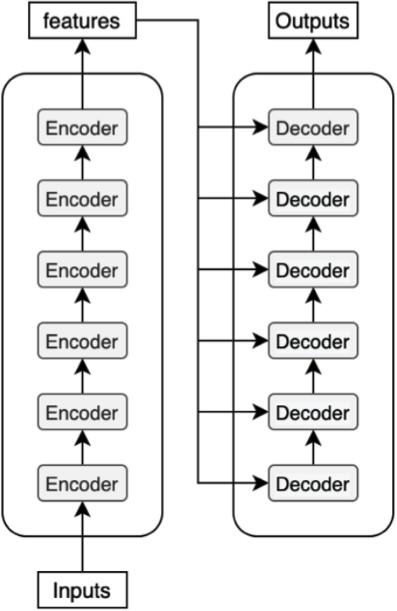
Kiến trúc thì khá giống encoder những chỉ có thêm khối masked multi-head attention ở vị trí đầu tiên. Ta sẽ tìm hiểu sâu hơn các thành phần trong transformer.



*Hình 3.15: Hình mô tả cấu trúc của mô hình Transformer.*

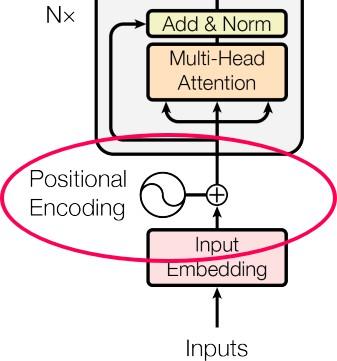
Với kiến trúc của Transformer có 2 phần chính là encoder và decoder:

* Encoders: gồm bộ lớp encoder giống hệt nhau và xếp chồng lên nhau. Mỗi lớp có 2 khối chính đó là: attention block và feed-forward block, feed-forward network được kết nối đầy đủ theo vị trí ( positionwide fully connected feed-forward network ).
* Decoders: Mỗi khối decoder nhận được các features từ encoder. Chúng ta có thể hình dung được sự liên kết giữa decoder và encoder



Hình 3.16: Hình ảnh mô tả sự liên kết giữa decoder và encoder.

**Positional Encoding:**



*Hình 3.17: Vị trí của positional encoder trong mô hình Transformer.*

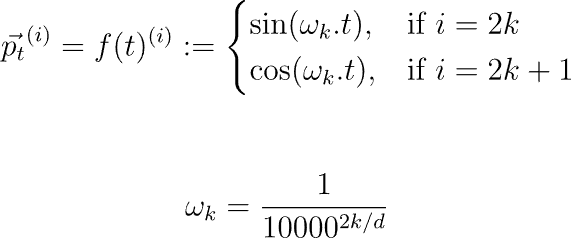
Khác với việc học tuần tự của RNN, Transformer loại bỏ cơ chế học tuần tự để học song song dựa trên multi-head self-attention, nên máy tính không hiểu được cấu trúc của câu dẫn đến việc sai về mặt ngữ nghĩa. Do đó cần một cách để giúp máy tính hiểu được thứ tự các từ trong câu, ý tưởng là ta sẽ đánh trọng số cho các từ trong câu.

Cách tiếp cận ngây thơ để đánh trọng trọng số là thêm giá trị mỗi từ theo tuyến tính. Có nghĩa là với từ đầu tiên thêm 1, từ thứ hai thêm hai, cứ như vậy cho đến hết. Với cách tiếp cần này sẽ có vài vấn đề như: các giá trị sẽ trở lên quá lớn, mô hình mất nhiều thời gian để hội tụ hơn (lý do các bạn có thể tìm kiếm với từ khóa “feature scaling”)

Một cách tiếp cận khác là thêm các giá trị trong khoảng [0, 1]; với giá trị 0 cho từ đầu tiên và 1 cho từ cuối cùng. Một trong những vấn đề đối với cách tiếp cần này là bạn không thể tìm ra có bao nhiêu từ hiện diện trong một phạm vi cụ thể.

Nói cách khác, giá trị cộng thêm không có ý nghĩa nhất quán trong các câu khác nhau. Cách tiếp cận tốt nhất nên thỏa mãn các yêu cầu sau:

* Giá trị được thêm nên là duy nhất với mỗi time-step.
* Khoảng cách giữa các từ trong câu nên thống nhất với mọi câu.
* Giá trị thêm vào cần phải được giới hạn.
* Giá trị thêm vào cần phải làm rõ được vị trí các từ trong câu. Công thức tính toán positional encoding trong Transformer là:



Trong đó :

* d là số chiều dài của vector từ.
* k là chỉ số của chiều trong vector từ
* là vị trí từ trong câu

Mỗi thành phần trong ma trận positional encoding sẽ có giá trị sin hoặc cos của một hàm sóng với tần số khác nhau. Khi kết hợp với ma trận từ nhúng, positional encoding sẽ giúp mô hình Transformer biết được vị trí tương đối của từng từ trong câu.

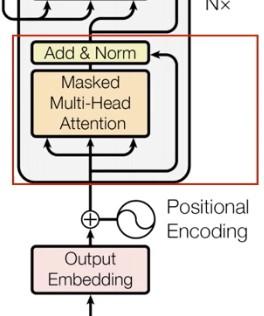
Ví dụ: Giả sử chúng ta có một câu gồm các từ "I", "love", "cats". Đầu tiên, mỗi từ sẽ được biểu diễn bằng vector từ nhúng. Tiếp theo, positional encoding sẽ được thêm vào các vector từ để đưa thông tin vị trí vào mô hình Transformer.

* Đầu vào ban đầu: [vector\_embed("I"), vector\_embed("love"), vector\_embed("cats")]
* Sau khi thêm positional encoding: [vector\_embed("I")+positional\_encoding("I"), vector\_embed("love")+positional\_encoding("love"), vector\_embed("cats")+positional\_encoding("cats")]

Positional encoding giúp mô hình Transformer phân biệt được vị trí và thứ tự các từ trong câu, làm cơ sở cho việc xử lý thông tin chuỗi trong mô hình.

**Masked Multi-Head Attention**

Một layer tương tự như Multi-Head Attention, điểm khác biệt là ở lúc train, khi đã biết rõ output mong muốn để đưa vào làm input của time-step tiếp theo. Để hiểu rõ điều này trước tiên bạn cần biết cách Decoder vận hành, tôi sẽ lấy ví dụ trong bài toán text generate với câu “Tall house with wide door”.

Cũng giống như RNN, Decoder sử dụng output của time-step trước làm input của time-step hiện tại. Tức với ví dụ trên ta sẽ có các bước decode sau:

Hình 3.18: Hình ảnh vị trí của lớp masked multi head attention.

* input = “” qua Masked Multi-Head Attention, nhận thêm thông tin từ Encoder để đưa ra thông tin đầu tiên output = “Tall”.
* input = “Tall” → output = “Tall house”.
* input = “Tall house” → output = “Tall house with”.
* input = “Tall house with” → output = “Tall house with wide”.
* input = “Tall house with wide” → output = “Tall house with wide door”.

Mọi thứ sẽ như trên nếu mô hình đã được train xong, nhưng trong quá trình train output ở time-step trước có thể sai khiến output của time-step hiện tại cũng có thể sai theo. Và đương nhiên việc này có thể dần cải thiện trong quá trình train, nhưng sẽ rất tốn tài nguyên. Vì vậy giải pháp được đưa ra là sử dụng output mà chúng ta muốn đưa vào làm input của từng time-step. Mỗi time-step sẽ nhập input tương ứng nên có ở thời điểm đó, tức sẽ cho biết các từ nên được generate ở thời đó và che đi phần còn lại. Để máy tính hiểu được cách làm trên ta sử dụng Look-ahead Mask là ma trận tam giác dưới với các giá trị 1 biểu thị từ không muốn che, và 0 cho từ muốn che.

**Add & Norm layers và Feed-forward layers**

Normalization layers: Trong cấu trúc của transformer có các lớp “Add và Norm” thì “Norm” ở đây thể hiện cho Normalization layer. Lớp này đơn giản là sẽ chuẩn hóa lại đầu ra của multi-head attention, mang lại hiệu quả cho việc nâng cao khả năng hội tụ.

Khối Feed-Forward: Sau khi thực hiện tính toán ở khối attention cho mỗi lớp thì khối tiếp theo là FFN. Có thể hiểu là cơ chế attention giúp thu thập thông tin từ những tokens đầu vào thì FFN là khối xử lý những thông tin đó.

**Tổng quan về việc sử dụng mô hình Transformer.**

Việc áp dụng Transformer vào bài toán nhận diện chữ giúp mô hình có khả năng mô hình hóa quan hệ phức tạp giữa hình ảnh và văn bản. Các cơ chế attention trong Transformer giúp mô hình tập trung vào các phần quan trọng của dữ liệu đầu vào và giúp cải thiện chính xác của hệ thống trích xuất thông tin.

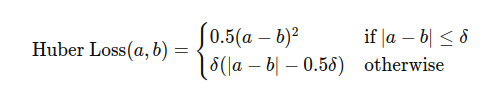
## **3.2. Hàm mất mát (Loss Function).**

### **3.2.1. Định nghĩa.**

Loss function (hàm mất mát) là một hàm toán học được sử dụng để đo lường sự khác nhau giữa giá trị dự đoán của mô hình và giá trị đích (ground truth). Mục tiêu của hàm mất mát là tối thiểu hóa sự khác nhau này, từ đó cung cấp cho mô hình một phản hồi về chất lượng của dự đoán và hướng dẫn cho mô hình để cải thiện hiệu suất của dự đoán.

Trong deep learning, hàm mất mát là một thành phần rất quan trọng dùng để đánh giá mức độ sai lệch giữa dự đoán của mô hình và giá trị thực tế. Các loại hàm mất mát khác nhau sẽ phù hợp với các dạng bài toán khác nhau. Dưới đây là một số hàm mất mát phổ biến :

* Mean Square Error (MSE) : Dùng để đo lường sự khác biệt giữa dự đoán và giá trị thực tế trong các bài toán hồi quy/ Ví dụ như bài toán hồi quy (regression), như dự đoán giá nhà, dự đoán nhiệt độ, … Công thức như sau:
* Mean Absolute Error (MAE): Tương tự như MSE, nhưng có thêm giá trị tuyệt đối thay vì bình phương, giúp giảm thiểu ảnh hưởng của các giá trị ngoại lệ. Thường sử dụng cho bài toán hồi quy khi có nhiều ngoại lệ. Công thức như sau :
* Huber Loss: là sự kết hợp ưu điểm của MSE và MAE, ít nhạy cảm với các giá trị ngoại lệ hơn MSE, nhưng lại mượt mà hơn MAE, phù hợp các bài toán hồi quy với dữ liệu có ngoại lệ. Công thức như sau :



* Binary Cross-Entropy (Log loss): Dùng để đo lường sự khác nhau giữa phân phối xác suất dự đoán và nhãn thực tế trong bài toán phân loại nhị phân. Ví dụ như phân loại xem email có phải spam hay không spam. Công thức như sau :

* Categorical Cross-Entropy : Tương tự như Binary Cross Entropy Loss nhưng cho các bài toán phân loại đa lớp. Ví dụ như phân loại hình ảnh , phân loại bệnh, … Công thức như sau :

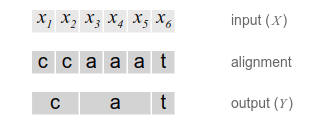
* Kullback-Leibler DIvergence (KL Divergence): Được sử dụng để đo lường sự khác biệt giữa hai phân phối xác suất. Thường được sử dụng trong mô hình Variational Autoencoder (VAE). Công thức như sau

Mỗi loại hàm mất mát đều có ưu nhược điểm riêng, và lựa chọn hàm mất mát phù hợp sẽ ảnh hưởng lớn đến hiệu suất và khả năng tổng quát hóa của mô hình. Khi thiết kế mô hình deep learning, việc lựa chọn hàm mất mát cần phải dựa vào các đặc điểm của bài toán và dữ liệu cụ thể. Trong thuật toán cho bài toán text detection (ví dụ như thuật toán DB - Differentiable Binarization) người ta thường sử dụng Binary Cross-Entropy, ngược lại trong bài toán text recognition, người ta sẽ thường hay sử dụng hàm mất mát là Connectionist Temporal Classification Loss (hay còn gọi là CTC loss).

### **3.2.2. Connectionist Temporal Classification Loss (CTC Loss).**

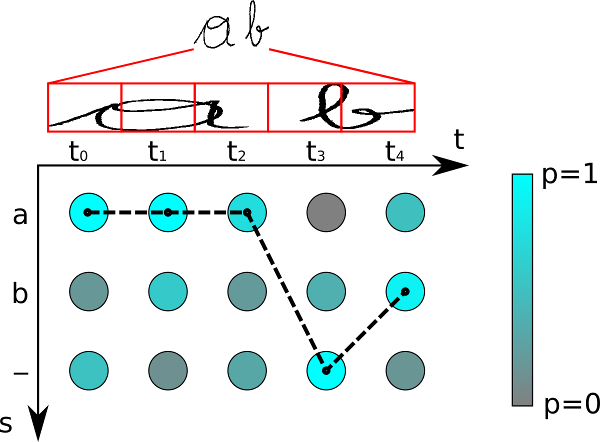
CTC loss được thiết kế để huấn luyện mô hình nhận diện chữ (text recognition) cho phép các mạng nơ-ron có dự đoán đầu ra các chuỗi mà không cần dữ liệu đào tạo đã được phân đoạn trước. Trong các nhiệm vụ dự đoán chuỗi điển hình, sự liên kết giữa chuỗi đầu vào (một đoạn hình ảnh) và đầu ra (văn bản) thường không được biết trước. CTC xử lý vấn đề này bằng cách giới thiệu một bước trung gian, từ đó tự động liên kết đầu vào với đầu ra.

CTC loss giúp giải quyết các vấn đề như phân biệt giữa các ký tự liền kề, xác định khoảng cách giữa các ký tự trống, và các quy tắc biểu diễn đầu ra khác. Nó cũng cho phép mô hình tạo ra các chuỗi đầu ra có độ dài khác nhau từ các chuỗi dự đoán ban đầu.

CTC đơn giản là một loss function được sử dụng để huấn luyện các Deep Learning model. Mục tiêu của nó là tìm ra cách ánh xạ (alignment) giữa một input X và output Y. Nó không yêu cầu Aligned Data (dữ liệu được gán nhãn cụ thể cho mỗi timestep như thuật toán CRNN, hay seq2seq, …), bởi vì nó có thể đưa ra được xác suất cho mỗi khả năng Align từ X sang Y. Đối với bài toán OCR, nó chỉ yêu cầu đầu vào là một hình ảnh (ma trận feature của hình ảnh) và đoạn text tương ứng với hình ảnh đó.

Hình 3.19: Hình ảnh mô tả CTC loss

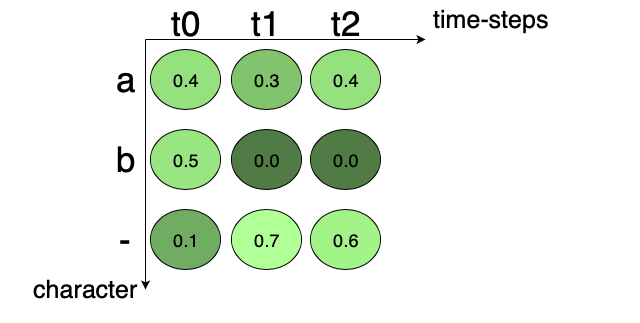
Thực chất, bản thân CTC cũng không biết chính xác cách alignment giữa X và Y. Nếu đưa vào mô hình là X thì nó sẽ trả lại cho ta tất cả các khả năng của Y, kèm theo xác suất của mỗi khả năng đó. Chúng ta cần huấn luyện mô hình sao cho:



Hình 3.20: Hình ảnh minh họa cho CTC loss

trong đó Y\* là kết đầu ra cuối cùng mà ta mong đợi, càng giống với nhãn càng tốt.

Quá trình làm việc của CTC trong bài toán nhận diện chữ hay bài toán OCR gồm 3 bước:

* **Encoding text (Mã hóa)**: Lý tường thì tại mỗi timestep sẽ tương ứng với một ký tự. Nhưng nếu không phải vậy và xuất hiện một ký trong cả 2 timestep thì sao ? Khi đó, kết quả sẽ xuất hiện các ký tự trùng nhau. CTC giải quyết các vấn đề này bằng cách gộp tất cả các ký tự trùng nhau thành một. Ví dụ: aaabb -> ab . Tuy nhiên, nếu cứ gộp tất cả các ký tự trùng nhau vào thì có những từ mà bản thân những từ đó có các ký tự trùng nhau thì phải làm thế nào ? Ví dụ các từ như hello, sorry, … Để xử lý vấn đề nảy sinh này, CTC sử dụng một ký tự giải, hay còn gọi là “blank” và ký hiệu là ‘-’. Trong khi mã hóa Text, nếu gặp 2 ký tự trung nhau, CTC sẽ chèn thêm ký tự blank này vào giữa chúng. Ví dụ: hello -> hh-e-lll-ll-o. Trong quá trình decoding text, nếu gặp ký tự blank này thì CTC hiểu rằng phải giữa lại cả 2 ký tự 2 bên ký hiệu đó.
* **Tính toán Loss**: Loss được tính toán cho mỗi training sample (gồm một cặp ảnh và nhãn text tương ứng). Nó là tổng tất cả scores của tất cả các khả năng Alignment của nhãn text. Giả sử chúng ta có một ma trận score là output của CRNN như sau : 

Hình 3.21: Hình ảnh mô tả ma trận xác suất.

Ma trận này gồm 2 timestep, và nhãn text có 3 ký tự gồm a, b và ký tự blank (-). Tổng score time tại mỗi timestep bằng 1.

* Các khả năng Alignment của ký tự a là : aaa, a--, -a-, aa-, -aa, --a → Tổng score của a = 0.4x0.3x0.4 + 0.4x0.7x0.6 + 0.1x0.4x0.7 + 0.4x0.3x0.6 + 0.1x0.3x0.4 + 0.1x0.7x0.4 = 0.608 → Loss = – log 0.6084 = 0.216
* Các khả năng Alignment của ký tự b là : bbb, b–, b-, bb-, -bb, –b → Tổng score của b = 0.5x0.0x0.0 + 0.5x0.7x0.6 + 0.5x0.7 + 0.5x0.0x0.6 + 0.1x0.0x0.0 + 0.1x0.7x0.0 = 0.56 → Loss = – log 0.56 = 0.25.
* Các khả năng Alignment của ký tự blank: - - - → Tổng score của blank = 0.1 x 0.7 x 0.6 = 0.042 → Loss = – Log (0.42) = 1.377

Tổng Loss = 0.216 + 0.25 + 1.377 = 1.843. Chúng ta cần tối thiểu hóa

giá trị này trong quá trình huấn luyện model

* **Decoding Text (Giải mã)** : Quá trình giải mã một hình ảnh đầu vào diễn ra như sau : Tìm đường đi tối ưu nhất từ score matrix bằng cách chọn các ký tự có score cao nhất tại mỗi timestep sau đó xóa bỏ các ký tự trống, ký tự trùng lặp. Ví dụ như ở tính loss trên thì kết quả sẽ là : b- - **.** Tuy nhiên kết quả “b - -” chưa chắc đã đúng và là chuỗi có xác suất cao nhất. Từ đó xuất hiện thêm các thuật toán khác để tìm ra chuỗi có xác suất cao nhất mà tôi sẽ giới thiệu ở các phần sau.

Tổng quan, Quá trình lan truyền ngược (backpropagation) được sử dụng để tính toán đạo hàm của hàm mất mát CTC theo các tham số của mô hình. Sau đó các tham số đó được cập nhật bằng các phương pháp tối ưu hóa như gradient descent để tối ưu hóa hàm mất mát. Đó là cách CTC loss giúp mô hình học hiểu được mối liên hệ giữa đầu vào X và đầu ra Y, từ đó cải thiện dự đoán của model.

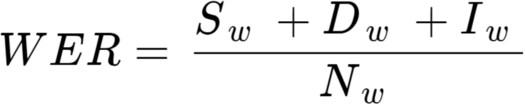
## **3.3. Các phương pháp đánh giá.**

Trong bài toán nhận diện chữ, để đánh giá độ chính xác của hệ thống, các phương pháp đánh giá sau đây thường được sử dụng : Word Error Rate (WER), Character Error Rate (CER).

Các phương pháp đánh giá này đều cung cấp thông tin quan trọng về độ chính xác của hệ thống nhận diện chữ. Thông thường càng thấp các chỉ số về WER, CER thì hệ thống nhận diện càng tốt. Đối với mỗi bài toán cụ thể, phương pháp đánh giá phù hợp sẽ được lựa chọn tùy thuộc vào yêu cầu và đặc điểm của tập dữ liệu.

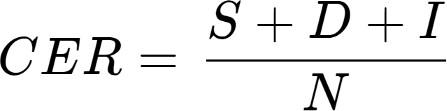
### **3.3.1. Word Error Rate (WER).**

* Word Error Rate hay WER là một phương pháp đánh giá phổ biến trong text recognition. Nó đo lường tỷ lệ lỗi giữa chuỗi văn bản được nhận dạng và chuỗi văn bản đúng. WER được tính bằng tổng số từ được chèn (insertions), xóa (deletions) và thay đổi (substitutions) chia cho tổng số từ trong chuỗi đúng. Mục tiêu là tìm cách giảm WER càng thấp càng tốt.
* Ví dụ: Văn bản nhãn là : “Tôi là sinh viên” và đầu ra mô hình là “Toi la sinh viên” thì ở đây tỷ lệ lỗi của WER là 50% vì 2 trên 4 từ trong câu trên đã bị sai. Dưới đây là công thức tỷ lệ lỗi của WER:



### **3.3.2. Character Error Rate (CER).**

* Character Error Rate (CER): CER là một phương pháp đánh giá tương tự như WER, nhưng thay vì đánh giá trên từng từ, nó đo lường tỷ lệ lỗi giữa chuỗi ký tự nhận dạng và chuỗi ký tự đúng. CER cũng được tính bằng tổng số ký tự chèn, xóa và thay đổi chia cho tổng số ký tự trong chuỗi đúng.

Trong đó: 

* S là Số lần thay thế
* D là số lần xuất
* I là số lần chèn.
* N là tổng số ký tự trong văn bản tham chiếu.

Mẫu số N có thể được tính theo công thức: N = S + D + C (trong đó C = Số ký tự đúng). Kết quả của phương trình này đại diện cho tỷ lệ phần trăm ký tự trong đầu ra OCR không chính xác so với văn bản tham chiếu đầu vào. Giá trị CER càng thấp (mô hình hoàn hảo khi CER=0), hiệu suất của mô hình OCR càng tốt.

Tùy thuộc trường hợp sử dụng và điều kiện liên quan, tỷ lệ lỗi ký tự CER được sử dụng để đánh giá mức độ hiệu quả của OCR tương ứng. Các tình huống và độ phức tạp khác nhau (ví dụ: văn bản in so với văn bản viết tay, loại nội dung, v.v.) có thể dẫn đến các hiệu suất OCR khác nhau. Tuy nhiên, chúng ta có thể tham khảo đánh giá độ chính xác của một số tổ chức uy tín tại Úc như sau (đối với văn bản in):

* Độ chính xác OCR tốt: CER 1‐2% (tức là chính xác 98–99%)
* Độ chính xác OCR trung bình: CER 2-10%
* Độ chính xác OCR kém: CER> 10% (tức là độ chính xác dưới 90%)

Đối với các trường hợp phức tạp liên quan đến văn bản viết tay có nội dung không đồng nhất và xa rời từ vựng (ví dụ: đơn đăng ký), giá trị CER cao khoảng 20% có thể được coi là đạt yêu cầu.

### **3.3.3. Beam-Search Output.**

Beam search là một thuật toán được sử dụng trong các tác vụ tạo ngôn ngữ tự nhiên, bao gồm cả bài toán nhận diện chữ. Beam search được sử dụng để tìm kiếm qua không gian đề xuất chuỗi kết quả có thể, với mục tiêu tìm ra chuỗi có xác suất cao nhất.

Khi áp dụng Beam Search trong bài toán nhận diện chữ, thuật toán sẽ duyệt qua các chuỗi đầu ra có khả năng từ các mô hình hình ảnh thành chữ và tìm ra chuỗi có xác suất cao nhất dựa trên mô hình đã huấn luyện.

Beam Search hoạt động theo các như sau:

* Ban đầu, thuật toán khởi tạo một số lượng k chuỗi đầu ra gọi là beam width với các chuỗi rỗng.
* Tiếp theo, thuật toán duyệt qua các chuỗi trong beam hiện tại và mở rộng chúng bằng cách thêm các từ mới từ không gian từ vựng hoặc thông qua quy tắc ngữ cảnh.
* Mỗi chuỗi dược gán kèm với một giá trị xác suất dự đoán dựa trên mô hình text recognition.
* Beam Search tiếp tục lựa chọn và mở rộng và chuỗi có xác suất cao nhất và cắt giảm beam width lại để chỉ giữ lại k chuỗi có xác suất cao nhất.
* Quá trình lựa chọn và mở rộng được lặp lại cho đến khi đạt được điều kiện dừng, chẳng hạn khi đạt tới độ dài tối đa cho kết quả hoặc khi không còn chuỗi nào để cải thiện đáng kể xác suất.

Beam

Beam Search sẽ trả về kết quả cuối cùng là chuỗi có xác suất cao nhất trong beam hoặc sau khi đạt được điều kiện dừng. Điều này giúp cải thiện chất lượng của kết quả đầu ra so với việc chỉ lựa chọn xác suất cao nhất tại mỗi bước thời gian như trong thuật toán đầu ra của CTC đã phân tích ở trên.

# C**HƯƠNG 4: CÁC MÔ-ĐUN THỰC HIỆN BÀI TOÁN MÔ TẢ HÌNH ẢNH.**

Ở chương 2, chúng ta đã có góc nhìn tổng quan về bài toán trích xuất thông tin, về các khái niệm, ứng dụng, các dạng, tính đặc trưng và các phương pháp tiếp cận. Thêm với chương 3, chúng ta đã điểm qua rất nhiều các lý thuyết nền tảng để xây dựng các mô hình học sâu một cách mạnh mẽ, đạt hiệu suất cao. Ở chúng 4, chúng ta sẽ đi sâu hơn về các mô đun mà bài toán trích xuất thông tin sẽ sử dụng:

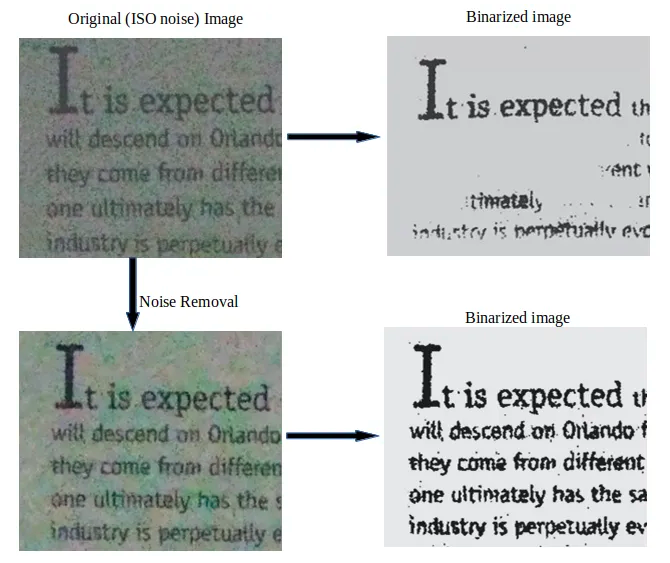
* Pre-processing module: Module này sẽ có nhiệm vụ tiền xử lý dữ liệu ảnh trước khi đưa vào các mô hình phát hiện và nhận diện chữ. Module này bao gồm các bước như lọc nhiễu, tăng độ phân giải, căn chỉnh hướng chiều của văn bản, xử lý ảnh mờ.
* Text detection module: Module này sẽ chịu trách nhiệm phát hiện các chữ (text) trong ảnh. Các phương pháp phổ biến trong module này thường là các phương pháp deep neural networks.
* Text Recognition module: Module này sử dụng các mô hình trích xuất đặc trưng từ hình ảnh chữ sau khi đã được đi qua text detection module. Sau khi trích xuất đặc trưng xong, mô hình sử dụng mô hình ngôn ngữ để mô hình hóa và dự đoán từ các đặc trưng đã được trích xuất thành các chữ cụ thể.
* Post-processing module: Module này có nhiệm vụ xử lý kết quả đầu ra của hệ thống trích xuất thông tin. Lấy ra các thông tin mà hệ thống cần trích xuất, ví dụ như ở mẫu Ielts, thông tin cần lấy ra là họ tên, ngày tháng thi, điểm của từng mục, … Module này sẽ xử lý được thiết kế như một rule based để trích xuất thông tin cần lấy một cách nhanh chóng, hiệu quả cũng như đảm bảo được chất lượng.

Trong chương này, chúng ta sẽ tìm hiểu các module trên.

## **4.1. Tiền xử lý dữ liệu hình ảnh (Pre-processing image).**

Pre-processing module trong bài toán trích xuất thông tin là một bước quan trọng để chuẩn bị dữ liệu hình ảnh trước khi đưa vào các module ở phía sau. Module này thường bao gồm các bước sau:

* Loại bỏ nhiễu và xử lý ảnh mờ: Dữ liệu hình ảnh thường bị ảnh hưởng bởi nhiễu từ các nguồn bên ngoài như thông tin về độ sáng (quá mạnh hoặc quá yếu) trong hình ảnh có thể khiến cho hình ảnh trở lên khó đọc hơn. Ở đây có thể có thể sử dụng các kỹ thuật làm mờ, làm mịn ảnh (Blurring or Smoothing) giúp loại bỏ các “ngoại lệ” điểm ảnh (outlier pixel) có thể xảy ra trong hình ảnh. Có nhiều các bộ lọc khác nhau đều có ưu điểm và nhược điểm riêng:
* Gaussian Blur:Sử dụng Gaussian kernel để thực hiện tích chập và giúp loại bỏ nhiễu Gaussian ra khỏi hình ảnh. Nó nhanh hơn nhiều so với các kỹ thuật làm mờ khác nhưng không giữ được các cạnh và điều này có thể ảnh hưởng đến đầu ra của text recognition module.
* Median Blur: Thay thế phần tử trung tâm của vùng kernel bằng giá trị trung bình của các điểm ảnh (pixels) trong vùng kernel. Nó rất tốt trong việc loại bỏ “salt and pepper noise” (nhiễu muối) khỏi tài liệu scanned.
* Bilateral Filtering: Nó có hiệu quả cao trong việc loại bỏ nhiễu trong khi vẫn giữ được các cạnh sắc nét. Cùng với việc sử dụng bộ lọc Gaussian trong không gian, nó còn có một bộ lọc Gaussian khác là hàm chênh lệch pixel. Hàm Gaussian trong không gian đảm bảo rằng chỉ các pixel lân cận mới được xem xét để làm mờ, trong khi hàm Gaussian về chênh lệch cường độ đảm bảo rằng chỉ những pixel có cường độ tương tự với pixel trung tâm mới được xem xét để làm mờ. Vì vậy, nó đảm bảo các cạnh vì các pixel ở các cạnh sẽ có sự thay đổi lớn về cường độ.



Hình 4.1: Hình ảnh mô tả tiền xử lý dữ liệu.

* Xử lý nhiễu và cải thiện chất lượng ảnh bằng autoencoders: Với sự phát triển của các mô hình học sâu trong thị giác máy tính, đã có rất nhiều nghiên cứu về nâng cao chất lượng hình ảnh, loại bỏ nhiễu trong đó nổi bật có mô hình autoencoders. AutoEncoders bao gồm một bộ kiến trúc mạng encoder và decoder. Trong khi encoder làm nhiệm vụ nén dữ liệu đầu vào thành biểu diễn có chiều thấp hơn thì decoders làm nhiệm vụ tái tạo lại biểu diễn để thu được đầu ra mô phỏng càng giống với đầu vào càng tốt. Khi làm như vậy, autoencoders sẽ tìm cách giữ lại các đặc trưng nổi bật và đồng thời bỏ đi các nhiễu của đầu vào.

Ngoài các phương pháp trên vẫn còn nhiều các phương pháp nhau dùng để cải thiện chất lượng ảnh đầu vào. Tất cả các phương pháp tiền xử lý dữ liệu này đều có một mục tiêu chung là giúp dữ liệu hình ảnh được chuẩn bị tốt hơn khi đưa vào các module sau, từ đó cải thiện độ chính xác và hiệu suất của hệ thống trích xuất thông tin.

## **4.2. Mô-đun phát hiện chữ (Text Detection Module).**

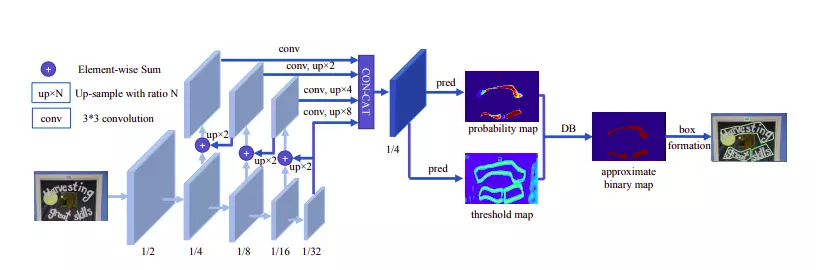
Bài toán phát hiện chữ trong ảnh (text detection) là bài toán xác định vị trí vùng có chữ trong ảnh đã trở nên phổ biến về mặt ứng dụng và nghiên cứu vì khả năng ứng dụng của nó. Đến hiện tại, bài toán phát hiện chữ trong ảnh thường được tiếp cận theo các hướng:

* **Mô hình phát hiện chữ dựa trên cơ chế hồi quy** (Regression Based Method) : Mô hình này giống như các mô hình dùng để phát hiện vật thể trong ảnh (object detection) và vật thể ở đây là chữ. Tuy nhiên do chữ có kích thước và hình dáng đa dạng như chữ cong, chữ nghiêng, … Do đó các mô hình phát hiện chữ dựa trên mô hình hồi quy như SSD, YOLO, … đều gặp hạn chế trong biểu diễn tất cả các trường hợp có thể gặp trong thực tế.
* **Mô hình phát hiện chữ dựa trên cơ chế phân đoạn** (Segmentation Based Method) : Mô hình này hoạt động dựa trên phương pháp phân đoạn ý nghĩa trong ảnh (segmentation), dự đoán các pixel ở cùng một đối tượng rồi sử dụng các thuật toán hậu xử lý để lấy ra vị trí các đối tượng vì vậy có thể xử lý tốt đối với độ đa dạng về hình ảnh cũng như kích thước của chữ. Có rất nhiều các phương pháp đã được nghiên cứu như phương pháp dự đoán trên mức ký tự CRAFT (Character Region Awareness for Text Detection), hay dự đoán trên mức từ như DB (Differentiable Binarization) hoặc Multi-scale FCN. Việc sử dụng trên mức ký tự sẽ không phải quan tâm nhiều tới vấn đề các từ sẽ được chia tách như thế nào vì việc chia tách như thế nào vì việc chia tách dựa trên nhiều tiêu chí như ngữ nghĩa, khoảng cách, màu sắc, … Nhưng lại gặp nhược điểm là việc chuẩn bị dữ liệu cho mức ký tự khá khó khăn.
* **Mô hình phát hiện chữ end to end** : Phương pháp end to end kết hợp cả hai mô hình phát hiện và nhận dạng chữ, do đó có thể tăng cường độ chính xác của mô hình phát hiện chữ thông qua kết quả của mô hình nhận dạng chữ. Có một số một hình đã được nghiên cứu như FOST, …

Trong phạm vi nghiên cứu của đề tài, ở đây chúng ta sẽ tập trung phân tích chủ yếu về 2 thuật toán là DB và DB++

### **4.2.1. Mô hình Differentiable Binarization**

Chúng ta sẽ cùng nhau đi tìm hiểu về mô hình **DB (Differentiable Binarization)**, một trong những mô hình nền tảng mạnh mẽ trong nhiệm vụ phát hiện chữ. Như trong bài báo “*Real-time Scene Text Detection with Differentiable Binarization*” có đề cập với kiến trúc mô hình và các đặc điểm của mô hình như sau :



Hình 4.2: Kiến trúc của mạng DB.

Các mô hình trong bài toán phát hiện chữ trước đây được nghiên cứu 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 đề xuất một ý tưởng cho mô hình học luôn cách đặt giá trị ngưỡng này thông qua việc huấn luyện.

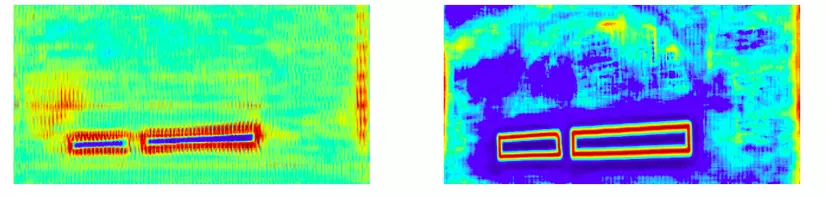
Kiến trúc mô hình được mô tả như hình trên. Mô hình sẽ nhận ảnh đầu vào và cho qua một bộ trích xuất đặc trưng sinh ra bản đồ đặc trưng F. Trong quá trình huấn luyện, bản đồ đặc trưng này được dùng để dự đoán bản đồ xác suất P và bản đồ ngưỡng T. Cuối cùng, sử dụng hai bản đồ này để tính ra bản đồ nhị 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 suất hoặc phản đồ nhị phân.

Kiến trúc của mô hình DB có hai ưu điểm nổi bật so với các kiến trúc khác 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 dựa trên phương pháp gọi là Differentiable Binarization.

**Differentiable 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 là bản đồ ngưỡng động được học từ mô hình, k là một chỉ số khuếch đại. P và T lần lượt là bản đồ xác suất và bản đồ ngưỡng.



Hình 4.3: Bản đồ ngưỡng động.

**Adaptive threshold**:hay còn gọi là ngưỡng động (hình ….). 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 tượng hay không thì bản đồ ngưỡng động thể hiện các điểm ảnh này có phải thuộc cạnh bao quanh đối tượng đó không ? Ở trong bài báo tác giả cũng so sánh bản đồ ngưỡng được học không giám sát ở ảnh trái và có giám sát ở ảnh phải. Kết quả huấn luyện có giám sát tốt hơn hẳn so với phương pháp không giám sát.

**Loss Function**: hay còn gọi là hàm mất mát. Theo bài báo, hàm mất mát tác giả sử dụng là tổng có trọng số của ba hàm mất mát cho bản đồ xác suất , bản đồ nhị phân và bản đồ ngưỡng :

trong đó lần lượt là trọng số ưu tiên của hàm mất mát cho bản đồ nhị phân và bản đồ ngưỡng. Do số lượng các điểm ảnh không phải chữ thường chiếm rất nhiều trong ảnh nên để giảm hiện tượng mất cân bằng giữa các lớp, tác giả 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 hàm mất mát được gọi là

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

trong đó là tập hợp các vị trí bên trong vùng chữ, là dự đoán của mô hình, là nhãn cho bản đồ ngưỡng.

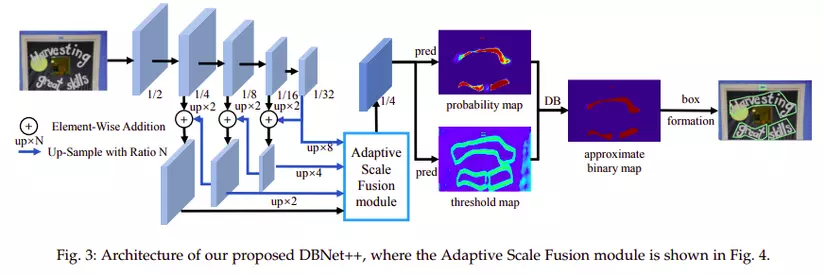
### **4.3.2. Mô hình Differentiable Binarization với Adaptive Scale Fusion.**

Mô hình Differentiable Binarization với Adaptive Scale Fusion hay còn gọi là mô hình DB++, ngay từ tên gọi chúng ta đã thấy đây là một phiên bản cải tiến dựa trên mô hình DB đã được trình bày ở mục trước. Mô hình DB trước đó đã làm đơn giản hóa quá trình tính toán và hậu xử lý hơn đáng kể so với các phương pháp trước đó như CRAFT, … bằng cách thay vì dùng một ngưỡng cố định thì sinh ra một bản đồ ngưỡng động cho từng pixels ở bản đồ xác suất được tối ưu cùng trong quá trình huấn luyện. Điều đó đã giúp mô hình DB đạt được độ chính xác cao hơn so với nhiều phương pháp trước đó.

Để tiếp tục cải thiện chất lượng của mô hình DB, nhóm tác giả của bài báo “*Real-Time Scene Text Detection with Differentiable Binarization and Adaptive Scale Fusion*” đã tiếp tục cải tiến chất lượng của mô hình DB, đề xuất sử dụng một kiến trúc mới sử dụng module Adaptive Scale Fusion (ASF) giúp mô hình có thể tổng hợp đặc trưng ở nhiều tỷ lệ khác nhau. Bây giờ chúng ta cùng đi vào chi tiết của thuật toán DB++ này.

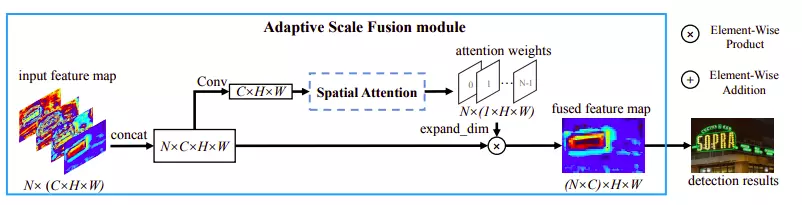
Về tư tưởng thiết kế mô hình, tác giả đề cập đến 3 ưu điểm giúp mô hình DB++ vượt trội so với các kiến trúc khác như sau :

* Sử dụng module DB được đề xuất trong bài báo trước đó giúp linh hoạt và tối ưu cùng lúc với mô hình.
* Module DB được loại bỏ trong quá trình suy luận do đó không phát sinh thêm thời gian xử lý.
* Khả năng biểu diễn mạnh mẽ nhờ khả năng tổng hợp đặc trưng ở nhiều mức độ tỷ lệ dưới sự hỗ trợ của Adaptive Scale Fusion.

Dưới đây là một số cải tiến của DB++ so với mô hình DB đã được trình bày trước đó: 

Hình 4.4: Kiến trúc tổng quan của mô hình DB++.

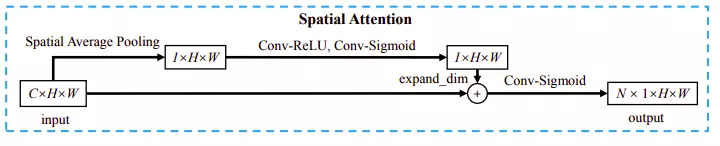
**Kiến trúc tổng quan:**

Mô hình DB++ được đề cập như ảnh trên. Ảnh đầu vào được cho qua một mô đun trích xuất đặc trưng phân cấp. Sau đó, các bản đồ đặc trưng được up sample về kích thước các bản đồ đặc trưng của lớp ngay trước đó rồi thực hiện tổng hợp đặc trưng của hai lớp này cùng với nhau và đưa vào mô đun Adaptive Scale Fusion (ASF) để tăng cường, sinh ra bản đồ đặc trưng F. Bản đồ đặc trưng F sau đó dùng để sinh ra bản đồ xác suất trên từng pixels P và bản đồ ngưỡng T. Cuối cùng bản đồ nhị phân sinh ra được tính toán dựa trên P và T.

Hình 4.5: Kiến trúc tổng quan của Adaptive Scale Fusion

**Adaptive Scale Fusion:**

Kiến trúc ASF được mô tả như hình trên. Thiết kế kiến trúc mô hình dựa trên một lý thuyết về mối liên hệ giữa hai khái niệm là context và scale. Context giúp mô hình tiếp nhận được vùng có kích thước lớn hơn trong khi ở mỗi tỷ lệ giúp mô hình có những biểu diễn đa dạng khác nhau ở các vùng tiếp nhận khác nhau và kích thước khác nhau nên đa dạng hơn. Tổng hợp đặc trưng ở nhiều tỷ lệ khác nhau giúp tăng cường khả năng biểu diễn của mô hình. Ý tưởng này cũng đã được đề cập ở bài báo Multi-Scale Feature Fusion hay trong U-Net cũng được được nói đến. Tuy nhiên ở kiến trúc DB++ được đề xuất ở bài báo có điểm đặc biệt. Thay vì chỉ cộng hay nối tất cả các bản đồ đặc trưng ở các scale vào với nhau, thì mô hình DB++ thực hiện ghép nối và cho đi qua mô đun ASF kết hợp với cơ chế **Spatial** **Attention** để có thể tổng hợp một cách linh hoạt đặc trưng ở nhiều tỷ lệ khác nhau.

Theo như kiến trúc được đề xuất trong bài báo, có tổng cộng 4 bản đồ đặc trưng ở các tỷ lệ tương ứng là 1/4, 1/8, 1/16 và 1/32. Bốn bản đồ đặc trưng này upscale về cùng kích thước và ghép nối vào nhau gọi là với N là 4. Bản đồ đặc trưng thu được cho qua lớp tích chập đưa về kích thước NxHxW để thực hiện tính Spatial Attention ra trọng số chú ý có kích thước Nx(1xHxW). Kết quả này sau đó nhân với bản đồ đặc trưng X ban đầu để ra kết quả cuối cùng. 

*Hình 4.6:Mô hình kiến trúc của Spatial Attention*

**Spatial Attention:**

Kiến trúc của Spatial Attention được mô tả như hình trên. Kiến trúc bao gồm 2 nhánh. Nhánh đầu tiên gồm các lớp Spatial Average Pooling, Conv-Relu, Conv-Sigmoid. Lớp Spatial Average Pooling giúp tổng hợp giá trị trung bình đặc trưng trên từng vị trí ở nhiều tỷ lệ khác nhau. Lớp này khiến mô đun Spatial Attention không những học được tính quan trọng của các giá trị ở vị trí khác nhau mà còn trên nhiều tỷ lệ khác nhau. Sau đó output đi qua lớp Conv-Relu, Conv-Sigmoid. Nhánh này thực hiện nhiệm vụ giúp tăng cường đặc trưng biểu diễn input đầu vào bằng cách cộng kết quả vào bản đồ đặc trưng ban đầu. Kết quả sau khi được cộng vào tiếp tục cho qua lớp tích chập kết hợp với hàm kích hoạt là sigmoid.

## **4.3. Mô-đun nhận diện chữ (Text Recognition Module).**

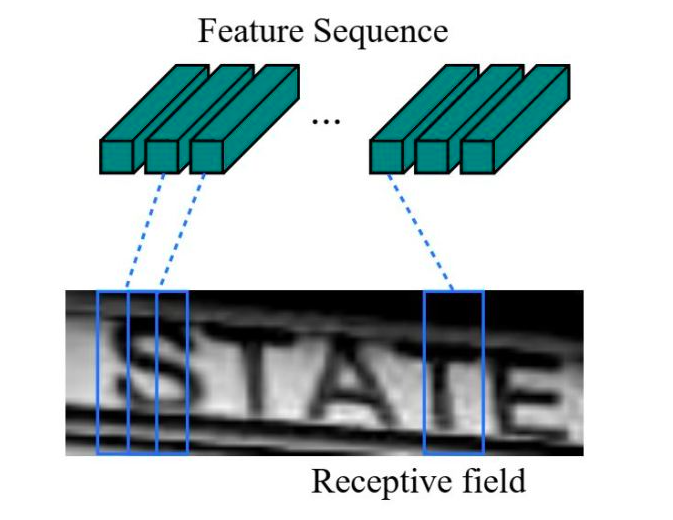
Bài toán nhận diện chữ hay text recognition là một bài toán không mới lạ khi nhắc đến trích xuất thông tin. Trái ngược với text detection, đầu vào của text recognition cũng là ảnh nhưng đầu ra là văn bản hay text. Trải qua một thời gian nghiên cứu và phát triển, bài toán nhận diện chữ cũng rất được các nhà nghiên cứu quan tâm, từ đó phát triển ra các thuật toán khác nhau và đạt hiệu suất cũng như độ chính xác cao trên nhiều dạng chữ khác nhau. Nói chung, về mặt mô hình hầu hết các thuật toán đều sử dụng một mạng tích chập (thường là các mạng cơ bản như vgg, resnet, mobilenet, …) để làm nhiệm vụ trích xuất dữ liệu từ ảnh, và sau đó tùy từng thuật toán mà sẽ sử dụng những mô hình hồi quy, hay transformer, hay các kỹ thuật khác nhau để ánh xạ các đặc trưng và dự đoán văn bản mà ảnh đó thể hiện. Tiếp theo đây chúng ta sẽ đi tìm hiểu các thuật toán làm từ cơ bản đến phức tạp và đạt được độ chính xác cao trên những bộ dữ liệu khác nhau. Ở đây chúng ta sẽ tìm hiểu xoay quanh thuật toán CRNN và SVTR.

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

CRNN là một kiến trúc được thiết chuyên biệt để giải quyết nhiệm vụ text recognition trong bài toán OCR. Xuất hiện trong bài báo “ *An End-to-End Trainable Neural Network for Image-based Sequence Recognition and Its Application to Scene Text Recognition*” xuất bản năm 2015, nhưng cho đến nay, nó vẫn được coi là một trong những mô hình nền tảng mạnh mẽ, tốc độ cao và đạt được hiệu suất mạnh trong việc nhận diễn chữ.

Mạng nơ-ron tích chập hồi quy (hay CRNN) là một kiến trúc học sâu kết hợp sức mạnh của mạng nơ-ron tích chập (CNNs) và mạng nơ-ron hồi quy (RNNs). Các mạng này đặc biệt hiệu quả cho các nhiệm vụ liên quan đến dữ liệu tuần tự có phụ thuộc không gian, chẳng hạn phân tích video, nhận diện giọng nói và nhận dạng chữ viết tay. CRNN bao gồm 3 phần riêng biệt: Lớp tích chập (CNN layers), lớp hồi quy (RNN layers), lớp hậu xử lý (Transcription layer).

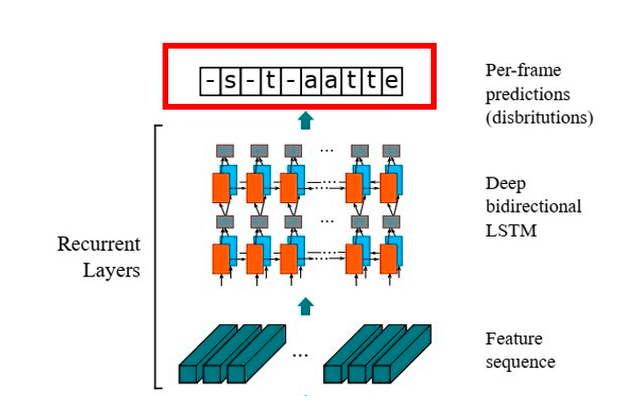
**Lớp tích chập (Convolutional Layers):**

Chức năng chính của lớp này là giúp trích xuất các đặc trưng không gian từ dữ liệu đầu vào. Ảnh đầu vào được cho đi qua các lớp tích chập, được tổng hợp rồi sinh ra các lớp đặc trưng (feature maps). Các feature maps sau đó lại được chia thành các chuỗi của các Feature Vectors (chính là đầu vào của các timestep cho lớp hồi quy) gọi là feature sequence.

*Hình 4.7: Minh họa cho lớp tích chập CNNs.*

**Lớp hồi quy (Recurrent Layers)**

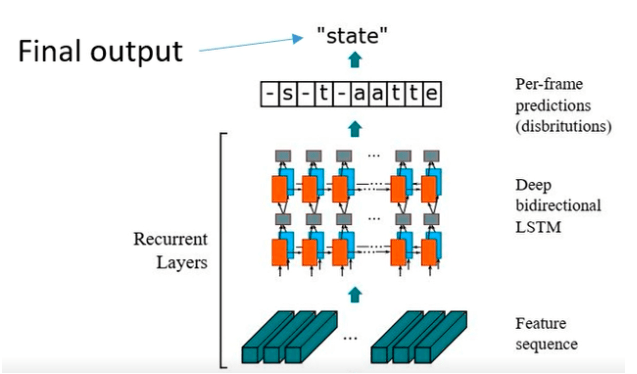
Feature Sequence được tổng hợp và chia ở lớp trước sẽ được đưa vào các mô hình hồi quy như RNN, LSTM, Bidirectional LSTM, … sinh ra một chuỗi các ký tự (Seq2Seq), mà mỗi một ký tự sẽ tương ứng với một timestep trong feature sequence. Về lý thuyết thì đây chính là văn bản đầu ra cần xác định. Tuy nhiên, Feature maps không phải lúc nào cũng chia chính xác thành các Feature Vectors, mà mỗi Feature Vector chứa đúng một ký tự cần nhận diện nên đầu ra của LSTM cũng rất lộn xộn, trùng lặp, ký tự trống, blank, … Từ đó chúng ta phải cần thêm một lớp hậu xử lý các ký tự này hay còn gọi là lớp Transcription Layers.



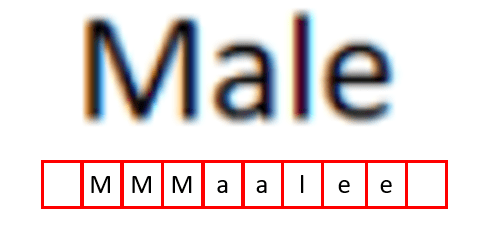
*Hình 4.8: Hình mô tả hoạt động của lớp hồi quy RNN.*

**Lớp hậu xử lý (Transcription Layers)**

Lớp Transcription là lớp xử lý cuối cùng của mô hình text recognition, xử lý đầu ra của lớp hồi quy trước đó, nhiệm vụ chính sắp xếp các ký tự, loại bỏ các lỗi tồn tại (alignment) để đưa ra kết quả cuối cùng.

Thường thì sẽ có một số hướng tiếp cận của Transcription Layers như sau :

Hình 4.9: Hình minh hoạ cho lớp Transcription.

* Cách 1: Quy định mỗi ký tự tương ứng với một số cố định time steps. Ví dụ ta có 10 time steps, tương ứng với Out là “State” thì ta sẽ sắp xếp các timestep như sau: “S, S, T, T, A, A, T, T, E, E”. Cách này đơn giản nhưng chỉ đúng khi kích thước, kiểu chữ của văn bản không đổi.
* Cách 2: Đánh nhãn cho mỗi timestep như hình bên dưới, rồi huấn luyện model sử dụng dữ liệu này. Tuy nhiên lại tốn rất nhiều thời gian để chuẩn bị dữ liệu và không đạt được hiệu quả cao.

Hình 4.10: Hình mô tả transcription layers.

* Cách 3: Sử dụng Connectionist Temporal Classification (CTC) như đã trình bày ở chương III, hàm mất mát CTC.

### **4.3.2. Scene Text Recognition With a Single Visual Model (SVTR).**

Điểm qua một chút bài toán phân dạng chữ thì bài toán có thể chia làm 2 dạng chính: nhận dạng chữ thông thường (regular text recognition) và nhận dạng chữ có biến dạng (irregular text recognition).

**Regular text recognition:**

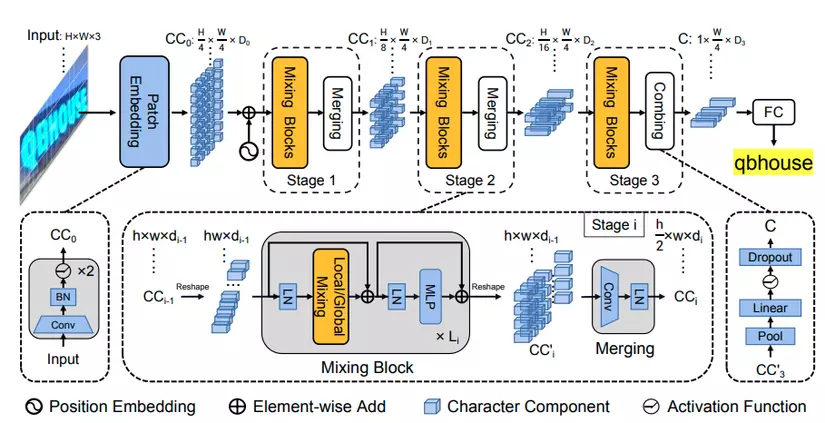
Như đã giới thiệu trước, phương pháp nhận dạng chữ thông thường bằng cách nhận dạng từng ký tự bằng các mô hình hồi quy như CRNN, hay sử dụng các dạng mô hình Attention OCR, Transformer OCR, … Mô hình sử dụng kiến trúc RNN hoặc Transformer kết hợp với CNN để biểu diễn mối quan hệ giữa các ký tự. Các mô hình coi ảnh đầu vào và chữ đầu ra là hai chuỗi và xử lý như bài toán sequence-to-sequence đã đạt được độ chính xác cao trên nhiều bộ dữ liệu. Tuy nhiên vẫn còn xuất hiện các nhược điểm, như RNN có nhược điểm tính toán tuần tự nên thời gian huấn luyện sẽ bị đẩy lên rất lâu và học phụ thuộc xa đối với những câu dài kém hơn. Trong khi đó mô hình Transformer có khả năng học những phụ thuộc xa tốt hơn nhưng việc tính toán trên các thiết bị cpu chậm hơn đáng kể.

**Irregular text recognition:**

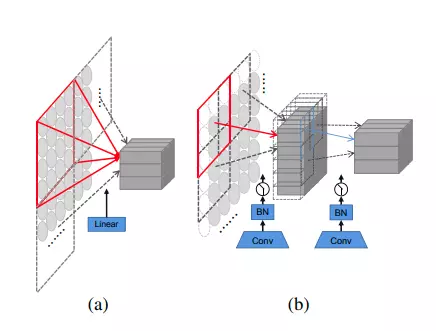
Về chữ biến dạng, có nhiều các biến thể như chữ nghiêng, chữ cong hoặc bị mờ, thậm chí bị méo do nhiều yếu tố. Để giải quyết vấn đề này, cũng có nhiều phương pháp ra đời ví dụ như bài báo “*Robust scene text recognition with automatic rectification*” có đề xuất sử dụng một mạng học khả năng xoay ảnh đầu vào là Spatial Transformer Network hoặc là mô hình Self-Attention Text Recognition Network nhưng vẫn chưa đạt được sự đột phá quá lớn.

**Tổng quan về cấu trúc mạng SVTR:**

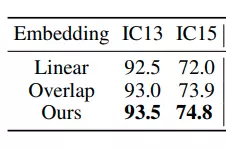
Như đã trình bày ở đầu mô đun, các phương pháp phổ biến cho mô hình text recognition bao gồm 2 phần. Một phần là kiến trúc CNN đóng vai trò như một bộ trích xuất đặc trưng từ ảnh đầu vào (feature extraction) và phần 2 là kiến trúc chuyển bản đồ đặc trưng thành dạng chuỗi. Phần 2 này các mô hình thường sử dụng các lớp RNN hoặc Transformer. Tuy nhiên cả 2 phần CNN, RNN hoặc Transformer thì đều có những ưu điểm, nhược điểm nhất định. CNN thì biểu diễn tốt cho mối liên hệ địa phương (local correlation) còn Transformer lại có thể biểu diễn tốt cho những mối liên hệ toàn cục. SVTR NET ra đời thừa hưởng ưu điểm của cả 2 mô hình trên.

Kiến trúc SVTR nhận đầu vào là ảnh sau đó cho qua một lớp Patch Embedding (được lấy ý tưởng từ Patch embedding của mô hình Vision Transformer) để chia nhỏ ảnh đầu vào thành dạng chuỗi. Việc chia nhỏ ảnh thành các phần tử nhỏ hơn giúp chúng ta dễ dàng xử lý ở dạng chuỗi như các thuật toán trong bài toán xử lý ngôn ngữ thông thường. Các phần tử này sau đó được cho qua 3 tầng. Mỗi tầng gồm 2 mô đun chính là Mixing Blocks và Merging. Hai mô đun này đóng vai trò quan trọng tạo nên hiệu quả của mô hình. Các tầng được thiết kế để trích xuất thông tin đặc trưng trên nhiều scale khác nhau, ngăn ngừa hiện tượng biểu diễn thông tin thừa qua các tầng. Tiếp theo hãy đi sâu hơn vào chi tiết từng kiến trúc và ý nghĩa của từng mô đun.

Hình 4.11: Hình mô tả kiến trúc của SVTR NET.

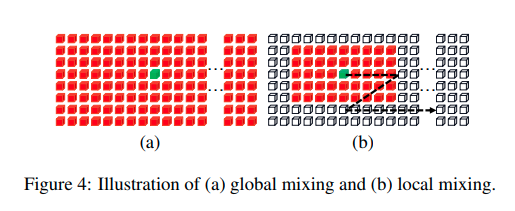
**Patch Embedding:** 

Hình 4.12: Mô tả kiến trúc của Patch Embedding. (a) là phép chiếu tuyến tính trong mô hình vision transformer. (b) là cấu trúc patch embedding chồng chéo của mô hình SVTR

Mô đun Patch Embedding nhận đầu vào là ma trận ảnh có kích thước HxWx3 và chuyển thành phần tử có chiều sâu là . Mỗi phần tử đóng vai trò như các ký tự trong các bài toán xử lý ngôn ngữ. Kiến trúc Patch Embedding bao gồm 2 lớp convolution có kernel 3x3, stride 2 theo sau đó là lớp Batch Normalization. Theo nghiên cứu tác giả, các chiến thuật Patch Embedding khác nhau sẽ ảnh hưởng đến hiệu quả của mô hình.

Hình 4.13: Thông số khi sử dụng Path Embedding

Theo như bài báo, Patch Embedding được đề xuất trong mô hình giúp tăng hiệu suất lên 0.78% và 2.8% so với các phương pháp còn lại.

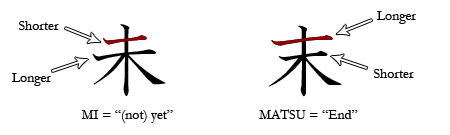
**Mixing Blocks:**

*Hình 4.14: Mô tả cấu trúc của Mixing Blocks*

Mixing Blocks gồm có hai loại là :

* Local mixing block
* Global mixing block.

Động lực của hai mô đun này xuất phát từ ý tưởng: một mô hình nhận diện chữ tốt ngoài biểu diễn tốt được mối liên hệ giwuax các chữ tức là thông tin toàn cục như phương pháp CRNN kết hợp với attention đang làm thì còn phải biểu diễn tốt tương quan giữa các chi tiết trong cùng một ký tự. Khác nhau giữa các nét chấm, phẩy cũng tạo nên sự khác nhau giữa chữ cái. Ví dụ chữ o, ơ, ô, ố, …

Global mixing : dùng để biểu diễn quan hệ giữa các phần tử non-text và text qua đó biểu diễn phụ thuộc xa giữa các ký tự với nhau. Biểu diễn phụ thuộc xa được tác giả sử dụng kiến trúc self-attention kết hợp với một số lớp như LayerNorm và MLP. 

Hình 4.15: Hình mô tả Global mixing, Local mixing.

Local mixing : dùng để biểu diễn mối quan hệ giữa các nét trong cùng một ký tự. Điều này đặc biệt ý nghĩa đối với các chữ có ký tự phức tạp như tiếng trung, tiếng nhật, … Như ảnh mô tả ở trên, chỉ với một chút khác nhau về độ dài ngắn giữa các nét cũng đủ để tạo ra một chữ khác. Local mixing cũng sử dụng window slide trượt trên các vùng kích thước 7x11 và tính toán mối liên hệ giữa các phần tử trong vùng cửa sổ đó. Các phần tử ở đây chính là các phần tử được chia qua lớp Patch Embedding.

Thông qua global và local mixing, thì kiến trúc mô hình có thể biểu diễn tốt sự khác biệt giữa các chữ đến từng chi tiết nhỏ và sẽ giúp cho quá trình phân biệt bằng ctc loss trở nên tốt hơn sau này.

**Mergin và Combining:**

Kiến trúc Merging này có chức năng trích xuất đặc trưng trên nhiều các tỷ lệ khác nhau loại bỏ hiện tượng biểu diễn thừa thông tin. Để thực hiện điều này, sau mỗi lớp Mixing Blocks, tác giả sử dụng một lớp Convolution có kích thước kernel 3x3, bước nhảy 2 theo chiều cao và 1 theo chiều rộng. Như vậy với một đầu có kích thước sẽ cho ra đầu ra có kích thước . Chiều cao sẽ được giảm đi một nửa tuy nhiên chiều rộng bản đồ đặc trưng sẽ được giữa nguyên giúp giảm chi phí tính toán và các lớp ở tầng khác nhau không biểu diễn cùng một thông tin. Điều này có ý nghĩa vì các ảnh cho bài toán nhận dạng chữ có kích thước chiều rộng lớn hơn chiều cao rất nhiều.

Kiến trúc Combining được sử dụng ở tầng cuối cùng của mô hình thay thế cho kiến trúc merging đưa kích thước chiều cao về 1 bằng lớp pooling. Theo sau đó là lớp fully connected và activation. Việc sử dụng lớp Combining ở cuối thay vì Merging giúp tránh việc sử dụng lớp tích chập đối với các ma trận đặc trưng quá nhỏ gây mất đặc trưng ban đầu.

## **4.4. Mô-đun hậu xử lý (Post processing module)**

Ở module cuối này, tùy vào công nghệ và mục đích mà hệ thống sử dụng lại có nhiều cách xử lý khác nhau. Ở đề tài này, chúng ta sẽ tập chung triển khai mô đun hậu xử lý như một rule-based dựa trên thông tin của hai module trước đó là text detection và text recognition có được vị trí và giá trị của text.

Từ các thông tin trên, ta hoàn toàn có thể xây dựng lại được các cấu trúc của ảnh đầu vào, từ đó xây dựng các rule based như : tiền tố, hậu tố, sử dụng regex, tọa độ, mức độ ưu tiên, xuống dòng, phân dòng, dòng mới, mức độ sai sót, trích xuất dữ liệu xuất hiện ở nhiều nơi.

Ở đây chúng ta sẽ đặt ra nhiều khái niệm để kiểm soát và lấy ra giá trị một cách chính xác nhất có thể. Ví dụ tiền tố đây là một tập hợp các giá trị thường xuất hiện trước trường dữ liệu mà chúng ta cần lấy nó ra, hậu tố cũng tương tự nhưng mà khác ở chỗ là xuất hiện ở ngay phía sau trường dữ liệu. Từ đó chúng ta có thể lấy được giá trị mà chúng ta cần. Ví trụ ở trong tài liệu về ielts thì điểm reading thì tiền tố sẽ là “reading” sau đó là giá trị chúng ta cần lấy, hậu tố sẽ là giá trị mà hay xuất hiện phía sau của điểm, thường sẽ là “writing”. Ngoài ra để tăng mức độ chính xác ở module này còn được sử dụng thêm nhiều các rule based khác, ví dụ như rule tọa độ, hoặc sử dụng regex để xác định xem là số hay là ký tự, là ngày tháng hay là giá trị đặc biệt, … Còn có sử dụng mức độ ưu tiên cho các rule based đối với từng trường dữ liệu cụ thể, mức độ sai sót của giá trị khi mà tìm kiếm tiền tố, hậu tố có thể được cho phép.

Tổng quan, ở module này, chúng ta đang cố gắng để xây dựng một hệ thống bao gồm rất nhiều rule based khác nhau, phục vụ cho từng trường hợp cụ thể, xử lý các ngoại lệ có thể xảy ra, … Ngoài ra cố gắng tăng mức linh hoạt nhiều nhất cho các trường dữ liệu cần lấy trong hình ảnh cũng như các loại văn bản khác nhau, có cấu trúc gần giống nhau hoặc khác nhau hoàn toàn. Chúng ta sẽ hiểu rõ hơn về các quy tắc này khi đến phần xây dựng hệ thống ở chương sau.

# **CHƯƠNG 5: THỰC NGHIỆM**

## **5.1. Môi trường Anaconda.**

Anaconda là một môi trường phát triển và quản lý gói phần mềm mã nguồn mở, được sử dụng rộng rãi trong việc phân tích dữ liệu và phát triển các ứng dụng khoa học dữ liệu. Nó cung cấp một bộ công cụ linh hoạt và mạnh mẽ để làm việc với Python và các ngôn ngữ khác như R và Julia.

Dưới đây là một số đặc điểm chính của môi trường Anaconda:

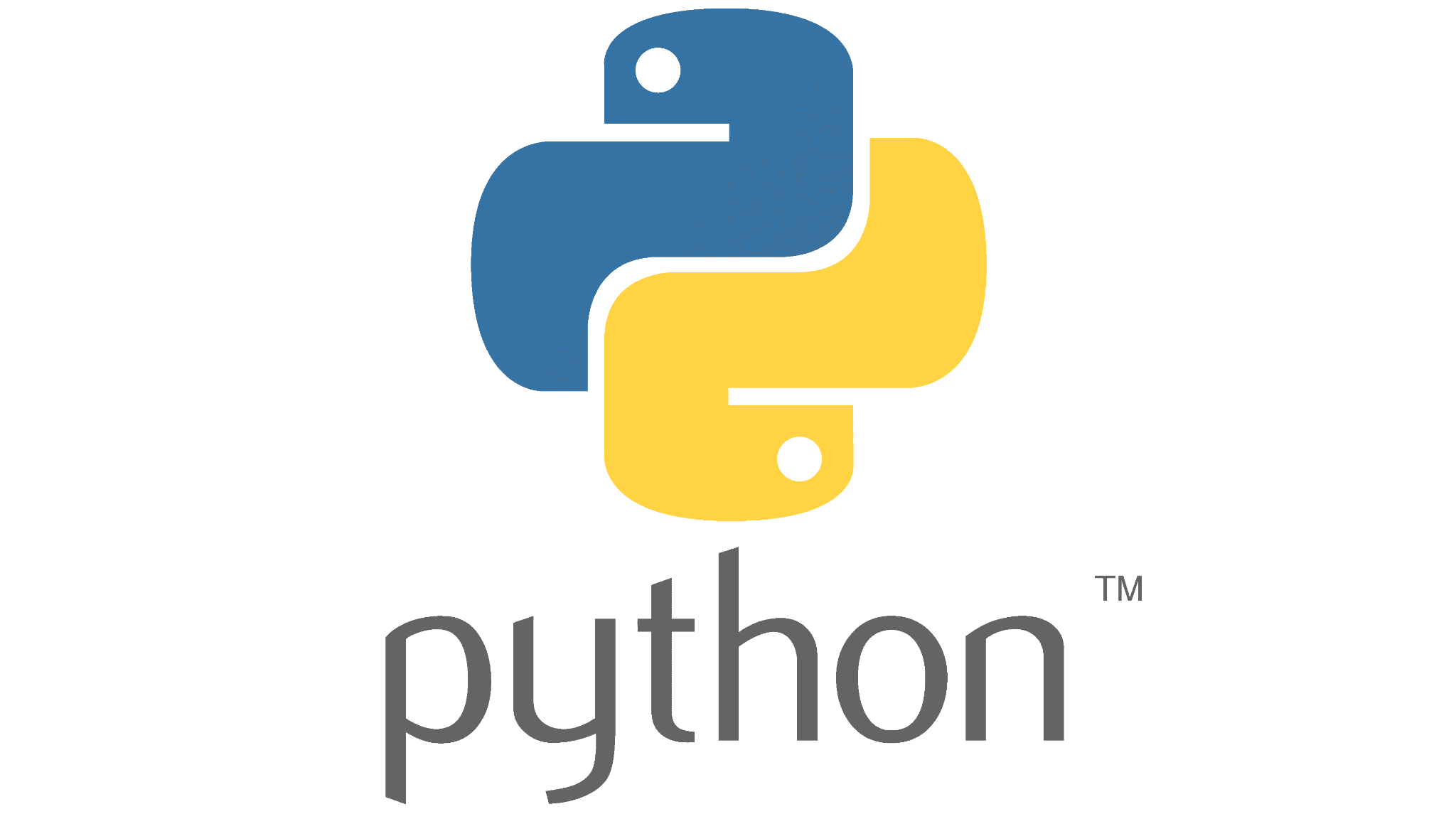
Hình 5.1: Logo của Anaconda

* Quản lý gói phần mềm: Anaconda cung cấp một hệ thống quản lý gói phần mềm mạnh mẽ cho python và các ngôn ngữ khác. Người dùng có thể dễ dàng cài đặt, cập nhật và xóa bỏ các gói phần mềm từ một kho lưu trữ trực tuyến.
* Môi trường ảo: Anaconda cho phép bạn tạo và quản lý các môi trường ảo, giúp tách biệt các phiên bản và phụ thuộc của gói phần mềm. Điều này cho phép bạn cài đặt các phiên bản khác nhau của các gói và thử nghiệm ứng dụng mà không ảnh hưởng đến môi trường hệ thống chính.
* Công cụ tích hợp: Anaconda đi kèm với một số công cụ tích hợp hữu ích như Jupyter Notebook, JupyterLab và Spyder. Điều này cho phép bạn làm việc trực tiếp trong môi trường phát triển tích hợp, cung cấp khả năng tương tác và khám phá dữ liệu thuận tiện.
* Hỗ trợ đa nền tảng: Anaconda hỗ trợ đa nền tảng, cho phép bạn sử dụng môi trường và các công cụ của nó trên các hệ điều hành phổ biến như Windows, macOS và Linux.

Conda giúp chúng ta quản lý thư viện một cách dễ dàng hơn, kiểm tra các phiên bản sao cho phù hợp và có thể tách mỗi project là một môi trường hoàn toàn độc lập với nhau.

## **5.2. Ngôn ngữ Python.**

Python là một ngôn ngữ lập trình thông dịch, được phát triển bởi Guido van Rossum và ra mắt lần đầu vào năm 1991. Nó có cú pháp đơn giản, dễ đọc và dễ hiểu, làm cho việc viết mã Python trở nên dễ dàng cho cả người mới học lập trình và những người có kinh nghiệm.

 Dưới đây là một vài đặc biệt và tính năng chính của Python:

Hình 5.2: Logo của Python

* Đa mục đích: Python là một ngôn ngữ đa mục đích, có thể được sử dụng để phát triển ứng dụng web, ứng dụng di động, phân tích dữ liệu, máy học (machine learning), trí tuệ nhân tạo (artificial intelligence), và nhiều lĩnh vực lập trình khác.
* Cú pháp đơn giản: Python có cú pháp rõ ràng và đơn giản, với việc sử dụng khoảng trắng để định dạng mã nguồn. Điều này làm cho mã Python dễ đọc và dễ hiểu, giúp tăng tính tương tác và sáng tạo của các nhà lập trình.
* Hỗ trợ thư viện phong phú: Python đi kèm với một số thư viện và module chuẩn mạnh mẽ, cho phép bạn thực hiện nhiều tác vụ thông qua các hàm sẵn có mà không cần phải viết lại từ đầu. Ngoài ra, Python còn có cộng đồng lớn và năng động, sản sinh ra nhiều thư viện bên thứ ba phong phú và mạnh mẽ, từ đó hỗ trợ cho nhiều lĩnh vực và ứng dụng khác nhau.
* Hỗ trợ đa nền tảng: Python là ngôn ngữ đa nền tảng, có thể chạy trên các hệ điều hành phổ biến như Windows, macOS và Linux. Điều này cho phép bạn phát triển ứng dụng Python một cách linh hoạt trên nhiều môi trường.
* Mạnh trong phân tích dữ liệu và máy học: Python là một trong những ngôn ngữ phổ biến nhất trong lĩnh vực phân tích dữ liệu và máy học. Có các thư viện mạnh mẽ như NumPy, Pandas, Matplotlib, scikit-learn và TensorFlow hỗ trợ việc xử lý dữ liệu, phân tích, và xây dựng mô hình máy học.
* Học và sử dụng dễ dàng: Python là một ngôn ngữ lập trình rất dễ học và sử dụng. Cú pháp đơn giản và rõ ràng giúp người mới học nắm bắt nhanh chóng, và cộng đồng Python rất hướng dẫn và hỗ trợ. Ngoài ra, Python cũng có tài liệu phong phú và rất nhiều nguồn tài nguyên trực tuyến, giúp bạn giải quyết vấn đề và tìm kiếm thông tin dễ dàng.
* Tích hợp và mở rộng: Python có khả năng tích hợp với các ngôn ngữ lập trình khác như C/C++, Java, và .NET. Bạn có thể sử dụng Python để viết các phần mở rộng cho ứng dụng sử dụng các ngôn ngữ khác hoặc sử dụng các thư viện có sẵn để tận dụng các tính năng mạnh mẽ từ các ngôn ngữ khác.
* Độ tin cậy và ổn định: Python là một ngôn ngữ ổn định và đáng tin cậy. Nó có cộng đồng lớn và sự hỗ trợ liên tục, giúp người dùng giải quyết các vấn đề và tìm kiếm giải pháp khi gặp lỗi.

Tổng quan, Python là một ngôn ngữ lập trình phổ biến và mạnh mẽ, với cú pháp đơn giản, hỗ trợ thư viện phong phú, và rất dễ học và sử dụng. Nó đã trở thành một công cụ quan trọng trong nhiều lĩnh vực, từ phân tích dữ liệu đến phát triển web và trí tuệ nhân tạo.

## **5.3. Các thư viện chính sử dụng trong bài toán.**

### **5.3.1. Thư viện Pytorch.**



Hình 5.3: Logo của Pytorch

PyTorch là một thư viện mã nguồn mở cho việc xây dựng và huấn luyện mô hình học sâu. Nó được phát triển bởi Facebook AI Research và hỗ trợ các tính năng quan trọng để triển khai các mô hình học máy và trí tuệ nhân tạo. Dưới đây là một tổng quan về các tính năng chính của PyTorch:

* Dynamic Computation Graph: PyTorch sử dụng đồ thị tính toán động, cho phép người dùng định nghĩa và thay đổi các phép tính trong quá trình chạy mà không cần xây dựng lại đồ thị từ đầu. Điều này tạo điều kiện thuận lợi cho việc debug và tạo ra các mô hình phức tạp.
* Tích hợp với Python: PyTorch được thiết kế để tương tác mượt mà với Python và các thư viện phổ biến khác như NumPy. Điều này giúp người dùng dễ dàng sử dụng và tích hợp PyTorch vào quy trình làm việc hiện có của mình.
* Cung cấp các lớp và hàm cơ bản: PyTorch cung cấp một loạt các lớp và hàm cơ bản để xây dựng mô hình học sâu, bao gồm các lớp và hàm cho việc tạo mạng nơ-ron, hàm kích hoạt, lớp tích chập, lớp pooling, và nhiều hơn nữa. Điều này giúp tiết kiệm thời gian và công sức trong việc triển khai mô hình.
* Hỗ trợ GPU: PyTorch hỗ trợ tính toán trên GPU, cho phép tăng tốc độ huấn luyện và triển khai mô hình lên GPU. Người dùng có thể dễ dàng di chuyển dữ liệu và mô hình giữa CPU và GPU sử dụng PyTorch.

PyTorch cung cấp một giao diện trực quan và dễ sử dụng cho việc xây dựng và huấn luyện mô hình học sâu. Nó đã trở thành một trong những thư viện phổ biến và được ưa chuộng trong cộng đồng nghiên cứu và ứng dụng học sâu. Điểm mạnh của PyTorch bao gồm:

* Tính linh hoạt: PyTorch cho phép người dùng tạo và tùy chỉnh các mô hình học sâu theo ý muốn. Người dùng có thể dễ dàng xác định các lớp, hàm kích hoạt, hàm mất mát và tối ưu hóa, và thậm chí tạo ra các kiến trúc mô hình phức tạp.
* Hỗ trợ tính toán tự động (Autograd): PyTorch tự động tính đạo hàm cho các phép tính trong mô hình, giúp người dùng dễ dàng thực hiện lan truyền ngược và cập nhật trọng số mô hình. Điều này giúp rút ngắn quá trình triển khai mô hình và tối ưu hóa.
* Giao diện dễ sử dụng: PyTorch có cú pháp đơn giản và trực quan, giúp người dùng dễ dàng hiểu và sử dụng các hàm và lớp của thư viện. Nó cũng cung cấp nhiều hàm tiện ích và công cụ hỗ trợ cho việc xử lý dữ liệu và triển khai mô hình.
* Tích hợp với các thư viện khác: PyTorch được tích hợp tốt với các thư viện phổ biến khác trong hệ sinh thái Python như NumPy và SciPy. Điều này giúp người dùng dễ dàng làm việc với dữ liệu số, thực hiện các phép tính mảng và tận dụng các công cụ phân tích dữ liệu mạnh mẽ.
* Hỗ trợ đa nền tảng: PyTorch có thể hoạt động trên nhiều nền tảng khác nhau, bao gồm Linux, Windows và macOS. Nó cũng hỗ trợ các kiến trúc máy tính khác nhau như CPU và GPU.

Tổng quan, PyTorch là một thư viện mạnh mẽ và linh hoạt cho việc xây dựng và huấn luyện mô hình học sâu. Nó đã thu hút sự quan tâm và đóng góp của một cộng đồng lớn, và tiếp tục phát triển và cải tiến để đáp ứng các yêu cầu của người dùng.

### **5.3.2. Thư viện Transformer.**

Thư viện Transformer là một thư viện Python được phát triển để triển khai và sử dụng các mô hình Transformer trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) và học sâu. Nó cung cấp các công cụ và chức năng để xây dựng, huấn luyện và sử dụng các mô hình Transformer phổ biến như BERT, GPT và Transformer Encoder/Decoder.



Hình 5.4: Logo của Transformer.

\

Dưới đây là một số tính năng chính của thư viện Transformer:

* Xây dựng mô hình Transformer: Thư viện Transformer cung cấp các lớp và công cụ để xây dựng các kiến trúc mô hình Transformer, bao gồm Transformer Encoder và Transformer Decoder. Bạn có thể tùy chỉnh số lớp, số đầu ra, kích thước embedding và các siêu tham số khác của mô hình.
* Tích hợp với các kiến trúc mô hình phổ biến: Thư viện hỗ trợ các kiến trúc mô hình phổ biến dựa trên Transformer như BERT, GPT và Transformer Encoder/Decoder. Bạn có thể sử dụng các kiến trúc này để thực hiện các tác vụ như phân loại văn bản, sinh văn bản và dịch máy.
* Hỗ trợ huấn luyện mô hình: Thư viện cung cấp các công cụ để huấn luyện mô hình Transformer trên các tập dữ liệu. Bạn có thể tùy chỉnh hàm mất mát, thuật toán tối ưu và các tham số huấn luyện khác. Nó cũng hỗ trợ huấn luyện mô hình trên nhiều GPU để tăng tốc quá trình huấn luyện.
* Tích hợp với PyTorch và TensorFlow: Thư viện Transformer được thiết kế để tích hợp tốt với các thư viện học sâu phổ biến như PyTorch và TensorFlow. Bạn có thể sử dụng Transformer cùng với các tính năng và công cụ của các thư viện này, bao gồm tích hợp với các lớp mô hình khác, tối ưu hóa và tri
* Tích hợp với các tác vụ NLP: Thư viện Transformer cung cấp các chức năng tiện ích để thực hiện các tác vụ NLP thông qua các mô hình Transformer. Bạn có thể sử dụng nó để thực hiện phân loại văn bản, phân tích cảm xúc, trích xuất thông tin, dịch máy, tự động tạo tiêu đề và nhiều tác vụ NLP khác.
* Tích hợp với mô hình ngôn ngữ tiền huấn luyện: Thư viện Transformer hỗ trợ tích hợp với các mô hình ngôn ngữ tiền huấn luyện như GPT và BERT. Bạn có thể tải xuống và sử dụng các mô hình này trong các tác vụ xử lý ngôn ngữ tự nhiên và học sâu.
* Tài liệu và ví dụ mã mẫu: Thư viện Transformer cung cấp tài liệu và ví dụ mã mẫu để giúp bạn bắt đầu sử dụng và hiểu rõ hơn về cách sử dụng các mô hình Transformer trong các tác vụ NLP. Có các hướng dẫn chi tiết, ví dụ mã và tài liệu tham khảo để bạn học tập và triển khai các ứng dụng của riêng mình.
* Cộng đồng và hỗ trợ: Thư viện Transformer được hỗ trợ bởi một cộng đồng lớn và sôi nổi, trong đó bạn có thể nhận được sự giúp đỡ, chia sẻ kiến thức và thảo luận với những người khác sử dụng thư viện này. Có các diễn đàn, nhóm thảo luận và nhiều nguồn tài nguyên trực tuyến khác để bạn có thể tương tác và học tập từ cộng đồng này.

Tổng quan về thư viện Transformer chỉ ra rằng nó là một công cụ mạnh mẽ để triển khai và sử dụng các mô hình Transformer trong lĩnh vực xử lý ngôn ngữ tự nhiên và học sâu. Bằng cách sử dụng thư viện này, bạn có thể xây dựng các mô hình mạnh mẽ và hiệu quả cho các tác vụ NLP và khám phá nhiều ứng dụng mới trong lĩnh vực này.

### **5.3.3. Thư viện PaddlePaddle.**

PaddlePaddle (PArallel Distributed Deep Learning) là một nền tảng học sâu mã nguồn mở được phát hành bởi Baidu. PaddlePaddle được thiết kế để hỗ trợ các tác vụ học sâu từ đơn giản đến phức tạp, với mục tiêu giúp các nhà phát triển nghiên cứu dễ dàng xây dựng, huấn luyện và triển khai các mô hình học sâu hiệu quả

Dưới đây là một số tính năng mạnh mẽ được paddlepaddle hỗ trợ:

Hình 5.5: Logo của PaddlePaddle

* PaddlePaddle được thiết kế để hỗ trợ việc huấn luyện mô hình phân tán trên nhiều GPU và nhiều máy, giúp tăng tốc quá trình huấn luyện và khả năng mở rộng của mô hình. Điều này rất hữu ích khi làm việc với các mô hình lớn và dữ liệu khổng lồ.
* Hỗ trợ nhiều loại mô hình : Paddle hỗ trợ nhiều loại mô hình học sâu khác nhau, bao gồm : Mạng CNN (thường được sử dụng cho các tác vụ liên quan đến hình ảnh), Mạng RNN (thích hợp cho các tác vụ liên quan đến chuỗi thời gian hoặc dữ liệu tuần tự), Mạng GAN (thường dùng cho việc tạo dữ liệu mới dựa trên dữ liệu hiện có), Mạng Transformer (thường được sử dụng cho các tác vụ xử lý ngôn ngữ tự nhiên), …
* API dễ dàng sử dụng: PaddlePaddle cung cấp API dễ sử dụng và linh hoạt, cho phép các nhà phát triển dễ dàng xây dựng và huấn luyện các mô hình học sâu. API của Paddle thân thiện với người dùng, hỗ trợ cả các lập trình viên mới bắt đầu lẫn các chuyên gia.
* Hỗ trợ triển khai mô hình: PaddlePaddle không chỉ hỗ trợ việc xây dựng và huấn luyện mô hình mà còn cung cấp các công cụ mạnh mẽ để triển khai mô hình trong môi trường sản xuất. Paddle Paddle hỗ trợ triển khai trên nhiều nền tảng khác nhau, bao gồm cả các thiết bị di động và nhúng.
* Thư viện phong phú và tài liệu đầy đủ : PaddlePaddle đi kèm với một thư viện phong phú các mô hình học sâu đã được huấn luyện sẵn và các bộ công cụ để đánh giá và tinh chỉnh mô hình. Bên cạnh đó, tài liệu của PaddlePaddle rất chi tiết và đầy đủ, giúp người dùng dễ dàng tiếp cận và sử dụng nền tảng này.
* Cộng đồng và hỗ trợ mạnh mẽ : Là một nền tảng mã nguồn mở, PaddlePaddle có một cộng đồng người dùng và nhà phát triển rộng lớn. Baidu và cộng đồng thường xuyên cập nhật và cải tiến nền tảng, cung cấp hỗ trợ thông qua các diễn đàn, tài liệu, và ví dụ mẫu.

Tổng quan, PaddlePaddle là một nền tảng học sâu mạnh mẽ và linh hoạt, cung cấp đầy đủ các công cụ cần thiết cho việc xây dựng, huấn luyện, và triển khai các mô hình học sâu. Với khả năng phân tán mạnh mẽ, hỗ trợ nhiều loại mô hình, API dễ sử dụng, và cộng đồng hỗ trợ rộng lớn, PaddlePaddle là một lựa chọn lý tưởng cho các nhà nghiên cứu và phát triển trong lĩnh vực trí tuệ nhân tạo và học máy.

### **5.3.4. Thư viện PaddleOCR.**

PaddleOCR là một dự án mã nguồn mở trong hệ sinh thái PaddlePaddle, được thiết kế để cung cấp các công cụ và mô hình mạnh mẽ cho các tác vụ nhận dạng ký tự quang học (OCR - Optical Character Recognition). PaddleOCR hỗ trợ nhận dạng văn bản từ hình ảnh trong nhiều ngôn ngữ khác nhau, bao gồm tiếng Anh, tiếng Trung, và nhiều ngôn ngữ khác. Dự án này không chỉ cung cấp các mô hình đã được huấn luyện sẵn mà còn các công cụ để tinh chỉnh và triển khai các mô hình OCR cho các ứng dụng thực tế.

Dưới đây là một số tính năng mạnh mẽ mà PaddleOCR hỗ trợ

Hình 5.6: Logo của PaddleOCR

* Đa ngôn ngữ: PaddleOCR hỗ trợ nhiều ngôn ngữ khác nhau, bao gồm nhưng không giới hạn ở tiếng Anh, tiếng Trung, tiếng Nhật, tiếng Hàn, và nhiều ngôn ngữ châu Âu. Điều này giúp PaddleOCR trở nên linh hoạt và hữu ích trong nhiều ứng dụng quốc tế.
* Pipeline OCR hoàn chỉnh :
* Text Detection: Phát hiện các vùng văn bản trong hình ảnh.
* Text Recognition: Nhận dạng các ký tự trong các vùng văn bản đã phát hiện
* Post-Processing: Xử lý hậu kỳ để cải thiện độ chính xác của kết quả.
* Hiệu suất cao và chính xác : Các mô hình OCR trong PaddleOCR được thiết kế và huấn luyện để đạt được độ chính xác cao, đồng thời tối ưu hóa hiệu suất để có thể xử lý nhanh chóng các hình ảnh đầu vào. Điều này làm cho PaddleOCR phù hợp với các ứng dụng yêu cầu xử lý thời gian thực.
* Mô hình đã được huấn luyện sẵn : PaddleOCR cung cấp nhiều mô hình đã được huấn luyện sẵn trên các bộ dữ liệu lớn, cho phép người dùng dễ dàng triển khai và sử dụng mà không cần phải huấn luyện lại từ đầu. Điều này giúp tiết kiệm thời gian và công sức, đồng thời đảm bảo kết quả chính xác và ổn định.
* Dễ sử dụng và tùy chỉnh : PaddleOCR cung cấp API và công cụ dễ sử dụng, cho phép người dùng nhanh chóng bắt đầu với các tác vụ OCR. Ngoài ra, người dùng cũng có thể tùy chỉnh và tinh chỉnh các mô hình theo nhu cầu cụ thể của họ.
* Tích hợp mạnh mẽ với PaddlePaddle: Là một phần của hệ sinh thái PaddlePaddle, PaddleOCR tận dụng toàn bộ các tính năng và lợi ích của nền tảng này, bao gồm khả năng huấn luyện phân tán, tối ưu hóa hiệu suất, và hỗ trợ triển khai trên nhiều nền tảng khác nhau.
* Hỗ trợ cộng đồng và tài liệu phong phú PaddleOCR có một cộng đồng người dùng rộng lớn và hoạt động tích cực. Dự án này cũng có tài liệu chi tiết và đầy đủ, bao gồm các hướng dẫn, ví dụ mẫu, và các bài viết chuyên sâu, giúp người dùng dễ dàng tiếp cận và sử dụng.
* Khả năng triển khai dễ dàng: PaddleOCR hỗ trợ triển khai mô hình trên nhiều nền tảng khác nhau, bao gồm các thiết bị nhúng, điện thoại di động, và các dịch vụ đám mây. Điều này giúp người dùng dễ dàng tích hợp OCR vào các ứng dụng thực tế.
* Mã nguồn mở và liên tục cải tiến : PaddleOCR là một dự án mã nguồn mở, liên tục được cập nhật và cải tiến bởi cộng đồng và các nhà phát triển từ Baidu. Điều này đảm bảo rằng các tính năng mới và các cải tiến hiệu suất luôn được thêm vào, giúp dự án ngày càng hoàn thiện.

Tổng quan, PaddleOCR là một giải pháp OCR mạnh mẽ và linh hoạt trong hệ sinh thái PaddlePaddle, cung cấp một pipeline hoàn chỉnh cho các tác vụ nhận dạng văn bản từ hình ảnh. Với hỗ trợ đa ngôn ngữ, hiệu suất cao, các mô hình đã được huấn luyện sẵn, và khả năng tùy chỉnh dễ dàng, PaddleOCR là một lựa chọn lý tưởng cho các nhà phát triển và nghiên cứu trong lĩnh vực nhận dạng ký tự quang học.

### **5.3.5. Thư viện FastAPI.**

FastAPI là một framework web hiện đại, nhanh và hiệu quả để xây dựng các API với Python 3.7+ dựa trên các tiêu chuẩn, như OpenAPI và JSON Schema. Nó được thiết kế để dễ sử dụng và hiệu suất cao, giúp các nhà phát triển tạo ra các ứng dụng web và API một cách nhanh chóng và dễ dàng.

 Dưới đây là một số chức năng và ưu điểm chính của FastAPI.

Hình 5.7: Logo của FastAPI

* Hiệu suất cao : FastAPI được xây dựng trên nền tảng ASGI (Asynchronous Server Gateway Interface), cho phép xử lý các yêu cầu đồng thời và không đồng bộ một cách hiệu quả. Nó sử dụng Starlette làm framework web cơ bản và Pydantic để xác thực dữ liệu và tạo các mô hình dữ liệu.
* Tự động tạo tài liệu API: FastAPI tự động tạo ra tài liệu API theo chuẩn OpenAPI và cung cấp giao diện web để kiểm thử API thông qua Swagger UI và ReDoc. Điều này giúp nhà phát triển dễ dàng kiểm thử và tương tác với API mà không cần phải viết tài liệu thủ công. Khi chạy ứng dụng, bạn có thể truy cập vào /docs để xem tài liệu Swagger UI hoặc /redoc để xem tài liệu ReDoc.
* Hỗ trợ xử lý đồng thời và không đồng bộ: FastAPI hỗ trợ cả synchronous và asynchronous code, cho phép tận dụng tối đa khả năng của các mô hình xử lý hiện đại, từ đó cải thiện hiệu suất và khả năng mở rộng của ứng dụng.
* Tích hợp với Pydantic để xác thực dữ liệu: FastAPI sử dụng Pydantic để xác thực dữ liệu đầu vào, giúp đảm bảo rằng dữ liệu được gửi tới API luôn đúng định dạng và hợp lệ. Điều này giúp giảm thiểu lỗi và tăng tính bảo mật cho ứng dụng.
* Hỗ trợ Dependency Injection: FastAPI có cơ chế dependency injection mạnh mẽ, cho phép dễ dàng quản lý các phụ thuộc và tái sử dụng các thành phần trong ứng dụng.
* Cộng đồng và tài liệu phong phú: FastAPI có một cộng đồng phát triển mạnh mẽ và tài liệu chi tiết, giúp người dùng dễ dàng tìm kiếm hỗ trợ và tài liệu tham khảo. Các ví dụ và hướng dẫn phong phú cũng giúp người dùng mới nhanh chóng nắm bắt và sử dụng FastAPI một cách hiệu quả.

Tổng quan, FastAPI là một framework web hiện đại và mạnh mẽ cho Python, cung cấp các công cụ và tính năng cần thiết để xây dựng các API hiệu suất cao và dễ sử dụng. Với các tính năng như tự động tạo tài liệu API, hỗ trợ xử lý không đồng bộ, và tích hợp với Pydantic để xác thực dữ liệu, FastAPI là một lựa chọn lý tưởng cho các nhà phát triển muốn xây dựng các ứng dụng web và API phức tạp một cách nhanh chóng và hiệu quả.

## **5.4. Bộ dữ liệu huấn luyên.**

### **5.4.1. Bộ dữ liệu ICDAR2015**

ICDAR2015 là một bộ dữ liệu quan trọng và phổ biến trong lĩnh vực nhận dạng và phát hiện văn bản trong hình ảnh, được phát triển và sử dụng trong hội nghị International Conference on Document Analysis and Recognition (ICDAR) năm 2015. Bộ dữ liệu này thường được sử dụng để đánh giá hiệu suất của các thuật toán phát hiện và nhận dạng văn bản trong các ứng dụng thực tế, chẳng hạn như OCR (Optical Character Recognition) và các hệ thống xử lý văn bản từ hình ảnh.



Hình 5.8: Hình ảnh minh họa ICDAR2015 datasets

Dưới đây là một số thông tin tổng quan từ bộ dữ liệu:

* Kích thước của bộ dữ liệu: ICDAR2015 gồm 2 phần chính. Training set bao gồm 1000 hình ảnh được chú thích với thông tin về vị trí và nội dung đoạn văn bản trong hình. Test set bao gồm 500 hình ảnh với chú thích tương tự như trong tập huấn luyện. Các hình ảnh trong bộ dữ liệu này được chụp trong các điều kiện thực tế, với nhiều loại biến thể khác nhau về độ sáng, độ tương phản, góc chụp và độ phân giải, giúp đảm bảo rằng các thuật toán được huấn luyện và kiểm thử trên bộ dữ liệu này có thể hoạt động tốt trong các điều kiện thực tế đa dạng.
* Phạm vi và tác dụng của bộ dữ liệu: Bộ dữ liệu ICDAR2015 chủ yếu được sử dụng trong các bài toán sau:
* Phát hiện văn bản (Text Detection): Xác định vị trí của các đoạn văn bản trong hình ảnh.
* Nhận dạng văn bản (Text Recognition): Chuyển đổi các đoạn văn bản đã được phát hiện từ ảnh sang dạng ký tự (chữ) số.
* End-To-End Text Recognition: Kết hợp cả hai nhiệm vụ phát hiện và nhận dạng văn bản trong một hệ thống duy nhất.
* Mức độ đa dạng: ICDAR2015 là một trong những bộ dữ liệu tiêu chuẩn và phổ biến nhất trong cộng đồng nghiên cứu về xử lý văn bản. Một số lý do khiến bộ dữ liệu này được sử dụng rộng rãi bao gồm:
* Độ đa dạng và thực tế của dữ liệu: Các hình ảnh trong ICDAR2015 được chụp từ các môi trường thực tế với nhiều biến thể về điều kiện ánh sáng và góc chụp, giúp đảm bảo rằng các thuật toán được kiểm tra và phát triển trên bộ dữ liệu này có thể hoạt động tốt trong các điều kiện khác nhau.
* Sự công nhận từ cộng đồng: Được sử dụng trong các cuộc thi tại hội nghị ICDAR, bộ dữ liệu này đã trở thành một tiêu chuẩn để so sánh hiệu suất của các thuật toán và hệ thống mới.
* Sự hỗ trợ và tài liệu phong phú: Có nhiều nghiên cứu và bài báo đã sử dụng bộ dữ liệu ICDAR2015, cung cấp tài liệu phong phú và các ví dụ thực tế để các nhà nghiên cứu và phát triển tham khảo học hỏi.

Tổng quan, bộ dữ liệu ICDAR2015 đóng vai trò quan trọng trong việc thúc đẩy nghiên cứu và phát triển các công nghệ nhận dạng và phát hiện văn bản trong hình ảnh. Với độ đa dạng và tính thực tế của các hình ảnh, cùng với sự công nhận rộng rãi từ cộng đồng nghiên cứu, ICDAR2015 đã trở thành một bộ dữ liệu không thể thiếu cho các nhà nghiên cứu và phát triển trong lĩnh vực này.

### **5.4.2. Bộ dữ liệu Total-Text Dataset.**

Total-Text là một bộ dữ liệu tiên tiến và được sử dụng rộng rãi trong lĩnh vực phát hiện và nhận dạng văn bản trong hình ảnh. Được công bố bởi nhà nghiên cứu tại trung tâm đa phương tiện và xử lý hình ảnh (CVPR) năm 2017. Total Text đã nhanh chóng trở thành một trong những bộ dữ liệu tiêu chuẩn cho các bài toán liên quan đến văn bản trong hình ảnh nhờ vào sự đa dạng và phức tạp của nó.

Dưới đây là một số thông tin tổng quan của bộ dữ liệu Total-Text:

Hình 5.9: Hình ảnh minh họa cho Toal Text Datasets

* Kích thước của bộ dữ liệu Total-Text: Tổng cộng có 1555 hình ảnh (bộ training set có 1255 ảnh, bộ test set có 300 ảnh), mỗi hình ảnh đều được chú thích cẩn thận với các thông tin về vị trí và nội dung của các đoạn văn bản trong hình. Đặc biệt, các đoạn văn bản trong Total-Text có thể có nhiều hình dạng khác nhau, bao gồm văn bản thẳng, cong và dạng hỗn hợp, giúp tạo ra một thách thức lớn hơn cho các thuật toán phát hiện và nhận dạng văn bản.
* Phạm vi và tác dụng của bộ dữ liệu: Bộ dữ liệu Total-Text chủ yếu được sử dụng trong các bài toán sau:
* Phát hiện văn bản (Text Detection): Xác định vị trí của các đoạn văn bản trong hình ảnh.
* Nhận dạng văn bản (Text Recognition): Chuyển đổi các đoạn văn bản đã được phát hiện từ ảnh sang dạng ký tự (chữ) số.
* End-To-End Text Recognition: Kết hợp cả hai nhiệm vụ phát hiện và nhận dạng văn bản trong một hệ thống duy nhất.
* Sự phổ biến của bộ dữ liệu, Total-Text là một trong những bộ dữ liệu được ưa chuộng nhất trong cộng đồng nghiên cứu về nhận dạng văn bản trong hình ảnh, nhờ vào các đặc điểm sau:
* Sự phức tạp và đa dạng: Bộ dữ liệu này bao gồm các đoạn văn bản với nhiều hình dáng khác nhau, từ thẳng, cong cho đến hỗn hợp, tạo ra một thách thức lớn cho các thuật toán và từ đó giúp cải thiện hiệu suất của các hệ thống phát hiện và nhận dạng chữ.
* Tính thực tế: Các hình ảnh trong Total-Text được thu thập từ môi trường thực tế, đảm bảo rằng các thuật toán huấn luyện trên bộ dữ liệu này có thể hoạt động tốt trong các tình huống thực tế.
* Sự công nhận từ cộng đồng: Total-Text được sử dụng rộng rãi trong các nghiên cứu học thuật và các cuộc thi, giúp nó trở thành một tiêu chuẩn để so sánh hiệu suất của các thuật toán mới.
* Một số nghiên cứu tiêu biểu sử dụng Total-Text Nhiều nghiên cứu và hệ thống nổi bật trong lĩnh vực phát hiện và nhận dạng văn bản đã sử dụng bộ dữ liệu Total-Text để đánh giá hiệu suất, bao gồm: Mask TextSpotter, TextSnake, …

Tổng quan, Total-Text là một bộ dữ liệu quan trọng và không thể thiếu trong nghiên cứu và phát triển các công nghệ liên quan đến phát hiện và nhận dạng văn bản trong hình ảnh. Với sự đa dạng và phức tạp của các đoạn văn bản trong bộ dữ liệu, Total-Text cung cấp một thách thức lớn nhưng cũng là cơ hội để các nhà nghiên cứu phát triển các thuật toán và hệ thống có khả năng hoạt động tốt trong các điều kiện thực tế. Sự công nhận rộng rãi và tính ứng dụng cao của Total-Text trong cộng đồng nghiên cứu đã giúp nó trở thành một bộ dữ liệu tiêu chuẩn trong lĩnh vực này.

### **5.4.3. Bộ dữ liệu CTW1500.**

Bộ dữ liệu CTW1500 (Chinese Text in the Wild) là một bộ dữ liệu nổi bật và được sử dụng rộng rãi trong lĩnh vực phát hiện và nhận dạng văn bản trong hình ảnh. Được giới thiệu vào năm 2017, CTW1500 đã trở thành một tiêu chuẩn trong nghiên cứu và phát triển các thuật toán phát hiện và nhận dạng văn bản, đặc biệt là các văn bản có hình dạng cong và phức tạp.

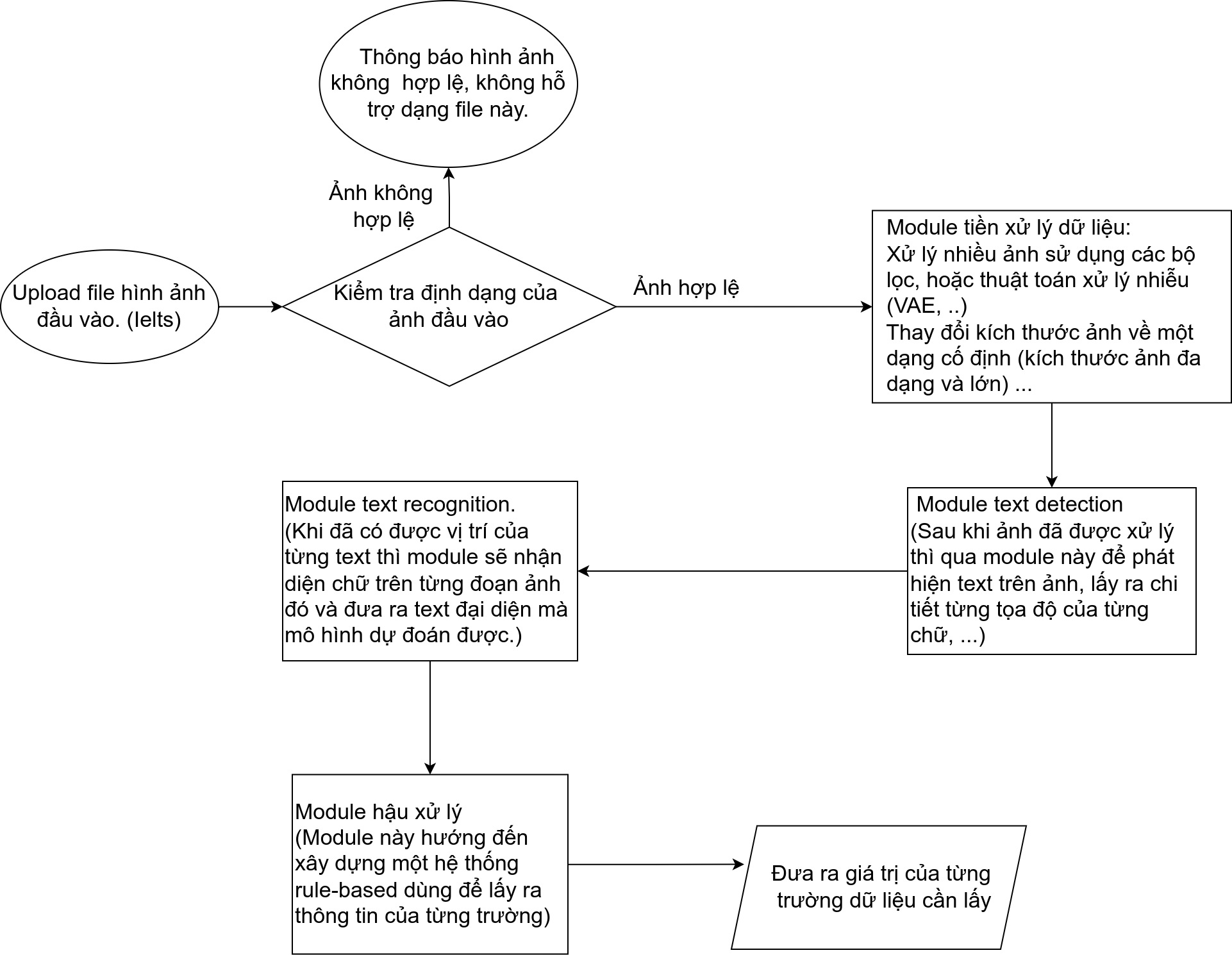
Hình 5.10: Hình ảnh minh họa cho CTW1500 Datasets

Dưới đây là một số thông tin tổng quan về bộ dữ liệu CTW1500:

* Kích thước của bộ dữ liệu: Tổng cộng có 1500 hình ảnh, chia thành 2 bộ dữ liệu, bộ huấn luyện (training set) gồm 1000 ảnh và bộ kiểm tra (test set) gồm 500 ảnh. Mỗi hình ảnh đều được chú thích chi tiết với thông tin về vị trí và nội dung của các đoạn văn bản. Các đoạn văn bản trong CTW1500 không chỉ bao gồm các văn bản thẳng mà còn bao gồm nhiều văn bản cong và có hình dạng phức tạp, giúp tăng cường khả năng phát triển và kiểm tra các thuật toán phát hiện và nhận dạng văn bản.
* Phạm vi và tác dụng của bộ dữ liệu: Bộ dữ liệu Total-Text chủ yếu được sử dụng trong các bài toán sau:
* Phát hiện văn bản (Text Detection): Xác định vị trí của các đoạn văn bản trong hình ảnh.
* Nhận dạng văn bản (Text Recognition): Chuyển đổi các đoạn văn bản đã được phát hiện từ ảnh sang dạng ký tự (chữ) số.
* End-To-End Text Recognition: Kết hợp cả hai nhiệm vụ phát hiện và nhận dạng văn bản trong một hệ thống duy nhất.
* Một số nghiên cứu tiêu biểu sử dụng bộ dữ liệu CTW1500:
* PSENet (Progessive Scale Expansion Network): Một thuật toán phát hiện văn bản tiên tiến sử dụng mạng nơ-ron tích chập để phát hiện văn bản với các hình dạng phức tạp, đã sử dụng CTW1500 để đánh gái khả năng phát hiện văn bản cong và không đều nhau.
* CTPN (Connectionist Text Proposal Network): Một mô hình đề xuất văn bản kết nối, được thiết kế để phát hiện các đoạn văn bản trong hình ảnh tự nhiên.

Tổng quan, Bộ dữ liệu CTW1500 đóng vai trò quan trọng trong việc thúc đẩy nghiên cứu và phát triển các công nghệ nhận dạng và phát hiện văn bản trong hình ảnh. Với sự đa dạng và phức tạp của các đoạn văn bản trong bộ dữ liệu, CTW1500 cung cấp một thách thức lớn nhưng cũng là cơ hội để các nhà nghiên cứu phát triển các thuật toán và hệ thống có khả năng hoạt động tốt trong các điều kiện thực tế. Sự công nhận rộng rãi và tính ứng dụng cao của CTW1500 trong cộng đồng nghiên cứu đã giúp nó trở thành một bộ dữ liệu tiêu chuẩn trong lĩnh vực này.

## **5.5. Xây dựng hệ thống.**



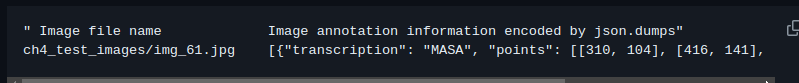
Hình 5.11: Mô tả kiến trúc và luồng của hệ thống.

Nhiệm vụ chính của bài toán này là phát hiện, nhận diện chữ và trích xuất ra thông tin từ ảnh thành văn bản. Với đặc điểm nổi bật chính là dữ liệu hình ảnh, chúng ta cần phải phân tích, trích xuất để có thể lấy ra thông tin một cách nhanh chóng đồng thời cũng phải đảm bảo được độ chính xác. Từ luồng xử lý dữ liệu ở ảnh trên chúng ta đã thấy được dữ liệu sẽ được đi qua 4 module chính, và các module này đã được giới thiệu chi tiết trong chương IV. Tiếp theo ở dưới đây chúng ta sẽ tìm hiểu cách huấn luyện mô hình, và triển khai hệ thống thành backend với thư viện FastAPI.

### **5.5.1. Huấn luyện mô hình.**

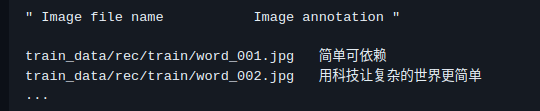
Về việc huấn luyện mô hình phát hiện và nhận diện chữ (text detection và text recognition) thì chúng ta sẽ sử dụng một thư viện nổi tiếng và hỗ trợ nhiều thuật toán khác nhau là PaddleOCR. Khi cần huấn luyện các model thì đầu tiên chúng ta sẽ cần 1 file config - là file mà thư viện PaddleOCR cung cấp, đây là file cấu hình tất cả những thông số thông tin, mô hình bên trong của PaddleOCR sẽ sử dụng.

**Về text detection:** Ngoài các bộ dữ liệu đã có sẵn nhãn cho phần phát hiện chữ thì chúng ta có thể huấn luyện thêm (transfer learning) dựa trên bộ dữ liệu mà chúng ta cần để mô hình có thể bắt chuẩn xác hơn, tập trung hơn vào lĩnh vực ảnh mà chúng ta cần xử lý. Giả sử bộ pretrained model được huấn luyện trên tập dữ liệu ảnh ngoài trời, hay ảnh đường phố. Tuy nhiên khi áp dụng vào các bài toán có bộ dữ liệu khác hoặc gần giống với dataset của bộ dữ liệu pretrained thì để cải thiện độ chính xác chúng ta nên huấn luyện thêm cho mô hình. Chúng ta cần một thư mục chứa toàn bộ ảnh sẽ đem vào để huấn luyện Cấu trúc bộ dữ liệu huấn luyện mà mô hình cung cấp sẽ như sau :

tên ảnh \t [{“transcription”: “text”, [[x1, y1], [x2, y2], [x3, y3], [x4, y4]]}]. Mỗi một ảnh sẽ được nằm trên 1 dòng trong file label, đồng thời x1, x2, x3, x4 là tọa độ x của 4 góc của text trong hình ảnh mà chúng ta đang gán nhãn. 

Hình 5.12: Hình ảnh mô tả label cho text detection

**Về text recognition:** Tương tự text detection, tuy đã có những bộ dữ liệu lớn đã tạo ra những bộ pretrained model đạt được độ chính xác cao trên nhiều tập đánh giá. Tuy nhiên, khi sử dụng nếu muốn model đạt được độ chính xác cao hơn trên phạm vi lĩnh vực mình cần làm (hay còn gọi là domain field) thì chúng ta vẫn cần huấn luyện thêm cho mô hình (transfer learning). Và đương nhiên, PaddleOCR cũng hỗ trợ về vấn đề huấn luyện cho mô hình. Chúng ta cũng cần một thư mục chứa toàn bộ ảnh, và file labels.txt mà ta sẽ dùng để train sẽ có cấu trúc như sau : “Đường dẫn đến ảnh \t nhãn của ảnh đó”. Khá đơn giản nhưng chú ý là chúng ta phải luôn luôn có “**\t**” để thư viện phân biệt được tên ảnh và nhãn dán. Mỗi ảnh và nhãn sẽ nằm trên 1 dòng duy nhất.

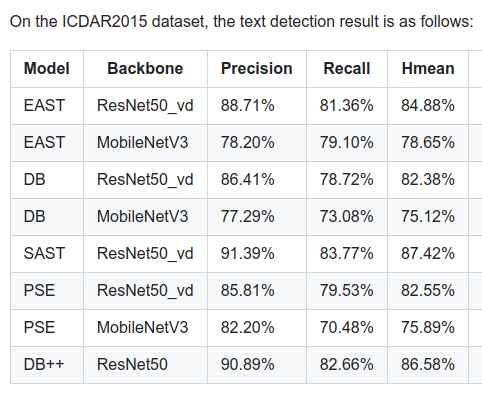
Tổng quan, thư viện PaddleOCR hỗ trợ rất tốt các nhiệm vụ liên quan đến trích xuất thông tin, rất đa dạng và linh hoạt khi triển khai huấn luyện các mô hình. Tuy nhiên có điểm yếu của thư viện PaddleOCR, là khá phức tạp khi cài đặt thư viện để thư viện có thể tương thích với card GPU và phần cứng của máy (hoặc server) từ đó huấn luyện trên GPU sẽ nhanh hơn rất nhiều lần so với trên CPU hay các nơi cung cấp tài nguyên GPU miễn phí như google colab hay kaggle, … Mặc dù vậy, PaddleOCR cũng là một trong những thư viện tiên phong và đi đầu và có cộng đồng lớn mạnh trong lĩnh vực OCR tính đến thời điểm hiện tại.

Hình 5.13: Hình ảnh mô tả label cho text recognition.

### **5.5.2. Kết quả đạt được của các thuật toán trên bộ dữ liệu công khai.**

**Về text detection:** Về cơ bản trên bộ dữ liệu ICDAR2015, 2 mô hình DB và DB++ dựa trên resnet50 và mobileNetv3 đều đạt được độ chính xác cao, khi so sánh với các thuật toán còn lại. Ở mô hình DB++ Precision đạt 90.89%, Recall đạt 82.66% , còn Hmean cũng đạt 86.58%. Mặc dù không phải cao nhất, nhưng cũng không thể phủ nhận rằng mô hình DB++ cũng đã đạt được sota (state of the art) trong lĩnh vực phát hiện chữ.

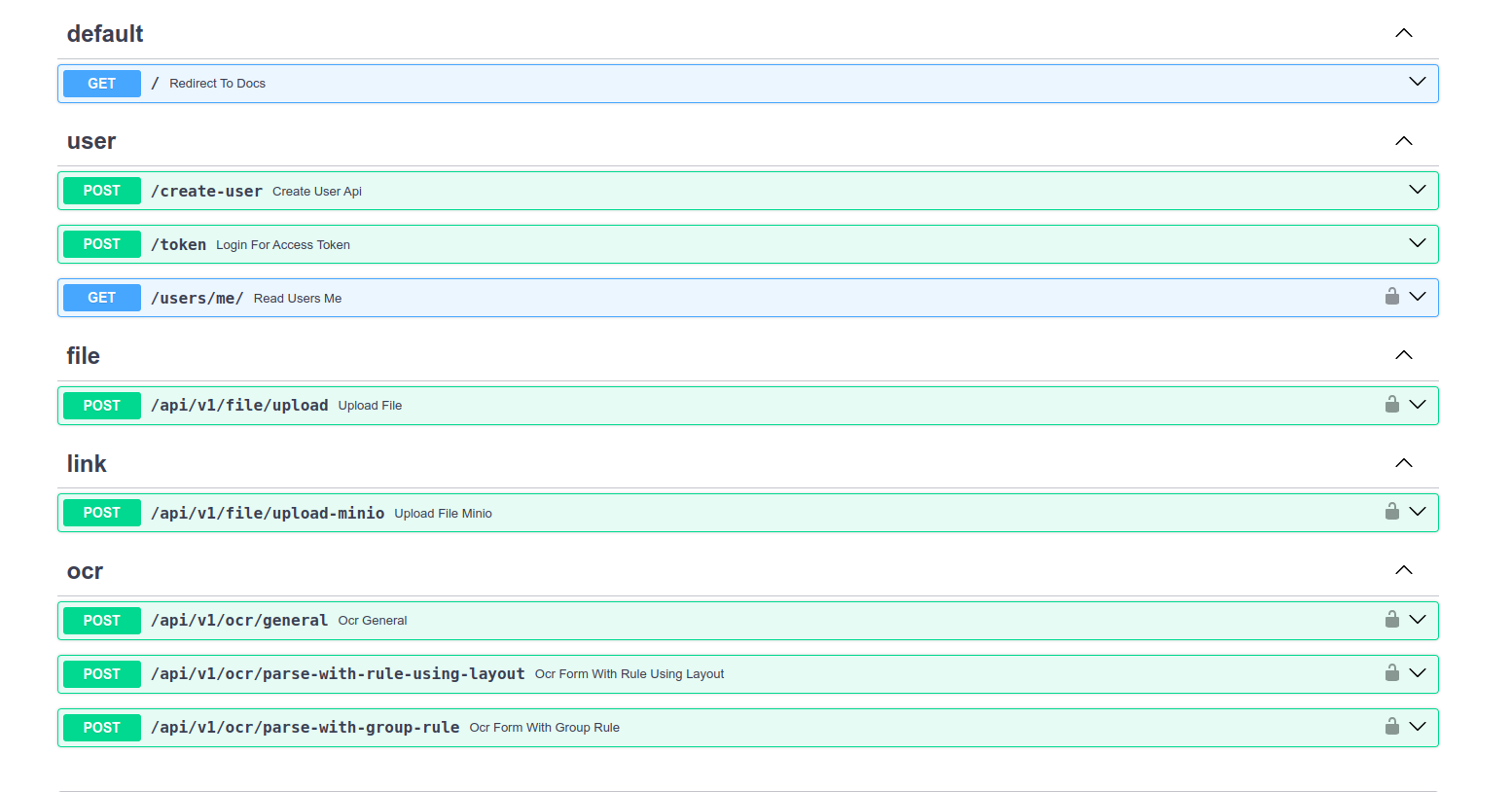
Hình 5.14: Hình mô tả độ chính xác của các thuật toán text detection.



**Về text recognition:** Mô hình SVTR cũng đạt được giá trị khá cao trong các bộ dữ liệu public như IC15, SVTP, … Mô hình cũng đạt được độ chính xác là 89.25% với cấu trúc của mô hình là svtr tiny. Cao hơn khá nhiều so với thuật toán CRNN lên 8.21%.

Hình 5.15: Hình mô tả độ chính xác của text recognition.

### **5.5.3. Triển khai các module và tạo backend với FastAPI.**

Ở đây chúng ta triển khai hệ thống trích xuất thông tin với thư viện FastAPI làm backend chính, dưới đây là một số API được viết để phục vụ các nhiệm vụ khác nhau của hệ thống: 

Hình 5.16: Hình ảnh mô tả một số API được thiết kế.

* Hệ thống được chia theo nhiều endpoint để thực hiện song song nhiều nhiệm vụ cùng lúc
* Tuy nhiên ở đây chúng ta sẽ có một số api chính: api upload file, api general, và api parse-with-group-rule.

**Api upload** cho phép upload file, không chỉ ở các dạng ảnh khác nhau mà còn hỗ trợ tải lên các file dạng pdf. Vì file có thể rất nặng, do nhiều tài liệu gộp thành 1 bộ, 1 file dữ liệu nên cần tách riêng api upload ra để server có thể chạy song song để tải các file lên server đồng thời chứ không phải chờ các api khác làm xong mới bắt đầu tải. Điều này giúp cho server có thể hoạt đông liên tục, tăng tốc độ và thời gian đợi xử lý của mô hình.

**API ocr/general** là api chính được thiết kế để làm nhiệm vụ tích hợp nhiều model để phát hiện, nhận diện chữ, … Từ đó tạo ra file json, chứa toàn bộ thông tin của file mà mô hình đang chạy. Nó chiếm lượng lớn tài nguyên và thời gian chạy của server.

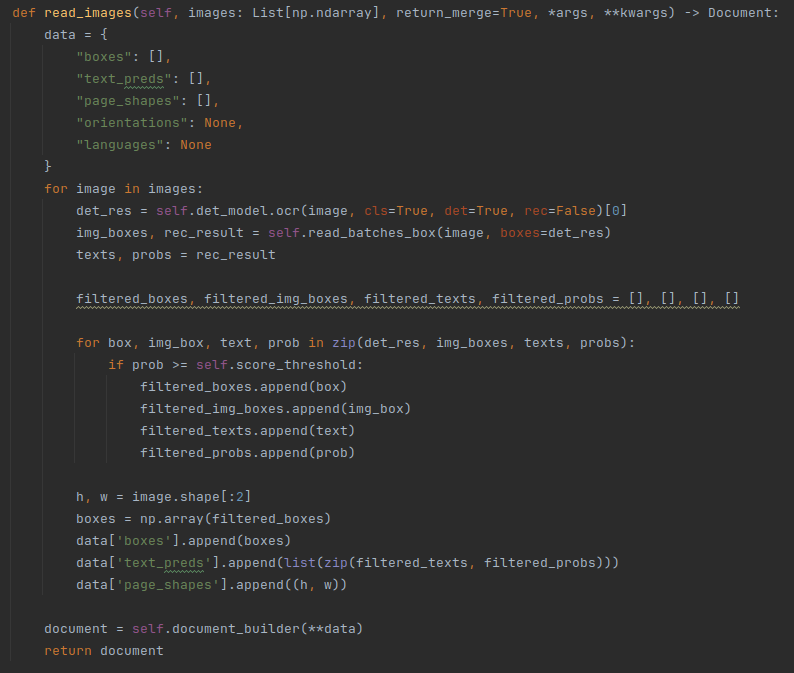
**API parse-with-group-rule** là api làm nhiệm vụ hậu xử lý, hay đây là module post processing, nhận file json mà API ocr/general trả ra, đồng thời nhận file rule-based được thiết kế để trích xuất các trường thông tin dữ liệu có trong file và đã được định nghĩa trong file rule-based.

**Tổng quan,** ở đây cũng có thêm nhiều các api khác để phục vụ các mục đích khác nhau, có những api để lấy thông tin của file đã gửi lên như trong tag files, có những api để test mô hình với các phiên bản nhận diện khác nhau mà server đang triển khai như ở tag OCR trên docs của FastAPI.

Ở phía trên, chúng ta đã nói sơ qua tác dụng chính của các API cần thiết để triển khai mô hình. Bây giờ chúng ta sẽ đi sâu hơn một chút về code triển khai các module mà hệ thống sử dụng.

**Đầu tiên là module text detection:** 

Hình 5.17: Hình mô tả text detection module.

Đầu tiên chúng ta sẽ khởi tạo các tham số của mô hình, sử dụng model db++ mà paddleocr đã cung cấp và lưu vào 1 biến det\_model.

Hình 5.18: Hình mô tả hàm read images

Sau đó chúng ta sẽ viết hàm read\_images với đầu vào là một list các ảnh và trả ra 1 biến document chứa toàn bộ thông tin về tọa độ của các chữ.

**Theo sau là module text recognition:** 

Hình 5.19: Hình mô tả text recognition module.

Tương tự như module text detection, ở module này chúng ta cũng sẽ khởi tạo các thông tin cần thiết của mô hình, như model\_path là đường dẫn đến model text recognition mà chúng ta đã huấn luyện (hoặc đường dẫn đến bộ trọng số mà chúng ta sử dụng). Ở đây có hai điều cần chú ý. Đầu tiên, mặc định sẵn PaddleOCR sẽ tự khởi tạo các thông số cho model khi chúng ta không tự cài đặt thông số, do đó nếu như lệch phiên bản hoặc file thông số (config) thì PaddleOCR sẽ ngay lập tức báo lỗi và không thể khởi tạo, vì vậy cần cài đặt chuẩn các thông số khi khởi tạo mô hình để chạy. Điều thứ 2 là file vocab của model sử dụng (là file chứa các ký tự mà mô hình sẽ học cho ngôn ngữ đó). Trong code trên chính là biến rec\_char\_dict\_path, mô hình sẽ đọc biến này kiểm tra xem có tồn tại hay không và đọc vào trong mô hình.

**Cuối cùng là module hậu xử lý:**

Module này bao gồm rất nhiều các rule-based để có thể giúp hệ thống lấy ra được giá trị của trường dữ liệu. Các trường dữ liệu này sẽ được khai báo sẵn trong file rule-based khi gửi lên hệ thống. Các rule-based này bao gồm : tiền tố, hậu tố, regex, dòng mới, chia các dòng, tỷ lệ chính xác cho phép, mức độ ưu tiên của các trường dữ liệu, … Và rất nhiều các rule based khác để có thể lấy dữ liệu ra một cách chính xác và nhanh chóng hơn. Tất cả các code xử lý logic và đọc file rule rule-based đều được đẩy lên github đồ án tốt nghiệp ở link sau :

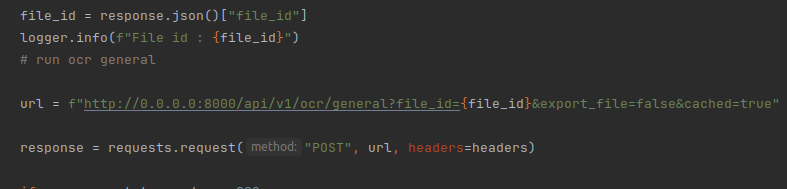
https://github.com/anhalu/DATN-BackEnd

### **5.5.4. Triển khai hệ thống với giao diện người dùng đơn giản.**

Mục tiêu của phần này là triển khai hệ thống trích xuất thông tin với một vài giao diện người dùng đơn giản, thể hiện trực quan, cách thức hoạt động của hệ thống. Về web, ta sử dụng thư viện Gradio trong python. Gradio là một thư viện python mạnh mẽ và dễ sử dụng, được thiết kế để tạo giao diện người dùng cho các mô hình học máy và các ứng dụng dữ liệu một cách nhanh chóng và dễ dàng. Gradio cho phép người dùng nhập dữ liệu và nhận kết quả từ mô hình học máy mà không cần viết mã phức tạp. Đồng thời Gradio cũng hỗ trợ nhiều loại đầu vào và đầu ra (ảnh, chữ, âm thanh, …). Ngoài ra thư viện cũng hỗ trợ chạy cục bộ hoặc trên đám mây, dễ dàng tích hợp với nhiều thư viện học máy.

Bây giờ, chúng ta sẽ đi sâu hơn một chút về code triển khai mô hình bằng Gradio. Giống như đã liệt kê ở phần trước, chúng ta sẽ sử dụng 3 API chính, đầu tiên là upload file. 

Hình 5.20: Mô tả cách sử dụng API upload

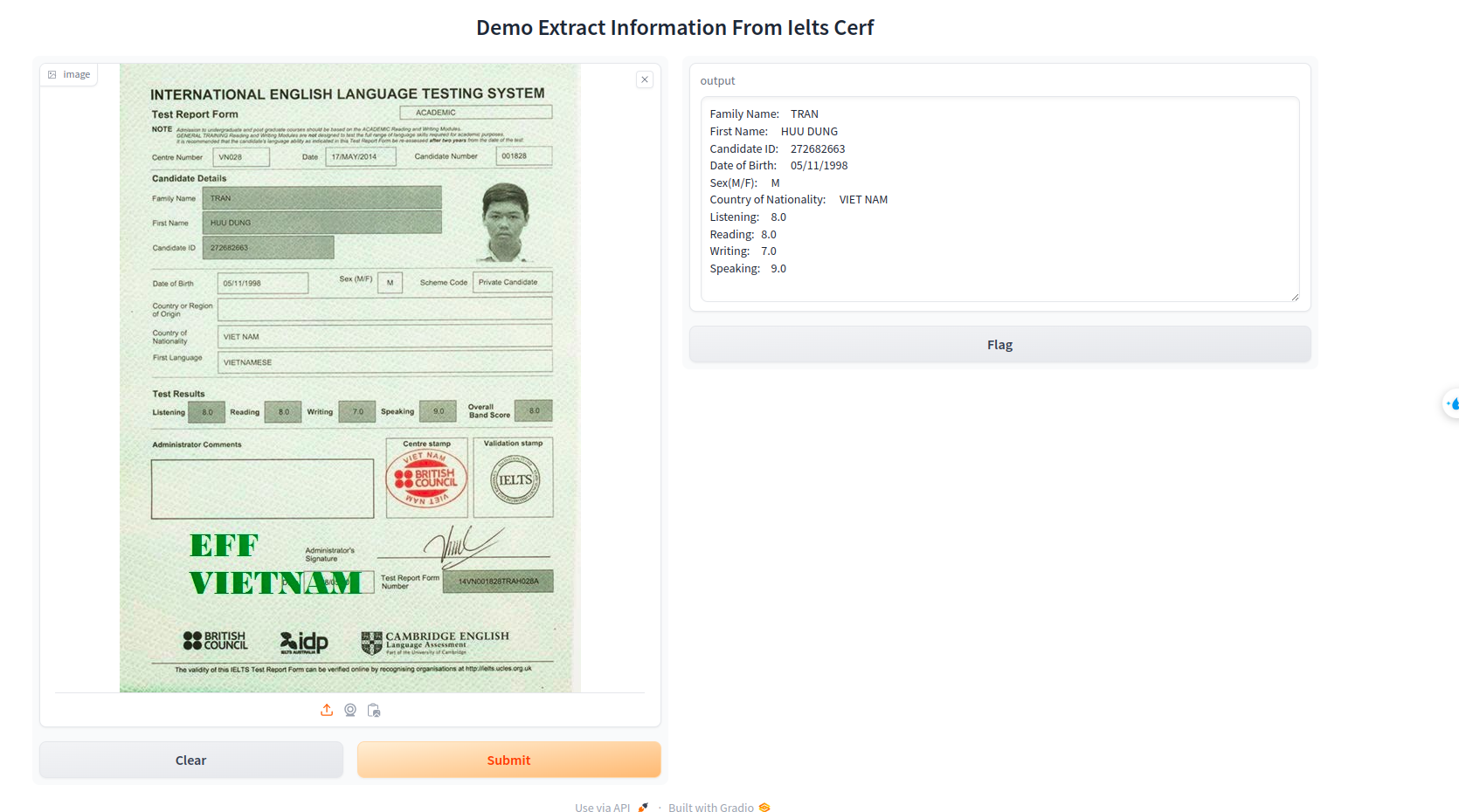
Sau đó chúng ta cần gửi yêu cầu lên API ocr/general để server chạy các mô hình học text detection, text recognition trên file mà chúng ta gửi lên ở api upload.

Hình 5.21: Mô tả cách sử dụng API general

 Cuối cùng chúng ta sử dụng API parse-with-group-rule để thực hiện các rule based để lấy ra các thông tin của file. Đồng thời đưa về lại kết quả ra web.

Hình 5.22: Mô tả cách sử dụng API parse-with-group-rule

Kết quả sau khi chạy mô hình và gửi ảnh lên sẽ như sau :



Hình 5.23: Mô tả giao diện người dùng.

# **KẾT LUẬN**

## **Kết quả đạt được.**

Trong thời đại 4.0, AI đang là một xu hướng cực kỳ mạnh mẽ và đang rất được quan tâm. Công nghệ AI đang được ứng dụng khắp mọi nơi và bài toán “*trích xuất thông tin*”cũng đang được đầu tư rất nhiều để cho bài toán trở nên hoàn thiện và khả thi hơn. Với mục tiêu học tập và nghiên cứu, các hướng giải quyết để bài toán trở nên gần gũi và dễ tiếp cận hơn. Đề tài “*Nghiên cứu sử dụng trí tuệ nhân tạo trích xuất thông tin tự động từ chứng chỉ IELTs*” đã được em nghiên cứu và ấp ủ từ lâu, vì mức độ thử thách của bài toán này khiến em trở nên thích thú và muốn chinh phục.

Đề tài hoàn toàn có thể ứng dụng được trong công cuộc chuyển đổi số, tích hợp trích xuất thông tin, đóng vai trò như một công cụ hữu ích giúp tối ưu hóa thời gian xử lý, phân tích và lọc dữ liệu, đặc biệt là các dữ liệu dạng ảnh hay pdf,… Qua quá trình học tập và rèn luyện tại công ty TNHH SUN ASTERISK VIET NAM đã giúp em trau dồi kiến thức, bên cạnh đó là sự giúp đỡ tận tình của thầy Phạm Bá Tuấn Chung, đã hướng dẫn cho em tận tình để hoàn thành môn đồ án một cách tốt nhất. Và chính bản thân em cũng đã có cho mình được lượng kiến thức nhất định: Hoàn thiện được mục tiêu đặt ra ban đầu của đề tài, xây dựng mô hình, cải tiến, triển khai với API. Hoàn thiện kỹ năng đọc hiểu và làm chủ sản phẩm của mình, tự giác chủ động trong công việc. Nắm chắc về kiến thức Deep Learning nói chung và Computer Vision nói riêng. Tính thực tiễn cao, bước đầu trong quá trình đưa lên sử dụng.

## **Hướng phát triển.**

Vì kỹ năng còn hạn chế và kinh nghiệm chưa đủ nhiều, còn một số chức năng mà em chưa kịp hoàn thiện, em rất mong được quý thầy cô đóng góp ý kiến để em hoàn thiện đề tài của mình trở nên tốt hơn. Hướng phát triển ra các thiết bị edge device, cải thiện tốc dự đoán của mô hình, nhúng source code vào thiết bị, cũng như xây dựng mô hình nhận diện theo thời gian thực. Đồng thời xây dựng hệ thống dùng để thiết kế rule-based, hoặc các công nghệ đọc hiểu ngữ nghĩa của dữ liệu từ đó giúp trích xuất nhiều loại thông tin có cấu trúc và không có cấu trúc khác nhau

# **TÀI LIỆU THAM KHẢO**

[1] Ashish,Vaswani, (2017),“Attention Is All You Need.”, *arXiv.*

[2] Baidu, “PaddleOCR”. Github Repositories

[*https://github.com/PaddlePaddle/PaddleOCR*](https://github.com/PaddlePaddle/PaddleOCR)

[3] Baoguang Shi, Xiang Bai, (2015), “An End to End Trainable

Neural Network for Image-based Sequence Recognition and Its

Application to Scene Text Recognition ”, *arXiv.*

[4] IBM, (2023), “What are Recurrent Neural Networks?”

[5] Keiron O'shea and Ryan Nash, (2015), “An Introduction to

Convolutional Neural Networks”, *arXiv.*

[6] Minghui Liao1, Zhaoyi Wan2, Cong Yao2, Kai Chen, Xiang

Bai, (2022), “Real-time Scene Text Detection with Differentiable

Binarization”, *arXiv.*

[7] Minghui Liao, Cong Yao, Xiang Bai, (2022), “Real-Time Scene

Text Detection with Differentiable Binarization and Adaptive

Scale Fusion”, *arXiv.*

[8] Mindee, “DocTR (Document Text Recognition)” Github

Repositories [*https://github.com/mindee/doctr*](https://github.com/mindee/doctr)

[9] IIya Sutskever, Oriol Vinyals, Quoc V.Le, (2014), “Sequence to

Sequence Learning with Neural Network”, *arXiv.*

[10] Poatek, (2023), “Understanding OCR: Your entertaining Guide

To Optical Character Recognition”, *medium.*

[11] Rina Buoy, Sokchea Kor, Nguonly Taing, (2021), “An End to

End Khmer Optical Character Recognition using Sequence to

Sequence With Attention”.. *arXiv.*

[12] Sun\*, website “Viblo”, *https://viblo.asia/*

[13] Vu Huu Tiep, Blog “Machine Learning co ban”

[*https://machinelearningcoban.com/*](https://machinelearningcoban.com/)

[14] Yongkun Du, Zhineng Chen, (2022), “SVTR: Scene Text

Recognition with a Single Visual Model” *arXiv.*