

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT



ĐỒ ÁN TỐT NGHIỆP  
NGHÀNH  
CÔNG NGHỆ THÔNG TIN

HÀ NỘI - 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO  
TRƯỜNG ĐẠI HỌC MỎ - ĐỊA CHẤT

# ĐỒ ÁN TỐT NGHIỆP

CHUYÊN NGHÀNH  
KHOA HỌC MÁY TÍNH - ỨNG DỤNG

## ĐỀ TÀI

Nghiên cứu ứng dụng công nghệ OCR  
nhận dạng hóa đơn

SINH VIÊN THỰC HIỆN  
HỒ VĂN ĐỨC  
DCCTKH64A

CÁN BỘ HƯỚNG DẪN  
ThS. ĐẶNG VĂN NAM  
BỘ MÔN KHOA HỌC  
MÁY TÍNH

HÀ NỘI - 2023

## Mục lục

<b>Mục lục</b>	<b>1</b>
<b>Danh sách hình vẽ</b>	<b>4</b>
<b>Danh sách bảng</b>	<b>6</b>
<b>Danh mục các từ viết tắt, dịch thuật</b>	<b>7</b>
<b>1 Tổng quan về đề tài</b>	<b>11</b>
1.1 Lý do chọn đề tài . . . . .	11
1.2 Tính cấp thiết của đề tài . . . . .	13
1.3 Mục Tiêu . . . . .	14
1.4 Phạm vi đề tài . . . . .	15
1.5 Bố cục chương . . . . .	15
<b>2 Cơ sở lý thuyết liên quan</b>	<b>17</b>
2.1 Nhận dạng ký tự quang học . . . . .	17
2.1.1 Nhận dạng ký tự quang học là gì? . . . . .	17
2.1.2 Lịch sử của OCR . . . . .	19
2.1.3 Nguyên tắc hoạt động của OCR . . . . .	21
2.2 Các thuật toán OCR . . . . .	22
2.2.1 Phát hiện văn bản . . . . .	22
2.2.2 Nhận dạng văn bản . . . . .	23
2.2.3 Nhận dạng cấu trúc tài liệu . . . . .	25
2.3 OCR dựa trên mẫu và OCR dựa trên AI . . . . .	29
2.3.1 OCR dựa trên mẫu . . . . .	29
2.3.2 OCR Dựa trên AI . . . . .	30

2.4	Học sâu . . . . .	31
2.4.1	Học sâu là gì? . . . . .	32
2.4.2	Kiến trúc mạng nơ-ron nhân tạo . . . . .	33
2.4.3	ResNet . . . . .	34
2.4.4	Transformer . . . . .	37
2.4.5	LayoutXLM . . . . .	42
2.4.6	DBNet . . . . .	48
2.4.7	TransformerOCR . . . . .	51
2.4.8	Ứng dụng trong OCR . . . . .	53
<b>3</b>	<b>Công cụ và môi trường thực hiện</b> . . . . .	<b>55</b>
3.1	Phầm mềm và công cụ hỗ trợ . . . . .	55
3.1.1	Google Colab . . . . .	55
3.1.2	Vast.ai . . . . .	56
3.1.3	CUDA Toolkit . . . . .	57
3.1.4	cuDNN . . . . .	58
3.1.5	PaddleOCR . . . . .	58
3.1.6	PPOCRLLabel . . . . .	59
3.1.7	Pytorch . . . . .	60
3.1.8	Gradio . . . . .	61
3.2	Cấu hình phần cứng . . . . .	62
3.3	Tạo môi trường và cài công cụ cần thiết . . . . .	63
3.3.1	Cài đặt Miniconda: . . . . .	63
3.3.2	Cài đặt CUDA Toolkit: . . . . .	64
3.3.3	Cài đặt cuDNN . . . . .	64
3.3.4	Cài đặt PaddleOCR . . . . .	64
3.3.5	Cài đặt PPOCRLLabel . . . . .	65
<b>4</b>	<b>Thu thập dữ liệu và Huấn luyện mô hình</b> . . . . .	<b>66</b>
4.1	Thu thập và chuẩn bị dữ liệu huấn luyện . . . . .	66
4.1.1	Lựa chọn hóa đơn . . . . .	66
4.1.2	Chuẩn bị dữ liệu . . . . .	67

4.1.3	Tổng quan về dữ liệu đào tạo . . . . .	70
4.2	Huấn luyện mô hình . . . . .	71
4.2.1	Phát hiện văn bản . . . . .	71
4.2.2	Phát hiện văn bản . . . . .	72
4.2.3	Key information extraction . . . . .	73
4.3	Kết quả mô hình . . . . .	74
<b>5</b>	<b>Xây dựng chương trình và Kết quả</b>	<b>76</b>
5.1	Ý tưởng . . . . .	76
5.2	Xây dựng chương trình . . . . .	78
5.2.1	Tiền xử lý ảnh . . . . .	78
5.2.2	Phát hiện vùng văn bản . . . . .	78
5.2.3	Nhận dạng văn bản . . . . .	79
5.2.4	KIE . . . . .	79
5.2.5	Post-process . . . . .	80
5.3	Kết quả . . . . .	83
<b>Kết luận</b>		<b>87</b>
<b>Tài Liệu Tham Khảo</b>		<b>90</b>

## Danh sách hình vẽ

2.1	Ví dụ về nhiệm vụ phát hiện văn bản . . . . .	23
2.2	Tổng quan về thuật toán phát hiện văn bản . . . . .	24
2.3	Văn bản đều đặn (trái) và Văn bản không đều đặn (phải) . . . . .	25
2.4	CTC-based recognition algorithm VS. Attention-based recognition algorithm . . . . .	26
2.5	Thuật toán nhận dạng dựa trên phân vùng ký tự . . . . .	26
2.6	RE(trái) VS SER(phải) . . . . .	28
2.7	Biểu đồ Venn về Trí tuệ nhân tạo . . . . .	32
2.8	Kiến trúc một mạng Nơ-ron . . . . .	34
2.9	Một khối residual . . . . .	36
2.10	Kiến trúc tổng thể của Transformer [13] . . . . .	38
2.11	Position encoding 128 chiều cho một câu có độ dài tối đa là 50	42
2.12	Kiến trúc của mô hình LayoutXLM . . . . .	43
2.13	Pipeline truyền thống (luồng màu xanh) và pipeline của DB (luồng màu đỏ) . . . . .	48
2.14	Minh họa về phân đoạn khác biệt và đạo hàm của nó. (a) So sánh số liệu giữa phương pháp nhị phân tiêu chuẩn (SB) và phân đoạn khác biệt (DB). (b) Đạo hàm của $l_+$ . (c) Đạo hàm của $l_-$ . [16] . . . . .	50
2.15	Kiến trúc của phương pháp DB . . . . .	51
2.16	Kiến trúc tổng qua của TransformerOCR [17] . . . . .	52
3.1	Môi trường đã được kích hoạt . . . . .	63
4.1	Hóa đơn Okono(trái) và VinCommerce(phải) . . . . .	67
4.2	Giao diện PPOCRLabel . . . . .	68

4.3	Danh sách ảnh PPOCRLabel	68
4.4	Gán nhãn ảnh với PPOCRLabel	69
4.5	Dữ liệu hóa đơn được export từ PPOCRLabel	70
5.1	Lưu đồ của chương trình nhận dạng hóa đơn	77
5.2	Kết quả hình ảnh hóa đơn 1	83
5.3	Kết quả Excel của hóa đơn 1	84
5.4	Kết quả hình ảnh hóa đơn 2	84
5.5	Kết quả Excel của hóa đơn 2	85
5.6	Kết quả hình ảnh hóa đơn 3	85
5.7	Kết quả Excel của hóa đơn 3	86

## Danh sách bảng

3.1 So sánh cấu hình phần cứng huấn luyện của từng tác vụ OCR	62
4.1 Độ chính xác của cả 3 mô hình . . . . .	74

## Danh mục các từ viết tắt, dịch thuật

STT	Tên Tiếng Anh/Viết tắt	Ý nghĩa
1	Artificial intelligence (AI)	Trí tuệ nhân tạo
2	Optical character recognition (OCR)	Nhận dạng ký tự quang học
3	Key Information Extraction (KIE)	Trích xuất thông tin
4	SSD	Single Shot MultiBox Detector
5	CTPN	Connectionist Text Proposal Network
6	EAST	An Efficient and Accurate Scene Text Detector
7	PSENet	Progressive Scale Expansion Network
8	DBNet	Differentiable Binarization Network
9	CTC	Connectionist Temporal Classification
10	STAR-Net	SpaTial Attention Residue Network
11	TPS	Thin-Plate-Spline
12	RARE	Robust text recognizer with Automatic REctification
13	Convolution Neural Network (CNN)	Mạng nơ-ron tích chập
14	Visual Question Answering (VQA)	Trả lời câu hỏi trực quan
15	SER	Semantic Entity Recognition
16	RE	Relation Extraction

<b>STT</b>	<b>Tên Tiếng Anh/Viết tắt</b>	<b>Ý nghĩa</b>
17	Named Entity Recognition (NER)	Nhận dạng tên thực thể
18	ResNet	Residual Neural Network
19	BERT	Bidirectional Encoder Representations from Transformers
20	GPT	Generative pre-trained transformers
21	XFUND	A Multilingual Form Understanding Benchmark
22	MVLM	Masked Visual-Language Modeling
23	TIA	Text-Image Alignment
24	TIM	Text-Image Matching

## Lời cảm ơn

## Mở đầu

## Chương 1

# Tổng quan về đề tài

### 1.1 Lý do chọn đề tài

Cuộc sống hiện nay việc mua bán trao đổi hàng hóa được diễn ra thường xuyên giữa người mua và người bán. Ban đầu hóa đơn có giá trị làm bằng chứng chứng nhận cho việc chuyển nhượng hàng hóa giữa hai bên. Mọi việc tranh chấp trong mua bán hàng hoá hai bên tự giải quyết.

Trong quá trình phát triển xã hội, hóa đơn được phổ biến dần trong một cộng đồng khi được cộng đồng chấp nhận một cách tự nguyện. Các cộng đồng có thể là các Phường hội hoặc các định chế làng, xã. Những tranh chấp trong việc mua bán hàng hoá được các cộng đồng xử lý trên cơ sở dân sự. Khi nhà nước tham dự vào quản lý mua bán hàng hoá và xử lý những tranh chấp về hàng hoá dựa trên pháp luật dân sự và hình sự thì hóa đơn được nhà nước quy định để làm căn cứ pháp lý chứng minh cho việc chuyển nhượng hàng hoá giữa các bên và làm căn cứ để xác nhận quyền sở hữu hợp pháp của người có hàng hoá. Do đó hóa đơn là một loại tài liệu quan trọng trong các giao dịch. Nó được sử dụng để ghi lại các giao dịch mua bán hàng hóa và dịch vụ. Thông tin trên hóa đơn bao gồm tên của người bán, tên của người mua, ngày lập hóa đơn, số lượng hàng hóa hoặc dịch vụ, giá cả, tổng số tiền phải thanh toán.v.v...

Hiện nay, hóa đơn thông thường được lập dưới dạng tài liệu giấy, có thể là hóa đơn giá trị gia tăng, hóa đơn bán hàng, tem, vé, thẻ, phiếu thu tiền bảo

hiếm... Hóa đơn giấy có thể được phát hành theo các hình thức như hóa đơn đặt in, hóa đơn tự in, hóa đơn mua của cơ quan thuế. Điều này gây ra một số khó khăn trong việc quản lý hóa đơn, chẳng hạn như:

- Quá nhiều hóa đơn: Các doanh nghiệp có thể phát sinh một số lượng lớn hóa đơn, từ các nhà cung cấp, khách hàng và các bên liên quan khác. Việc quản lý nhiều hóa đơn có thể là một thách thức, đặc biệt nếu chúng không được tổ chức và lưu trữ một cách hiệu quả.
- Sai sót và mất mát: Quản lý hóa đơn thủ công có thể gặp phải sai sót và mất mát hóa đơn, đặc biệt khi các hóa đơn được lưu trữ và xử lý bằng tay. Điều này có thể dẫn đến việc đòi tiền sai, không thu được tiền đúng lúc hoặc mất cơ hội thu hồi tiền nợ.
- Tìm kiếm hóa đơn: Khi cần tìm một hóa đơn cụ thể, việc tìm kiếm nó có thể là một thách thức nếu nó không được tổ chức và lưu trữ một cách hiệu quả. Điều này có thể dẫn đến chậm trễ trong quá trình thanh toán hóa đơn hoặc thậm chí mất hóa đơn.
- Lưu trữ hóa đơn. Các hóa đơn phải được lưu trữ trong một thời gian nhất định theo quy định của pháp luật. Điều này có thể là một thách thức nếu không có một quy trình lưu trữ hóa đơn hiệu quả.
- Tuân thủ luật pháp: Việc tuân thủ các quy định và luật pháp về hóa đơn là rất quan trọng. Nếu không tuân thủ đúng, doanh nghiệp có thể phải đối mặt với các vấn đề pháp lý và hậu quả tài chính nghiêm trọng.
- Thay đổi trong quy định thuế: Thay đổi trong quy định thuế và các quy tắc về hóa đơn có thể làm cho việc quản lý hóa đơn trở nên phức tạp hơn, đòi hỏi doanh nghiệp phải cập nhật và điều chỉnh quy trình của mình thường xuyên.

Để giải quyết các vấn đề này ta có thể cân nhắc sử dụng các phần mềm quản lý hóa đơn hiện đại. OCR có thể giải quyết vấn đề này bằng cách tự động hóa quá trình nhập liệu của hóa đơn một cách đơn giản và dễ dàng.

## 1.2 Tính cấp thiết của đề tài

OCR là một công nghệ có thể giải quyết các vấn đề trên bằng cách tự động trích xuất thông tin từ hóa đơn. Đây là một công nghệ cho phép máy tính nhận dạng và chuyển đổi văn bản từ hình ảnh chứa văn bản thành dạng văn bản có thể chỉnh sửa, tìm kiếm và lưu trữ. Áp dụng OCR trong việc quản lý hóa đơn có thể giúp giải quyết một số vấn đề như sau:

- **Tiết kiệm thời gian:** OCR giúp doanh nghiệp tiết kiệm lượng lớn thời gian so với quá trình nhập dữ liệu thủ công. Với công cụ OCR, thông tin có thể dễ dàng được trích xuất sang các định dạng kỹ thuật số theo nhu cầu chỉ bằng việc chụp và tải ảnh lên. Không chỉ vậy, dữ liệu khi được trích xuất có thể dễ dàng được tìm kiếm, chỉnh sửa và thực hiện nhiều tác vụ khác, hỗ trợ quy trình xử lý tài liệu dễ dàng và thuận tiện hơn. Trên thực tế, nghiên cứu đã phát hiện ra rằng lượng thời gian dành cho công việc giấy tờ có thể giảm 75% khi sử dụng OCR. Trung bình, thời gian để trích xuất một tài liệu sang dạng số chỉ từ 0.5 – 2 giây với công cụ OCR, một sự tối ưu đáng kể so với thời gian trung bình 1– 5 phút khi sử dụng phương pháp nhập liệu truyền thống. [1]
- **Cải thiện độ chính xác:** Việc nhập liệu bằng tay không chỉ tốn nhiều thời gian, nguồn lực mà còn có mức độ rủi ro cao trong sai sót nhập. Nhất là với các loại tài liệu bao gồm nhiều trường thông tin bằng số, địa chỉ email, địa chỉ nhà,... việc nhập tay thủ công khó có thể chính xác 100%. Những lỗi sai thông tin ngay từ bước đầu sẽ khiến kho dữ liệu doanh nghiệp không được “sạch” và chính xác.
- **Hỗ trợ tuân thủ luật pháp:** Sử dụng OCR giúp đảm bảo tính chính xác và toàn vẹn của dữ liệu trên hóa đơn, từ đó đảm bảo tuân thủ các quy định về hóa đơn và thuế.
- **Quản lý hóa đơn điện tử:** Kết hợp OCR với hóa đơn điện tử giúp tự động tạo và lưu trữ các hóa đơn điện tử, giảm thiểu việc sử dụng giấy tờ truyền thống và tiết kiệm không gian lưu trữ.

Nhìn chung, OCR là một công nghệ có nhiều tiềm năng ứng dụng trong lĩnh vực kế toán và tài chính. OCR có thể giúp các doanh nghiệp tiết kiệm thời gian, tăng cường độ chính xác và cải thiện khả năng truy xuất thông tin hóa đơn. Tuy nhiên, để OCR hiệu quả, ta cần đảm bảo rằng hóa đơn được quét và lưu trữ ở định dạng tốt, đủ để đảm bảo hiệu suất nhận dạng của OCR cao nhất.

### 1.3 Mục Tiêu

Dựa vào những vấn đề của hóa đơn và các giải pháp của OCR ở mục 1.1 và 1.2, mục tiêu của đề tài “**Nghiên cứu ứng dụng công nghệ OCR nhận dạng hóa đơn**” là tìm hiểu, đánh giá khả năng ứng dụng của công nghệ OCR hiện nay trong việc quản lý hóa đơn. Cụ thể đề tài tập trung vào các mục tiêu sau:

- Tìm hiểu về công nghệ OCR: Nghiên cứu các nguyên lý hoạt động của OCR, các phương pháp và thuật toán phổ biến trong việc nhận dạng văn bản từ hình ảnh.
- Phân tích hiệu quả và lợi ích của ứng dụng OCR trong quản lý hóa đơn: So sánh các phương pháp truyền thống và ứng dụng OCR trong việc quản lý hóa đơn, đánh giá hiệu quả và lợi ích mà OCR mang lại, bao gồm tối ưu hóa thời gian, giảm thiểu sai sót, tiết kiệm chi phí và tăng cường khả năng xử lý lượng hóa đơn lớn.
- Đề xuất giải pháp và quy trình triển khai OCR: Dựa trên kết quả nghiên cứu, đề xuất các giải pháp và quy trình triển khai OCR trong việc quản lý hóa đơn, bao gồm lựa chọn phần mềm OCR phù hợp, quy trình xử lý hóa đơn, quản lý dữ liệu và bảo đảm tính an toàn thông tin.
- Đánh giá hiệu quả thực tế: Tiến hành thử nghiệm ứng dụng OCR trong môi trường thực tế của doanh nghiệp hoặc tổ chức để đánh giá hiệu quả, tính ổn định và khả năng mở rộng của giải pháp OCR.

Dựa trên kết quả đánh giá, đề xuất các cải tiến và phát triển tương lai của

công nghệ OCR trong việc quản lý hóa đơn, nhằm nâng cao hiệu quả và khả năng ứng dụng của nó trong thực tế

### 1.4 Phạm vi đề tài

Phạm vi của đề tài “Nghiên cứu ứng dụng OCR nhận dạng hóa đơn” tập trung vào việc áp dụng công nghệ Nhận dạng ký tự quang học để tự động nhận dạng và trích xuất thông tin từ hình ảnh của hóa đơn. Giải quyết các vấn đề liên quan đến việc tự động hóa quá trình xử lý hóa đơn, giảm thiểu công việc thủ công và tối ưu hóa hiệu suất làm việc trong việc quản lý tài liệu.

Hiện nay, có rất nhiều mẫu hóa đơn chúng đều không theo một quy chuẩn cụ thể nào cả mỗi cửa hàng, doanh nghiệp lại có một loại hóa đơn riêng nên trong đề tài này em sẽ sử dụng hai loại hóa đơn chính để thực hiện đề tài này, mẫu hóa đơn của cửa hàng tiện lợi Okono và VinCom để thực hiện trích xuất các thông tin như tên cửa hàng, địa chỉ, thời gian mua hàng, tên nhân viên, sản phẩm đã mua, tổng tiền hàng,... để thực hiện đề tài này.

### 1.5 Bố cục chương

#### **Chương 1: Tổng quan về đề tài**

Chương này giới thiệu sơ lược qua sự hình thành và phát triển của hóa đơn, nêu qua vấn đề của hóa đơn trong cuộc sống hiện tại và cách OCR ứng dụng vào để giải quyết các vấn đề đó. Trình bày mục tiêu, giải pháp và phạm vi nghiên cứu của đề tài

#### **Chương 2: Cơ sở lý thuyết liên quan**

Dưa ra các khái niệm cơ bản về OCR, Học sâu, lịch sử phát triển của và ứng dụng của Học sâu vào OCR, các kiến trúc tiêu biểu. Giới thiệu về các thuật toán OCR như phát hiện văn bản, nhận diện văn bản và nhận dạng cấu trúc tài liệu.

#### **Chương 3: Công cụ và môi trường thực hiện**

Trình bày các phương pháp thực hiện đề tài, giới thiệu về các công cụ và môi trường cần thiết, các bước cấu hình môi trường và cài đặt công cụ.

**Chương 4: Thu thập dữ liệu và Huấn luyện mô hình**

Mô tả loại hóa đơn, trình bày cách thu thập và chuẩn bị dữ liệu và huấn luyện mô hình, kết quả huấn luyện.

**Chương 5: Xây dựng chương trình và Kết quả**

Nêu ý tưởng và trình bày quá trình xây dựng cho bài đề tài. Trình bày quá trình xây dựng và kết quả của chương trình

## Chương 2

# Cơ sở lý thuyết liên quan

Với sự phát triển không ngừng của công nghệ, việc chuyển đổi thông tin từ tài liệu giấy trở thành dạng điện tử đã trở nên cực kỳ quan trọng đối với doanh nghiệp và cá nhân. Trước đây, việc nhập liệu thủ công từ hóa đơn và tài liệu tương tự tồn rất nhiều thời gian, công sức và có nguy cơ sai sót cao. Tuy nhiên, với sự xuất hiện của công nghệ OCR, quá trình này đã trở nên tự động và hiệu quả hơn, giúp tiết kiệm thời gian và tối ưu hóa quá trình làm việc.

Trong chương này, ta sẽ bắt đầu bằng việc tìm hiểu về cơ bản của công nghệ OCR, một công nghệ quan trọng đã thay đổi cách chúng ta xử lý và quản lý thông tin, khám phá cách OCR có thể phân tích hình ảnh, xác định ký tự và từ, và biến đổi chúng thành dạng văn bản có thể xử lý.

Qua đó, chương cơ sở lý thuyết này sẽ tạo nền tảng kiến thức quan trọng cho việc hiểu rõ hơn về công nghệ OCR và tầm quan trọng của nó trong việc tối ưu hóa quá trình nhận dạng hóa đơn.

### 2.1 Nhận dạng ký tự quang học

#### 2.1.1 Nhận dạng ký tự quang học là gì?

Nhận dạng ký tự quang học hay còn gọi là OCR đây là quá trình chuyển đổi một hình ảnh văn bản viết tay, đánh máy hoặc in thành định dạng văn bản mà máy có thể hiểu được. Nó được sử dụng rộng rãi để nhận dạng và tìm kiếm văn bản từ các tài liệu điện tử hoặc để xuất bản văn bản trên một trang

web. [2], [3]

OCR được sử dụng rộng rãi như một hình thức nhập dữ liệu từ các bản ghi dữ liệu giấy in - cho dù đó là tài liệu hộ chiếu, hóa đơn, sao kê ngân hàng, biên lai vi tính hóa, danh thiếp, thư, dữ liệu in hoặc bất kỳ tài liệu phù hợp nào - đó là một phương pháp phổ biến để số hóa các văn bản in sao cho chúng có thể được chỉnh sửa, tìm kiếm, lưu trữ bằng điện tử, hiển thị trực tuyến và được sử dụng trong các quy trình máy như điện toán nhận thức, dịch máy, chuyển văn bản thành giọng nói (trích xuất), dữ liệu chính và khai thác văn bản. OCR là một lĩnh vực nghiên cứu về nhận dạng mẫu, trí tuệ nhân tạo và thị giác máy tính.[4]

Nhận dạng ký tự quang học đã được áp dụng vào nhiều ứng dụng khác nhau. Dưới đây là một số ứng dụng của OCR: [3]

- **Nhận dạng chữ viết tay:** Máy tính để nhận và diễn dịch thông tin viết tay rõ ràng từ các nguồn như tài liệu giấy, ảnh, màn hình cảm ứng và các thiết bị khác. Hình ảnh văn bản viết có thể được cảm nhận "ngoại tuyến" từ tờ giấy thông qua quét quang học hoặc nhận dạng từ thông minh. Một cách khác, các chuyển động của đầu bút viết có thể được cảm nhận "trực tuyến", ví dụ như bề mặt màn hình máy tính dựa trên bút viết.
- **Ngân hàng:** Được sử dụng để xử lý séc mà không cần sự tham gia của con người. Một tờ séc có thể được đặt vào máy, trong đó hệ thống quét số tiền cần phát hành và số tiền chính xác sẽ được chuyển khoản. Công nghệ này đã gần như được hoàn thiện cho các séc được in ấn và cũng khá chính xác đối với các séc viết tay, giảm thiểu thời gian chờ đợi tại ngân hàng.
- **Chăm sóc sức khỏe:** Các chuyên gia y tế luôn phải đối mặt với số lượng lớn các biểu mẫu cho mỗi bệnh nhân, bao gồm cả biểu mẫu bảo hiểm cũng như các biểu mẫu sức khỏe chung. Để theo kịp với tất cả thông tin này, việc nhập dữ liệu liên quan vào một cơ sở dữ liệu điện tử có thể được truy cập khi cần thiết. Các công cụ xử lý biểu mẫu, được cung cấp bởi công nghệ OCR, có khả năng trích xuất thông tin từ các biểu mẫu và đưa

vào cơ sở dữ liệu, để mỗi dữ liệu bệnh nhân được ghi lại đúng thời điểm.

- **Captcha:** Trong CAPTCHA, một hình ảnh gồm các ký tự hoặc số được tạo ra, bị mờ đi bằng các kỹ thuật biến dạng hình ảnh, biến đổi kích thước và phông chữ, phông nền gây xao lâng, đoạn ngẫu nhiên, đánh dấu và nhiễu trong hình ảnh. Hệ thống này có thể được sử dụng để loại bỏ nhiễu và phân đoạn hình ảnh để làm cho hình ảnh dễ xử lý cho các hệ thống OCR
- **Ảnh hóa đơn:** Được sử dụng rộng rãi trong nhiều ứng dụng kinh doanh để theo dõi hồ sơ tài chính và ngăn chặn việc tích lũy các khoản thanh toán chồng chất.
- **Nhận dạng biển số xe:** Sử dụng để tự động nhận dạng và ghi nhận biển số xe trên các hình ảnh hoặc video.
- ...

Từ những ứng dụng trên ta có thể thấy rằng OCR đang được sử dụng rộng rãi trong cuộc sống hàng ngày, nó đang đóng vai trò quan trọng trong việc chuyển đổi số hiện nay. Điều này rất quan trọng để tối ưu hóa quá trình làm việc với thông tin trong thời đại công nghệ thông tin.

### 2.1.2 Lịch sử của OCR

OCR được ra đời và cuối thế kỷ 19, được cấp bằng sáng chế tại Mỹ vào ngày 31 tháng 12 năm 1935 của Gustav Tauschek đến từ Viên, Áo, đây là một trong những phát minh sớm nhất liên quan đến OCR. OCR ban đầu được sử dụng để số hóa các văn bản in và cho phép chúng có thể đọc được bằng máy. Khi công nghệ OCR tiếp tục phát triển, nó đã được sử dụng rộng rãi trong các ngành công nghiệp khác nhau.

Sự khởi đầu thực sự của những hệ thống OCR ban đầu thực sự bắt đầu vào những năm 1960 và 1970. Các hệ thống này được thiết kế cho các trường hợp sử dụng cụ thể, chẳng hạn như phân loại thư dựa trên mã zip hoặc đọc số viết tay. Phông chữ có thể đọc bằng máy quang học đầu tiên OCR-A được

phát triển vào năm 1968 bởi nhà thiết kế kiểu chữ người Thụy Sĩ Adrian Frutiger.

Trong suốt những năm 1980, công nghệ OCR đã đạt được những bước tiến đáng kể với sự phát triển của các thuật toán mới và các máy tính mạnh hơn. Các hệ thống OCR có thể nhận dạng nhiều loại phông chữ hơn và có thể xử lý các hình ảnh phức tạp hơn, khiến chúng trở nên chính xác và hữu ích hơn cho nhiều ứng dụng hơn.

Vào những năm 1990, việc sử dụng rộng rãi máy tính cá nhân và internet đã dẫn đến sự gia tăng đáng kể trong việc sử dụng công nghệ OCR. Các hệ thống OCR được sử dụng để số hóa sách, tạp chí và các tài liệu in khác, giúp tìm kiếm và truy cập thông tin dễ dàng hơn. Công nghệ này cũng được sử dụng để tự động hóa các quy trình nhập dữ liệu trong các ngành như tài chính, chăm sóc sức khỏe và chính phủ.

Vào đầu những năm 2000, lịch sử của công nghệ OCR đã phát triển với việc giới thiệu các thuật toán mới và phần cứng được cải tiến. Các hệ thống OCR trở nên chính xác hơn và có thể nhận dạng nhiều loại ký tự và ngôn ngữ hơn. Điều này đã mở đường cho việc áp dụng rộng rãi công nghệ OCR trong nhiều ngành và ứng dụng khác nhau, chẳng hạn như quản lý tài liệu và xử lý hóa đơn. Trong khung thời gian này, Google cũng nổi tiếng (và gây tranh cãi) đã ra mắt Google Sách, có tên mã là Dự án Đại dương, sử dụng OCR để số hóa hàng chục triệu cuốn sách và làm cho văn bản của chúng có thể tìm kiếm được.

Ngày nay, công nghệ OCR tiên tiến và phức tạp hơn bao giờ hết. Các hệ thống OCR có thể nhận dạng nhiều loại ký tự và ngôn ngữ, chữ viết tay và các hình ảnh phức tạp khác. Công nghệ OCR đang tiếp tục phát triển và những tiến bộ mới nhất về trí tuệ nhân tạo và máy học đang dẫn đến các hệ thống thậm chí còn phức tạp và chính xác hơn.

Lịch sử OCR bắt đầu với những phát minh mang tính cách mạng được thiết kế để cải thiện chất lượng cuộc sống cho nhân loại. Nhiều thập kỷ sau, công nghệ này vẫn đang trải qua quá trình phát triển và cải tiến liên tục, đồng thời là một yếu tố quyết định quan trọng của thời đại kỹ thuật số. OCR

đã trải qua một chặng đường dài và đang thực sự cải thiện chất lượng cuộc sống của phần lớn nhân loại. Ngày nay, nhiều ngành công nghiệp và ứng dụng sử dụng OCR. Trong những thập kỷ tới, nó sẽ đóng một vai trò quan trọng trong quá trình chuyển đổi kỹ thuật số toàn cầu.<sup>[5]</sup>

### 2.1.3 Nguyên tắc hoạt động của OCR

OCR hoạt động bằng cách phân tích hình ảnh văn bản và sau đó tạo ra một bản sao văn bản kỹ thuật số của hình ảnh đó. Quá trình này thường được thực hiện theo các bước sau:

- **Quét tài liệu:** Tài liệu được quét bằng máy quét để tạo ra một hình ảnh kỹ thuật số của tài liệu.
- **Phân tích và xử lý hình ảnh:** Trước khi nhận dạng văn bản, ảnh được tiền xử lý để làm sạch và cải thiện chất lượng. Điều này có thể bao gồm việc điều chỉnh độ tương phản, loại bỏ nhiễu, cắt biên và xoay ảnh để đảm bảo văn bản nằm ngang. Sau đó hình ảnh được phân tích để xác định các vùng văn bản.
- **Nhận dạng ký tự:** Trong bước này, hình ảnh được chuyển đổi thành dạng dữ liệu văn bản bằng cách nhận dạng các ký tự riêng lẻ. Các thuật toán và mô hình máy học được sử dụng để so khớp các đặc trưng trong hình ảnh với các ký tự đã biết từ bộ dữ liệu huấn luyện.
- **Phân tích cấu trúc:** Sau khi xác định được các ký tự, công cụ OCR cũng cố gắng xác định cấu trúc của văn bản, bao gồm việc xác định các đoạn, đoạn văn bản, tiêu đề, danh sách và các yếu tố cấu trúc khác.
- **Sửa lỗi và kiểm tra:** Sau khi nhận dạng, dữ liệu văn bản thường cần được kiểm tra lại và sửa lỗi do các lỗi nhận dạng có thể xảy ra. Điều này có thể thực hiện tự động hoặc thông qua giao diện người dùng để đảm bảo tính chính xác của kết quả.
- **Tạo bản sao văn bản kỹ thuật số:** Một bản sao văn bản kỹ thuật số của hình ảnh được tạo ra bằng cách kết hợp các ký tự đã được nhận dạng.

Các công nghệ OCR ngày càng phát triển, sử dụng các mô hình học sâu và học máy để cải thiện khả năng nhận dạng và xử lý ngôn ngữ tự nhiên, tạo ra kết quả chính xác hơn và phức tạp hơn.

## 2.2 Các thuật toán OCR

Mặc dù OCR tương đối cũ kỹ, nhưng nó liên quan đến nhiều khía cạnh của công nghệ, bao gồm phát hiện văn bản, nhận dạng văn bản, nhận dạng văn bản từ đầu đến cuối, phân tích tài liệu, v.v. Nghiên cứu học thuật về các công nghệ liên quan của OCR phát triển mạnh mẽ. Phần này đây sẽ giới thiệu sơ lược về một số công nghệ chính trong tác vụ OCR.

### 2.2.1 Phát hiện văn bản

Công việc phát hiện văn bản là để xác định vùng chứa văn bản trên ảnh đầu vào. Trong những năm gần đây, có nhiều nghiên cứu học thuật về phát hiện văn bản. Một lớp phương pháp coi việc phát hiện văn bản như một tình huống cụ thể trong việc phát hiện mục tiêu, và điều chỉnh các thuật toán phát hiện mục tiêu chung để phù hợp với việc phát hiện văn bản. Ví dụ, TextBoxes dựa trên một bộ phát hiện mục tiêu một giai đoạn là SSD. Thuật toán điều chỉnh khung mục tiêu để vừa với các dòng văn bản có tỷ lệ khía cạnh cực đoan, trong khi CTPN được phát triển từ Faster RCNN. Tuy nhiên, vẫn có một số khác biệt giữa phát hiện văn bản và phát hiện mục tiêu về thông tin mục tiêu và nhiệm vụ chính. Ví dụ, văn bản thường dài và trông giống "vạch", khoảng cách giữa các dòng nhỏ, văn bản có thể uốn cong, v.v. Do đó, nhiều thuật toán đặc biệt cho việc phát hiện văn bản đã được phát triển, như EAST, PSENet, DBNet, và nhiều thuật toán khác [6].

Hiện tại, một số thuật toán phát hiện văn bản phổ biến có thể được chia ra một cách đại khái thành hai loại: **Thuật toán dựa trên Hồi quy** và **Thuật toán dựa trên Phân đoạn**. Cũng có một số thuật toán kết hợp cả hai loại này. Các thuật toán dựa trên hồi quy lấy cảm hứng từ các thuật toán phát hiện đối tượng chung, thực hiện việc hồi quy hộp phát hiện bằng cách đặt các anchor, hoặc thậm chí trực tiếp thực hiện hồi quy điểm ảnh. Loại

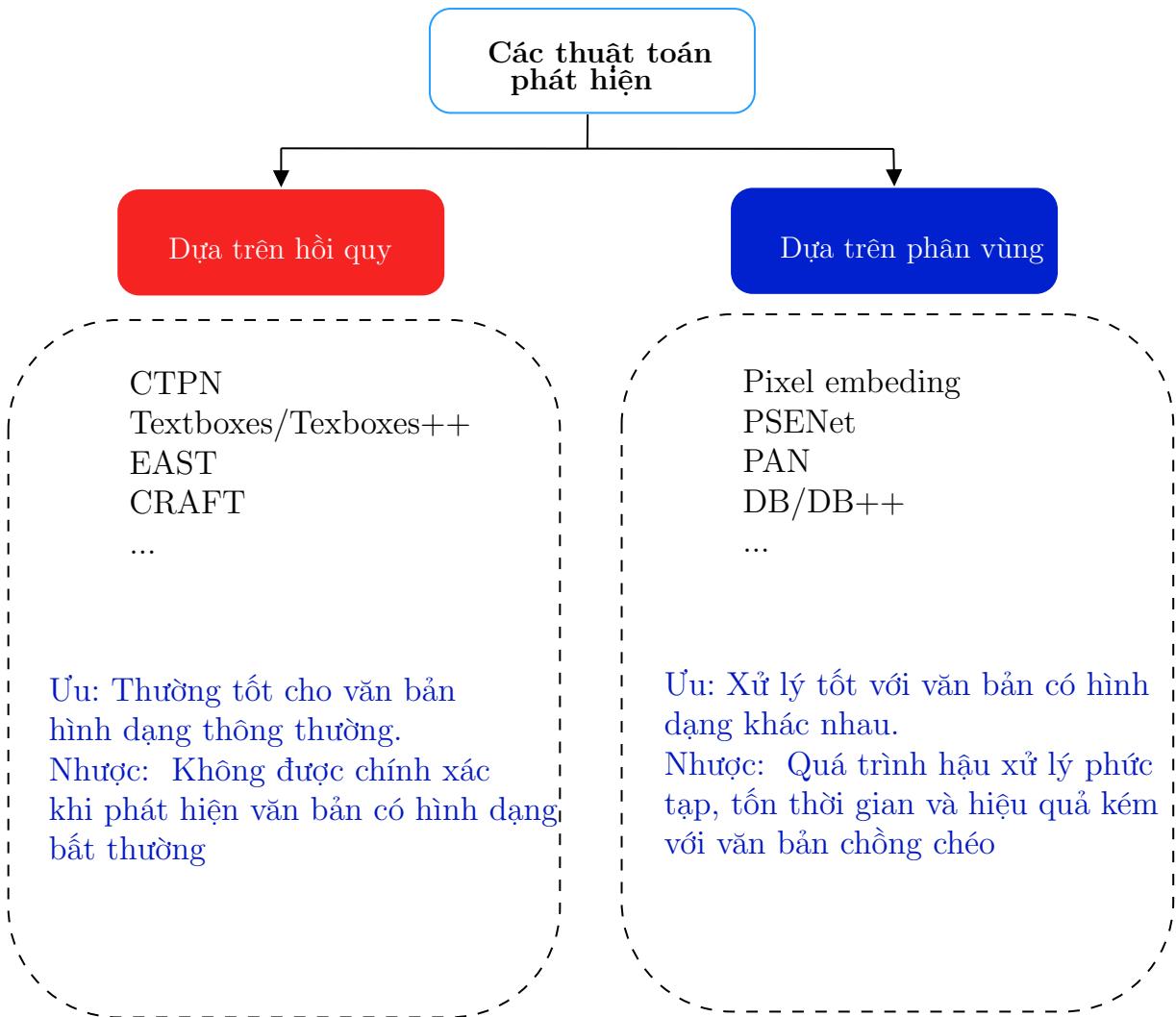


Hình 2.1: Ví dụ về nhiệm vụ phát hiện văn bản

phương pháp này hoạt động tốt trong việc phân biệt văn bản có hình dạng đều đặn, nhưng kém trong việc phát hiện văn bản có hình dạng không đều. Ví dụ, CTPN tốt trong việc nhận dạng văn bản ngang, nhưng kém trong việc phát hiện văn bản uốn cong và xoắn. SegLink phù hợp hơn với văn bản dài, nhưng không thích hợp cho việc phát hiện văn bản phân tán thưa thớt. Các thuật toán dựa trên phân đoạn giới thiệu Mask-RCNN, loại thuật toán này có thể hoạt động tốt hơn trong việc phát hiện trong các tình huống và văn bản có các hình dạng khác nhau, nhưng hạn chế là việc xử lý sau cùng phức tạp, vì vậy có thể chậm về tốc độ và không thể phát hiện được văn bản chồng lấn [6].

### 2.2.2 Nhận dạng văn bản

Nhận dạng văn bản là việc nhận biết nội dung văn bản trong hình ảnh, và đầu vào thường là từ phần vùng chứa văn bản của ảnh được cắt ra bằng hộp văn bản được tạo ra từ việc phát hiện văn bản. Nhận dạng văn bản có thể được chia thành hai loại chính: **Nhận dạng Văn bản Đầu đặn** và **Nhận dạng Văn bản Không đều đặn** dựa trên đường viền của văn bản cần nhận dạng.



Hình 2.2: Tổng quan về thuật toán phát hiện văn bản

Văn bản đều đặn chủ yếu đề cập đến các phông chữ in, văn bản được quét, và các nguồn tương tự có hướng chính đều. Văn bản không đều đặn thường không nằm trong tư thế ngang, thường uốn cong, bị che khuất và mờ mờ. Các tình huống văn bản không đều đặn thách thức rất lớn, và đó cũng là hướng nghiên cứu chính trong việc nhận dạng văn bản.

Các thuật toán nhận dạng văn bản đều đặn có thể được chia thành hai loại dựa trên các phương pháp giải mã khác nhau: Thuật toán dựa trên CTC và Thuật toán dựa trên Sequence2Sequence. Chúng khác nhau trong cách chuyển đổi các đặc trưng chuỗi mà mạng học học được thành kết quả nhận dạng cuối cùng. Một ví dụ đại diện cho thuật toán dựa trên CTC là CRNN cổ điển.

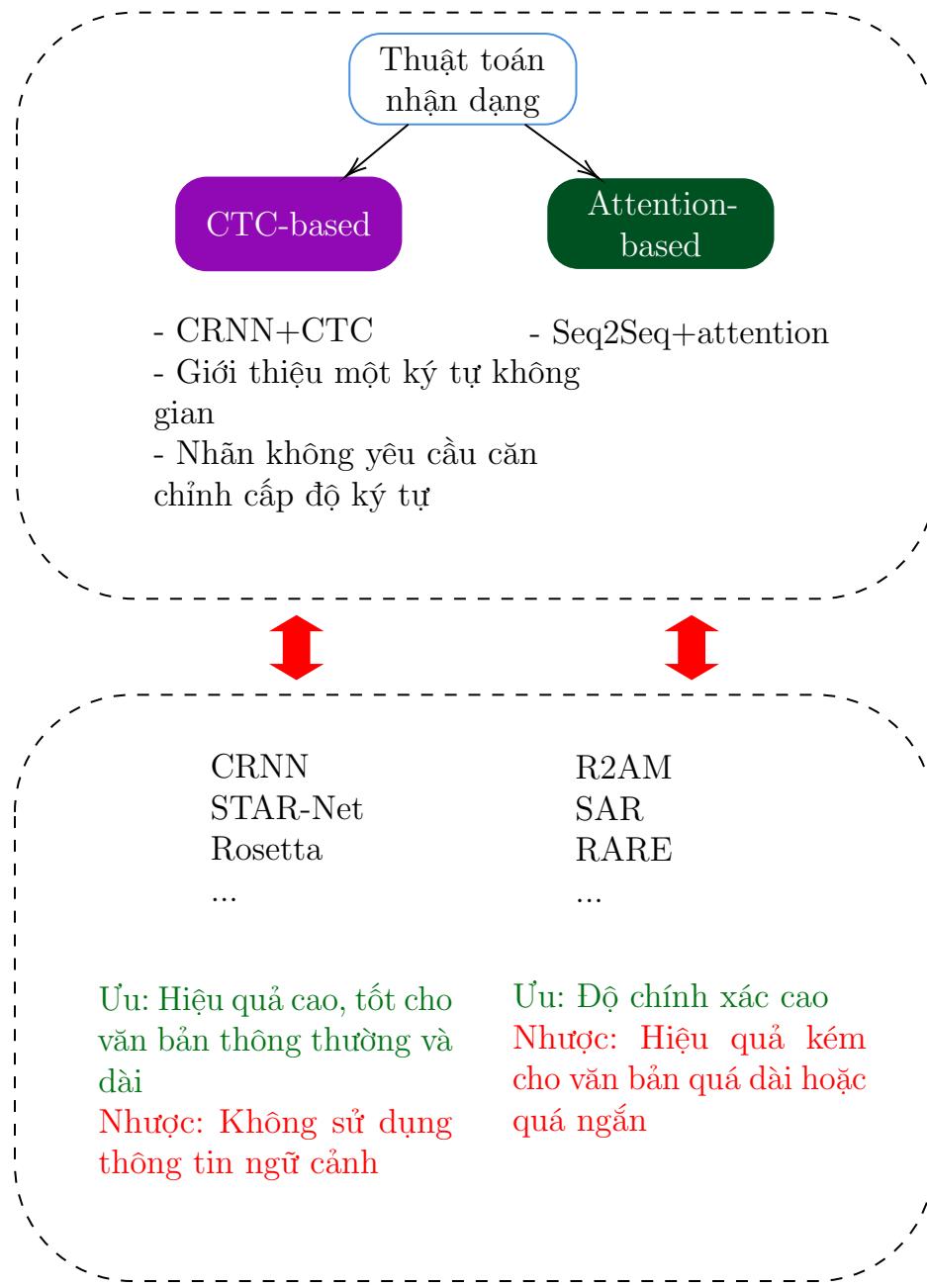


Hình 2.3: Văn bản đều đặn (trái) và Văn bản không đều đặn (phải)

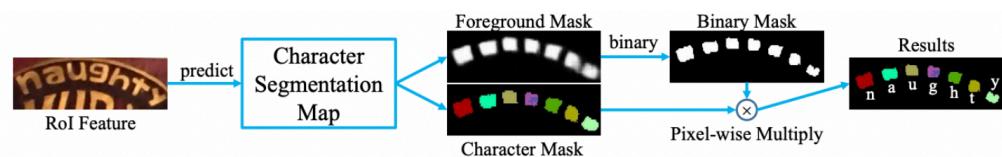
Các thuật toán nhận dạng cho văn bản không đều đặn phong phú hơn. Các phương pháp như STAR-Net sửa chữa đường viền của văn bản không đều đặn thành các hình chữ nhật đều đặn bằng cách thêm các mô-đun sửa chữa như TPS trước khi thực hiện việc nhận dạng. Các phương pháp dựa trên chú ý như RARE chú trọng hơn đến mối quan hệ giữa các phần trong chuỗi. Các phương pháp dựa trên phân đoạn xử lý mỗi ký tự trên dòng văn bản như một đơn vị riêng lẻ, làm cho việc nhận dạng ký tự đã phân đoạn dễ dàng hơn so với việc nhận dạng toàn bộ dòng văn bản sau khi sửa chữa. Ngoài ra, với sự phát triển nhanh chóng của Transformer và hiệu quả đã được xác minh trong các nhiệm vụ khác nhau trong những năm gần đây, nhiều thuật toán nhận dạng văn bản dựa trên transformer đã phát triển mạnh mẽ. Loại giải pháp này sử dụng cấu trúc transformer để giải quyết việc mô hình hóa sự phụ thuộc lâu dài trong CNN và đã đạt được kết quả tốt.

### 2.2.3 Nhận dạng cấu trúc tài liệu

Công nghệ OCR có thể đáp ứng yêu cầu về phát hiện và nhận dạng văn bản. Tuy nhiên, trong các tình huống thực tế, điều chúng ta thường cần là thông tin có cấu trúc, chẳng hạn như trích xuất thông tin từ thẻ ID và hóa đơn, xác định có cấu trúc của bảng, và vân vân. Các tình huống ứng dụng của công nghệ OCR chủ yếu là trích xuất tài liệu nhanh, so sánh nội dung hợp đồng, so sánh thông tin tài chính trên các tài liệu cần thanh toán, và xác định tài liệu vận chuyển. Kết quả OCR + xử lý sau cùng là một kế hoạch cấu trúc thường được sử dụng, nhưng phức tạp và cần thiết phải được thiết kế cẩn thận, và thiếu sự tổng quát. Với sự phát triển liên tục của công nghệ OCR và nhu cầu về trích xuất thông tin có cấu trúc đang gia tăng, các công



Hình 2.4: CTC-based recognition algorithm VS. Attention-based recognition algorithm



Hình 2.5: Thuật toán nhận dạng dựa trên phân vùng ký tự

nghệ liên quan đến phân tích tài liệu thông minh, như phân tích bô cục, nhận dạng bảng, và trích xuất thông tin quan trọng, đã nhận được sự chú ý ngày càng tăng.

### **Phân tích bô cục**

Phân tích bô cục được thực hiện để phân loại nội dung của hình ảnh tài liệu thành các loại như văn bản thuần túy, tiêu đề, bảng biểu, hình ảnh, v.v. Các phương pháp hiện tại thường thực hiện việc phát hiện hoặc phân đoạn chúng một cách riêng biệt. Ví dụ, Soto Carlos sử dụng thông tin ngữ cảnh và vị trí tự nhiên của nội dung tài liệu để cải thiện hiệu suất phát hiện vùng dựa trên thuật toán phát hiện mục tiêu Faster R-CNN. Sarkar Mausoom và đồng nghiệp đề xuất một cơ chế phân đoạn dựa trên tiên biết để huấn luyện mô hình phân đoạn tài liệu với các hình ảnh có độ phân giải cao, giải quyết vấn đề rằng các cấu trúc khác nhau trong các khu vực dày đặc không thể phân biệt và hợp nhất do việc giảm quá mức của hình ảnh gốc.

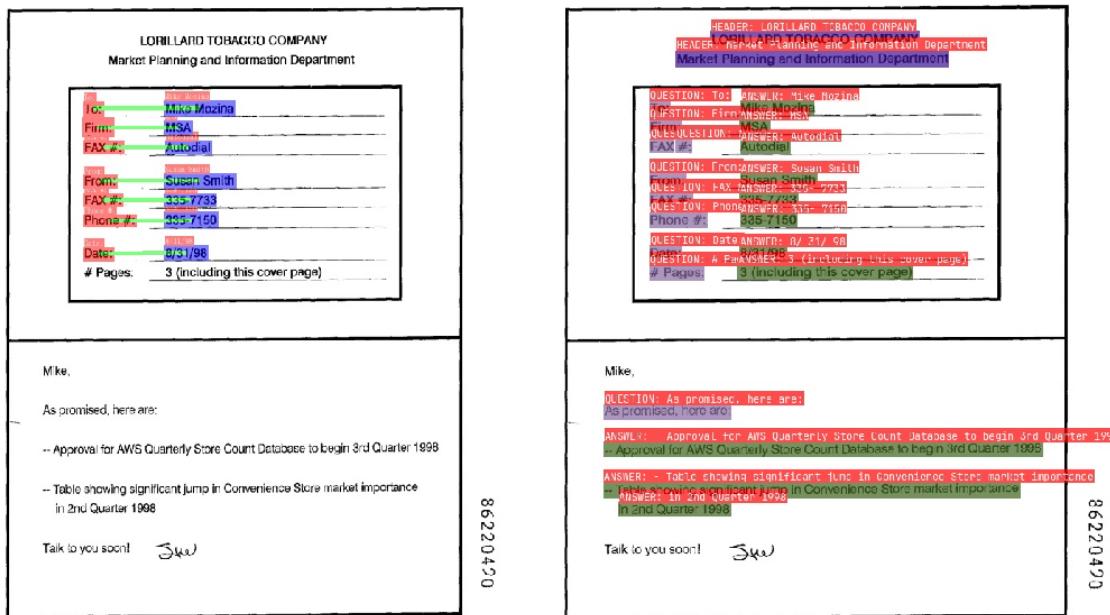
### **Nhận dạng bảng**

Nhận dạng bảng là việc xác định và chuyển thông tin bảng của tài liệu thành một tệp Excel. Có nhiều loại và phong cách bảng khác nhau trong hình ảnh văn bản, chẳng hạn như các hàng và cột khác nhau và các loại văn bản khác nhau. Ngoài ra, phong cách của tài liệu và môi trường ánh sáng khi chụp ảnh đã đặt ra những thách thức lớn cho việc nhận dạng bảng, làm cho việc nhận dạng bảng trở thành một vấn đề nghiên cứu khó khăn trong việc hiểu tài liệu. Có nhiều phương pháp nhận dạng bảng. Ví dụ, vào những ngày đầu tiên, có các thuật toán truyền thống dựa trên các quy tắc heuristics, như thuật toán T-Rect được đề xuất bởi Kieninger và cộng sự, thường sử dụng quy tắc thiết kế thủ công và phát hiện và phân tích miền kết nối. Trong những năm gần đây, khi học sâu tiếp tục phát triển, một số thuật toán nhận dạng cấu trúc bảng dựa trên mạng CNN đã xuất hiện, như DeepTabStR được đề xuất bởi Siddiqui Shoaib Ahmed và cộng sự và TabStruct-Net được đề xuất bởi Raja Sachin và cộng sự. Ngoài ra, với sự gia tăng của Mạng Neural Đò thị,

một số nhà nghiên cứu đã thử áp dụng Mạng Neural Đồ thị vào việc nhận dạng cấu trúc bảng và coi việc nhận dạng bảng như một vấn đề tái tạo đồ thị dựa trên Mạng Neural Đồ thị. Đây là cách mà TGRNet được đề xuất bởi Xue Wenyuan và cộng sự hoạt động. Hơn nữa, có các giải pháp end-to-end có kết quả đầu ra cấu trúc bảng dưới dạng HTML bằng mạng. Hầu hết trong số này áp dụng Seq2Seq để dự đoán cấu trúc bảng như những thuật toán dựa trên attention hoặc transformer, bao gồm TableMaster.

### Trích xuất thông tin chính

Trích xuất thông tin quan trọng (KIE) là một nhiệm vụ quan trọng trong Hỏi và Trả lời Văn bản (Document VQA). Nó liên quan đến việc trích xuất thông tin cần thiết từ hình ảnh, chẳng hạn như tên và số ID từ thẻ ID. Thông tin như vậy thường được xác định trong một nhiệm vụ, nhưng khác nhau giữa các nhiệm vụ khác nhau.



Hình 2.6: RE(trái) VS SER(phải)

KIE thường được chia thành hai phần nhiệm vụ con để nghiên cứu (Hình 2.6):

- SER: Đây là việc nhận dạng thực thể ngữ nghĩa, phân loại từng đoạn văn

bản được phát hiện. Ví dụ, nó chia văn bản thành tên và số thẻ ID như hình dưới đây.

- RE: Đây là việc trích xuất mối quan hệ, phân loại từng đoạn văn bản. Ví dụ, nó có thể phân loại văn bản thành câu hỏi và câu trả lời, sau đó tìm câu trả lời tương ứng cho mỗi câu hỏi. Như hình dưới đây, các hộp đỏ và đen đại diện cho câu hỏi và câu trả lời tương ứng, và các mũi tên màu vàng chỉ sự tương ứng giữa câu hỏi và câu trả lời.

Phương pháp KIE thông thường được phát triển dựa trên nhận dạng thực thể đặt tên (NER), nhưng loại phương pháp này chỉ sử dụng thông tin văn bản trong hình ảnh mà không sử dụng thông tin hình ảnh và cấu trúc. Do đó, nó không đạt độ chính xác cao. Trong những năm gần đây, nhiều giải pháp đã bắt đầu kết hợp thông tin hình ảnh và cấu trúc với thông tin văn bản. Do sử dụng các nguyên tắc khác nhau trong việc kết hợp thông tin đa tầng, các phương pháp này có thể được chia thành bốn loại:

- Phương pháp dựa trên lưới
- Phương pháp dựa trên token
- Phương pháp dựa trên Graph Convolutional Network
- Phương pháp end-to-end

## 2.3 OCR dựa trên mẫu và OCR dựa trên AI

Với số lượng ngày càng tăng của các giải pháp OCR có sẵn trên thị trường, điều cần thiết là phải hiểu các kỹ thuật chính được sử dụng bởi các công cụ này, cụ thể là OCR dựa trên mẫu và OCR dựa trên AI.

### 2.3.1 OCR dựa trên mẫu

OCR dựa trên mẫu là một cách tiếp cận cũ hơn, truyền thống hơn đối với OCR dựa trên các mẫu được xác định trước để nhận dạng và trích xuất văn bản từ tài liệu. Phương pháp này hoạt động tốt nhất khi xử lý các tài liệu

có cấu trúc, chẳng hạn như hóa đơn, biểu mẫu hoặc biên lai, có bố cục nhất quán. [7]

### **Ưu điểm:**

1. Độ chính xác cao: Khi được sử dụng với các tài liệu có cấu trúc tuân theo định dạng nhất định, OCR dựa trên mẫu có thể đạt được tỷ lệ chính xác tuyệt vời.
2. Thời gian xử lý thấp hơn: Vì OCR dựa trên mẫu dựa trên các mẫu đựng sẵn, nó không yêu cầu nhiều sức mạnh xử lý hoặc thời gian để nhận dạng ký tự.
3. Có thể tùy chỉnh: Người dùng có thể tạo các mẫu tùy chỉnh cho các loại tài liệu cụ thể, đảm bảo kết quả chính xác.

### **Nhược điểm**

1. Tính linh hoạt hạn chế: OCR dựa trên mẫu gặp khó khăn khi xử lý các tài liệu không tuân theo định dạng nhất quán hoặc có bố cục phức tạp.
2. Thiết lập tốn thời gian: Tạo các mẫu tùy chỉnh có thể là một quá trình tẻ nhạt, đặc biệt là khi xử lý nhiều loại tài liệu.
3. Không hiệu quả đối với văn bản viết tay: OCR dựa trên mẫu thường hoạt động kém khi xử lý văn bản viết tay hoặc phông chữ.

Các trường hợp sử dụng tốt nhất cho OCR dựa trên mẫu: Cách tiếp cận này phù hợp nhất với các tổ chức xử lý khối lượng lớn tài liệu có cấu trúc với bố cục nhất quán, chẳng hạn như hóa đơn hoặc biểu mẫu.

#### **2.3.2 OCR Dựa trên AI**

OCR dựa trên AI tận dụng trí tuệ nhân tạo, máy học và mạng thần kinh để nhận dạng và trích xuất văn bản từ tài liệu. Cách tiếp cận này nâng cao hơn và có thể xử lý nhiều loại tài liệu, bao gồm cả tài liệu phi cấu trúc và bán cấu trúc. [7]

### **Ưu điểm:**

1. Độ chính xác và tính linh hoạt cao: OCR dựa trên AI có thể thích ứng với nhiều loại tài liệu và bố cục khác nhau, mang lại kết quả chính xác ngay cả khi xử lý các bố cục phức tạp hoặc không nhất quán.
2. Xử lý văn bản viết tay: OCR dựa trên AI được trang bị tốt hơn để nhận dạng và trích xuất phông chữ văn bản viết tay hoặc tập lệnh, làm cho nó linh hoạt hơn so với OCR dựa trên mẫu.
3. Cải tiến liên tục: Các thuật toán máy học cho phép OCR dựa trên AI cải thiện độ chính xác của nó theo thời gian khi nó xử lý nhiều tài liệu hơn.
4. Thiết lập nhanh hơn: OCR dựa trên AI không yêu cầu tạo mẫu tùy chỉnh, cho phép triển khai nhanh hơn.

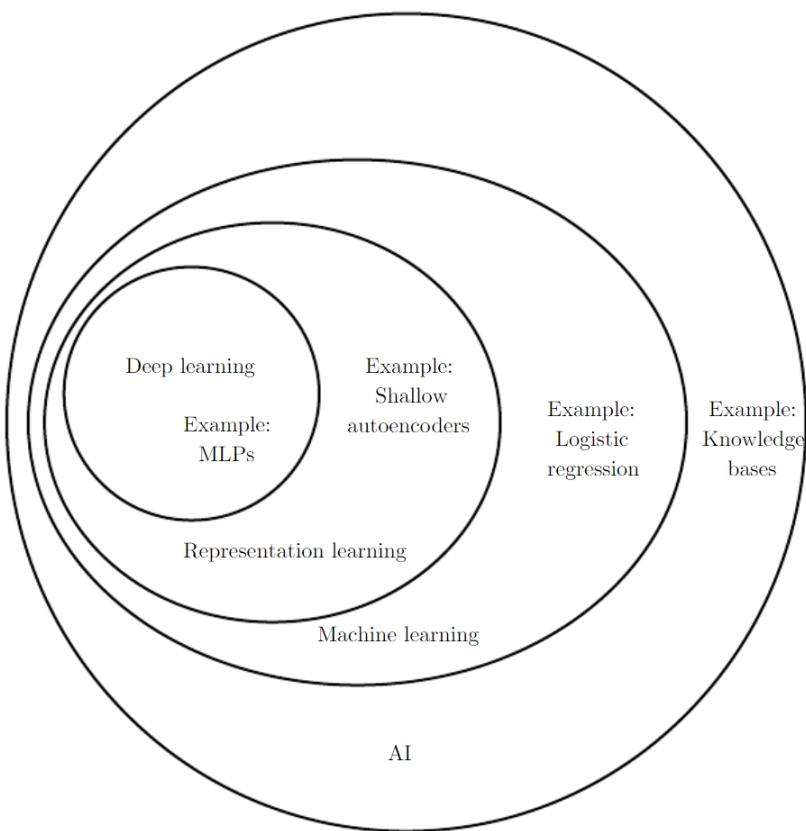
### **Nhược điểm**

1. Thời gian xử lý cao hơn: OCR dựa trên AI thường yêu cầu nhiều thời gian và sức mạnh xử lý hơn so với OCR dựa trên mẫu, vì nó phân tích tài liệu toàn diện hơn.
2. Chi phí cao hơn: Các giải pháp OCR dựa trên AI có thể đắt hơn do công nghệ tiên tiến và sự phát triển liên tục có liên quan.

Các trường hợp sử dụng tốt nhất cho OCR dựa trên AI: Phương pháp này lý tưởng cho các tổ chức xử lý nhiều loại tài liệu, bao gồm cả tài liệu phi cấu trúc và bán cấu trúc hoặc yêu cầu trích xuất văn bản viết tay

## **2.4 Học sâu**

Trong việc ứng dụng OCR để nhận dạng hóa đơn, mạng học sâu đã chơi một vai trò quan trọng và mang lại những cải tiến đáng kể cho quá trình này. Trước khi sự xuất hiện của học sâu, các hệ thống nhận dạng dựa trên các phương pháp truyền thống thường gặp khó khăn trong việc xử lý các biến thể phức tạp của hình ảnh hóa đơn và khả năng xử lý đa dạng của chúng. Nhưng



Hình 2.7: Biểu đồ Venn về Trí tuệ nhân tạo

với mạng nơron học sâu, khả năng học và tự điều chỉnh của mô hình đã mở ra những cánh cửa mới cho việc nhận dạng hóa đơn hiệu quả hơn.

#### 2.4.1 Học sâu là gì?

Học sâu là một các tiếp cận của Trí tuệ nhân tạo. Cụ thể thì nó là một kiểu của học máy (Hình 2.7), một kỹ thuật mà cho phép hệ thống máy tính tự học từ trải nghiệm và dữ liệu, nó sở hữu sức mạnh và sự linh hoạt tuyệt vời thông qua việc học cách biểu diễn như một hệ phân cấp khái niệm trong đó mỗi khái niệm được định nghĩa từ những khái niệm đơn giản hơn, và mỗi biểu diễn được tính toán từ những biểu diễn kém trừu tượng hơn. [8]

Một điểm đáng chú ý là học sâu cần một lượng lớn dữ liệu để huấn luyện mô hình một cách hiệu quả [9]. Trong trường hợp OCR và nhận dạng hóa đơn, mạng nơron học sâu có khả năng học từ hàng nghìn hoặc thậm chí hàng triệu hình ảnh hóa đơn, điều này giúp mô hình hiểu rõ các đặc trưng và biểu diễn

của dữ liệu hơn.

Hiện nay các kiến trúc học sâu như mạng nơ-ron sâu, mạng niềm tin sâu, học tăng cường sâu, mạng nơ-ron tái phát, mạng nơ-ron tích chập và máy biến áp đã được áp dụng cho các lĩnh vực bao gồm thị giác máy tính, nhận dạng giọng nói, xử lý ngôn ngữ tự nhiên, dịch máy, tin sinh học, thiết kế thuốc, Phân tích hình ảnh y tế, khoa học khí hậu, kiểm tra vật liệu và các chương trình trò chơi trên bàn cờ, nơi chúng đã tạo ra kết quả tương đương và trong một số trường hợp vượt qua hiệu suất chuyên gia của con người.

#### 2.4.2 Kiến trúc mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo là một mô hình được lấy cảm hứng từ mạng nơ-ron thần kinh. Kết hợp với các kỹ thuật học sâu, nó đang trở thành một công cụ rất mạnh mẽ mang lại hiệu quả tốt nhất cho nhiều bài toán khó như nhận dạng ảnh, giọng nói hay xử lý ngôn ngữ tự nhiên. Một mạng nơ-ron được cấu thành bởi các nơ-ron đơn lẻ được gọi là các perceptron.

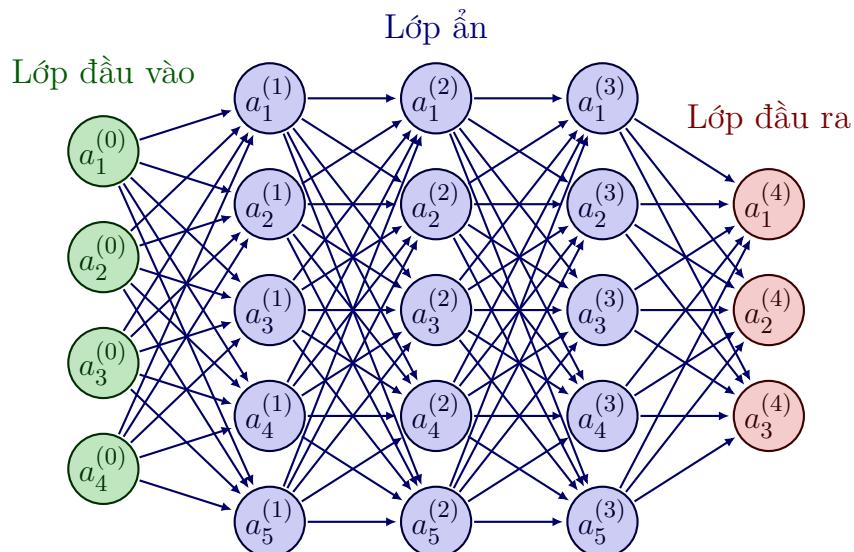
Một perceptron sẽ nhận một hoặc nhiều đầu  $x$  vào dạng nhị phân và cho ra một kết quả  $o$  dạng nhị phân duy nhất. Các đầu vào được điều phối tầm ảnh hưởng bởi các tham số trọng lượng tương ứng  $w$  của nó, còn kết quả đầu ra được quyết định dựa vào một ngưỡng quyết định  $b$  nào đó:

$$o = \begin{cases} 0 & \text{if } \sum_i w_i x_i + b \leq 0 \\ 1 & \text{if } \sum_i w_i x_i + b \geq 0 \end{cases}$$

Mạng nơ-ron là sự kết hợp của các tầng perceptron hay còn được gọi là perceptron đa tầng như hình vẽ bên dưới:

Một mạng NN sẽ có 3 kiểu tầng [10]:

- **Lớp đầu vào:** Một mạng nơ-ron nhân tạo sẽ có một số nút để nhập dữ liệu đầu vào. Các nút này tạo nên lớp đầu vào của hệ thống.
- **Lớp ẩn:** Lớp đầu vào xử lý và chuyển dữ liệu đến các lớp sâu hơn trong mạng nơ-ron. Các lớp ẩn này xử lý thông tin ở các cấp độ khác nhau, thích ứng với hành vi của mình khi nhận được thông tin mới. Các mạng



Hình 2.8: Kiến trúc một mạng Nơ-ron

học sâu có hàng trăm lớp ẩn có thể được dùng để phân tích một vấn đề từ nhiều góc độ khác nhau.

- **Lớp đầu ra:** Lớp đầu ra bao gồm các nút xuất dữ liệu. Các mô hình học sâu xuất ra đáp án "có" hoặc "không" chỉ có hai nút trong lớp đầu ra. Mặt khác, các mô hình xuất ra nhiều đáp án hơn sẽ có nhiều nút hơn.

Trong mạng Nơ-ron, mỗi nút mạng là một sigmoid nơ-ron nhưng hàm kích hoạt của chúng có thể khác nhau. Tuy nhiên trong thực tế người ta thường để chúng cùng dạng với nhau để tính toán cho thuận lợi.

Ở mỗi tầng, số lượng các nút mạng (nơ-ron) có thể khác nhau tùy thuộc vào bài toán và cách giải quyết. Nhưng thường khi làm việc người ta để các tầng ẩn có số lượng nơ-ron bằng nhau. Ngoài ra, các nơ-ron ở các tầng thường được liên kết đôi một với nhau tạo thành mạng kết nối đầy đủ. Khi đó ta có thể tính được kích cỡ của mạng dựa vào số tầng và số nơ-ron.

#### 2.4.3 ResNet

ResNet, viết tắt của “Residual Network”, là một kiến trúc mạng nơ-ron sâu được giới thiệu bởi Kaiming He và đồng nghiệp vào năm 2015. ResNet đã đạt được thành công lớn trong việc giải quyết vấn đề đào tạo mạng nơ-ron sâu với

số lượng lớp ngày càng tăng mà không gặp vấn đề về độ sâu. Kiến trúc của ResNet đã giúp giải quyết vấn đề triệt tiêu gradient và giúp việc huấn luyện mạng nơ-ron sâu trở nên dễ dàng hơn.

Một vấn đề thường gặp khi huấn luyện mạng nơ-ron sâu là hiện tượng triệt tiêu độ dốc (vanishing gradient) và bùng nổ độ dốc(exploding gradient), đặc biệt là khi mạng có nhiều lớp. Điều này dẫn đến khó khăn trong việc lan truyền ngược độ dốc trong quá trình học. ResNet giải quyết vấn đề này bằng cách sử dụng khối “residual”.

## Identity Mapping

Identity mapping hay còn gọi là ánh xạ đồng nhất là một khái niệm quan trọng trong toán học và xử lý tín hiệu, đặc biệt là trong lĩnh vực của mạng nơ-ron và học sâu. Đơn giản, ánh xạ đồng nhất đề cập đến việc ánh xạ một giá trị đầu vào sang giá trị đầu ra mà không thay đổi giá trị hoặc cấu trúc của dữ liệu. Trong các mô hình học sâu, việc sử dụng ánh xạ đồng nhất có thể giúp cho việc huấn luyện mô hình trở nên hiệu quả hơn.

Khi áp dụng trong mạng nơ-ron, ánh xạ đồng nhất thường được sử dụng như là một thành phần của các khối residual trong kiến trúc ResNet. Trong ResNet cho phép thông tin từ tầng trước được truyền thẳng qua mà không qua bất kỳ biến đổi nào, thông qua phép toán “identity shortcut”. Điều này giúp cho việc huấn luyện mô hình dễ dàng hơn và cải thiện khả năng học của mạng, đặc biệt là khi mạng trở nên rất sâu [11].

## Residual learning

Phương pháp tiếp cận này được gọi là “Residual Learning” được sử dụng trong việc đào tạo mạng nơ-ron sâu . Thay vì cố gắng trực tiếp xấp xỉ một phép ánh xạ mong muốn  $\mathcal{H}(x)$  bằng cách sử dụng các lớp xếp chồng lên nhau, Mô hình đề xuất một phương pháp là xấp xỉ một hàm residual  $\mathcal{F}(x)$  là hiệu của  $\mathcal{H}(x)$  và đầu vào  $x$ . Điều này có nghĩa là phép ánh xạ ban đầu  $\mathcal{H}(x)$  có thể được xây dựng lại như  $\mathcal{F}(x) + x$  [12].

Trong [12] tác giả lập luận rằng vấn đề suy giảm trong mạng sâu, khi mô

hình sâu hơn hoạt động kém hơn so với các mô hình nông, có thể do khó khăn trong việc xấp xỉ các phép ánh xạ đồng nhất(identity mapping) bằng nhiều lớp phi tuyến. Trong trường hợp các lớp được thêm vào hoạt động như các phép ánh xạ đồng nhất, một mạng sâu hơn nên hoạt động ít nhất cũng tốt như một mạng cạn. Tuy nhiên, điều này thường không được quan sát.

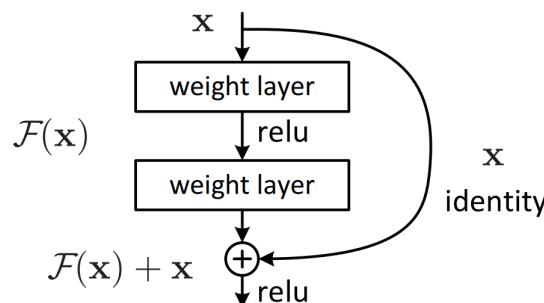
Bằng cách sắp xếp lại vấn đề thành việc xấp xỉ một hàm residual, quá trình tối ưu hóa có thể tập trung vào việc điều chỉnh trọng số của các lớp phi tuyến về gần không. Sự sắp xếp lại này cũng giúp trong các trường hợp mà hàm tối ưu gần với phép ánh xạ đồng nhất, làm cho quá trình tối ưu hóa dễ dàng hơn trong việc tìm các biến đổi liên quan đến phép ánh xạ đồng nhất thay vì học một hàm hoàn toàn mới.

#### 2.4.3.1 Identity Mapping by Shortcuts

Residual learning được áp dụng cho một vài lớp xếp chồng lên nhau. Một khối xây dựng minh họa trong Hình 2.9. Một cách chính thức, trong bài báo [12] tác giả xem xét một khối xây dựng được định nghĩa như sau:

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{W_i\}) + \mathbf{x}$$

Ở đây  $\mathbf{x}$  và  $\mathbf{y}$  là vectơ đầu vào và đầu ra của các lớp được xem xét. Hàm  $\mathcal{F}(\mathbf{x}, \{W_i\})$  biểu diễn sự ánh xạ dư cần học. Với ví dụ trong Hình 2.9 có hai lớp,  $\mathcal{F} = W_2\sigma(W_1\mathbf{x})$  trong đó  $\sigma$  biểu thị ReLU. Phép toán  $\mathcal{F} + \mathbf{x}$  được thực hiện bởi một kết nối tắt và phép cộng từng phần tử.



Hình 2.9: Một khối residual

Kết nối tắt thực hiện ánh xạ đồng nhất giữa đầu vào và đầu ra của một khối xây dựng trong mạng thần kinh. Phương trình này không thêm tham số hay phức tạp tính toán. Điều này quan trọng trong so sánh giữa mạng thuần túy và mạng dư thừa. Giúp giải quyết vấn đề suy giảm gradient trong mạng sâu, cho phép huấn luyện hiệu quả mạng sâu hơn.

#### 2.4.4 Transformer

Transformer là một kiến trúc mạng nơ-ron sâu được giới thiệu bởi Vaswani et al. trong bài báo “Attention Is All You Need [13]” vào năm 2017. Đây là một trong những tiến bộ quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên và máy dịch dẫn đến những cải tiến đáng kể trong các ứng dụng như dịch máy, tổng hợp văn bản và nhiều nhiệm vụ khác liên quan đến ngôn ngữ.

Kiến trúc Transformer dựa trên cơ chế attention (chú ý), cho phép mạng có khả năng xem xét toàn bộ các phần của dữ liệu đầu vào cùng một lúc, thay vì như các kiến trúc trước đây phải đi qua từng bước theo thứ tự. Kiến trúc này có thể hoạt động với cả dữ liệu tuỳ chỉnh dài ngắn mà không cần giới hạn về chiều dài chuỗi đầu vào.

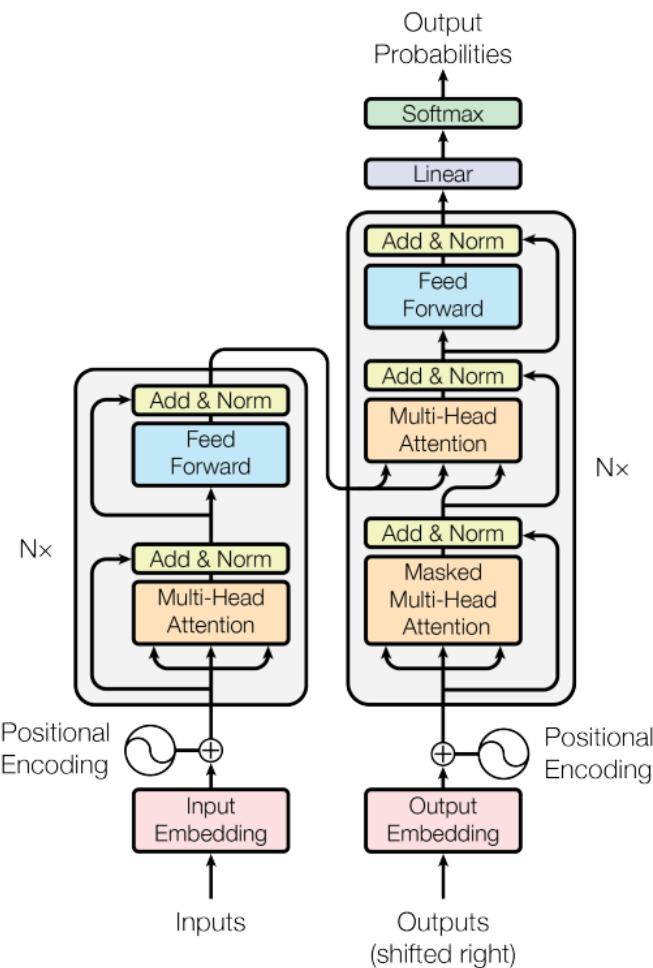
Transformer bao gồm hai phần chính: Bộ mã hóa (Encoder) và Bộ giải mã (Decoder). Cả hai phần đều sử dụng nhiều lớp tự chú ý (self-attention) để xác định sự quan hệ giữa các phần tử trong chuỗi đầu vào hoặc đầu ra. Sự chú ý tự cho phép mạng tập trung vào các phần tử quan trọng trong chuỗi và hiểu được mối quan hệ giữa chúng.

Transformer đã trở thành cơ sở cho nhiều kiến trúc mạng nơ-ron tiên tiến trong xử lý ngôn ngữ tự nhiên và thậm chí trong các ứng dụng khác như xử lý hình ảnh. Ví dụ nổi tiếng nhất có thể kể đến là “BERT” và “GPT”, cả hai đều là các biến thể của kiến trúc Transformer và đã đạt được những kết quả ấn tượng trong nhiều nhiệm vụ liên quan đến xử lý ngôn ngữ tự nhiên.

### Kiến trúc Transformer

Transformer tuân theo kiến trúc tổng thể này bằng cách sử dụng các lớp tự chú ý xếp chồng và các lớp kết nối đầy đủ theo điểm cho cả bộ mã hóa và

bộ giải mã, như được thể hiện ở nửa trái và nửa phải của Hình 2.10.



Hình 2.10: Kiến trúc tổng thể của Transformer [13]

## Ngăn xếp Encoder và Decoder

**Bộ mã hóa:** Bộ giải mã có một ngăn xếp gồm  $N = 6$  lớp tương tự nhau. Mỗi lớp bao gồm hai lớp con. Lớp đầu tiên là cơ chế tự chú ý đa đầu(multi-head self-attention), nơi mà mỗi từ trong câu đều “tương tác” với tất cả các từ khác để tạo ra sự chú ý chung trong ngữ cảnh. Lớp thứ hai là một mạng truyền thẳng kết nối đầy đủ theo từng vị trí, tức là thông tin từ mỗi vị trí trong câu được xử lý độc lập.

Để duy trì thông tin và hỗ trợ quá trình học, sử dụng kết nối residual xung quanh cả hai lớp con. Kết nối residual cho phép thông tin truyền từ đầu vào của lớp này đến đầu ra một cách dễ dàng. Sau đó, chúng ta thực hiện chuẩn

hóa lớp để điều chỉnh phạm vi giá trị đầu ra. Cụ thể, đầu ra của mỗi lớp con được chuẩn hóa thông qua phép tính  $\text{LayerNorm}(x + \text{Sublayer}(x))$ , trong đó  $\text{Sublayer}(x)$  là chức năng được thực hiện bởi chính lớp con đó.

Để đảm bảo tính nhất quán và khả năng kết nối residual, tất cả các lớp con trong mô hình cùng với các lớp nhúng đều tạo ra đầu ra với chiều  $d_{model} = 512$ , tức là có cùng kích thước đặc trưng để xử lý và truyền thông tin.

**Bộ giải mã:** Bộ giải mã cũng được tạo thành từ một ngăn xếp gồm  $N = 6$  lớp tương tự nhau. Bên cạnh hai lớp con trong mỗi lớp mã hóa, bộ giải mã thêm một lớp con thứ ba, thực hiện chú ý đa đầu (multi-head attention) qua đầu ra của ngăn xếp bộ mã hóa. Tương tự như bộ mã hóa, chúng ta sử dụng kết nối residual xung quanh mỗi lớp con, sau đó là chuẩn hóa lớp. Điều chỉnh lớp con tự chú ý trong ngăn xếp bộ giải mã để ngăn các vị trí tập trung vào các vị trí tiếp theo. Thao tác này, kết hợp với việc lớp nhúng đầu ra được dịch chuyển một vị trí, đảm bảo rằng các dự đoán cho vị trí  $i$  chỉ phụ thuộc vào đầu ra đã biết tại các vị trí nhỏ hơn  $i$ .

## Attention

Một hàm chú ý có thể được mô tả như việc ánh xạ một truy vấn và một tập hợp các cặp khóa-giá trị thành một đầu ra, trong đó truy vấn, khóa, giá trị và đầu ra đều là các vector. Đầu ra được tính toán dưới dạng tổng có trọng số của các giá trị, trong đó trọng số được gán cho mỗi giá trị được tính bằng một hàm tương thích của truy vấn với khóa tương ứng.

### Scaled Dot-Product Attention

Scaled Dot-Product Attention là một phần quan trọng của kiến trúc Transformer và cơ chế chú ý tự. Đầu vào của phương pháp này gồm các truy vấn (queries) và các khóa (keys) có kích thước  $dk$ , cùng với các giá trị (values) có kích thước  $dv$ . Tích tích vô hướng giữa truy vấn và tất cả các khóa, chia từng kết quả cho  $dk$ , sau đó áp dụng hàm softmax để thu được trọng số cho các giá trị. Thực hiện phép attention trên một tập hợp các truy vấn cùng lúc, gói chúng lại thành một ma trận  $Q$ . Các khóa và giá trị cũng được gói lại

thành các ma trận  $K$  và  $V$ . Kết quả đầu ra được tính bằng cách:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Bởi vì ảnh hưởng của tích vô hướng trở nên lớn về độ lớn, đẩy hàm softmax vào các vùng có độ dốc rất nhỏ. Để chống lại tác động này, sử dụng tỷ lệ các tích vô hướng bằng  $\frac{1}{\sqrt{d_k}}$ .

## Multi-Head Attention

Multi-Head Attention là một phần quan trọng trong kiến trúc Transformer. Trong chú ý đa đầu, một lớp chú ý thông thường được áp dụng nhiều lần với các trọng số khác nhau. Mỗi lần áp dụng này tạo ra một Đầu chú ý riêng biệt. Mỗi đầu chú ý có thể tập trung vào các phần khác nhau của thông tin đầu vào và tạo ra các biểu diễn khác nhau. Sau đó, đầu ra của các đầu chú ý này được kết hợp để tạo ra đầu ra cuối cùng của chú ý đa đầu.

Một tầng Chú ý đa đầu cho phép mô hình cùng lúc chú ý đến thông tin từ các không gian biểu diễn khác nhau tại các vị trí khác nhau. Với một đầu chú ý duy nhất, việc lấy trung bình ức chế khả năng này[13].

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O$$

$$\text{where } \text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

Ở đây phép chiếu là tham số của ma trận  $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$ ,  $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$ ,  $W_i^V \in \mathbb{R}^{d_{model} \times d_k}$  và  $W_i^O \in \mathbb{R}^{hd_v \times d_{model}}$

Quá trình này giúp mô hình học cách tập trung vào các khía cạnh khác nhau của dữ liệu đầu vào và xử lý chúng một cách đa dạng. Chú ý đa đầu giúp kiến trúc Transformer hiệu quả hơn trong việc học các mối quan hệ và tương tác phức tạp trong dữ liệu ngôn ngữ và hình ảnh.

## Position-wise Feed-Forward Networks

Ngoài các lớp con chú ý, mỗi lớp trong bộ mã hóa và giải mã của còn chứa một mạng lan truyền thẳng kết nối đầy đủ, được áp dụng độc lập và giống nhau cho mỗi vị trí. Điều này bao gồm hai phép biến đổi tuyến tính với một hàm kích hoạt ReLU ở giữa.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

Trong khi các phép biến đổi tuyến tính giống nhau ở các vị trí khác nhau, chúng sử dụng các tham số khác nhau từ lớp này sang lớp khác. Một cách khác để mô tả điều này là như hai phép tích chập với kích thước kernel là 1. Không gian chiều của đầu vào và đầu ra là  $d_{model} = 512$ , và lớp bên trong có chiều  $d_{ff} = 2048$ .

## Position Encoding

Vì mô hình của Transformer không chứa hồi quy và không có tích chập, để mô hình có thể sử dụng thứ tự của chuỗi, ta cần phải đưa vào một số thông tin về vị trí tương đối hoặc tuyệt đối của các token trong chuỗi. Để làm điều này, mô hình thêm mã hóa vị trí vào lớp nhúng đầu vào ở phía dưới của các bộ mã hóa và giải mã. Các mã hóa vị trí có cùng kích thước  $d_{model}$  như các lớp nhúng, để hai thành phần này có thể được tổng hợp. Có nhiều lựa chọn cho các mã hóa vị trí, có thể được học và cố định.

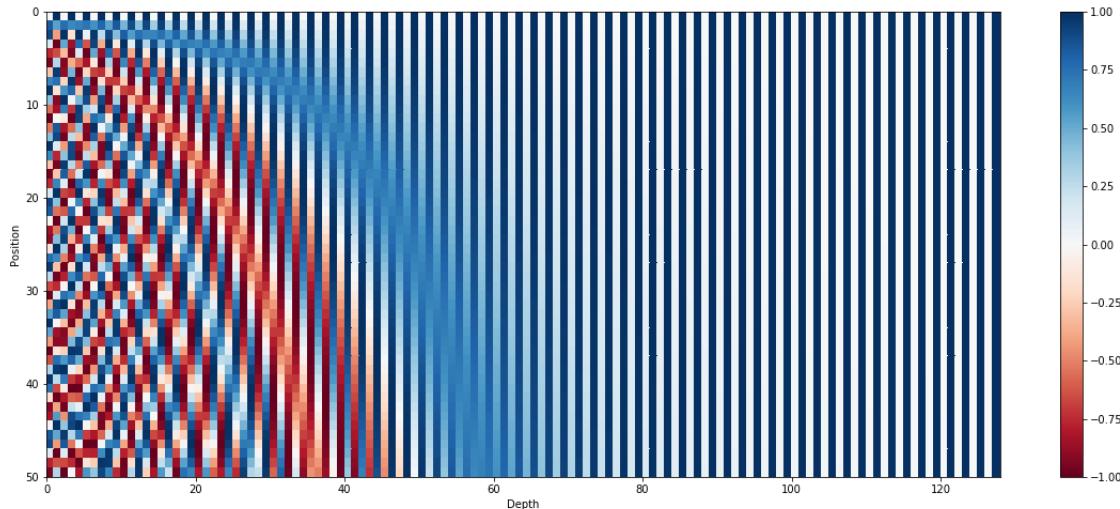
Trong mô hình này sử dụng các hàm sine và cosine với các tần số khác nhau.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{\frac{2i+1}{d_{model}}}}\right)$$

Ở đây  $pos$  là vị trí và  $i$  là chiều. Mỗi chiều của mã hóa vị trí tương ứng với một hàm sin. Các bước sóng tạo thành một dãy hình học từ  $2\pi$  đến  $10000 \times 2\pi$ . Hàm này được lựa chọn bởi vì tác giả [13], giả định rằng nó sẽ cho phép mô

hình dễ dàng học cách chú ý đến vị trí tương đối, vì đối với bất kỳ độ lệch cố định  $k$  nào,  $PE_{pos+k}$  có thể được biểu diễn dưới dạng một hàm tuyến tính của  $PE_{pos}$ [2.11](#).



Hình 2.11: Position encoding 128 chiều cho một câu có độ dài tối đa là 50

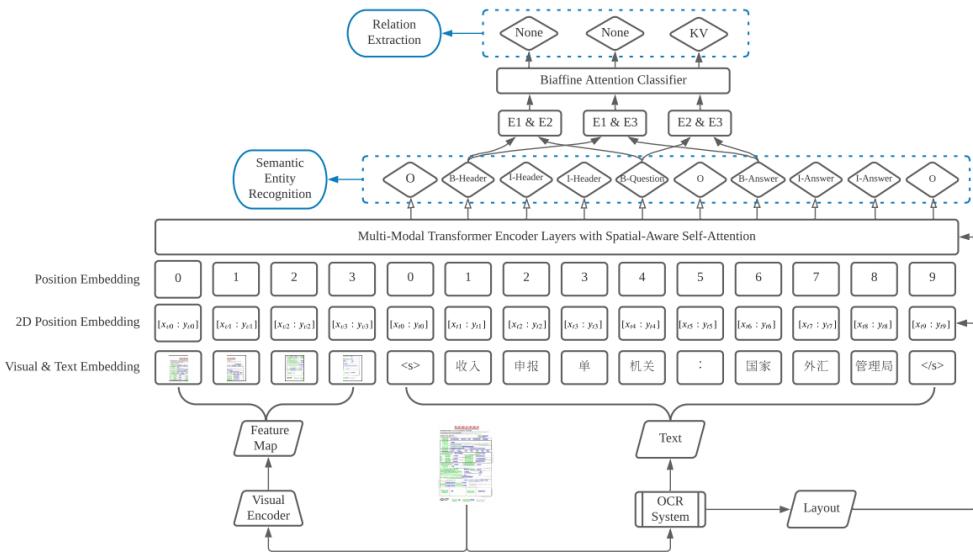
#### 2.4.5 LayoutXLM

LayoutXLM [14], một mô hình đã được pre-trained đa phương thức để hiểu văn bản trong nhiều ngôn ngữ khác nhau. Mục tiêu chính của mô hình này là vượt qua rào cản ngôn ngữ trong việc hiểu các văn bản có chứa thông tin hình ảnh đa dạng. Điều đặc đáo của LayoutXLM nằm ở việc nó kết hợp thông tin từ văn bản, cấu trúc bố cục và cả hình ảnh để huấn luyện một mô hình Transformer để hiểu tài liệu.

Mô hình đã được huấn luyện bằng cách sử dụng 30 triệu tài liệu số từ 53 ngôn ngữ khác nhau. Kết quả thể hiện rằng LayoutXLM vượt trội hơn so với các mô hình đa ngôn ngữ khác khi được đánh giá trên bộ dữ liệu hiểu biểu mẫu XFUND trong 7 ngôn ngữ khác nhau.

Tóm lại, đây thực sự là một bước đột phá trong nghiên cứu về việc huấn luyện mô hình đa phương tiện và đa ngôn ngữ để hiểu các tài liệu phức tạp chứa cả hình ảnh.

Mô hình LayoutXLM được xây dựng với một kiến trúc Transformer đa



Hình 2.12: Kiến trúc của mô hình LayoutXLM

phương tiện. Kiến trúc được thể hiện trong Hình 2.12. Mô hình chấp nhận thông tin từ ba phương thức khác nhau, bao gồm văn bản, bố cục và hình ảnh, được mã hóa lần lượt bằng các lớp nhúng văn bản, nhúng bố cục và nhúng hình ảnh. Các nhúng văn bản và hình ảnh được nối lại, sau đó cộng với nhúng bố cục để thu được nhúng đầu vào. Những nhúng đầu vào được mã hóa bởi một Transformer đa phương tiện với cơ chế tự-chú ý có thông tin vị trí. Cuối cùng, biểu diễn ngữ cảnh đầu ra có thể được sử dụng cho các lớp cụ thể cho các nhiệm vụ tiếp theo.

#### 2.4.5.1 Kiến trúc mô hình

##### Text Embedding

Tuân theo thực tiễn thông thường, mô hình sử dụng WordPiece để tách thành các từ trong dãy văn bản từ OCR và gán mỗi từ vào một phân đoạn cụ thể  $s_i \in \{[A], [B]\}$ . Sau đó, thêm [CLS] ở đầu của dãy và [SEP] ở cuối mỗi phân đoạn văn bản. Các [PAD] bổ sung được đặt ở cuối để đảm bảo độ dài cuối cùng của chuỗi chính xác là độ dài tối đa của chuỗi  $L$ . Nhúng văn bản cuối cùng là tổng của ba embedding. Token embedding đại diện cho chính token, position embedding 1D đại diện cho chỉ số của token, và embedding phân đoạn được

sử dụng để phân biệt các phân đoạn văn bản khác nhau. Một cách chính xác hơn, mô hình có text embedding thứ  $i$  ( $0 \leq i < L$ ) như sau [15]:

$$\mathbf{t}_i = \text{TokEmb}(w_i) + \text{PosEmb1D}(i) + \text{SegEmb}(s_i)$$

## Visual Embedding

Do mô hình gặp khó khăn trong việc bắt kịp các chi tiết trong hình ảnh toàn bộ trang, nên cần chuyển đổi hình ảnh thành chuỗi có độ dài cố định bằng cách sử dụng bản đồ đặc trưng của mã hóa hình ảnh CNN. Mô hình dùng kiến trúc ResNeXt-FPN làm nền tảng mã hóa hình ảnh. Hình ảnh được chuẩn hóa kích thước và đưa vào mã hóa. Bản đồ đặc trưng đầu ra được làm phẳng thành chuỗi embedding hình ảnh. Một lớp chiếu áp dụng cho mỗi embedding để đồng nhất kích thước với text embedding. Thêm position embedding 1D để bổ sung thông tin vị trí. Tất cả embedding hình ảnh gán vào phân đoạn hình ảnh.

Cho một hình ảnh trang tài liệu  $I$ , nó được điều chỉnh kích thước thành  $224 \times 224$  rồi được đưa vào bộ mã hóa hình ảnh. Sau đó, bản đồ đặc trưng đầu ra lấy trung bình để có kích thước cố định với chiều rộng là  $W$  và chiều cao là  $H$ . Tiếp theo, nó được làm phẳng thành một chuỗi embedding hình ảnh có độ dài là  $W \times H$ . Chuỗi này được gọi là  $\text{VisTokEmb}(I)$ . Một lớp chiếu tuyến tính được áp dụng cho mỗi token embedding hình ảnh để đồng nhất kích thước chiều với text embedding. Vì bộ mã hóa hình ảnh dựa trên CNN không thể bắt được thông tin vị trí, chúng tôi cũng thêm position embedding 1D vào các token embedding hình ảnh này. position embedding 1D được sử dụng chung với lớp text embedding. Đối với embedding phân đoạn, chúng tôi gán tất cả các token embedding hình ảnh vào phân đoạn hình ảnh  $[C]$ . embedding hình ảnh thứ  $i$  ( $0 \leq i < WH$ ) có thể được biểu diễn như sau [15]:

$$\mathbf{v}_i = \text{Proj}(\text{VisTokEmb}(I)_i) + \text{PosEmb1D}(i) + \text{SegEmb}([C])$$

## Layout Embedding

Lớp Layout Embedding dùng để nhúng thông tin bối cảnh không gian được biểu diễn bởi các khung chữ nhật của các token từ kết quả OCR, trong đó chiều rộng, chiều cao khung cùng với tọa độ góc được xác định. Tất cả các tọa độ được chuẩn hóa và định lượng thành các số nguyên trong khoảng [0, 1000], và sử dụng 2 lớp embedding riêng biệt để nhúng các đặc trưng trực  $x$  và trực  $y$ . Với hộp giới hạn chuẩn hóa của token văn bản/thị giác thứ  $i$  ( $0 \leq i < WH + L$ ) là  $\text{box}_i = (x_{min}, x_{max}, y_{min}, y_{max}, width, height)$ , lớp layout embedding nối tất cả 6 đặc trưng của hộp giới hạn thành một vector position embedding 2D ở cấp độ token [15].

$$\mathbf{l}_i = \text{Concat}(\text{PosEmb2D}_x(x_{min}, x_{max}, width), \text{PosEmb2D}_y(y_{min}, y_{max}, height))$$

Như vậy, lớp nhúng bối cảnh dùng để mã hóa thông tin vị trí không gian của các token dựa trên kết quả OCR thành các vector nhúng, giúp mô hình hiểu được mối quan hệ không gian giữa các token.

## Multi-modal Encoder with Spatial-Aware Self-Attention Mechanism

Bộ mã hóa kết hợp nhúng hình ảnh và văn bản thành một chuỗi đơn, tích hợp thông tin không gian thông qua nhúng bối cảnh. Bộ mã hóa hoạt động giống như kiến trúc Transformer, sử dụng nhiều lớp tự chú ý liên tiếp theo bởi các mạng tiền thảng. Để giải quyết việc hiểu quả việc nắm bắt mối quan hệ bối cảnh trong bối cảnh tài liệu, một cơ chế tự chú ý nhận thức vị trí được giới thiệu. Khác với tự chú ý ban đầu, phụ thuộc vào thông tin vị trí tuyệt đối, tự chú ý nhận thức vị trí xem xét thông tin vị trí tương đối một cách rõ ràng. Cơ chế này liên quan đến một đầu vào duy nhất trong một lớp tự chú ý, với kích thước ẩn  $d_{head}$  và các ma trận chiều  $WQ$ ,  $WK$ , và  $WV$ . Điểm chú ý giữa truy vấn  $xi$  và khóa  $xj$  được tính toán bằng quá trình chiếu và chuẩn hóa.

$$\alpha_{ij} = \frac{1}{\sqrt{d_{head}}} (\mathbf{x}_i \mathbf{W}^Q) (\mathbf{x}_j \mathbf{W}^K)^T$$

Xem xét khoảng cách rộng lớn của các vị trí, mô hình hóa vị trí tương quan ngữ nghĩa và vị trí tương quan không gian dưới dạng các thành phần bias để ngăn việc thêm quá nhiều tham số. Thực hành tương tự đã được chứng minh hiệu quả trong các kiến trúc Transformer chỉ có văn bản [13]. Giả sử  $\mathbf{b}^{(1D)}$ ,  $\mathbf{b}^{(2D_x)}$  và  $\mathbf{b}^{(2D_y)}$  là các bias vị trí tương quan 1D và 2D có thể học được tương ứng. Các bias này khác nhau giữa các đầu chú ý nhưng được chia sẻ trong tất cả các lớp mã hóa. Giả sử  $(x_i, y_i)$  là tọa độ góc trên bên trái của hộp giới hạn thứ  $i$ , chúng ta thu được điểm chú ý nhận thức vị trí không gian như sau:

$$\alpha'_{ij} = \alpha_{ij} + \mathbf{b}_{j-i}^{(1D)} + \mathbf{b}_{x_j-x_i}^{(2D_x)} + \mathbf{b}_{y_j-y_i}^{(2D_y)}$$

Cuối cùng, các vector đầu ra được biểu diễn dưới dạng trung bình có trọng số của tất cả các vector giá trị đã được chiếu tương ứng với các điểm chú ý nhận thức vị trí không gian đã được chuẩn hóa [15].

$$\mathbf{h}_i = \sum_j \frac{\exp(\alpha'_{ij})}{\sum_k \exp(\alpha'_{ik})} \mathbf{x}_j \mathbf{W}^V$$

#### 2.4.5.2 Nhiệm vụ tiền huấn luyện

##### Masked Visual-Language Modeling

Masked Visual-Language Modeling giúp mô hình học tốt hơn trong phần ngôn ngữ với các dấu vết chéo-modality. Mô hình ngẫu nhiên che đi một số token văn bản và yêu cầu mô hình phục hồi các token bị che. Trong khi đó, thông tin về bộ cục vẫn không thay đổi, điều này có nghĩa là mô hình biết vị trí của từng mảnh thông báo bị che trên trang. Các biểu diễn đầu ra của các token bị che từ bộ mã hóa được đưa vào một bộ phân loại trên toàn bộ từ vựng, dựa trên một mất mát cross-entropy. Để tránh rò rỉ dấu vết hình ảnh, Mô hình che khu vực hình ảnh tương ứng với các token bị che trên đầu vào hình ảnh trang gốc trước khi đưa nó vào bộ mã hóa hình ảnh.

## Text-Image Alignment

Để giúp mô hình học được sự tương ứng vị trí không gian giữa hình ảnh và tọa độ của các hộp giới hạn, Mô hình đề xuất Nhiệm vụ Text-Image Alignment (TIA) như một nhiệm vụ căn chỉnh chéo chi tiết hơn. Trong nhiệm vụ TIA, một số dòng văn bản được chọn ngẫu nhiên và các khu vực hình ảnh tương ứng của chúng được che trên hình ảnh tài liệu. Thao tác này được gọi là “covering” để tránh sự nhầm lẫn với thao tác “masking” trong MVLM. Trong quá trình tiền huấn luyện, một lớp phân loại được xây dựng trên đầu ra của bộ mã hóa. Lớp này dự đoán một nhãn cho mỗi mã thông báo văn bản tùy thuộc vào việc nó được che, tức là [Covered] hoặc [Not Covered], và tính mất mát nhị phân cross-entropy. Xem xét độ phân giải của hình ảnh đầu vào có hạn, và một số yếu tố trong tài liệu như biển báo và thanh trong một hình ảnh có thể giống như các vùng văn bản bị che, nhiệm vụ tìm kiếm một vùng hình ảnh bị che cõi từ có thể gây nhiễu. Do đó, thao tác che được thực hiện ở mức dòng văn bản. Khi MVLM và TIA được thực hiện đồng thời, các mất mát TIA của các mã thông báo bị che trong MVLM không được tính vào. Điều này ngăn mô hình học được sự tương ứng không hữu ích nhưng trực tiếp từ [MASK] đến [Covered].

## Text-Image Matching

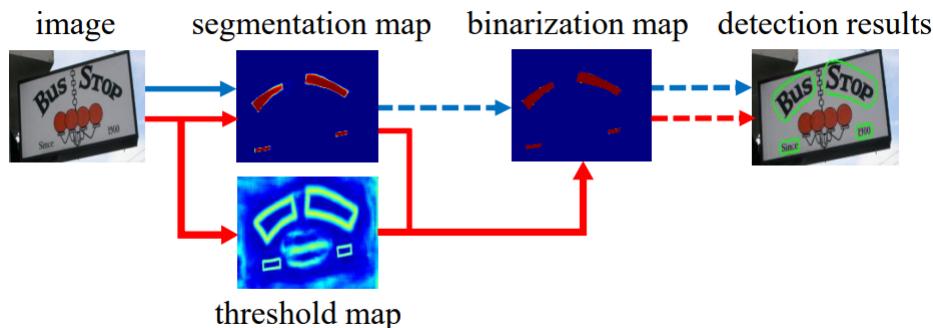
Hơn nữa, một nhiệm vụ căn chỉnh chéo chi tiết hơn, Nhiệm vụ Text-Image Matching (TIM), được áp dụng để giúp mô hình học sự tương ứng giữa hình ảnh tài liệu và nội dung văn bản. Chúng tôi đưa biểu diễn đầu ra tại [CLS] vào một bộ phân loại để dự đoán xem hình ảnh và văn bản có đến từ cùng một trang tài liệu hay không. Các đầu vào thông thường được sử dụng là các mẫu tích cực. Để xây dựng một mẫu tiêu cực, một hình ảnh có thể được thay thế bằng một hình ảnh trang từ tài liệu khác hoặc bị bỏ qua. Để ngăn mô hình gian lận bằng cách tìm các đặc trưng của nhiệm vụ, chúng tôi thực hiện các thao tác masking và covering tương tự cho các hình ảnh trong các mẫu tiêu cực. Nhãn mục tiêu TIM trong các mẫu tiêu cực được thiết lập là [Covered]

cho tất cả. Chúng tôi áp dụng mất mát nhị phân cross-entropy trong quá trình tối ưu hóa.

#### 2.4.6 DBNet

Differentiable Binarization Network (DBNet) [16], đây là một mô hình mạng nơ-ron sử dụng trong lĩnh vực phát hiện văn bản. Phương pháp này tập trung vào vấn đề chuyển đổi hình ảnh văn bản đa dạng thành hình ảnh nhị phân, đồng thời huấn luyện mạng theo cách có thể tối ưu hóa quá trình này. Phương pháp nhị phân hóa khác biệt của DBNet cho phép giữ lại thông tin quan trọng trong hình ảnh và loại bỏ thông tin không cần thiết, giúp cải thiện độ chính xác của quá trình nhận dạng ký tự.

DB là một thuật toán dựa trên phân đoạn để phát hiện văn bản, sử dụng một mô-đun Nhị phân hóa Nguồn Khả vi (DB) khác biệt để phân biệt vùng văn bản khỏi nền với một ngưỡng động.



Hình 2.13: Pipeline truyền thông (luồng màu xanh) và pipeline của DB (luồng màu đỏ)

Những mũi tên màu xanh trong hình minh họa cho quy trình của các thuật toán phát hiện văn bản dựa trên phân đoạn thông thường. Loại phương pháp này sử dụng một ngưỡng cố định để tạo ra bản đồ phân đoạn nhị phân sau khi phân đoạn, sau đó áp dụng các thuật toán heuristics như gom cụm pixel để có được vùng văn bản. Những mũi tên màu đỏ trong hình minh họa cho luồng của thuật toán DB. Sự khác biệt lớn nhất so với các giải pháp thông thường là DB có một bản đồ ngưỡng, và nó sẽ dự đoán ngưỡng tại mỗi điểm

pixel của hình ảnh thông qua mạng neural, thay vì chỉ định một giá trị cố định. Do đó, nó có thể phân biệt tốt hơn giữa nền và vùng văn bản.

Thuật toán DB có những ưu điểm sau:

1. Cấu trúc thuật toán đơn giản và không cần xử lý sau quá trình tính toán phức tạp
2. Dữ liệu nguồn mở của nó có độ chính xác và hiệu suất tốt

Sau khi có bản đồ xác suất, thuật toán truyền thống dựa trên phân đoạn hình ảnh sẽ đặt tất cả các điểm ảnh có giá trị thấp hơn ngưỡng  $t$  thành 0 và ngược lại thành 1. Công thức là:

$$B_{i,j} = \begin{cases} 1, & \text{if } P_{i,j} \geq t, \\ 0, & \text{otherwise} \end{cases}$$

Nhưng phương pháp nhị phân tiêu chuẩn không có khả năng khác biệt, từ đó khiến mạng không thể được huấn luyện theo kiểu end-to-end (tức là không thể tích hợp vào quá trình lan truyền ngược). Để giải quyết vấn đề này, thuật toán DB sử dụng Differentiable Binarization, giúp xấp xỉ step function của phương pháp nhị phân tiêu chuẩn. Nó sử dụng công thức khác:

$$\hat{B} = \frac{1}{1 + e^{-k(P_{i,j} - T_{i,j})}}$$

Ở trên,  $P$  đề cập đến bản đồ xác suất,  $T$  đề cập đến bản đồ ngưỡng, và  $k$  là hệ số tăng được thiết lập là 50 dưới một quy tắc thực tế trong thí nghiệm. Hình 2.14(a) dưới đây cho thấy sự khác biệt giữa phương pháp nhị phân tiêu chuẩn và phân đoạn khác biệt.

Khi sử dụng hàm mất mát cross-entropy, mất mát của các mẫu dương và mẫu âm lần lượt là  $l_+$  và  $l_-$ :

$$l_+ = -\log\left(\frac{1}{1 + e^{-k(P_{i,j} - T_{i,j})}}\right)$$

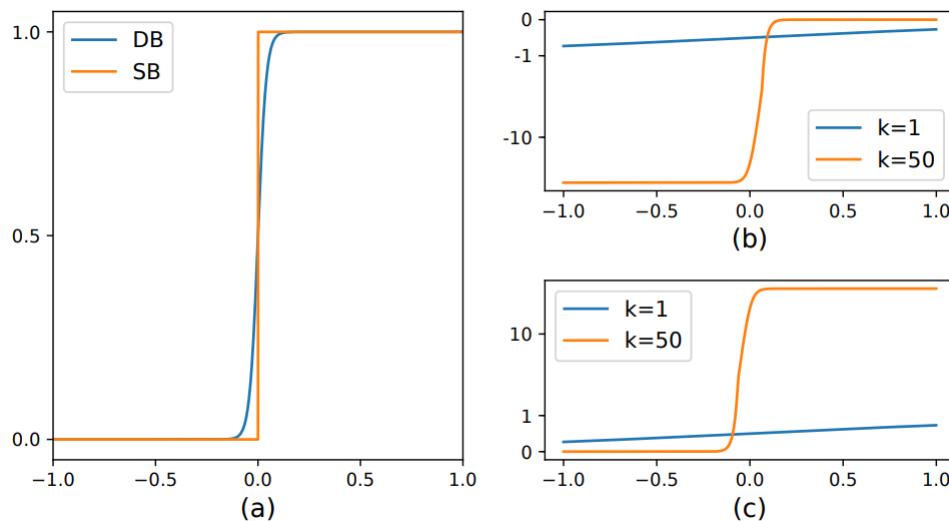
$$l_- = -\log\left(1 - \frac{1}{1 + e^{-k(P_{i,j} - T_{i,j})}}\right)$$

Nhập  $x$  vào để lấy đạo hàm riêng có thể dẫn đến:

$$\frac{\delta l_+}{\delta x} = -kf(x)e^{-kx}$$

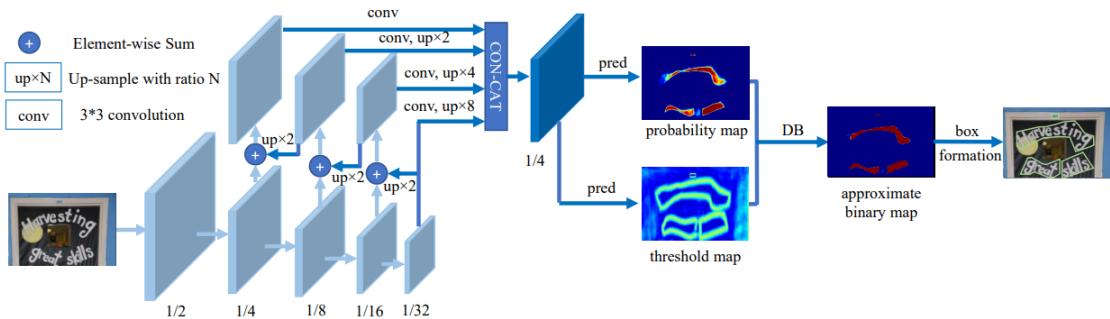
$$\frac{\delta l_-}{\delta x} = -kf(x)$$

Có thể thấy rằng hệ số tăng sẽ làm phình to độ dốc của dự đoán lỗi, từ đó tối ưu hóa mô hình để đạt được kết quả tốt hơn. Trong Hình 2.14(b), phần của  $x < 0$  đại diện cho trường hợp mẫu tích cực được dự đoán thành mẫu tiêu cực. Có thể thấy rằng hệ số tăng  $k$  làm phình to độ dốc. Hình 2.14(c) hiển thị  $x > 0$ , để cập đến trường hợp mẫu tiêu cực được dự đoán thành mẫu tích cực, và độ dốc cũng được phình to.



Hình 2.14: Minh họa về phân đoạn khác biệt và đạo hàm của nó. (a) So sánh số liệu giữa phương pháp nhị phân tiêu chuẩn (SB) và phân đoạn khác biệt (DB). (b) Đạo hàm của  $l_+$ . (c) Đạo hàm của  $l_-$ . [16]

Cấu trúc tổng quan của thuật toán DB (Hình 2.15), các đặc điểm của hình ảnh đầu vào được trích xuất thông qua mạng Backbone và FPN, sau đó chúng được nối liền để tạo ra một đặc điểm có kích thước là một phần tư của hình ảnh gốc. Sau đó, lớp tích chập được sử dụng để tạo ra bản đồ xác suất dự đoán và bản đồ ngưỡng, và sau đó tạo ra đường viền qua quá trình xử lý sau cùng của DB.



Hình 2.15: Kiến trúc của phương pháp DB

#### 2.4.7 TransformerOCR

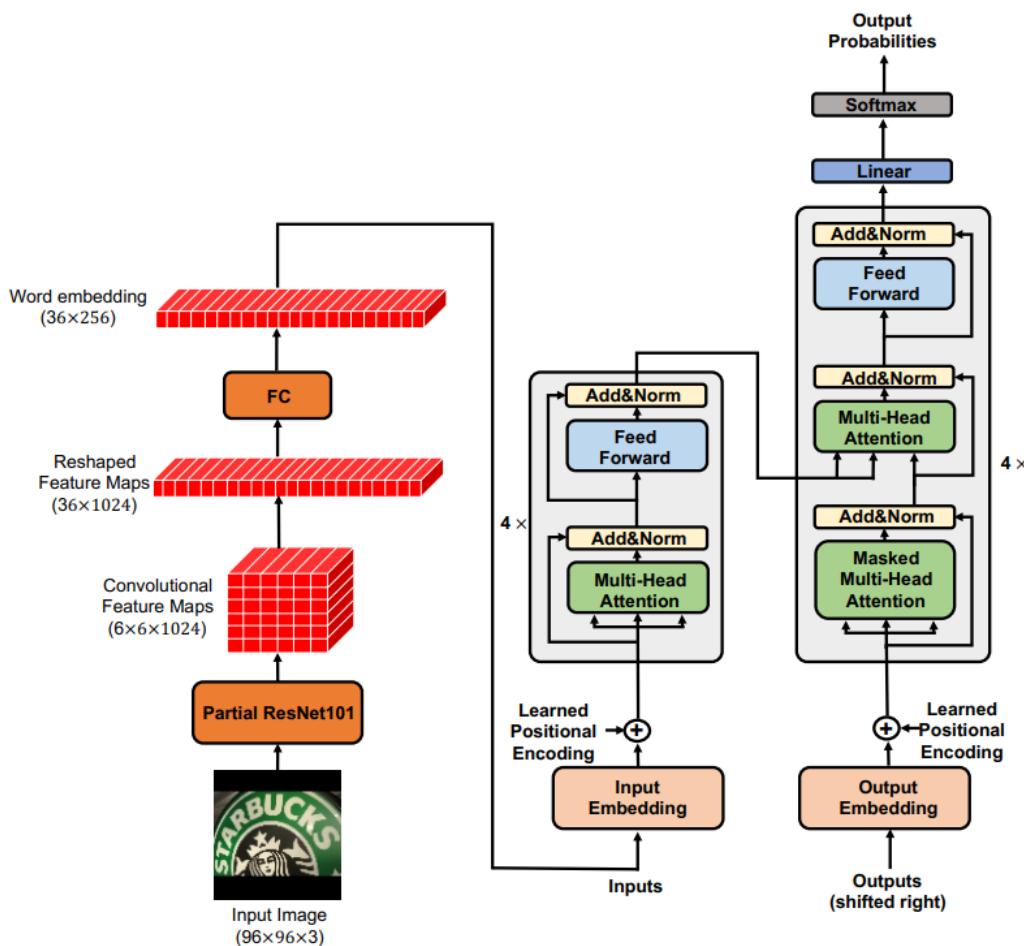
Để nhận dạng văn bản bằng tiếng việt, em sử dụng mã nguồn mở VietOCR đây là một mô hình được cài đặt trên Transformer OCR. Dùng để nhận dạng chữ viết tay, chữ đánh máy cho Tiếng Việt.

TransformerOCR là một sự kết hợp tuyệt vời giữa mô hình CNN và Transformer Hình 2.16, được áp dụng rộng rãi trong lĩnh vực nhận dạng ký tự quang học. Mô hình bao gồm hai mô-đun chính: mô-đun trích xuất đặc trưng và mô-đun transformer. Mô-đun trích xuất đặc trưng được sử dụng ban đầu để tạo ra các bản đồ đặc trưng từ một hình ảnh đầu vào. Sau đó, các bản đồ đặc trưng được đưa vào mô-đun transformer như là nhúng từ đầu vào.

#### Mô-đun Trích xuất đặc trưng

Có rất nhiều mạng CNN có thể được sử dụng để trích xuất đặc trưng. Mô hình CNN dùng trong bài toán OCR nhận đầu vào là một ảnh, thông thường có kích thước với chiều dài lớn hơn nhiều so với chiều rộng, do đó việc điều chỉnh tham số stride size của tầng pooling là cực kì quan trọng. Việc chọn kích thước stride size của các lớp pooling cuối cùng là  $w \times x = 2 \times 1$  trong mô hình OCR. Không thay đổi stride size phù hợp với kích thước ảnh thì sẽ dẫn đến kết quả nhận dạng của mô hình sẽ tệ.

Đối với mô hình VGG, việc thay đổi pooling size khá dễ do kiến trúc đơn giản, tuy nhiên đối với mô hình phức tạp khác như resnet việc điều chỉnh tham số pooling size hơi phức tạp do một ảnh bị downsampling không chỉ bởi



Hình 2.16: Kiến trúc tổng qua của TransformerOCR [17]

tầng pooling mà còn tại các tầng convolution khác.

Một ảnh qua mô hình CNN, sẽ cho một feature maps có kích thước (width  $\times$  height)  $\times$  batch  $\times$  channels, feature maps này sẽ trở thành đầu vào cho mô hình Transformer.

## Mô-đun Transformer

Transformer là một mô hình cách mạng trong xử lý ngôn ngữ tự nhiên. Nhờ cơ chế attention mạnh mẽ, transformer vượt trội hơn đáng kể so với hầu hết các mô hình attention dựa trên RNN. Khác với local attention dựa trên RNN, transformer có phạm vi global attention. Như được minh họa trong Hình 2.16, transformer có cấu trúc mã hóa-giải mã. Bộ mã hóa ánh xạ các bản đồ đặc trưng ( $x_1, \dots, x_n$ ) do mô-đun trích xuất đặc trưng tạo ra thành một

chuỗi các biểu diễn liên tục  $z = (z_1, \dots, z_n)$ . Dựa trên  $z$ , bộ giải mã sau đó tạo ra một chuỗi đầu ra  $(y_1, \dots, y_m)$  của các ký hiệu một phần tử tại một thời điểm. Ở mỗi bước, mô hình là tự hồi quy, tiêu thụ các ký hiệu được tạo ra trước đó như đầu vào bổ sung khi tạo ra ký hiệu tiếp theo [17].

Trong mô hình này, hình ảnh đầu vào có góc nhìn hai chiều. Do đó, mã hóa vị trí cố định một chiều không phù hợp. Thay vào đó, mô hình sử dụng nhúng vị trí có thể học được trong mô hình đề xuất. Nhờ cơ chế Attention mạnh mẽ của transformer, mô hình thực hiện lưu giữ đầu ra CNN dưới dạng Spatial attention. Tại mỗi bước giải mã, một mô-đun Attention nhìn thấy spatial attention. Bằng cách này, mô hình đạt được hiệu suất tốt nhất trên các bộ dữ liệu nhận dạng văn bản không đều và kết quả có thể so sánh trên các tập dữ liệu văn bản thông thường.

Tại sao không sử dụng các mô hình chú ý dựa trên RNN? Lý do chính là, RNN xử lý chuỗi đầu vào một cách tuần tự, cơ chế này khiến cho RNN có xu hướng thu thập thông tin cục bộ hơn. Cái tồi tệ hơn là RNN không có khả năng bù đắp cho sự mất mát thông tin không gian khi chuyển đổi bản đồ đặc trưng hai chiều thành một chiều. Nhưng transformer xử lý chuỗi đầu vào theo cách song song, mỗi đặc trưng đơn lẻ có thể có tất cả sự chú ý từ đầu vào. Mặc dù việc sắp xếp lại bản đồ đặc trưng từ hai chiều thành một chiều sẽ mất thông tin không gian, mã hóa vị trí học được có khả năng bù đắp cho sự mất mát này [17].

#### 2.4.8 Ứng dụng trong OCR

Học sâu đã được chứng minh là có hiệu quả hơn các phương pháp OCR truyền thống [18], đặc biệt là đối với các tài liệu có chất lượng thấp hoặc bị định dạng phức tạp.

Một số ứng dụng của Học sâu trong OCR bao gồm:

- Nhận dạng hóa đơn: Học sâu có thể được sử dụng để nhận dạng các trường thông tin quan trọng trên hóa đơn, chẳng hạn như tên người mua, người bán, ngày giao dịch, số lượng, giá và tổng số tiền. Điều này có thể giúp tiết kiệm thời gian và chi phí cho các doanh nghiệp, đồng thời cải

thiện độ chính xác của việc xử lý hóa đơn.

- Nhận dạng tài liệu y tế: Học sâu có thể được sử dụng để nhận dạng thông tin quan trọng trên tài liệu y tế, chẳng hạn như tên bệnh nhân, chẩn đoán, phương pháp điều trị và các loại thuốc được kê đơn. Điều này có thể giúp cải thiện chất lượng chăm sóc bệnh nhân và giảm chi phí chăm sóc sức khỏe.
- Nhận dạng tài liệu pháp lý: Học sâu có thể được sử dụng để nhận dạng thông tin quan trọng trên tài liệu pháp lý, chẳng hạn như tên các bên liên quan, ngày tháng, các điều khoản của thỏa thuận và các điều khoản của hợp đồng. Điều này có thể giúp các luật sư và chuyên gia pháp lý tìm kiếm thông tin nhanh chóng và dễ dàng hơn.
- Nhận dạng tài liệu tài chính: Học sâu có thể được sử dụng để nhận dạng thông tin quan trọng trên tài liệu tài chính, chẳng hạn như tên công ty, giá cổ phiếu, số lượng cổ phiếu và giá trị thị trường của cổ phiếu. Điều này có thể giúp các nhà đầu tư và các chuyên gia tài chính đưa ra quyết định đầu tư tốt hơn.

Học sâu là một công nghệ mạnh mẽ có thể được sử dụng để cải thiện độ chính xác và hiệu quả của OCR. Với sự phát triển của Học sâu, OCR sẽ trở nên dễ dàng và thuận tiện hơn trong tương lai.

Chương này em đã mô tả các lý thuyết liên quan đến OCR, học sâu và các thuật toán, phương pháp nổi bật của từng nhiệm vụ riêng lẻ. Trong chương tiếp theo em sẽ giới thiệu về các công cụ và thư viện cần thiết.

## Chương 3

# Công cụ và môi trường thực hiện

Trong chương này em sẽ trình bày những công cụ và môi trường cần thiết trong quá trình huấn luyện mô hình OCR. Dưới đây là các bước trình bày chi tiết:

- Phầm mềm và công cụ hỗ trợ:** Trình bày các phần mềm và công cụ cho quá trình chuẩn bị dữ liệu, huấn luyện và xây dựng mô hình.
- Tạo môi trường và cài công cụ cần thiết:** Trình bày các bước cài đặt môi trường cho đê tài và thiết lập một số môi trường cần thiết để chạy.

Các công cụ và môi trường là nền tảng quan trọng cho việc thực hiện nghiên cứu, giúp định hình cách tiến hành các bước quan trọng trong đê tài “Nghiên cứu ứng dụng công nghệ OCR nhận dạng hóa đơn”.

### 3.1 Phầm mềm và công cụ hỗ trợ

#### 3.1.1 Google Colab

Google Colab (viết tắt của Google Colaboratory) là một dịch vụ miễn phí của Google cho phép thực hiện và chia sẻ các tệp notebook Jupyter, cũng như code Python. Nó là một môi trường trực tuyến, cho phép người dùng viết và

chạy code Python một cách trực tiếp trên trình duyệt web mà không cần cài đặt bất kỳ môi trường phát triển nào trên máy tính.

Colab cung cấp sử dụng miễn phí cho CPU, GPU và RAM để người dùng có thể thực hiện các nhiệm vụ tính toán phức tạp mà không cần phải mua hoặc cầu hình phần cứng riêng. Colab còn được tích hợp sẵn với nhiều thư viện phổ biến cho khoa học dữ liệu, học máy và xử lý ảnh, giúp dễ dàng tiến hành các tác vụ phức tạp. Có thể tạo notebook Jupyter, trong đó bạn có thể viết code Python từng cell và thực thi chúng một cách tương tác. Điều này rất hữu ích cho việc thử nghiệm, phân tích dữ liệu và xây dựng mô hình máy học.

Colab có thể chia sẻ notebook của mình với người khác thông qua liên kết. Người khác có thể xem và chỉnh sửa notebook hoặc thậm chí làm việc chung với nhau trên cùng một notebook. Có thể lưu notebook và dữ liệu của mình trực tiếp vào Google Drive để truy cập dễ dàng và chia sẻ với các thiết bị khác.

Google Colab thường được sử dụng trong việc học, nghiên cứu và phát triển các dự án liên quan đến khoa học dữ liệu, học máy và trí tuệ nhân tạo mà không cần đầu tư nhiều vào cấu hình phần cứng.

### 3.1.2 Vast.ai

Vast.ai là một nền tảng tính toán phi tập trung cho phép bất cứ ai cho thuê công suất tính toán dư thừa của mình để kiếm tiền. Nó kết nối những người cần công suất tính toán bổ sung cho các tác vụ như học máy với những người có GPU hoặc CPU nhàn rỗi trên máy tính của họ. Một số tính năng chính của Vast.ai:

- **Mạng phi tập trung:** Vast.ai chạy trên mạng ngang hàng phi tập trung thay vì máy chủ tập trung. Điều này làm cho nó bền vững và minh bạch hơn.
- **Cho thuê công suất tính toán dư thừa:** Mọi người có thể cài đặt ứng dụng khách Vast.ai trên máy tính của họ để cho thuê bất kỳ chu kỳ GPU hoặc CPU không sử dụng nào và được trả bằng tiền điện tử. Điều này cho

phép mọi người kiếm tiền từ các tài nguyên tính toán dư thừa.

- Thuê công suất tính toán: Các nhà nghiên cứu, nhà phát triển và những người khác cần công suất tính toán bổ sung cho các ứng dụng như huấn luyện học máy có thể thuê GPU và CPU theo yêu cầu thông qua sàn giao dịch vast.ai. Giá được định giá động dựa trên cung và cầu.
- Mã nguồn mở: Vast.ai là một dự án nguồn mở được xây dựng trên các công nghệ phi tập trung hiện có. Điều này cho phép tính minh bạch và đóng góp của cộng đồng vào nền tảng.

Vast.ai tạo ra mô hình kinh tế chia sẻ cho công suất tính toán, kết nối những người có nguồn lực dư thừa với những người cần chúng theo cách ngang hàng phi tập trung. Nó cung cấp một cách dễ dàng để bất cứ ai có thể kiếm tiền từ công suất tính toán không sử dụng.

### 3.1.3 CUDA Toolkit

CUDA Toolkit (Compute Unified Device Architecture Toolkit) là một bộ công cụ và thư viện phát triển bởi NVIDIA dành cho việc phát triển ứng dụng và tính toán sử dụng các GPU của NVIDIA. CUDA là một mô hình lập trình và nền tảng tích hợp vào GPU để thực hiện tính toán đồng thời và xử lý dữ liệu song song. Điều này cho phép các ứng dụng thực hiện tính toán nhanh hơn trên GPU so với việc sử dụng CPU truyền thống.

CUDA Toolkit bao gồm các thành phần quan trọng sau đây:

1. **CUDA Runtime:** CUDA Runtime là một tập hợp các thư viện và API cho phép ứng dụng tương tác với GPU và thực hiện các phép tính trên GPU. Nó cho phép bạn chạy các chương trình CUDA trên GPU của NVIDIA.
2. **CUDA Compiler:** CUDA Toolkit đi kèm với trình biên dịch nvcc để biên dịch mã CUDA C/C++ thành mã máy GPU thực thi được.
3. **CUDA Libraries:** CUDA Toolkit cung cấp một loạt thư viện cho các loại tính toán khác nhau, bao gồm thư viện đại số tuyến tính, thư viện

xử lý ảnh, thư viện tính toán máy học (cuDNN, cuBLAS), và nhiều thư viện khác.

4. **CUDA SDK:** CUDA SDK cung cấp các ví dụ và tài liệu hướng dẫn để bạn có thể bắt đầu phát triển ứng dụng sử dụng CUDA.
5. **NVIDIA GPU Driver:** CUDA Toolkit yêu cầu cài đặt đúng phiên bản driver cho GPU của bạn để hoạt động.

CUDA Toolkit thường được sử dụng trong lĩnh vực deep learning, scientific computing, và các ứng dụng đòi hỏi tính toán song song và xử lý số lớn. Nó cho phép các nhà phát triển tận dụng sức mạnh của GPU để gia tăng hiệu suất tính toán và tốc độ xử lý dữ liệu cho các ứng dụng của họ.

### 3.1.4 cuDNN

Đây là một thư viện phần mềm do NVIDIA phát triển, được tối ưu hóa cho việc thực hiện các phép tính liên quan đến mạng nơ-ron trên GPU. Tên gọi "cuDNN" viết tắt của "CUDA Deep Neural Network library," cho thấy rằng nó được phát triển để hỗ trợ hiệu suất cao khi sử dụng GPU để huấn luyện và triển khai các mô hình mạng nơ-ron sâu (deep neural networks).

Thư viện cung cấp một loạt các chức năng và thư viện tối ưu hóa để thực hiện các phép tính phổ biến trong deep learning, bao gồm các phép tính như: Convolution, Pooling, Normalization, Activation, Recurrent, ...

cuDNN được tích hợp vào nhiều framework deep learning phổ biến như TensorFlow và PyTorch, giúp cải thiện hiệu suất huấn luyện và triển khai mô hình trên GPU. Nó đóng vai trò quan trọng trong việc tăng tốc quá trình huấn luyện và triển khai mô hình deep learning trên nền tảng GPU của NVIDIA.

### 3.1.5 PaddleOCR

PaddleOCR là một dự án mã nguồn mở do PaddlePaddle phát triển, nhằm cung cấp một giải pháp toàn diện cho các nhiệm vụ liên quan đến xử lý ảnh và văn bản, bao gồm cả nhận dạng ký tự, nhận dạng văn bản và các tác vụ liên quan đến OCR. Dự án này được xây dựng trên cơ sở của các mô hình học

sâu và sử dụng các thuật toán tiên tiến để giải quyết các thách thức trong việc xử lý ảnh và văn bản.

PaddleOCR hỗ trợ nhiều tác vụ liên quan đến OCR như nhận dạng ký tự, nhận dạng văn bản và phân loại chữ viết tay. Có thể được đào tạo và sử dụng cho nhiều ngôn ngữ khác nhau, giúp phát triển ứng dụng OCR toàn cầu. Người dùng có thể tùy chỉnh và đào tạo lại các mô hình của PaddleOCR cho phù hợp với nhu cầu cụ thể của dự án.

PaddleOCR có thể hoạt động trên nhiều nền tảng khác nhau, bao gồm máy tính cá nhân, máy chủ và các môi trường đám mây. Dự án cung cấp một loạt các mô hình học sâu đã được đào tạo trước để giúp giải quyết các vấn đề liên quan đến xử lý ảnh và văn bản.

Dự án được tối ưu hóa để đạt hiệu suất cao và đáp ứng yêu cầu xử lý ảnh và văn bản trong thời gian thực.

Hơn nữa PaddleOCR là một dự án mã nguồn mở và có cộng đồng hỗ trợ sẵn sàng chia sẻ kiến thức, giải đáp thắc mắc và cùng nhau phát triển.

Tóm lại, cung cấp một giải pháp mạnh mẽ và linh hoạt cho các ứng dụng liên quan đến xử lý ảnh và văn bản, đặc biệt là trong lĩnh vực OCR. Điều này giúp đơn giản hóa và tối ưu hóa quá trình xây dựng hệ thống nhận dạng văn bản và thông tin từ các hình ảnh hóa đơn và tài liệu khác.

### 3.1.6 PPOCRLLabel

PPOCRLLabel là một công cụ hỗ trợ trong lĩnh vực xử lý ảnh và trí tuệ nhân tạo, được sử dụng để thực hiện công việc nhận dạng và đánh dấu vùng chứa văn bản trên ảnh. Đây là một dự án mã nguồn mở của PaddlePaddle, một thư viện học máy phát triển bởi Baidu. PaddleOCR nhằm mục tiêu xây dựng các mô hình nhận dạng ký tự trên ảnh với hiệu suất cao.

PPOCRLLabel được tạo ra để hỗ trợ quá trình chuẩn bị dữ liệu cho việc huấn luyện mô hình nhận dạng văn bản. Việc chuẩn bị dữ liệu là một bước quan trọng trong quá trình phát triển mô hình học máy, và công cụ như PPOCRLLabel giúp đơn giản hóa và tăng cường hiệu suất của quá trình này. Dưới đây là một số tính năng chính của PPOCRLLabel [19]:

1. **Labeling vùng chứa văn bản:** PPOCRLabel cho phép người dùng vẽ các hộp giới hạn xung quanh các vùng chứa văn bản trên ảnh để đánh dấu vị trí của văn bản cần nhận dạng.
2. **Labeling trích xuất từ khóa:** Người dùng có thể gán nhãn thông tin từ khóa để cho bài toán trích xuất thông tin chính.
3. **Gắn nhãn văn bản:** Người dùng có thể gắn nhãn văn bản được nhận dạng trong các hộp giới hạn để cho biết nội dung của văn bản đó.
4. **Chú thích cho bảng:** PPOCRLabel cung cấp cho người dùng chức năng chú thích bảng nhằm mục đích bóc tách cấu trúc của bảng dưới dạng hình ảnh và chuyển sang định dạng Excel
5. **Chỉnh sửa và xem trước:** PPOCRLabel cung cấp giao diện để chỉnh sửa và xem trước dữ liệu đã được đánh dấu trên ảnh, đảm bảo rằng dữ liệu được chuẩn bị chính xác trước khi sử dụng để huấn luyện mô hình.
6. **Xuất dữ liệu:** Sau khi hoàn thành việc đánh dấu và chuẩn bị dữ liệu, PPOCRLabel cho phép bạn xuất dữ liệu trong các định dạng phổ biến để sử dụng trong quá trình huấn luyện mô hình.
7. **Tích hợp với PaddleOCR:** PPOCRLabel có thể liên kết với dự án PaddleOCR để tiện lợi trong việc sử dụng dữ liệu đã được chuẩn bị để huấn luyện các mô hình nhận dạng văn bản.

PPOCRLabel là một công cụ cực kỳ hữu ích trong quá trình chuẩn bị dữ liệu cho việc huấn luyện mô hình liên quan đến OCR, giúp tăng cường hiệu suất và chính xác của mô hình cuối cùng.

### 3.1.7 Pytorch

PyTorch là một thư viện mã nguồn mở được phát triển bởi Facebook's AI Research lab dùng cho việc xây dựng và huấn luyện các mạng nơ-ron. Nó được thiết kế đặc biệt để hỗ trợ tính toán trên mạng nơ-ron và dự án thực tế với sự linh hoạt và hiệu suất cao. PyTorch là một trong những thư viện phổ

biến nhất cho deep learning và machine learning trong cộng đồng nghiên cứu và phát triển AI.

Một số điểm mạnh của PyTorch bao gồm:

1. Tính năng tạo đồ thị động: PyTorch sử dụng một cơ chế đồ thị tính toán động, cho phép bạn xây dựng và điều chỉnh mô hình một cách linh hoạt trong quá trình chạy. Điều này rất hữu ích khi bạn cần xây dựng các mô hình phức tạp hoặc thực hiện các thay đổi động.
2. Hỗ trợ GPU: PyTorch có tích hợp sẵn hỗ trợ tính toán trên GPU, giúp gia tăng tốc độ huấn luyện mô hình đáng kể.
3. Cộng đồng phát triển mạnh mẽ: PyTorch có một cộng đồng người dùng và phát triển đông đảo, nên bạn có thể tìm thấy nhiều tài liệu học tập và hỗ trợ từ cộng đồng.
4. Sử dụng dễ dàng: PyTorch có một cú pháp Python thân thiện và dễ đọc, giúp người dùng dễ dàng hiểu và sử dụng.

PyTorch cung cấp các thành phần cơ bản để xây dựng và huấn luyện mạng nơ-ron, bao gồm tensor, autograd (tự động tính đạo hàm), và một loạt các lớp và hàm tiện ích để xây dựng mô hình. Nó đã trở thành một trong những công cụ quan trọng cho nghiên cứu và phát triển trong lĩnh vực deep learning.

### 3.1.8 Gradio

Gradio là một thư viện mã nguồn mở được sử dụng để xây dựng giao diện người dùng cho các mô hình học máy và học sâu một cách dễ dàng và nhanh chóng. Thư viện này giúp tạo ra các ứng dụng web hoặc desktop đơn giản để tương tác với các mô hình đã huấn luyện mà bạn đã xây dựng hoặc sử dụng.

Thư viện cung cấp một API Python dễ sử dụng để bạn có thể định nghĩa các giao diện người dùng tương tác cho các mô hình học máy và học sâu của mình. Với Gradio, ta có thể tạo các biểu đồ, hộp văn bản, nút bấm và các phần tử UI khác để tương tác với mô hình. Gradio cũng hỗ trợ việc hiển thị kết quả và dự đoán từ mô hình trực tiếp trong giao diện người dùng.

Gradio được thiết kế để đơn giản hóa việc tạo giao diện người dùng cho các mô hình học máy mà không cần kiến thức về phát triển web phức tạp. Tương thích với nhiều loại mô hình có thể sử dụng Gradio với các mô hình học máy, deep learning và thậm chí cả các mô hình không cần kết nối mạng. Hỗ trợ cả ứng dụng web và desktop, cho phép triển khai giao diện người dùng trên nhiều nền tảng.

Thư viện có khả năng tích hợp với các framework học máy và deep learning phổ biến như TensorFlow, PyTorch, và Scikit-Learn. Người dùng có thể tùy chỉnh giao diện người dùng của mình bằng cách thêm các phần tử UI và tùy chỉnh cách hiển thị kết quả.

Gradio giúp nhanh chóng tạo ra các ứng dụng tương tác cho các mô hình học máy của bạn mà không cần kiến thức chuyên sâu về phát triển giao diện người dùng. Điều này có thể rất hữu ích trong việc trình bày và thử nghiệm các mô hình với người dùng cuối.

### 3.2 Cấu hình phần cứng

Huấn luyện một mô hình học sâu đòi hỏi xử lý một lượng lớn dữ liệu và tính toán phức tạp. Cần phần cứng mạnh mẽ để xử lý nhanh chóng và hiệu quả. Phần cứng yếu sẽ làm chậm quá trình huấn luyện. Phần cứng chất lượng cao cho phép huấn luyện nhanh hơn, với dữ liệu lớn hơn và mô hình phức tạp hơn. Do đó với từng tác vụ khác nhau em sẽ sử dụng một cấu hình khác nhau. Dưới đây là bảng so sánh phần cứng của từng tác vụ:

	<b>Detection</b>	<b>KIE</b>	<b>Recognition</b>
<b>Server</b>	Colab	Colab	Vast.ai
<b>OS</b>	Ubuntu 22.04.2	Ubuntu 22.04.2	Ubuntu 22.04.2
<b>Python</b>	3.10	3.10	3.10
<b>CUDA</b>	12.0	12.0	11.7
<b>Framework</b>	PaddlePaddle	PaddlePaddle	Pytorch
<b>RAM</b>	13GB	13GB	64GB
<b>GPU</b>	NVIDIA T4	NVIDIA T4	NVIDIA RTX 3060

Bảng 3.1: So sánh cấu hình phần cứng huấn luyện của từng tác vụ OCR

### 3.3 Tạo môi trường và cài công cụ cần thiết

#### 3.3.1 Cài đặt Miniconda:

Miniconda là một công cụ quản lý môi trường và gói thư viện Python nhẹ và tối giản. Dưới đây là hướng dẫn cách tạo môi trường Miniconda trên hệ điều hành Ubuntu:

##### Bước 1: Tải Miniconda

```
1 wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

##### Bước 2: Cài đặt Miniconda

```
1 sh Miniconda3-latest-Linux-x86_64.sh
```

Chấp nhận tất cả các điều khoản

```
1 source ~/.bashrc
```

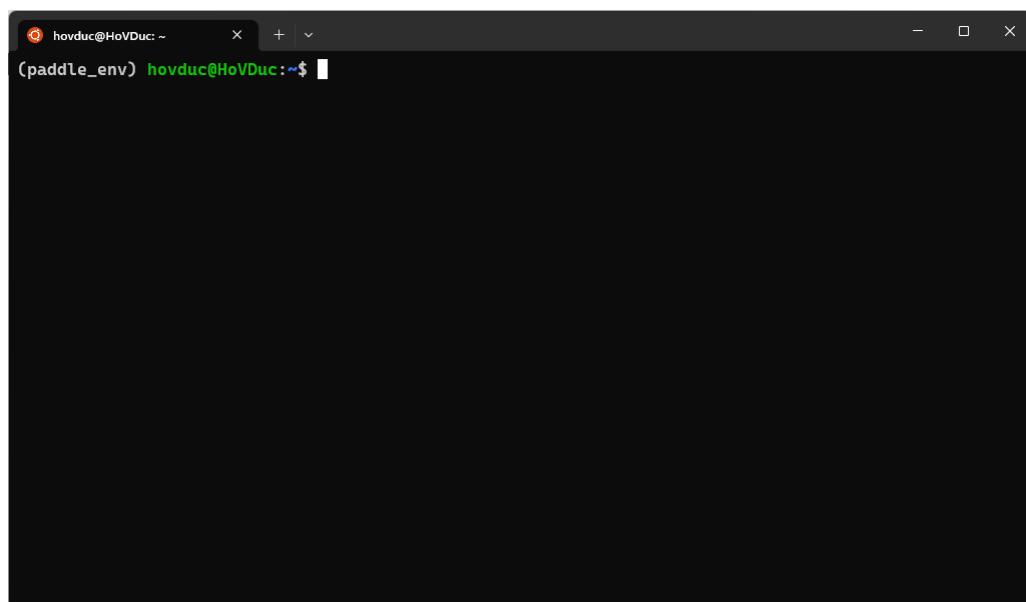
##### Bước 3: Tạo môi trường

```
1 conda create -n paddle_env python=3.8
```

Nhấn y để cài đặt

##### Bước 4: Kích hoạt môi trường

```
1 conda activate paddle_env
```



Hình 3.1: Môi trường đã được kích hoạt

### 3.3.2 Cài đặt CUDA Toolkit:

```

1 $ wget https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/
     x86_64/cuda-ubuntu2204.pin
2 $ sudo mv cuda-ubuntu2204.pin /etc/apt/preferences.d/cuda-repository-pin
     -600
3 $ wget https://developer.download.nvidia.com/compute/cuda/11.7.0/
     local_installers/cuda-repo-ubuntu2204-11-7-local_11.7.0-515.43.04-1
     _amd64.deb
4 $ sudo dpkg -i cuda-repo-ubuntu2204-11-7-local_11.7.0-515.43.04-1_amd64.deb
5 $ sudo cp /var/cuda-repo-ubuntu2204-11-7-local/cuda-*-keyring.gpg /usr/
     share/keyrings/
6 $ sudo apt-get update
7 $ sudo apt-get -y install cuda

```

### 3.3.3 Cài đặt cuDNN

#### Bước 1: Tải xuống cuDNN

```

1 wget https://developer.nvidia.com/downloads/compute/cudnn/secure/8.9.4/
     local_installers/11.x/cudnn-local-repo-ubuntu2204-8.9.4.25_1.0-1_amd64.
     deb

```

#### Bước 2: Cài đặt gói phần mềm

```

1 sudo dpkg -i cudnn-local-repo-ubuntu2204-8.9.4.25_1.0-1_amd64.deb

```

#### Bước 3: Nhập khóa CUDA GPG

```

1 sudo cp /var/cudnn-local-repo-*/cudnn-local-*-keyring.gpg /usr/share/
     keyrings/

```

#### Bước 4: Làm mới và cài đặt thư viện

```

1 sudo apt-get update
2 sudo apt-get install libcudnn8=8.9.4.25_1.0-1+cuda11.7
3 sudo apt-get install libcudnn8-dev=8.9.4.25_1.0-1+cuda11.7

```

### 3.3.4 Cài đặt PaddleOCR

#### Bước 1: Cài đặt PaddlePaddle

```

1 python3 -m pip install paddlepaddle -i https://mirror.baidu.com/pypi/
     simple

```

#### Bước 2: Clone PaddleOCR và cài đặt

```

1 git clone https://github.com/PaddlePaddle/PaddleOCR.git
2 pip install -r PaddleOCR/requirements.txt

```

### 3.3.5 Cài đặt PPOCRLLabel

```
1 pip3 install PPOCRLLabel  
2 pip3 install trash-cli
```

Trong chương này đã giới thiệu các công cụ và thư viện quan trọng trong đồ án. Phần tiếp theo em sẽ trình bày về dữ liệu và các bước huấn luyện mô hình OCR.

## Chương 4

# Thu thập dữ liệu và Huấn luyện mô hình

Trong chương này của báo cáo, em sẽ trình bày quá trình thu thập xử lý dữ liệu và quá trình huấn luyện mô hình OCR

### 4.1 Thu thập và chuẩn bị dữ liệu huấn luyện

#### 4.1.1 Lựa chọn hóa đơn

Loại hóa đơn đề tài lựa chọn là hóa đơn bán hàng ở cửa hàng tiện lợi và siêu thị. Đây một dạng thông tin thô có rất nhiều thông tin cần được lưu trữ và xử lý.

Hóa đơn bán hàng cửa hàng tiện lợi và siêu thị được lựa chọn vì nó thường chứa các thông tin cần thiết như địa chỉ, ngày mua, sản phẩm, số lượng, cũng như các khoản phí.... Loại hóa đơn này thường đa dạng về cấu trúc và kiểu dáng, bao gồm vùng văn bản in và cả phần hình ảnh với các dữ liệu chú thích. Do đó, ứng dụng công nghệ OCR để tự động nhận dạng và trích xuất thông tin từ loại hóa đơn này đem lại giá trị thực tiễn và hứa hẹn trong việc tối ưu hóa quá trình xử lý hóa đơn và quản lý tài liệu.

Để thực hiện đề tài này em sử dụng mẫu hóa đơn của cửa hàng tiện lợi Okono và VinMart (Hình 4.1) để thực hiện trích xuất thông tin.

Qua việc tập trung vào loại hóa đơn bán hàng cửa hàng tiện lợi và siêu thị,



Hình 4.1: Hóa đơn Okono(trái) và VinCommerce(phải)

mong muốn thực hiện một nghiên cứu chi tiết về cách ứng dụng công nghệ OCR vào việc nhận dạng và xử lý dữ liệu từ hóa đơn trong ngữ cảnh thực tế.

#### 4.1.2 Chuẩn bị dữ liệu

Chuẩn bị dữ liệu đây là một bước quan trọng để đảm bảo rằng mô hình OCR hoạt động tốt trên các dữ liệu thực tế. Trong phần này, em sẽ trình bày quá trình chuẩn bị và xử lý dữ liệu hình ảnh của hóa đơn. Tùy với nhiệm vụ khác nhau trong OCR ta có cách chuẩn bị dữ liệu khác nhau. Để thực hiện gán nhãn dữ liệu OCR, em sử dụng công cụ PPOCRLLabel. Dưới đây là các bước thực hiện:

**Bước 1:** Chuẩn bị tập dữ liệu cần gán nhãn Ở bước này em chuẩn bị hơn 75 ảnh hóa đơn của cửa hàng tiện lợi Okono và xác định số lớp cần gán nhãn cho bài toán KIE.

**Bước 2:** Chạy chương trình PPOCRLLabel

```
1 PPOCRLLabel --kie True # [KIE mode] for [detection + recognition +
keyword extraction] labeling
```

**Bước 3:** Nhấp vào nút "Open Dir" để mở thư mục chứa các hình ảnh cần gán nhãn. Các hình ảnh sẽ hiển thị trong danh sách.

**Bước 4:** Chọn một hình ảnh, sử dụng các công cụ vẽ để khoanh vùng văn bản và gán nhãn tương ứng. Có thể gán nhiều vùng văn bản cho một hình



Hình 4.2: Giao diện PPOCRLLabel



Hình 4.3: Danh sách ảnh PPOCRLabel

Ảnh.

- W: để tạo một hộp cho phát hiện văn bản.
  - Ctrl + E: Sửa nhãn của hộp.
  - Ctrl + X: Thay đổi lớp của nhãn dùng cho gán nhãn KIE.

Ở đây số lớp của bài toán KIE em xác định bao gồm các lớp sau:

- OTHER:
  - SELLER: Tên cửa hàng

- ADDRESS: Địa chỉ
- TIMESTAMP: Thời gian mua
- STAFF: Nhân viên bán hàng
- PRODUCT: Sản phẩm đã mua
- NUMBER: Số lượng sản phẩm
- PRICE: Giá của sản phẩm
- TOTAL\_COST: Tổng tiền thanh toán



Hình 4.4: Gán nhãn ảnh với PPOCRLLabel

**Bước 5:** Lưu các thay đổi bằng nút "Save" hoặc tự động lưu khi thoát chương trình.

**Bước 6:** Tiếp tục gán nhãn cho các hình ảnh còn lại.

**Bước 7:** Nhấn nút "Export Recognition Result" để cắt ảnh cho tác vụ training nhận dạng văn bản.

**Bước 8:** Kết thúc chương trình

Sau khi gán nhãn xong ta thu được một file Label.txt, file có format như sau:

```
1   image_files/IMG20230626141447.jpg [{"transcription": "Okono", "points": [[440, 195], [807, 201], [805, 333], [438, 327]], "difficult": false, "key_cls": "SELLER"}...]
```



Hình 4.5: Dữ liệu hóa đơn được export từ PPOCRLLabel

```
2     image_files/IMG20230628105753.jpg [{"transcription": "Okono", "points":  
    [[429, 469], [700, 469], [700, 542], [429, 542]], "difficult": true,  
    "key_cls": "SELLER"}...]
```

Các điểm trong các từ điển đó biểu thị tọa độ  $(x, y)$  của bốn điểm của hộp văn bản, được sắp xếp theo chiều kim đồng hồ từ điểm ở góc trên bên trái. Trong transcription đại diện cho văn bản trong hộp, và key\_cls đại diện cho lớp của văn bản.

Dữ liệu cho nhận dạng văn bản khi export từ PPOCRLabel có format như sau:

	#Tên ảnh	#Thông tin chú thích ảnh
1	IMG20230626141447_crop_0.jpg	Okono
2	IMG20230626141447_crop_2.jpg	PHIẾU BÁN HÀNG/ INVOICE
3	IMG20230626141447_crop_5.jpg	Mã số BBB
4	IMG20230626141447_crop_6.jpg	Tên hàng
5	IMG20230626141447_crop_7.jpg	SL
6	...	

Để cho đúng dữ liệu cho bài toán này ta cần đưa nó về chuẩn LMDB:

#### 4.1.3 Tổng quan về dữ liệu đào tạo

Dữ liệu cho nhiệm vụ phát hiện văn bản và KIE có cấu trúc như sau:

```
1 |-train_data  
2   |- invoice  
3     |- train  
4       |- image_files  
5         |- image_001.jpg
```

```

6             |- image_002.jpg
7             |
8             | ...
9             | train.txt
10            |- valid
11            |-
12            | image_files
13            |-
14            | image_001.jpg
15            |-
16            | image_002.jpg
17            | ...
18            | valid.txt

```

Bao gồm Dữ liệu cho nhiệm vụ phát hiện văn bản và KIE có cấu trúc như sau:

```

1   |-train_data
2       |- invoice-rec
3           |- images
4               |- image_001.jpg
5               |- image_002.jpg
6               |
7               | ...
8           |- train.txt
9           |- valid.txt

```

## 4.2 Huấn luyện mô hình

Để huấn luyện mô hình OCR cho nhiệm vụ nhận diện hóa đơn đây là một quá trình phức tạp đòi hỏi sự kết hợp giữa nhiều nhiệm vụ khác nhau, mỗi nhiệm vụ là một quá trình huấn luyện riêng biệt. Điều này đảm bảo chất lượng của quá trình nhận dạng như Phát hiện vùng văn bản, Nhận dạng văn bản và Hiểu tài liệu(Document Understanding).

### 4.2.1 Phát hiện văn bản

Ở tác vụ này em lựa chọn thuật toán DBNet để phát hiện vùng văn bản. Sau khi quyết định được mô hình, download pre-trained model đã được train sẵn với dataset tiếng anh là ICDAR2015 dataset. Với backbone là ResNet50\_vd

```

1 # download the pre-trained model ResNet50_vd
2 wget -P ./pretrain_models/ https://paddleocr.bj.bcebos.com/pretrained/
ResNet50_vd_ssld_pretrained.pdparams

```

Bắt đầu training

```

1 python3 tools/train.py -c configs/det/det_r50_vd_db.yml -o \

```

```

2     Global.pretrained_model='./pretrain_models/ResNet50_vd_ssld_pretrained.
3         pdparams' \
4             Train.dataset.data_dir='./train_data/Okono_3/train/image/' \
5             Train.dataset.label_file_list=['./train_data/Okono_3/train/train.txt'] \
6             \
7             Train.loader.batch_size_per_card=12 \
8             Eval.dataset.data_dir='./train_data/Okono_3/valid/image/' \
9             Eval.dataset.label_file_list=['./train_data/Okono_3/valid/valid.txt'] \
10            Global.save_model_dir='..../drive/MyDrive/save/' \
11            Global.checkpoints='/content/PaddleOCR/output/det_r50_vd' \
12            Global.eval_batch_step=100 \
13            Global.cal_metric_during_train=True \

```

```

1 [2023/08/25 11:14:07] ppocr INFO: Architecture :
2 [2023/08/25 11:14:07] ppocr INFO:      Backbone :
3 [2023/08/25 11:14:07] ppocr INFO:          layers : 50
4 [2023/08/25 11:14:07] ppocr INFO:          name : ResNet_vd
5 [2023/08/25 11:14:07] ppocr INFO:      Head :
6 [2023/08/25 11:14:07] ppocr INFO:          k : 50
7 [2023/08/25 11:14:07] ppocr INFO:          name : DBHead
8 [2023/08/25 11:14:07] ppocr INFO:      Neck :
9 [2023/08/25 11:14:07] ppocr INFO:          name : DBFPN
10 [2023/08/25 11:14:07] ppocr INFO:          out_channels : 256
11 [2023/08/25 11:14:07] ppocr INFO:      Transform : None
12 [2023/08/25 11:14:07] ppocr INFO:      algorithm : DB
13 [2023/08/25 11:14:07] ppocr INFO:      model_type : det
14 ...
15 eval model:: 100% 31/31 [00:03<00:00,  2.97it/s]
16 [2023/08/25 13:22:35] ppocr INFO: cur metric, precision:
17   0.2653061224489796, recall: 0.17067833698030635, hmean:
18   0.20772303595206393, fps: 14.512030517908638
17 [2023/08/25 13:22:35] ppocr INFO: best metric, hmean: 0.8958, is_float16:
18   False, start_epoch: 156, precision: 0.9292, recall: 0.8648, fps: 13.95,
19   best_epoch: 280

```

#### 4.2.2 Phát hiện văn bản

Với tác vụ nhận dạng văn bản em sử dụng VietOCR. Dưới đây là các bước huấn luyện mô hình:

```

1 from vietocr.tool.config import Cfg
2 from vietocr.model.trainer import Trainer

```

Load config VGG19 + Transformer và điều chỉnh tham số:

```

1 config = Cfg.load_config_from_name('vgg_transformer')
2 dataset_params = {
3     'name': 'hw',
4     'data_root': './train_data/invoice-rec/',

```

```

5         'train_annotation':'train.txt',
6         'valid_annotation':'valid.txt'
7     }
8
9     params = {
10         'print_every':200,
11         'valid_every':15*200,
12         'iters':20000,
13         'checkpoint': './checkpoint/transformerocr_checkpoint.pth',
14         'export': './weights/transformerocr.pth',
15         'metrics': 10000
16     }
17
18 config['trainer'].update(params)
19 config['dataset'].update(dataset_params)
20 config['device'] = 'cuda:0'

```

Bắt đầu huấn luyện mô hình:

```

1     trainer.train()

1 iter: 000200 - train loss: 2.627 - lr: 1.51e-05 - load time: 0.72 - gpu
    time: 97.80
2 iter: 000400 - train loss: 2.367 - lr: 2.45e-05 - load time: 0.06 - gpu
    time: 91.02
3 iter: 000600 - train loss: 2.212 - lr: 3.95e-05 - load time: 0.07 - gpu
    time: 93.54
4 ...
5 iter: 029600 - train loss: 0.526 - lr: 1.63e-07 - load time: 0.06 - gpu
    time: 84.60
6 iter: 029800 - train loss: 0.541 - lr: 4.14e-08 - load time: 0.06 - gpu
    time: 90.35
7 iter: 030000 - train loss: 0.550 - lr: 1.20e-09 - load time: 0.07 - gpu
    time: 90.14
8 iter: 030000 - valid loss: 0.551 - acc full seq: 0.8101 - acc per char:
    0.9543

```

#### 4.2.3 Key information extraction

Cuối cùng là mô hình KIE, ở đây em sử dụng LayoutXLM SER để huấn luyện. Dưới đây là các bước:

Download pretrained model và giải nén.

```

1     mkdir pretrained_model
2     cd pretrained_model
3     wget https://paddleocr.bj.bcebos.com/ppstructure/models/vi_layoutxlm/
        ser_vi_layoutxlm_xfund_pretrained.tar
4     tar -xf ser_vi_layoutxlm_xfund_pretrained.tar

```

Bắt đầu huấn luyện mô hình:

```

1 python tools/train.py -c ./configs/kie/vi_layoutxlm/
2   ser_vi_layoutxlm_xfund_zh.yml -o \
3     Architecture.Backbone=&num_classes \
4     Train.dataset.data_dir=train_data/invoice/train/image_files \
5     Eval.datasetdata_dir=train_data/invoice/valid/image_files \
6     PostProcess.class_path=train_data/invoice/class_list.txt

```

```

1 [2023/08/14 17:40:07] ppocr INFO: Architecture :
2 [2023/08/14 17:40:07] ppocr INFO:      Backbone :
3 [2023/08/14 17:40:07] ppocr INFO:          checkpoints : None
4 [2023/08/14 17:40:07] ppocr INFO:          mode : vi
5 [2023/08/14 17:40:07] ppocr INFO:          name : LayoutXLMForSer
6 [2023/08/14 17:40:07] ppocr INFO:          num_classes : 17
7 [2023/08/14 17:40:07] ppocr INFO:          pretrained : True
8 [2023/08/14 17:40:07] ppocr INFO:          Transform : None
9 [2023/08/14 17:40:07] ppocr INFO:          algorithm : LayoutXLM
10 [2023/08/14 17:40:07] ppocr INFO:          model_type : kie
11 ...
12 [2023/08/14 19:05:50] ppocr INFO: cur metric, precision: 0.9857, recall:
13   0.9928, hmean: 0.9892, fps: 24
14 [2023/08/14 19:08:10] ppocr INFO: save best model is to ./output/
    ser_vi_layoutxlm_xfund_zh/best_accuracy
[2023/08/14 19:08:10] ppocr INFO: best metric, hmean: 0.9857, precision:
  0.9928, recall: 0.9892, fps: 24, best_epoch: 57

```

### 4.3 Kết quả mô hình

Task	Model	Backbone	Hmean	Precision	Recall
Detection	DBNet	ResNet50vd	89.58%	92.92%	86.48%
KIE	LayoutXLM	LayoutXLM	98.57%	99.28%	98.92%
Task	Model	Backbone	Accuracy	Accuracy char	
Recognition	VietOCR	VGG19	81.01%	95.43%	

Bảng 4.1: Độ chính xác của cả 3 mô hình

Kết quả huấn luyện của mô hình đạt được đều thu được kết quả tốt, đặc biệt là với mô hình KIE cho ra độ chính xác cao nhất (Bảng 4.1). Tuy vậy ở tác vụ nhận dạng văn bản độ chính xác của mô hình trên toàn bộ chuỗi chỉ đạt 81.01%, điều này ảnh hưởng không ít đến thông tin được lưu trữ của đề tài.

Trong chương 4, em đã mô tả xong loại hóa đơn dùng cho đồ án, quá trình

thu thập và xử lý dữ liệu, các bước huấn luyện mô hình và kết quả huấn luyện của từng mô hình. Ở chương tiếp theo của báo cáo, em sẽ mô tả quá trình xây dựng chương trình hoàn chỉnh và trình bày các kết quả của chương trình.

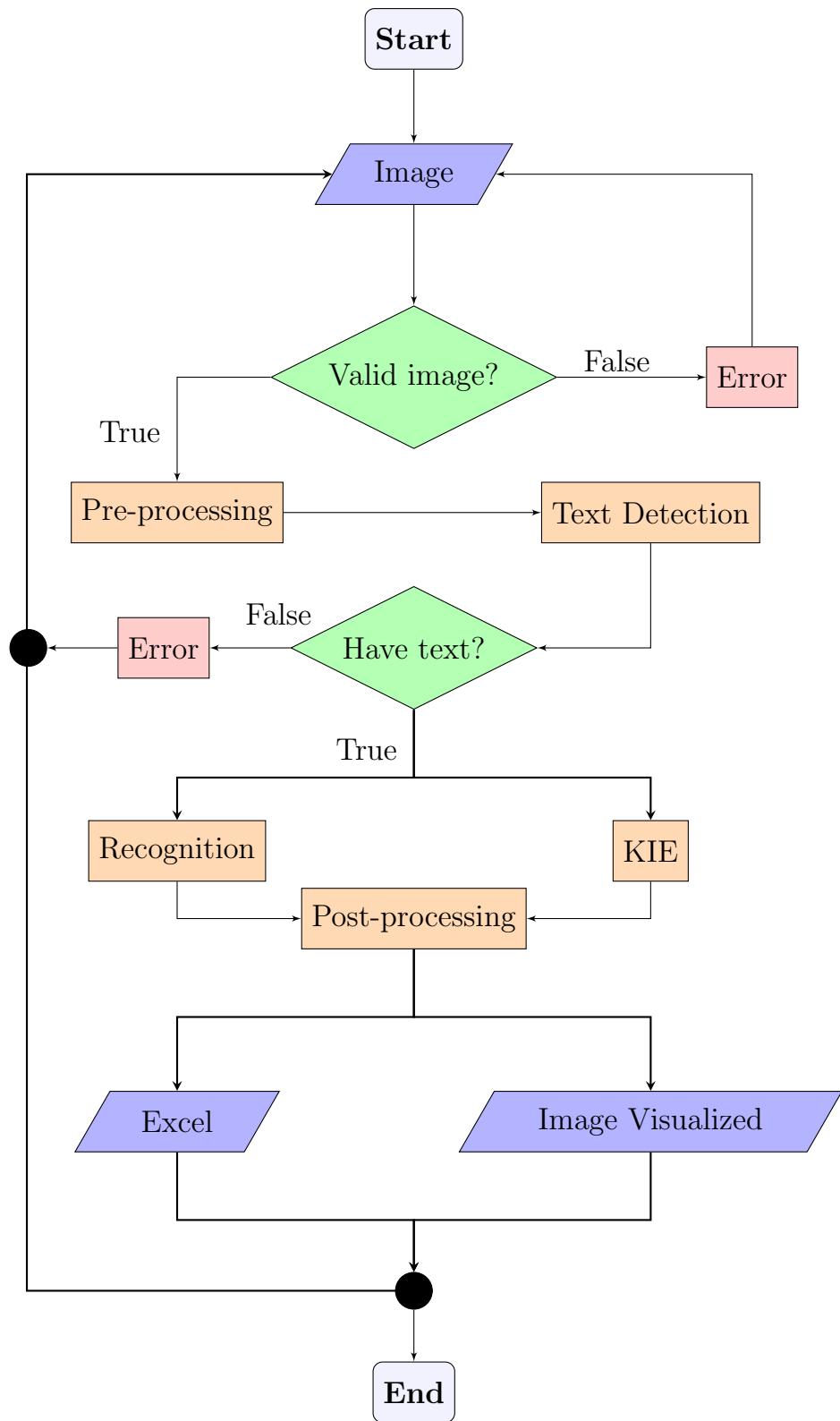
## Chương 5

# Xây dựng chương trình và Kết quả

Chương này, em mô tả quá trình xây dựng một chương trình ứng dụng hoàn chỉnh cho việc nhận dạng hóa đơn thông qua công nghệ OCR và kết quả đạt được của chương trình.

### 5.1 Ý tưởng

Để xây dựng được một chương trình hoàn chỉnh cho đề tài “**Nghiên cứu ứng dụng OCR nhận dạng hóa đơn**”, chương trình sẽ cần sự kết hợp của nhiều mô hình với các nhiệm vụ khác nhau để tạo thành một chương trình hoàn chỉnh Hình 5.1. Ý tưởng của chương trình này là từ bức ảnh đầu vào ảnh sẽ cho qua các bước tiền xử lý hình ảnh để tăng chất lượng ảnh đầu vào tiếp đến ảnh sẽ được cho qua bước phát hiện văn bản ở bước này đầu ra của ảnh sẽ là các 4 điểm tọa độ  $(x, y)$  của vùng văn bản đã được mô hình phát hiện, các điểm này sẽ được dùng làm đầu vào cho mô hình nhận dạng văn bản và KIE. Bước nhận dạng sẽ dự đoán văn bản trong vùng được phát hiện còn KIE sẽ thực hiện phân loại nhãn cho vùng văn bản. Sau đó cả hai sẽ được qua hàm hậu xử lý để cải thiện chất lượng và độ chính xác của văn bản dễ dàng cho các bước tiếp. Trích xuất các thông tin quan trọng và trực quan hóa hình ảnh, đưa về dữ liệu có cấu trúc ...



Hình 5.1: Lưu đồ của chương trình nhận dạng hóa đơn

## 5.2 Xây dựng chương trình

### 5.2.1 Tiền xử lý ảnh

```

1 def preprocess_image(_image):
2     _image = alpha_to_color(_image, alpha_color)
3     # Gaussian blur
4     _image = cv2.GaussianBlur(_image, (1,1), cv2.BORDER_DEFAULT)
5
6     # Image Sharpness
7     kernel = np.array([[-1,-1,-1],
8                         [-1, 9,-1],
9                         [-1,-1,-1]])
10    _image = cv2.filter2D(_image, -1, kernel)
11    if inv:
12        _image = cv2.bitwise_not(_image)
13    if bin:
14        _image = binarize_img(_image)
15    return _image

```

Ở đoạn code trên em sử dụng các phương pháp tiền xử lý ảnh cơ bản, để cải thiện chất lượng ảnh đầu vào cho chương trình.

### 5.2.2 Phát hiện vùng văn bản

Đoạn code dưới đây sử dụng thư viện PaddleOCR để xây dựng một ứng dụng phát hiện văn bản trong hình ảnh.

```

1 from paddleocr import PaddleOCR
2
3 self.ocr_engine = PaddleOCR(
4     use_angle_cls=False,
5     show_log=False,
6     det_model_dir=global_config.get("kie_det_model_dir", None),
7     use_gpu=global_config['use_gpu'])
8
9 det_result = self.ocr_engine(image)

```

Kết quả trả về là một danh sách các điểm tọa độ

```

1 [[27.0, 459.0], [136.0, 459.0], [136.0, 479.0], [27.0, 479.0]]
2 [[28.0, 429.0], [372.0, 429.0], [372.0, 445.0], [28.0, 445.0]]
3 .....

```

### 5.2.3 Nhận dạng văn bản

Với nhận dạng hóa đơn đầu vào của nhiệm vụ này là kết quả của 5.2.2. Hàm `recog` nhận 2 tham số đầu vào là ảnh đầu vào, và các điểm tọa độ, hàm sẽ thực hiện lấy các vùng văn bản và thực hiện dự đoán các văn bản trong từng `roi`.

```

1 from src.ocr.tools.predictor import Predictor
2 from src.ocr.tools.config import Cfg
3
4 self.config = Cfg.load_config_from_file(global_config['rec_config_path'])
5 self.config['predictor']['import'] = global_config['rec_weight']
6 self.config['predictor']['beamsearch'] = True
7 self.config['cnn']['pretrained'] = False
8 self.config['device'] = 'cuda' if global_config['use_gpu'] else 'cpu'
9 self.detector = Predictor(self.config)
10
11 def recog(image, boxes):
12     texts = []
13     for box in tqdm(boxes):
14         x, y, w, h = box
15         roi = Image.fromarray(image[y:h, x:w])
16         text = self.detector.predict(roi)
17         texts.append(text)
18     return texts
19
20 self.recog(image, det_result)

```

Kết quả trả về của nhiệm vụ này là list các văn bản đã được dự đoán như hình dưới đây.

```

1 [''Okono'', ''Số 85 Lê Văn Hiến, Đức Thắng, Bắc Từ Liêm'', ''PHIẾU BÁN HÀNG/
INVOICE'', ''Số:A3 1AA145550Ngày:18/05/2023 7:51:19PM'', ...]

```

### 5.2.4 KIE

Đoạn mã sau thực hiện quy trình dự đoán phân loại thông tin trong hình ảnh, từ việc chuẩn bị dữ liệu đến việc thực hiện dự đoán và xử lý kết quả sau khi dự đoán.

```

1 from ppocr.data import create_operators, transform
2 from ppocr.modeling.architectures import build_model
3 from ppocr.postprocess import build_post_process
4
5 self.post_process_class = build_post_process(config['PostProcess'],
6                                             global_config)

```

```

7 self.model = build_model(config['Architecture'])
8
9 transforms = []
10 for op in config['Eval']['dataset']['transforms']:
11     op_name = list(op)[0]
12     if 'Label' in op_name:
13         op[op_name]['ocr_engine'] = self.ocr_engine
14     elif op_name == 'KeepKeys':
15         op[op_name]['keep_keys'] = [
16             'input_ids', 'bbox', 'attention_mask', 'token_type_ids',
17             'image', 'labels', 'segment_offset_id', 'ocr_info',
18             'entities'
19         ]
20     transforms.append(op)
21
22 if config["Global"].get("infer_mode", None) is None:
23     global_config['infer_mode'] = True
24 self.ops = create_operators(config['Eval']['dataset']['transforms'],
25                             global_config)
26 self.model.eval()
27
28 batch = transform(data, self.ops)
29 batch = to_tensor(batch)
30 preds = self.model(batch)
31 post_result = self.post_process_class(
32     preds, segment_offset_ids=batch[6], ocr_infos=batch[7])

```

Kết quả sẽ trả về một danh sách các dictionary như sau:

```

1 [[[{'transcription': 'Okono', 'bbox': [435, 195, 811, 338], 'points':
2     [[437.0, 195.0], [811.0, 201.0], [808.0, 338.0], [435.0, 332.0]], 'pred_id': 1, 'pred': 'SELLER'}, {'transcription': 'Số 85 Lê Văn Hiến, Đ
3     ức Thắng, Bắc Từ Liêm', 'bbox': [309, 390, 890, 426], 'points':
4     [[309.0, 390.0], [890.0, 392.0], [890.0, 426.0], [309.0, 424.0]], 'pred_id': 3, 'pred': 'ADDRESS'}, ...]

```

### 5.2.5 Post-process

Đoạn mã sau có chức năng xử lý và trích xuất thông tin từ kết quả của quá trình nhận dạng hóa đơn, sau đó tạo một Pandas DataFrame để lưu trữ thông tin này và xuất ra một tệp Excel

```

1 def process_info(self, results):
2     info = {
3         'SELLER': '',
4         'ADDRESS': '',
5         'STAFF': '',
6         'TIMESTAMP': '',
7         'CODE': '',

```

```

8         'PRODUCTS': [],
9         'TOTAL_COST': 0
10    }
11
12    current_product = {
13        'PRODUCT': '',
14        'NUMBER': 0,
15        'PRICE': 0
16    }
17
18    products = []
19    for result in results:
20        label = result['pred']
21        transcription = result['transcription']
22
23        if label != '0':
24            if label in ['PRODUCT', 'NUMBER', 'PRICE']:
25                if label == 'PRODUCT':
26                    current_product['PRODUCT'] = transcription
27                    products.append(current_product.copy())
28                else:
29                    products[-1][label] = transcription
30            else:
31                if label == 'TIMESTAMP':
32                    text = transcription
33                    code, time = text.split('Ngày')
34                    info['CODE'] = code.strip()
35                    info['TIMESTAMP'] = time[1:]
36                else:
37                    info[label] = transcription
38
39    info['PRODUCTS'] = products
40
41    product_rows = []
42    for product in info_["PRODUCTS"]:
43        product_row = [info_["SELLER"], info_["ADDRESS"], info_["STAFF"],
44                      info_["TIMESTAMP"], info_["CODE"],
45                      product["PRODUCT"], product["NUMBER"], product["PRICE"]
46                      ], info_["TOTAL_COST"]]
47        product_rows.append(product_row)
48
49    # Create a Pandas DataFrame
50    columns = ["SELLER", "ADDRESS", "STAFF", "TIMESTAMP", "CODE", "PRODUCT",
51               "NUMBER", "PRICE", "TOTAL_COST"]
52    df = pd.DataFrame(product_rows, columns=columns)
53
54    # Create an Excel file
55    now = datetime.now()
56
57    current_time = now.strftime("%H-%M-%S")

```

```

55     output_file = "invoice-{}.xlsx".format(current_time)
56     df.to_excel(output_file, index=False, engine="openpyxl")
57
58     print(f"Excel file '{output_file}' created.")
59
60     return json.dumps(info, indent=1, ensure_ascii=False)

```

Đoạn mã sau vẽ kết quả của chương trình lên trên hình ảnh gốc.

```

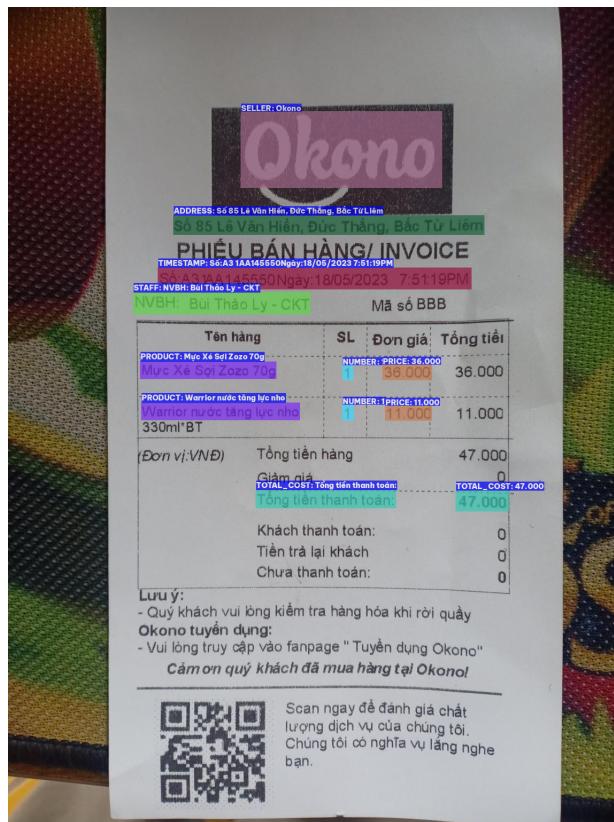
1 def draw_ser_results(image, ocr_results,
2                     font_path="doc/fonts/BeVietnamPro-Black.ttf",
3                     font_size=14):
4     np.random.seed(2021)
5     color = (np.random.permutation(range(255)),
6               np.random.permutation(range(255)),
7               np.random.permutation(range(255)))
8     color_map = {
9         idx: (color[0][idx], color[1][idx], color[2][idx])
10        for idx in range(1, 255)
11    }
12
13    if isinstance(image, np.ndarray):
14        image = Image.fromarray(image)
15    elif isinstance(image, str) and os.path.isfile(image):
16        image = Image.open(image).convert('RGB')
17    img_new = image.copy()
18    draw = ImageDraw.Draw(img_new)
19
20    font = ImageFont.truetype(font_path, font_size, encoding="utf-8")
21    for ocr_info in ocr_results:
22        if ocr_info["pred_id"] not in color_map:
23            continue
24        color = color_map[ocr_info["pred_id"]]
25        text = "{}: {}".format(ocr_info["pred"], ocr_info["transcription"])
26
27        if "bbox" in ocr_info:
28            bbox = ocr_info["bbox"]
29        else:
30            bbox = trans_poly_to_bbox(ocr_info["points"])
31        draw_box_txt(bbox, text, draw, font, font_size, color)
32
33    img_new = Image.blend(image, img_new, 0.7)
34    return np.array(img_new)

```

### 5.3 Kết quả

Dưới đây là một số kết quả đạt được của đề tài:

## Kết quả 1



Hình 5.2: Kết quả hình ảnh hóa đơn 1

```
1 {  
2     "SELLER": "Okono",  
3     "ADDRESS": "Số 85 Lê Văn Hiến, Đức Thắng, Bắc Từ Liêm",  
4     "STAFF": "NVBH: Bùi Thảo Ly - CKT",  
5     "TIMESTAMP": "18/05/2023 7:51:19PM",  
6     "CODE": "Số:A3 1AA145550",  
7     "PRODUCTS": [  
8         {  
9             "PRODUCT": "Mực Xé Sợi Zozo 70g",  
10            "NUMBER": "1",  
11            "PRICE": "36.000"  
12        },  
13        {  
14            "PRODUCT": "Warrior nước tăng lực nho",  
15            "NUMBER": "1",  
16            "PRICE": "11.000"  
17        }  
18    ]  
19 }
```

```

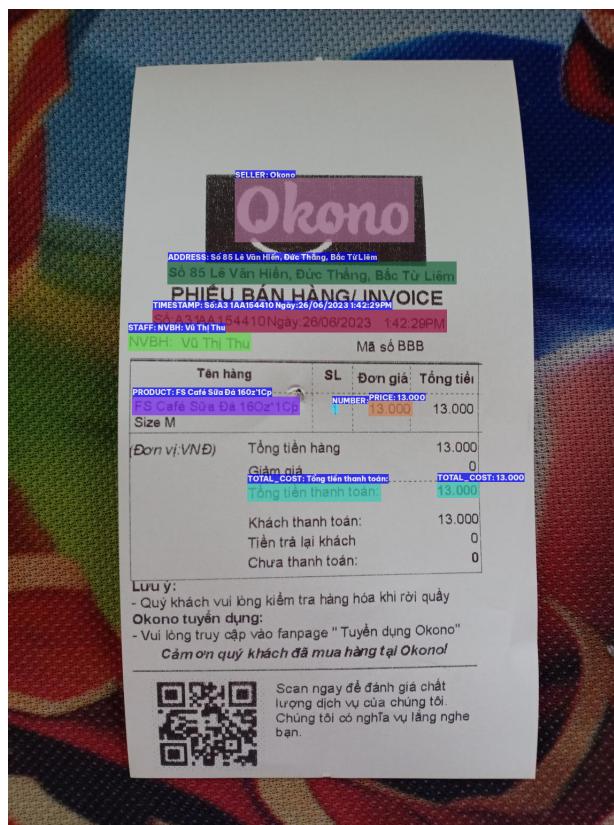
17     }
18   ],
19   "TOTAL_COST": "47.000"
20 }

```

SELLER	ADDRESS	STAFF	TIMESTAMP	CODE	PRODUCT	NUMBER	PRICE	TOTAL_COST
Okono	Số 85 Lê Văn Hiến, NVBH: Bùi Thảo Ly - 18/05/2023 7:51:19PM	Số:A3 1AA145550	Mực Xé Sợi Zozo 70g	1	36.000	47.000		
Okono	Số 85 Lê Văn Hiến, NVBH: Bùi Thảo Ly - 18/05/2023 7:51:19PM	Số:A3 1AA145550	Warrior nước tăng lực n1	1	11.000	47.000		

Hình 5.3: Kết quả Excel của hóa đơn 1

## Kết quả 2



Hình 5.4: Kết quả hình ảnh hóa đơn 2

```

1 {
2   "SELLER": "Okono",
3   "ADDRESS": "Số 85 Lê Văn Hiến, Đức Thắng, Bắc Từ Liêm",
4   "STAFF": "NVBH: Vũ Thị Thu",

```

```

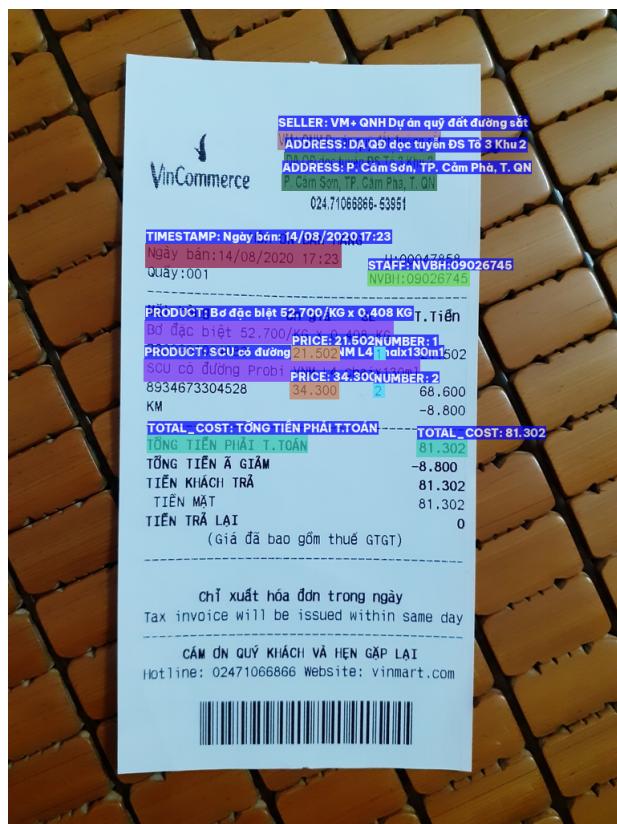
5   "TIMESTAMP": "26/06/2023 1:42:29PM",
6   "CODE": "Số:A3 1AA154410",
7   "PRODUCTS": [
8     {
9       "PRODUCT": "FS Café Sữa Đá 160z'1Cp",
10      "NUMBER": "1",
11      "PRICE": "13.000"
12    }
13  ],
14  "TOTAL_COST": "13.000"
15 }

```

SELLER	ADDRESS	STAFF	TIMESTAMP	CODE	PRODUCT	NUMBER	PRICE	TOTAL_COST
Okono	Số 85 Lê Văn Hiến, NVBH: Vũ Thị Thu	26/06/2023 1:42:29PM	Số:A3 1AA15 FS	Café Sữa Đá 1		1	13.000	13.000

Hình 5.5: Kết quả Excel của hóa đơn 2

### Kết quả 3



Hình 5.6: Kết quả hình ảnh hóa đơn 3

```

1 {
2   "SELLER": "VM+ QNH Dự án quỹ đất đường sắt",

```

```

3   "ADDRESS": "P. Cẩm Sơn, TP. Cẩm Phả, T. QN",
4   "STAFF": "NVBH:09026745",
5   "TIMESTAMP": "14/08/2020 17:23",
6   "CODE": "",
7   "PRODUCTS": [
8     {
9       "PRODUCT": "Bơ đặc biệt 52.700/KG x 0,408 KG",
10      "NUMBER": 0,
11      "PRICE": 0
12    },
13    {
14      "PRODUCT": "SCU có đường Probi VNM L4 chaix130ml",
15      "NUMBER": "2",
16      "PRICE": "34.300"
17    }
18  ],
19  "TOTAL_COST": "81.302"
20
21 ]

```

SELLER	ADDRESS	STAFF	TIMESTAMP	CODE	PRODUCT	NUMBER	PRICE	TOTAL_COST
VM+ QNH Dự án quý đất P. Cẩm Sơn, TP. NVBH:09026745 bán: 14/08/2020 17:23					Bơ đặc biệt 52.	0	0	81.302
VM+ QNH Dự án quý đất P. Cẩm Sơn, TP. NVBH:09026745 bán: 14/08/2020 17:23					SCU có đường	2	34.300	81.302

Hình 5.7: Kết quả Excel của hóa đơn 3

Điều này đã kết thúc chương 5, trong chương này em đã mô tả quá trình xây dựng chương trình hoàn chỉnh của đề tài bao gồm kết quả của chương trình. Kết quả đạt được của chương trình cho người dùng có một cái nhìn tổng quát về những gì mà mô hình trích xuất, và thông tin hóa đơn được chương trình lưu trữ lại dưới dạng dữ liệu có cấu trúc.

## Kết luận

Sau quá trình nghiên cứu và tìm hiểu các phương pháp, thuật toán liên quan đến OCR cùng với sự nỗ lực của bản thân em đã hoàn thành đề tài kịp tiến độ đề tài “**Nghiên cứu ứng dụng công nghệ OCR nhận dạng hóa đơn**” và đạt được những kết quả sau:

1. Tìm hiểu và áp dụng thành công các thuật toán “state-of-the-art” cho đề tài
2. Huấn luyện thành công các mô hình cho nhiệm vụ khác nhau phát hiện văn bản, nhận dạng văn bản, KIE và đạt hiệu quả tốt trên bộ dữ liệu thu thập được.
3. Xây dựng hoàn chỉnh chương trình trích xuất thông tin hóa đơn.
4. Thành công chuyển đổi ảnh hóa đơn thành dạng văn bản có thể xử lý được.
5. Trích xuất được các thông tin quan trọng của hóa đơn như tên cửa hàng, địa chỉ, ngày mua, sản phẩm, ... Và lưu trữ thông tin dưới dạng văn bản có cấu trúc.

Như vậy, đề tài đã hoàn thành mục tiêu đề ra và đạt được những kết quả quan trọng trong việc ứng dụng công nghệ OCR để nhận dạng và trích xuất thông tin từ hóa đơn.

### Những hạn chế của đề tài

Tuy vậy, đề tài còn một số hạn chế đáng chú ý như sau:

1. Đề ảnh hưởng bởi nhiều yếu tố như chất lượng hình ảnh, font chữ, định dạng hóa đơn
2. Mô-đun phát hiện văn bản vẫn chưa phát hiện được hầu hết văn bản. Điều này dẫn đến thiếu sót thông tin trong quá trình nhận diện và trích xuất thông tin.

3. Mô-đun nhận diện văn bản, dễ bị ảnh hưởng bởi chất lượng hình ảnh, và nhiễu. Điều này làm cho quá trình nhận dạng trở lên sai sót ảnh hưởng đến thông tin lưu trữ.
4. Mẫu hóa đơn chưa được đa dạng và phong phú, chỉ nhận dạng trên vào mẫu hóa đơn nhất định.

## Hướng phát triển

Trong tương lai, Em sẽ tiếp tục tìm hiểu và nghiên cứu thêm để khắc phục các vấn đề còn tồn tại của đề tài. Phát triển và mở rộng phạm vi nghiên cứu nhằm phát triển các kỹ năng và kiến thức để cải thiện bản thân. Dưới đây là những dự định phát triển:

1. Khắc phục các hạn chế ở mục hạn chế, tối ưu hóa độ chính xác, thời gian chạy và đa dạng mẫu hóa đơn ,và cải thiện khả năng nhận dạng trong các trường hợp phức tạp.
2. Xây dựng ứng dụng di động cho phép người dùng chụp ảnh hóa đơn bằng điện thoại di động và tự động đưa ra thông tin liên quan. Điều này sẽ giúp tạo ra một trải nghiệm tiện lợi và linh hoạt cho người dùng.
3. Tích hợp ứng dụng OCR vào hệ thống quản lý tài liệu và dữ liệu của doanh nghiệp. Điều này sẽ giúp tự động hóa quy trình làm việc, giảm thiểu thời gian và nguy cơ sai sót trong việc nhập liệu thủ công.

Trải qua quá trình nghiên cứu và thử nghiệm, Em nhận thấy rằng OCR có tiềm năng lớn trong việc tối ưu hóa quá trình nhận dạng và nhập liệu từ hóa đơn. Nhờ vào việc tự động nhận dạng ký tự, giảm thiểu sự phụ thuộc vào công việc thủ công, giảm nguy cơ sai sót từ việc nhập liệu thủ công và tăng khả năng đáp ứng nhanh chóng các yêu cầu kinh doanh.

Tổng kết lại, “nghiên cứu về ứng dụng OCR trong nhận dạng hóa đơn” là một bước tiến quan trọng trong việc áp dụng công nghệ vào thực tiễn kinh doanh. Tin rằng sự phát triển của OCR cùng với sự tương tác với các công

nghệ khác sẽ mở ra những cơ hội mới và đóng góp tích cực cho sự phát triển của nền kinh tế và xã hội.

## Tài Liệu Tham Khảo

- [1] FPT.AI. “Nâng cấp quy trình số hóa tài liệu của doanh nghiệp với công nghệ ocr.” (2020), [Online]. Available: <https://fpt.ai/vi/nang-cap-quy-trinh-so-hoa-tai-lieu-cua-doanh-nghiep-voi-cong-nghe-ocr> (visited on 07/08/2023).
- [2] AWS. “Ocr (nhận dạng ký tự quang học) là gì?” (), [Online]. Available: <https://aws.amazon.com/vi/what-is/ocr/> (visited on 07/08/2023).
- [3] A. Singh, K. Bacchuwar, and A. Bhasin, “A survey of ocr applications,” *International Journal of Machine Learning and Computing (IJMLC)*, Jan. 2012. DOI: [10.7763/IJMLC.2012.V2.137](https://doi.org/10.7763/IJMLC.2012.V2.137).
- [4] Wikipedia. “Optical character recognition.” (2002), [Online]. Available: [https://en.wikipedia.org/wiki/Optical\\_character\\_recognition](https://en.wikipedia.org/wiki/Optical_character_recognition) (visited on 07/08/2023).
- [5] D. V. Everen. “The history of ocr.” (2023), [Online]. Available: <https://www.veryfi.com/ocr-api-platform/history-of-ocr/> (visited on 07/08/2023).
- [6] C. Li, W. Liu, R. Guo, et al., *Dive into OCR*. 2022, [https://paddleocr.bj.bcebos.com/ebook/Dive\\_into\\_OCR.pdf](https://paddleocr.bj.bcebos.com/ebook/Dive_into_OCR.pdf).
- [7] K. Lopez-Nichol. “Template-based vs. ai-based ocr: Which is right for you?” (2023), [Online]. Available: <https://www.veryfi.com/technology/template-based-vs-ai-based-ocr/>.
- [8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [9] Wikipedia. “Deep learning.” (2012), [Online]. Available: [https://en.wikipedia.org/wiki/Deep\\_learning#Neural\\_networks](https://en.wikipedia.org/wiki/Deep_learning#Neural_networks) (visited on 10/08/2023).
- [10] AWS. “Học sâu là gì?” (), [Online]. Available: <https://aws.amazon.com/vi/what-is/deep-learning/> (visited on 10/08/2023).

- [11] K. He, X. Zhang, S. Ren, and J. Sun, *Identity mappings in deep residual networks*, 2016. arXiv: [1603.05027 \[cs.CV\]](https://arxiv.org/abs/1603.05027).
- [12] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: [1512.03385 \[cs.CV\]](https://arxiv.org/abs/1512.03385).
- [13] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: [1706.03762 \[cs.CL\]](https://arxiv.org/abs/1706.03762).
- [14] Y. Xu, T. Lv, L. Cui, *et al.*, *Layoutxlm: Multimodal pre-training for multilingual visually-rich document understanding*, 2021. arXiv: [2104.08836 \[cs.CL\]](https://arxiv.org/abs/2104.08836).
- [15] Y. Xu, Y. Xu, T. Lv, *et al.*, *Layoutlmv2: Multi-modal pre-training for visually-rich document understanding*, 2022. arXiv: [2012.14740 \[cs.CL\]](https://arxiv.org/abs/2012.14740).
- [16] M. Liao, Z. Wan, C. Yao, K. Chen, and X. Bai, *Real-time scene text detection with differentiable binarization*, 2019. arXiv: [1911.08947 \[cs.CV\]](https://arxiv.org/abs/1911.08947).
- [17] X. Feng, H. Yao, Y. Qi, J. Zhang, and S. Zhang, *Scene text recognition via transformer*, 2020. arXiv: [2003.08077 \[cs.CV\]](https://arxiv.org/abs/2003.08077).
- [18] J. W. Lum. “Ai vs traditional ocr.” (2020), [Online]. Available: <https://medium.com/staple-ai/artificial-intelligence-vs-traditional-ocr-how-staple-rides-the-wave-of-emerging-technologies-e60f2d295a26> (visited on 12/08/2023).
- [19] Evezerest. “Ppocrlabel.” (2022), [Online]. Available: <https://github.com/Evezerest/PPOCRLLabel>.