

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



**ĐOÀN NHẬT SANG , LÊ NGÔ MINH ĐỨC
TRƯƠNG VĂN KHẢI, ĐẶNG PHƯỚC SANG**

**SCENE TEXT DETECTION AND RECOGNITION ON
VIETNAMESE**

ĐỒ ÁN XỬ LÝ ẢNH VÀ ỨNG DỤNG

TP. HCM, 2023

ĐẠI HỌC QUỐC GIA TP. HCM
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



ĐOÀN NHẬT SANG – 21522542

LÊ NGÔ MINH ĐỨC – 21520195

TRƯỜNG VĂN KHẢI – 21520274

ĐẶNG PHƯỚC SANG – 21521377

**SCENE TEXT DETECTION AND RECOGNITION ON
VIETNAMESE**

ĐỒ ÁN XỬ LÝ ẢNH VÀ ỨNG DỤNG

GVHD

TH.S CÁP PHẠM ĐÌNH THĂNG

TP. HCM, 2023

LỜI CAM ĐOAN

Chúng tôi xin cam đoan nội dung được trình bày trong đồ án “SCENE TEXT DETECTION AND RECOGNITION ON VIETNAMESE” là do chúng tôi nghiên cứu, tìm hiểu và phát triển dưới sự dẫn dắt của ThS. Cáp Phạm Đình Thăng. Đồ án không sao chép từ các tài liệu, công trình nghiên cứu của người khác mà không ghi rõ trong tài liệu tham khảo. Tất cả những tham khảo từ các nghiên cứu liên quan đều được nêu nguồn gốc một cách rõ ràng từ danh mục tài liệu tham khảo trong luận văn. Chúng tôi xin chịu trách nhiệm về lời cam đoan này.

TP. HCM, ngày 19 tháng 12 năm 2023

LỜI CẢM ƠN

Chúng tôi xin chân thành cảm ơn thầy giáo, ThS. Cáp Phạm Đình Thăng, người đã định hướng, giúp đỡ, trực tiếp hướng dẫn và tận tình chỉ bảo tôi trong suốt quá trình nghiên cứu, xây dựng và hoàn thành đồ án này.

Chúng tôi cũng xin được cảm ơn tới gia đình, những người thân và bạn bè thường xuyên quan tâm, động viên, chia sẻ kinh nghiệm, cung cấp các tài liệu hữu ích trong thời gian học tập, nghiên cứu cũng như trong suốt quá trình thực hiện đồ án.

TP. HCM, ngày 19 tháng 12 năm 2023

MỤC LỤC

MỞ ĐẦU	1
CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN PHÁT HIÊN VÀ NHẬN DẠNG VĂN BẢN TIẾNG VIỆT TRONG CẢNH TRÊN	4
1.1 PHÁT BIỂU BÀI TOÁN	4
1.2 CÁC THÁCH THỨC CỦA BÀI TOÁN.....	4
1.2.1 CÁC THÁCH THỨC BÊN NGOÀI.....	5
1.2.2 CÁC THÁCH THỨC BÊN TRONG.....	8
1.2.3 TÔNG KẾT VỀ CÁC THÁCH THỨC CỦA BÀI TOÁN	9
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	11
2.1 CÁC PHƯƠNG PHÁP TĂNG CƯỜNG DỮ LIỆU.....	11
2.1.1 RANDOM ROTATION	11
2.1.2 RANDOM CROPPING	11
2.1.3 RANDOM FLIPPING	11
2.1.4 AUTO CONTRAST.....	12
2.1.5 EQUALIZE.....	12
2.1.6 SOLARIZEADD.....	13
2.1.7 COLOR.....	13
2.1.8 POSTERIZE.....	14
2.1.9 CONTRAST.....	14
2.1.10 BRIGHTNESS.....	15
2.1.11 SHEAR	15
2.1.12 TRANSLATE.....	16
2.1.13 INVERT	16
2.1.14 GAUSSIAN BLUR	17
2.1.15 POISSON NOISE	17
2.2 CÁC KĨ THUẬT DEEP LEARNING LIÊN QUAN	18
2.2.1 RESNET-50.....	18
2.2.2 TRANSPOSED CONVOLUTION	19
2.2.3 DEFORMABLE CONVOLUTION	20
2.2.4 CÓ CHÉ ATTENTION TRONG TRANSFORMER [4].....	21
2.2.5 VISION TRANSFORMER	29
2.3 CÁC PHƯƠNG PHÁP TRÊN BỘ DỮ LIỆU VINTEXT [6]	31
2.3.1 ABCNET [7].....	31
2.3.2 MASK TEXTSPOTTER V3 [8]	31
2.3.3 DICTIONARY-GUIDED SCENE TEXT RECOGNITION [6]	32
CHƯƠNG 3: BỘ DỮ LIỆU VINTEXT	34
3.1 KHẢO SÁT DỮ LIỆU	34
3.2 BỘ DỮ LIỆU VIN-TEXT [6]	34

CHƯƠNG 4: PHƯƠNG PHÁP ĐÁNH GIÁ.....	36
4.1 ĐÁNH GIÁ RIÊNG CHO DETECTION	36
4.2 ĐÁNH GIÁ RIÊNG CHO RECOGNITION	38
4.3 ĐÁNH GIÁ CHO MÔ HÌNH END TO END	39
CHƯƠNG 5: HUỐNG TIẾP CẬN	40
5.1 HUỐNG TIẾP CẬN CHUNG	40
5.2 HUỐNG TIẾP CẬN CỦA NHÓM.....	40
5.2.1 DBNETPP (DETECTION)	41
5.2.2 PARSEQ (RECOGNITION).....	50
CHƯƠNG 6: THỰC NGHIỆM.....	58
6.1 CÀI ĐẶT DBNETPP	58
6.1.1 THÔNG SỐ MÔ HÌNH	58
6.1.2 TĂNG CUỒNG DỮ LIỆU	61
6.1.3 MÔI TRƯỜNG HUÂN LUYỆN VÀ CÁC SIÊU THAM SỐ KHÁC	61
6.2 CÀI ĐẶT PARSEQ	62
6.2.1 THÔNG SỐ MÔ HÌNH	62
6.2.2 TĂNG CUỒNG DỮ LIỆU	62
6.2.3 MÔI TRƯỜNG HUÂN LUYỆN VÀ CÁC SIÊU THAM SỐ KHÁC	63
6.3 KẾT QUẢ THỰC NGHIỆM	64
6.3.1 CHỌN THRESHOLD CHO DBNETPP.....	64
6.3.2 KẾT QUẢ ĐÁNH GIÁ.....	65
6.4 PHÂN TÍCH LỖI SAI	66
6.4.1 DETECTION	66
6.4.2 RECOGNITION	70
6.5 NHẬN XÉT.....	73
CHƯƠNG 7: XÂY DỰNG DEMO.....	75
7.1 CÁC CÔNG CỤ CHÍNH HỖ TRỢ.....	75
7.1.1 FASTAPI	75
7.1.2 REACTJS	75
7.2 PHÂN TÍCH THIẾT KẾ HỆ THỐNG	76
7.3 THIẾT KẾ GIAO DIỆN	77
7.4 HUỐNG DẪN SỬ DỤNG.....	78
7.5 MÔI TRƯỜNG CÀI ĐẶT	79
KẾT LUẬN.....	80
TÀI LIỆU THAM KHẢO	82

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Tù viết tắt	Tù chuẩn	Điễn giải
AI	Artificial Intelligence	Trí tuệ nhân tạo
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
CV	Computer Vision	Thị giác máy tính
CNN	Convolution Neural Network	Mạng nơ-ron tích chập
MLP	Multi Layer Perceptron	Mạng perceptron nhiều lớp
DCN	Deformable Convolution Network	Mạng tích chập biến dạng
PLM	Permutation Language Modeling	Mô hình ngôn ngữ hoán vị
MHA	Multi-head Attention	Kỹ thuật attention nhiều head
MHSA	Multi-head Self Attention	Kỹ thuật self-attention nhiều head
SD	Standard Binarization	Nhị phân hoá tiêu chuẩn
DB	Differentiable Binarization	Nhị phân hoá khả vi
AR	Auto Regressive	Tự động hội quy
NAR	Non-Auto Regressive	Không tự động hội quy
IR	Iterative Refinement	Điều chỉnh lặp

DANH MỤC BẢNG

<i>Table 6.1: Thông số của các thao tác tăng cường dữ liệu cho PARSeq.....</i>	63
<i>Table 6.2: Kết quả đánh giá trên tập test (threshold = 0.7 cho detectoin và end to end).....</i>	65
<i>Table 6.3: Hmean của một số phương pháp trên tập VinText. Những phương pháp +D sử dụng kĩ thuật dictionary guided, + dictionary sử dụng bộ tự vựng trong quá trình suy luận.....</i>	66
<i>Table 6.4: Hình ảnh kích thước nhỏ, mờ, nhoè.....</i>	70
<i>Table 6.5: Nền phông tạp.....</i>	71
<i>Table 6.6: Phông chữ đặc biệt, chữ viết tay</i>	71
<i>Table 6.7: Văn bản mà người còn khó đọc</i>	72
<i>Table 6.8: Nhầm văn bản bình thường thành văn bản nghệ thuật</i>	72
<i>Table 6.9: Nhầm lẫn ký tự hoá – thường</i>	72
<i>Table 6.10: Nhầm lẫn giữa các ký tự gần giống nhau (do bản chất gần giống nhau hoặc do môi trường làm biến đổi ký tự trở nên gần giống với ký tự khác)</i>	73
<i>Table 6.11: Không nhận dạng được các ký tự đặc biệt</i>	73
<i>Table 6.12: Dữ liệu bị gán nhãn sai</i>	73

DANH MỤC HÌNH VẼ VÀ ĐỒ THỊ

Figure 1.1: Mô tả đầu vào và đầu ra của bài toán.....	4
Figure 1.2: Trường hợp văn bản chụp bị che khuất trong ảnh	5
Figure 1.3: Trường hợp văn bản bị mất chữ trong ảnh.....	6
Figure 1.4: Trường hợp chất lượng ảnh thấp.....	7
Figure 1.5: Trường hợp bối cảnh chụp phức tạp	7
Figure 1.6: Trường hợp góc chụp xa.....	8
Figure 1.7: Sự khác nhau về kích thước các ảnh đầu vào	9
Figure 1.8: Thách thức về cấu trúc của các trường văn bản	9
Figure 2.1: Ví dụ về random rotation	11
Figure 2.2: Ví dụ về random cropping	11
Figure 2.3: Ví dụ về random flipping	12
Figure 2.4: Ví dụ về auto contrast	12
Figure 2.5: Ví dụ về equalize histogram.....	13
Figure 2.6: Ví dụ về solarize add.....	13
Figure 2.7: Ví dụ về coloring.....	14
Figure 2.8: Ví dụ về posterizing.....	14
Figure 2.9: Ví dụ về constrast.....	15
Figure 2.10: Ví dụ về brightness.....	15
Figure 2.11: Ví dụ về shear.....	16
Figure 2.12: Ví dụ về translate	16
Figure 2.13: Ví dụ về invert	17
Figure 2.14: Ví dụ về gaussian blur	17
Figure 2.15: Ví dụ về Poison noise	18
Figure 2.16: Residual connection	18
Figure 2.17: Identity block.....	19
Figure 2.18: Các thao tác của transposed convolution.....	20
Figure 2.19: Minh họa thao tác của deformable convolution	21
Figure 2.20: Ché tệp trung với các từ khác của Self-Attention	22

<i>Figure 2.21: Cách hoạt động của Self-Attention</i>	24
<i>Figure 2.22: Self-Attention và kiến trúc Multi-head Attention</i>	25
<i>Figure 2.23: Cách hoạt động của Multi-head Self Attention.....</i>	26
<i>Figure 2.24: Kiến trúc của Encoder – Decoder</i>	28
<i>Figure 2.25: Kiến trúc của mô hình ViT.....</i>	29
<i>Figure 2.26: Cơ chế tập trung của mô hình ViT.....</i>	30
<i>Figure 2.27: Kiến trúc của ABCNet.....</i>	31
<i>Figure 2.28: Kiến trúc của khối recognition. Hình trên là kiến trúc thông thường. Hình dưới là kiến trúc do tác giả đề xuất</i>	33
<i>Figure 3.1: Một số mẫu dữ liệu trong VinText</i>	35
<i>Figure 4.1: Các polygon có pred_score < 0.5 bị lọc bỏ.....</i>	36
<i>Figure 4.2: Các Bbox trong GT có nhãn là ###</i>	37
<i>Figure 5.1: Biểu đồ so sánh DBNetpp với một vài phương pháp scene text detection trên dữ liệu MRSA-TD500 với batch size = 1 và một 1080ti GPU. Ours là DBNetpp. DBNetpp có sự trade-off lý tưởng giữa tốc độ và độ chính xác.....</i>	41
<i>Figure 5.2: Kiến trúc mô hình DBNetpp.....</i>	42
<i>Figure 5.3: Kiến trúc của ASF Module.....</i>	43
<i>Figure 5.4: Đồ thị của SB và DB</i>	45
<i>Figure 5.5: Đạo hàm của Eq5 (Trái) và Eq7 (phải).....</i>	47
<i>Figure 5.6: Threshold map có hoặc không có sự giám sát.....</i>	48
<i>Figure 5.7: Minh họa của mô hình NAR và IR (cloze) trong mối quan hệ với sự kết hợp của nhiều mô hình AR cho hình x với nhãn y gồm 3 thành phần.</i>	51
<i>Figure 5.8: Kiến trúc mô hình PARSeq</i>	52
<i>Figure 5.9: Kiến trúc ViT Encoder. Trái là encoder trong PARSeq. Phải là encoder trong paper ViT gốc.</i>	52
<i>Figure 5.10: Khối MHA đầu tiên trong Visio-lingual decoder</i>	53
<i>Figure 5.11: Khối MHA thứ hai và khối MLP của Visio-lingual decoder</i>	54
<i>Figure 5.12: Biểu đồ so sánh kích thước, độ chính xác và FLOPS giữa PARSeq với một số mô hình khác.....</i>	56

<i>Figure 6.1: Các kiến trúc của ResNet.....</i>	58
<i>Figure 6.2: Feature Pyramid Backbone</i>	59
<i>Figure 6.3: Mô tả chi tiết của khối ASF</i>	60
<i>Figure 6.4: Mô tả chi tiết của khối Pred.....</i>	61
<i>Figure 6.5: Hmean, precision, recall của detection trên tập val với threshold từ 0.1 đến 0.9</i>	64
<i>Figure 6.6: Hmean, precision, recall của end to end trên tập val với threshold từ 0.1 đến 0.9</i>	65
<i>Figure 6.7: Văn bản nhỏ, mờ bị mô hình bỏ qua trong lúc dự đoán</i>	67
<i>Figure 6.8: Số điện thoại bị dự đoán rời rạc thành hai hay nhiều polygon khác nhau</i>	67
<i>Figure 6.9: Nhầm biếu tượng với văn bản.....</i>	68
<i>Figure 6.10: Gộm các vùng văn bản với nhau thành một</i>	68
<i>Figure 6.11: Khó nhận dạng các văn bản có phông chữ đặc biệt hoặc chữ viết tay</i>	69
<i>Figure 6.12: Vùng văn bản nằm trên nền phông tạp như có nhiều đường gạch, màu nền gần giống màu văn bản,</i>	69
<i>Figure 6.13: Thiếu dấu</i>	70
<i>Figure 7.1: FastAPI</i>	75
<i>Figure 7.2: ReactJS.....</i>	76
<i>Figure 7.3: Sơ đồ hệ thống cho chức năng 1</i>	77
<i>Figure 7.4: Sơ đồ hệ thống cho chức năng 2</i>	77
<i>Figure 7.5: Giao diện.....</i>	78
<i>Figure 7.6: Giao diện sau khi gửi file zip</i>	78
<i>Figure 7.7: Giao diện sử dụng chức năng thay đổi kích thước ảnh</i>	79

MỞ ĐẦU

Bài toán phát hiện và nhận diện văn bản trong điều kiện ngoại cảnh vẫn đang là một trong những vấn đề quan trọng của lĩnh vực thị giác máy tính. Đặc biệt, bài toán này ngày càng được quan tâm bởi sự tích hợp của lĩnh vực xử lý ngôn ngữ tự nhiên vào nó. Bên cạnh đó, những ứng dụng thực tế của bài toán ngày càng được sử dụng rộng rãi như phát hiện và nhận diện biển số xe vi phạm; phân tích nội dung ảnh lan truyền trên mạng xã hội; trích xuất thông tin hoá đơn; số hóa dữ liệu giấy; các ứng dụng tìm kiếm bằng nội dung hình ảnh;...

Trong những năm gần đây, các phương pháp deep learning dần trở thành xu hướng để xử lý trong hướng giải quyết các bài toán phát hiện và nhận diện văn bản tiếng Việt trong ảnh ngoại cảnh. Các mô hình này đều đã được áp dụng hiệu quả để xây dựng các hệ thống tự động phát hiện và nhận diện văn bản.

Mặc dù các kiến thức áp dụng trong bài toán này đã và đang phát triển mạnh mẽ, song bài toán Phát hiện và Nhận diện văn bản vẫn đang đối mặt với nhiều thách thức. Trong đó, việc phát hiện và nhận diện văn bản trên các hình ảnh có kích thước và hình dạng khác nhau và vẫn đang đối mặt với sự phức tạp về hình dáng văn bản: văn bản nằm dọc, văn bản nằm nhiều hướng hay văn bản cong, kiểu chữ đa dạng trên các nền khác nhau là các thách thức chính. Ngoài ra việc xử lý văn bản viết bằng nhiều ngôn ngữ và xử lý các văn bản trong các điều kiện thời tiết khác nhau cũng đòi hỏi các cách xử lý ngôn ngữ tự nhiên và xử lý ảnh nâng cao.

Ý nghĩa thực tiễn của bài toán và sự phát triển mạnh mẽ của các mô hình deep learning đi cùng với những thách thức khó khăn đã chứng minh bài toán này có tiềm năng to lớn. Chính tiềm năng này đã tạo động lực cho chúng tôi để quyết định lựa chọn bài toán này để tìm hiểu, nghiên cứu và thực hiện đồ án cuối kì.

Mục tiêu nghiên cứu

- Tìm hiểu các phương pháp sử dụng deep learning trong 2 năm trở lại đây cho bài toán phát hiện và nhận dạng văn bản ngoại cảnh.
- Huấn luyện mô hình phát hiện và nhận dạng văn bản trên tập dữ liệu tiếng Việt và so sánh với các phương pháp đã được áp dụng trên bộ dữ liệu này.
- Xây dựng một web app demo tiện lợi, dễ dàng sử dụng.

Cấu trúc đồ án

MỞ ĐẦU: Nêu ra ý nghĩa thực tiễn của đề tài; động lực, mục tiêu nghiên cứu.

CHƯƠNG 1: Giới thiệu bài toán phát hiện và nhận diện văn bản tiếng Việt trong ngoại cảnh: Phát biểu bài toán và những khó khăn trong bài toán.

CHƯƠNG 2: Cơ sở lý thuyết: Cung cấp kiến thức về các kỹ thuật tăng cường dữ liệu; các kỹ thuật và mô hình học sâu có liên quan; các phương pháp trước đây được sử dụng trên bộ dữ liệu VinText

CHƯƠNG 3: Bộ dữ liệu VinText: Mô tả bộ dữ liệu VinText

CHƯƠNG 4: Phương pháp đánh giá: Nêu các thang đo phổ biến được sử dụng để đánh giá mô hình recognition, detection và end to end.

CHƯƠNG 5: Hướng tiếp cận: Đưa ra hướng tiếp cận chung và giải thích chi tiết hướng tiếp cận của nhóm.

CHƯƠNG 6: Thực nghiệm: Các thông số được sử dụng cho mô hình, tăng cường dữ liệu; Kết quả thực nghiệm và nhận xét

CHƯƠNG 7: Xây dựng demo: Xây dựng một ứng dụng web nhận ảnh đầu vào từ người dùng và trả về tập các văn bản nhận dạng được.

KẾT LUẬN: Đưa ra kết luận và hướng phát triển trong tương lai.

TÀI LIỆU THAM KHẢO: Tài liệu tham khảo và phụ lục.

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN PHÁT HIỆN VÀ NHẬN DẠNG VĂN BẢN TIẾNG VIỆT TRONG NGOẠI CẢNH

1.1 Phát biểu bài toán

Bài toán phát hiện và nhận văn bản tiếng Việt trong cảnh là một trong những bài toán từ rất lâu những vẫn đang được cộng đồng các nhà nghiên cứu dành sự quan tâm. Đầu vào và đầu ra của bài toán như sau:

Đầu vào (Input): Ảnh chứa nội dung văn bản bằng Tiếng Việt, với mỗi hình ảnh có thể chứa một hoặc nhiều đối tượng văn bản và với các biến dạng khác nhau như kích thước, màu sắc...

Đầu ra (Output): Các polygons xác định vùng chứa văn bản trong ảnh và chuỗi ký tự tương ứng. Mỗi polygon là một danh sách tọa độ (x, y) của các điểm.

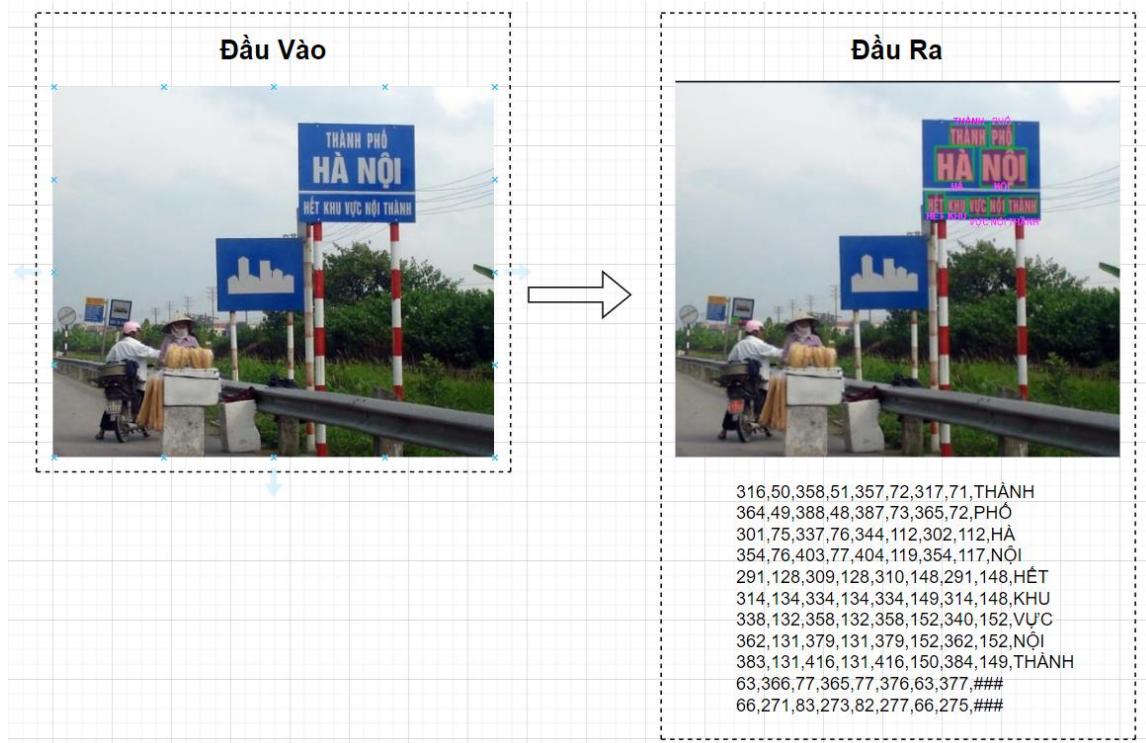


Figure 1.1: Mô tả đầu vào và đầu ra của bài toán

1.2 Các thách thức của bài toán

Trong bài toán Phát hiện và Nhận diện văn bản có rất nhiều thách thức đã và đang phải đổi mới. Các thách thức đó có thể được liệt kê ra là: Các thách thức bên ngoài và Các thách thức bên trong.

1.2.1 Các thách thức bên ngoài

Các thách thức bên ngoài còn có thể được đề cập như là các yếu tố ngoại cảnh. Đây là những yếu tố đến từ điều kiện bên ngoài như môi trường, con người, thời tiết, ... Đồng thời, các yếu tố này cũng ảnh hưởng trực tiếp đến hiệu suất của toàn bộ bài toán. Thách thức đầu tiên và cũng là thách thức tiêu biểu cho các bài toán phát hiện và nhận diện văn bản đó là ảnh chụp bị che khuất hay ảnh chụp với chất lượng không tốt. Theo điều kiện thực tế, không khó để bắt gặp các ảnh chụp bị các đối tượng ngẫu nhiên che mắt một phần văn bản dẫn đến việc khiến cho mô hình khó khăn hơn cho khâu phát hiện văn bản.



Figure 1.2: Trường hợp văn bản chụp bị che khuất trong ảnh

Việc các biển báo, các ảnh chụp bị mất chữ cũng giống trường hợp ảnh chụp có trường văn bản bị che khuất trên, khiến cho mô hình khó phát hiện và nhận diện các trường văn bản, từ đó làm giảm đáng kể hiệu suất của mô hình.



Figure 1.3: Trường hợp văn bản bị mất chữ trong ảnh

Các yếu tố về chất lượng ảnh thấp thông thường có thể được kể đến như màu ảnh không tốt, đường viền bị nhòe, hay bị nhiễu sáng có thể dễ thấy được ở các tấm bảng quảng cáo cũ, các áp phích cũ hoặc do các yếu tố thời tiết cũng gây khá nhiều trở ngại cho mô hình trong khâu dự đoán của từng giai đoạn.



Figure 1.4: Trường hợp chất lượng ảnh thấp

Tiếp theo, thách thức khó khăn có thể kể đến đó là độ phức tạp của nền ảnh. Vì đây là các tấm ảnh được chụp trong điều kiện ngoại cảnh, nên có vùng văn bản thường đi kèm với các bối cảnh rất phức tạp như trung tâm thương mại, đường phố đông đúc, ... Việc này làm cho các hướng giải quyết bài toán phải ngày càng phức tạp hơn để giải quyết được các vấn đề này.



Figure 1.5: Trường hợp bối cảnh chụp phức tạp

Cuối cùng, các yếu tố khác như hướng chụp, góc chụp cũng ảnh hưởng rất lớn đến việc dự đoán của mô hình. Nếu góc chụp là nghiêng thì kích thước các chữ cái sẽ có sự khác nhau, nếu góc chụp là ở xa thì dẫn đến các văn bản có thể quá nhỏ dẫn đến biến mất trong “mắt nhìn” của mô hình. Điều này đều dẫn đến việc mô hình có thể bỏ qua các trường văn bản đặc biệt là các trường văn bản nhỏ.



Figure 1.6: Trường hợp góc chụp xa

1.2.2 Các thách thức bên trong

Các yếu tố bên trong có thể dễ kề đến như kích thước của các ảnh đầu vào hay là cấu trúc của các trường văn bản có trong ảnh. Về yếu tố kích thước ảnh có sự khác nhau điều này yêu cầu các mô hình phải chuẩn bị trước một khâu để xử lý ảnh đầu vào. Từ đó nếu không xử lý tốt ở khâu này, có thể dẫn đến mất mát các thông tin ảnh quan trọng gây giảm hiệu suất của toàn bộ mô hình.



Figure 1.7: Sự khác nhau về kích thước các ảnh đầu vào

Thứ hai đó chính là cấu trúc của các trường văn bản có sự khác nhau. Các văn bản có có hướng (ngang, xiên, cong), phông chữ và kiểu chữ khác nhau trong hình ảnh gây rất nhiều sự nhọc nhằn cho mô hình trong việc phát hiện và nhận diện các đối tượng văn bản với nhau. Đặc biệt, có đối tượng văn bản với kiểu chữ nghệ thuật thường sẽ khiến các mô hình phát hiện sai hay bỏ qua các đối tượng này.



Figure 1.8: Thách thức về cấu trúc của các trường văn bản

1.2.3 Tổng kết về các thách thức của bài toán

Nhìn chung, dạng bài toán này thường được áp dụng vào các ứng dụng thực tế. Mà các ứng dụng thực tế không chỉ yêu cầu về độ chính xác mà còn có rất nhiều về tốc độ xử lý. Điều này phải yêu cầu các mô hình phải có khả năng xử lý trên thời gian thực.

Theo như đã phân tích thì thường các yếu tố bên ngoài và bên trong thì không hoàn toàn tách rời nhau, nó thường tác động và ảnh hưởng qua lại lẫn nhau. Vì vậy để giúp mô hình đạt hiệu suất tốt nhất, chúng ta phải linh hoạt sử dụng các phương pháp máy học, học sâu để giải quyết triệt để hay thách thức này, từ đó giúp chúng ta giải quyết thành công bài toán này.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Các phương pháp tăng cường dữ liệu

2.1.1 Random rotation

Xoay ảnh ngẫu nhiên với một góc $\in [-\alpha; \alpha]$. Nếu góc > 0 thì xoay ngược chiều kim đồng hồ, nếu góc < 0 thì xoay cùng chiều kim đồng hồ.



Figure 2.1: Ví dụ về random rotation

2.1.2 Random cropping

Cắt ra một phần hình chữ nhật ngẫu nhiên trong ảnh gốc.

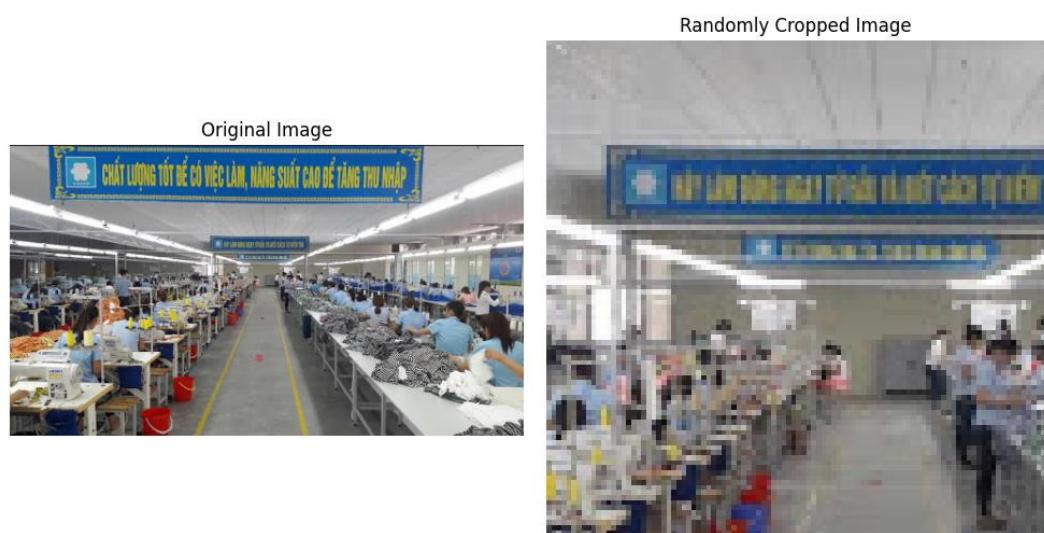


Figure 2.2: Ví dụ về random cropping

2.1.3 Random flipping

Lật ảnh ngẫu nhiên theo hướng ngang hoặc dọc.

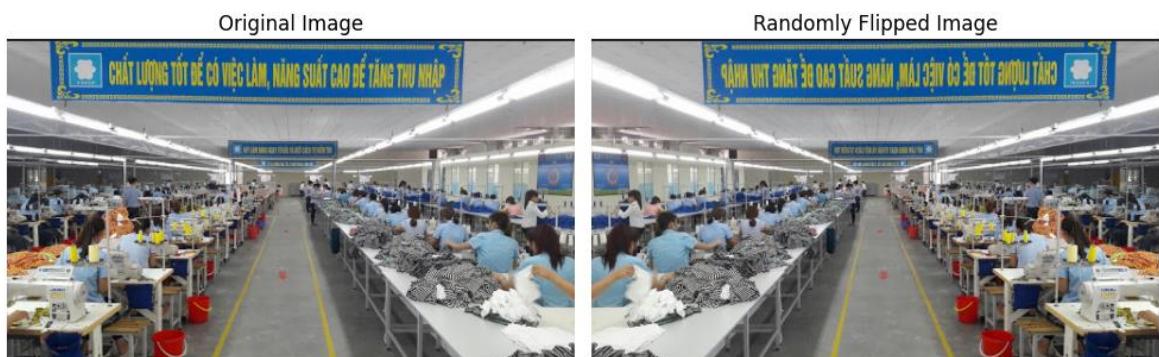


Figure 2.3: Ví dụ về random flipping

2.1.4 Auto contrast

Áp dụng tương phản ngẫu nhiên lên ảnh, tối đa hóa độ tương phản của hình ảnh, bằng cách làm cho điểm ảnh tối nhất trở nên đen và điểm ảnh sáng nhất trở nên trắng.



Figure 2.4: Ví dụ về auto contrast

2.1.5 Equalize

Cân bằng histogram



Figure 2.5: Ví dụ về equalize histogram

2.1.6 SolarizeAdd

Nghịch đảo tất cả giá trị pixel nhỏ hơn threshold, giá trị pixel mới được tính theo công thức $\min(255, \text{giá trị cũ} + add)$. Threshold và add là hai siêu tham số mà ta lựa chọn.



Figure 2.6: Ví dụ về solarize add

2.1.7 Color

Điều chỉnh cân bằng màu sắc của hình ảnh, theo cách tương tự như các nút điều khiển trên một tivi màu. Một giá trị (ta gọi là factor) có độ lớn bằng 0 sẽ tạo ra một hình ảnh đen trắng, trong khi giá trị độ lớn bằng 1 sẽ giữ nguyên hình ảnh.



Figure 2.7: Ví dụ về coloring

2.1.8 Posterize

Giảm số lượng bit cho mỗi kênh ảnh xuống số lượng được chỉ định.



Figure 2.8: Ví dụ về posterizing

2.1.9 Contrast

Điều chỉnh độ tương phản của hình ảnh. Một giá trị (ta gọi là factor) bằng 0 sẽ tạo ra một hình ảnh màu xám, trong khi giá trị độ lớn bằng 1 sẽ giữ nguyên hình ảnh gốc.



Figure 2.9: Ví dụ về contrast

2.1.10 Brightness

Điều chỉnh độ sáng của hình ảnh. Một giá trị (ta gọi là factor) bằng 0 sẽ tạo ra một hình ảnh màu đen, trong khi giá trị độ lớn bằng 1 sẽ giữ nguyên hình ảnh gốc.



Figure 2.10: Ví dụ về brightness

2.1.11 Shear

Shear là phương pháp làm biến dạng hình dạng và kích thước của một vật thể 2 chiều dọc theo trục x và y. Gọi \mathbf{P}, \mathbf{P}' lần lượt là ảnh đầu vào và ảnh đầu ra. ShearX sẽ giữ toạ độ y nhưng thay đổi toạ độ x theo công thức $\mathbf{P}'(x + Sh_x * y; y) = \mathbf{P}(x; y)$. ShearY sẽ giữ toạ độ x nhưng thay đổi toạ độ y theo công thức $\mathbf{P}'(x; y + Sh_y * x) = \mathbf{P}(x; y)$. Trong đó, Sh_x, Sh_y là các nhân tố (factor) mà ta có thể điều chỉnh.



Figure 2.11: Ví dụ về shear

2.1.12 Translate

Dịch chuyển hình ảnh theo hướng ngang (dọc) một khoảng bằng số lượng điểm ảnh độ lớn chỉ định. Nếu số lượng điểm ảnh là bất biến với các kích thước ảnh khác nhau thì đó là translate tuyệt đối, còn nếu số lượng điểm ảnh sẽ thay đổi phụ thuộc vào kích thước ảnh thì đó là translate tương đối. Để xác định số lượng điểm ảnh trong translate tương đối, ta sử dụng hàm ***pixels = pct x width (hoặc height)***, trong đó, ***pct*** là yếu tố mà ta có thể điều chỉnh.



Figure 2.12: Ví dụ về translate

2.1.13 Invert

Đảo ngược các điểm ảnh của hình ảnh.



Figure 2.13: Ví dụ về invert

2.1.14 Gaussian Blur

Làm mờ và loại bỏ nhiễu của ảnh với hệ số radius. Radius càng lớn thì ảnh càng mờ và ngược lại.



Figure 2.14: Ví dụ về gaussian blur

2.1.15 Poisson Noise

Nhiễu Poisson, hay nhiễu shot, là một dạng méo hình ảnh do sự dao động ngẫu nhiên thống kê của các photon hoặc hạt đập vào cảm biến hình ảnh.



Figure 2.15: Ví dụ về Poison noise

2.2 Các kĩ thuật deep learning liên quan

2.2.1 ResNet-50

ResNet-50 [1] là một mạng thần kinh tích chập có độ sâu 50 lớp được công bố bởi Microsoft năm 2015. Mô hình này được đào tạo trên hơn 1 triệu hình ảnh từ cơ sở dữ liệu ImageNet.

ResNet có khối tích chập (Convolutional Block, chính là Conv block trong hình) sử dụng bộ lọc kích thước 3×3 giống với của InceptionNet. Khối tích chập bao gồm 2 nhánh tích chập trong đó một nhánh áp dụng tích chập 1×1 trước khi cộng trực tiếp vào nhánh còn lại. Khối xác định (Identity block) thì không áp dụng tích chập 1×1 mà cộng trực tiếp giá trị của nhánh đó vào nhánh còn lại.

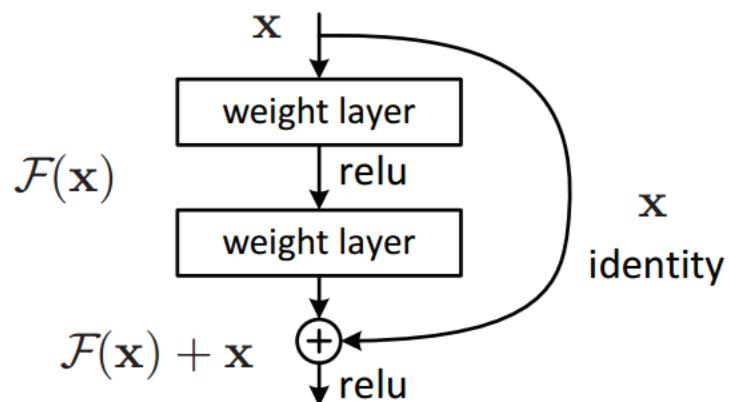


Figure 2.16: Residual connection

Giả sử với x là đầu vào của khối xác định, ta cần ánh xạ x thành hàm $f(x)$. Để tìm ra ánh xạ chuẩn xác tương đương với hàm $f(x)$ là điều khá khó, nhưng nếu cộng thêm ở đầu ra thành $x + f(x)$ thì ta có thể quy về tham số hóa độ lệch, tức là cần tham số hóa phần dư $f(x)$.

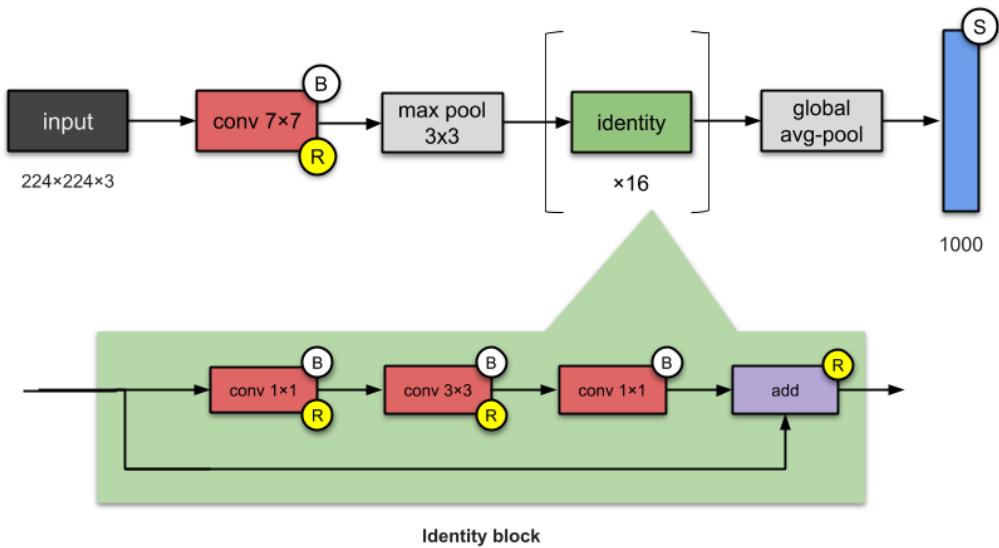


Figure 2.17: Identity block

2.2.2 Transposed convolution

Ngoài gia tăng kích thước thông qua Upsampling, chúng ta có thể thực hiện theo cách phức tạp hơn thông qua tích chập chuyển vị (Transposed Convolution [2] hoặc Conv2DTranspose). Vai trò của tích chập chuyển vị xuất phát từ nhu cầu biến đổi theo quá trình ngược lại của mạng tích chập thông thường, không phải là nghịch đảo về giá trị của tích chập thông thường, mà chỉ là kích thước không gian. Giả sử từ ma trận đầu vào có kích thước (w_1, h_1) , sau khi áp dụng phép tích chập thông thường ta thu được kích thước (w_2, h_2) . Tích chập chuyển vị sẽ biến đổi từ một ma trận có kích thước (w_2, h_2) của output sang ma trận có kích thước (w_1, h_1) của input.

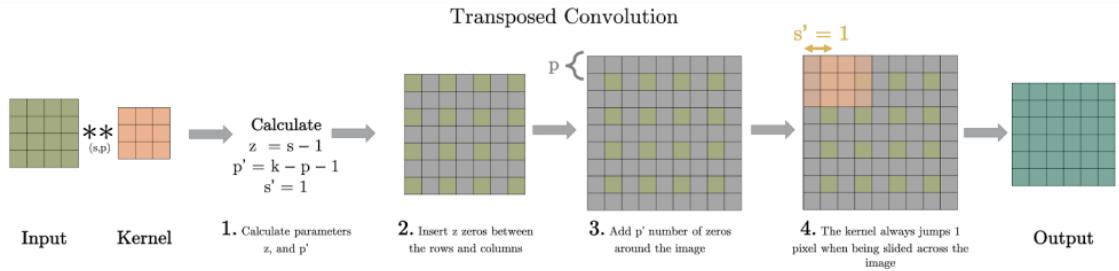


Figure 2.18: Các thao tác của transposed convolution

Trên thực tế thì có thể coi tích chập chuyên vị là một quá trình ngược của tích chập thông thường khi mỗi một đặc trưng (feature) được mapping sang các pixels ảnh thay vì ngược lại từ các pixels sang đặc trưng (feature).

2.2.3 Deformable convolution

Đối với convolution thông thường trong mạng nơ-ron tích chập (CNN), mỗi vùng của đầu vào được ánh xạ một cách đồng đều và cố định vào một vùng tương ứng trên đầu ra thông qua các bộ lọc (kernels). Tuy nhiên, trong một số trường hợp, điều này có thể không phản ánh chính xác cách thông tin phân phối trên ảnh, đặc biệt là khi có những biến động lớn trong đối tượng, như việc có sự biến đổi hình dạng (deformation) trong các đối tượng.

Deformable Convolution [3] xuất phát ý tưởng cộng thêm vào các vị trí lấy mẫu trong các lớp convolution truyền thống một mảng offsets 2 chiều. Điều này cho phép lớp tích chập lấy mẫu tại những vị trí đa dạng hơn. Đặc biệt các giá trị offsets này điều được học từ các bản đồ đặc trưng trước đó thông qua các lớp tích chập nhờ vậy các giá trị offsets linh hoạt thích ứng với dữ liệu đầu vào.

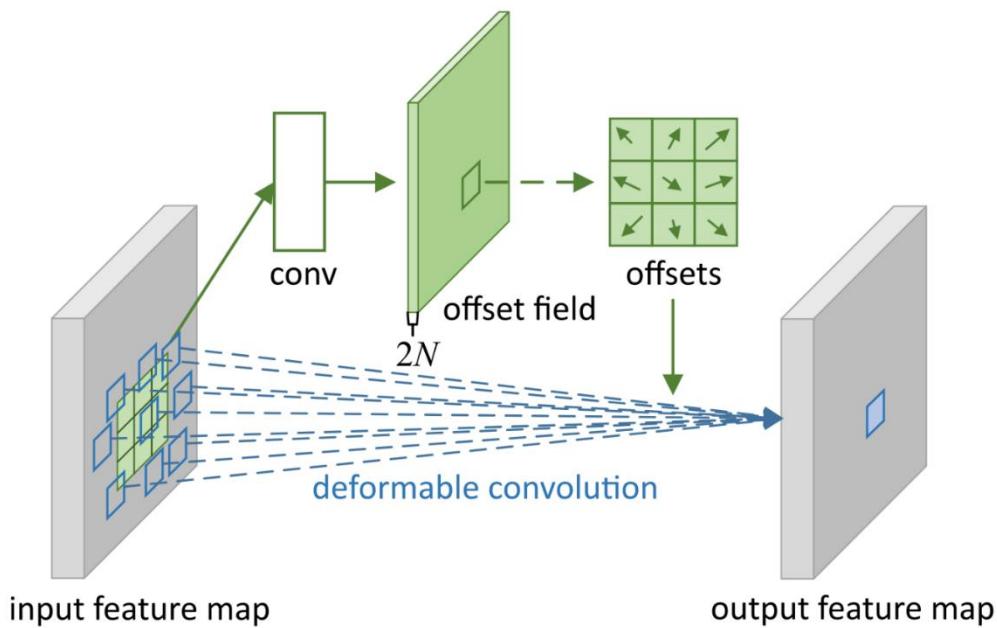


Figure 2.19: Minh họa thao tác của deformable convolution

2.2.4 Cơ chế attention trong Transformer [4]

2.2.4.1 Cơ chế Self-Attention

Đây được coi như là thành phần quan trọng nhất của mô hình Transformer. Self-Attention có thể coi như là một thuật toán tìm kiếm, khi mà khi ta đưa một câu đầu vào, mô hình sẽ phải chú ý tập trung đến có phần khác có liên quan đến phần đang xử lý. Ví dụ với một câu là “Con dê đang băng qua một con đường dài nhưng nó đã dừng lại nghỉ vì nó quá mệt.”. Thì với cơ chế Self-Attention, khi xử lý đến từ “nó” thì mô hình phải biết được từ nó ở đây là từ gì, là “Con dê” hay “con đường”? Điều này có vẻ dễ đối với con người nhưng ngược lại, để máy tính hiểu được điều này thì còn khó hơn nữa.

Chính vì vậy, cơ chế Self-Attention cho phép mô hình tập trung nhiều hơn đến từ “Con dê” nhiều hơn từ “con đường” khi nó đang xử lý từ “nó”. Có thể kết luận rằng, cơ chế Self-Attention có thể được coi là bộ mã hóa và nó có thể hiểu được mức độ liên quan của các từ ngữ đối với nhau, từ đó giúp nó có thể dự đoán chính xác hơn.

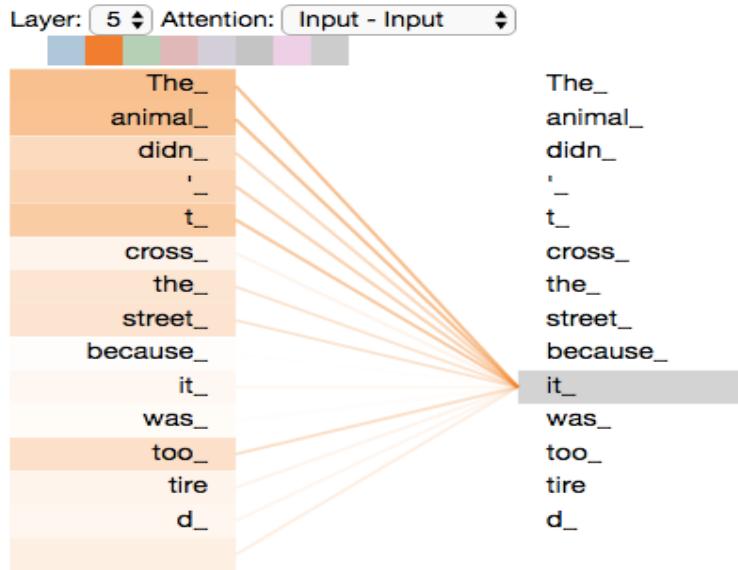


Figure 2.20: Ché tập trung với các từ khác của Self-Attention

Mô hình thường chú ý đến những từ để dễ quan sát hơn như sau:

- Chú ý đến từ kế trước của một từ
- Chú ý đến từ kế sau của một từ
- Chú ý đến những từ liên quan của một từ

Đầu vào của Self-Attention là 3 vector query, key và value. Các vector này có thể tính được bằng cách nhân đầu vào với các ma trận trọng số tương ứng với query, key và value. Có thể tóm tắt công dụng của 3 loại vector này như sau:

- Query: là vector dùng để chứa thông tin của từ đang xử lý ở bước hiện tại.
- Key: là vector dùng để biểu diễn thông tin các từ đang được so sánh với từ đang được xử lý ở trên.
- Value: là vector biểu diễn nội dung ý nghĩa của từ.

Vector attention là vector thể hiện độ tương quan giữa 3 loại vector Query, Key và Value. Vector này có thể được tính bằng cách nhân tích vô hướng giữa hai Vector Query và Key, sau đó cho kết quả đi qua hàm Softmax để chuẩn hóa dữ liệu. Cuối

cùng nhân với Vector Value để có được kết quả Vector Attention. Cụ thể từng bước như sau:

- Bước 1: Tính ra ba ma trận query, key và value bằng cách nhân đầu vào với ma trận trọng số tương ứng.
- Bước 2: Nhân ma trận query và key lại với nhau bằng phép nhân tích vô hướng. Sau đó đưa kết quả phép nhân này qua một hàm Softmax để chuẩn hóa dữ liệu về miền $[0,1]$. Phép nhân này thể hiện sự tương quan của hai vector key và value, nếu kết quả này càng cao sự tương quan giữa key và query càng cao và ngược lại.
- Bước 3: Kết quả cuối cùng sẽ được tạo ra bằng cách nhân kết quả của bước 2 với ma trận value.

Phép nhân ở bước hai được gọi là hàm dot-product attention, đặc điểm nổi bật của phép nhân này là sau khi nhân ma trận query và key, nó sẽ được chia cho $\sqrt{d_k}$, trong đó d_k là số chiều của vector key, trước khi đi qua một hàm softmax.

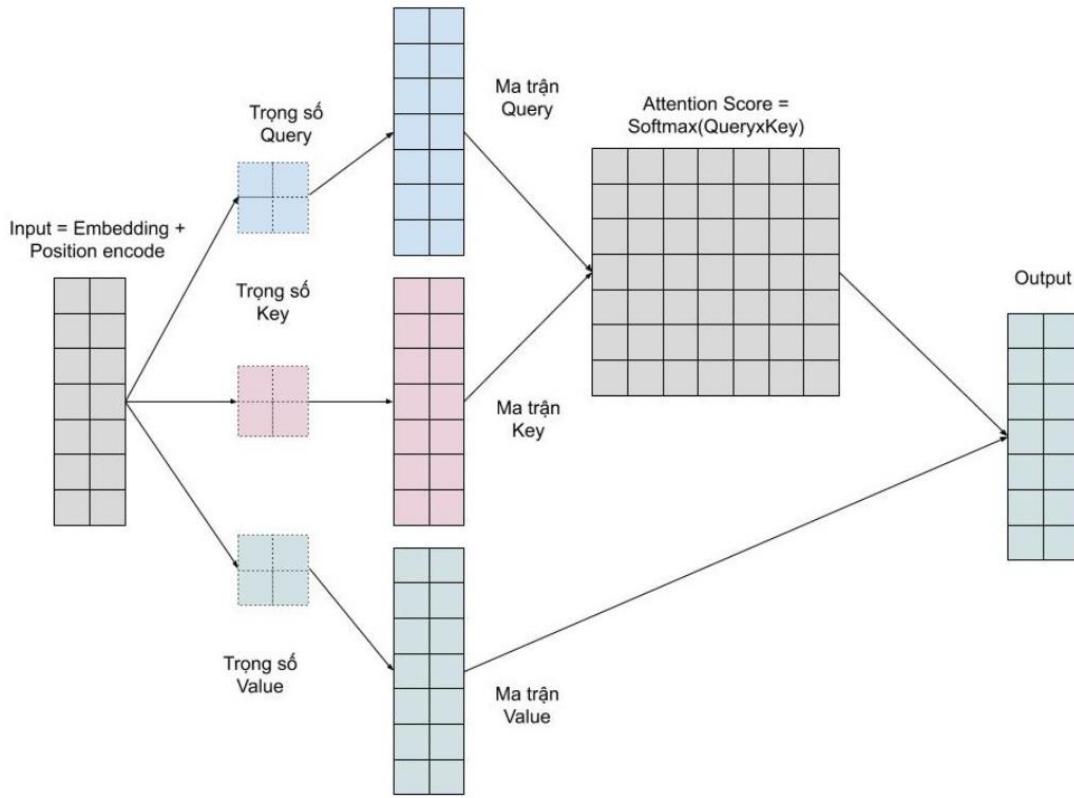


Figure 2.21: Cách hoạt động của Self-Attention

Vì vậy, Attention có thể được hiểu là cơ chế ánh xạ một query với một cặp key-value sang một đầu ra mới. Đầu ra này chính là tổng trọng số giữa các giá trị trong values, với trọng số được tính từ query với key tương ứng. Để đơn giản hơn, ta có thể xem query là một câu truy vấn từ để tìm được mã câu tương ứng (key), với nội dung của câu chính là value. Và query, key và value sẽ được kí hiệu lần lượt là Q, K, V.

2.2.4.2 Cơ chế Multi-head Self-Attention

Chúng ta muốn mô hình có thể học nhiều kiểu mối quan hệ giữa các từ với nhau. Với mỗi self-attention, chúng ta học được một kiểu mẫu, do đó để có thể mở rộng khả năng này, chúng ta đơn giản là thêm nhiều self-attention. Tức là chúng ta cần nhiều ma trận query, key, value mà thôi. Giờ đây ma trận trọng số key, query, value sẽ có thêm 1 chiều depth nữa. Do đó Multi-head Self Attention được sử dụng

như là một trong những cơ chế không thể thiếu của mô hình Transformer, nó sẽ xem xét một câu đầu vào với nhiều góc độ hơn.

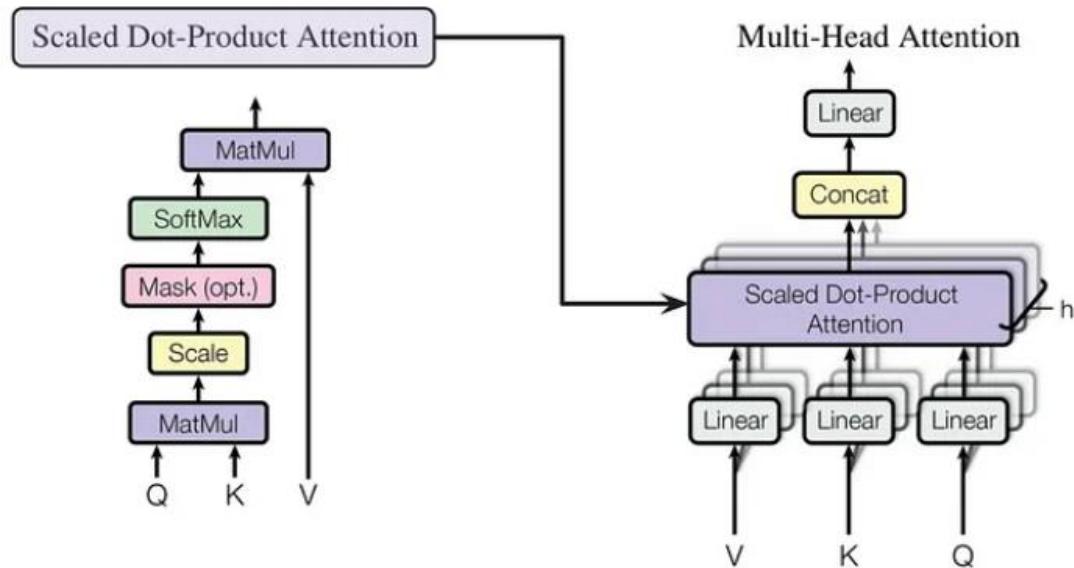


Figure 2.22: Self-Attention và kiến trúc Multi-head Attention

Trong kiến trúc Multi-head Attention, với mỗi đầu vào, ta sẽ có ba ma trận Q , K , V tương ứng được học từ phép biến đổi tuyến tính thông qua việc đi qua một lớp Linear (mỗi Q , K , V sẽ đi qua lớp Linear tương ứng với nó như hình trên). Ngoài ra, các Q , K , V này phải đi qua nhiều lớp Linear chồng nhau, nhằm mục đích đạt được Multi-head Attention. Sau khi đi qua nhiều lớp Linear chồng nhau, mô hình sẽ có khả năng hiểu được ý nghĩa của câu dưới nhiều ý nghĩa với góc nhìn khác nhau. Sau khi đi qua hết các lớp Linear, các trọng số attention sẽ được học lại thông qua phép Scaled Dot-Product Attention.

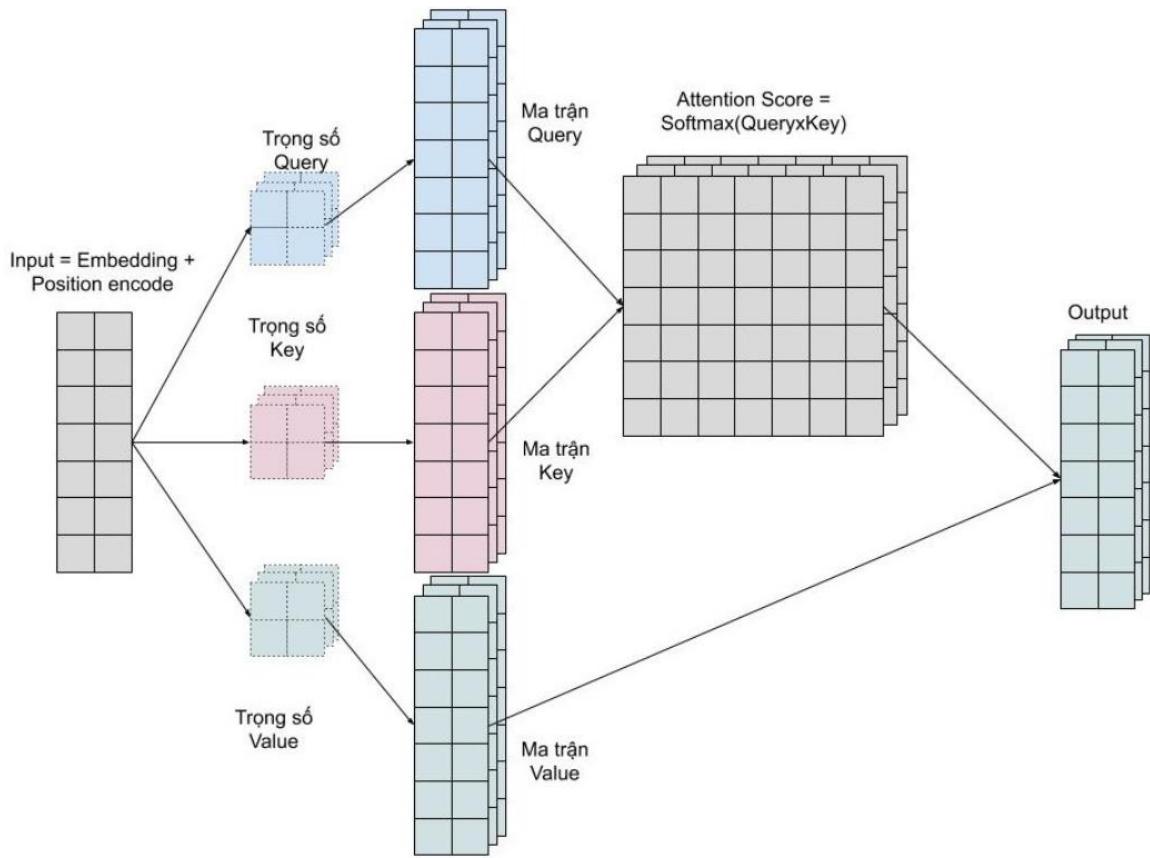


Figure 2.23: Cách hoạt động của Multi-head Self Attention

Scaled Dot-Product Attention với ý tưởng là lấy vector Q nhân với vector K để tìm ra các từ có liên quan đến nhau nhất. Và để tránh hiện tượng kết quả đạo hàm nhỏ (do phép dot product của Q và K trong công thức $qk = \sum_{i=1}^{d_k} q_i k_i$ sẽ có giá trị trung bình cực nhỏ và phương sai lớn tương ứng với d_k) khi số chiều của vector K tăng lên.

Thì kết quả của phép nhân ở trên sẽ được nhân với $\frac{1}{\sqrt{d_k}}$. Sau đó kết quả này sẽ được đi qua một hàm Softmax và nhân với V để giữ được các thông tin quan trọng. Kết quả đầu ra của phép nhân Scaled Dot-Product Attention sẽ được ghép lại qua phép ConCat và đi qua lớp Linear. Công thức tổng quát của toàn bộ quá trình trên:

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

2.2.4.3 Encoder-Decoder Attention

Encoder của mô hình transformer có thể bao gồm nhiều encoder layer tương tự nhau. Mỗi encoder layer của transformer lại bao gồm 2 thành phần chính là multi-head attention và feed-forward network, ngoài ra còn có cả skip connection và normalization layer.

Encoder đầu tiên sẽ nhận ma trận biểu diễn của các từ đã được cộng với thông tin vị trí thông qua positional encoding. Sau đó, ma trận này sẽ được xử lý bởi Multi Head Attention. Multi Head Attention thực chất là self-attention, nhưng mà để mô hình có thể có chú ý nhiều góc cạnh khác nhau, tác giả đơn giản là sử dụng nhiều self-attention. Cuối cùng, các ma trận này sẽ được đi qua một lớp Feed Forward và Add&Norm nữa để tạo đầu ra có kích thước bằng với đầu vào.

Trong decoder còn có một multi head attention khác có chức năng chú ý các từ ở mô hình encoder, layer này nhận vector key và value từ mô hình encoder, và output từ layer Masked Attention ở bên dưới. Đơn giản bởi vì chúng ta muốn so sánh sự tương quan giữa các tokens nguồn và đích.

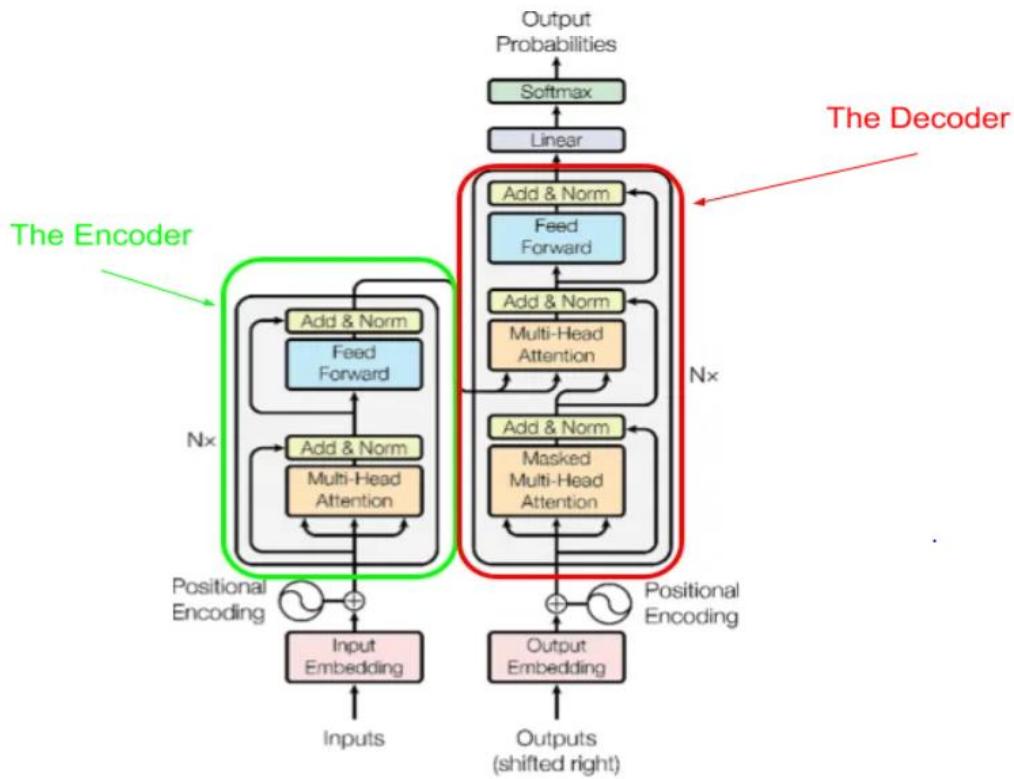


Figure 2.24: Kiến trúc của Encoder – Decoder

2.2.4.4 Masked Attention

Masked Attention hay còn gọi là Masked Multi Head Attention. Tuy nhiên, Masked Attention cũng là một multi-head attention mà như chúng ta đã nói nãy giờ ở trên, tuy nhiên, lúc cài đặt chúng ta cần phải che các tokens ở tương lai (tức là không tính đến attention của các tokens trong tương lai) chưa được mô hình dụng tới. Để làm việc này thì chúng đơn giản chỉ cần thực hiện theo công thức sau:

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

Trong đó, mask là một ma trận 0, $-\infty$. Trong đó $-\infty$ được dùng như một mặt nạ để che đi token ở tương lai.

2.2.5 Vision Transformer

Được ra mắt vào năm 2020, mô hình Vision Transformer [5] nhanh chóng trở nên nổi tiếng vì hiệu suất vượt trội của nó. Kiến trúc tổng quan của ViT như sau, về cơ bản ViT sử dụng lại Encoder của mô hình Transformer.

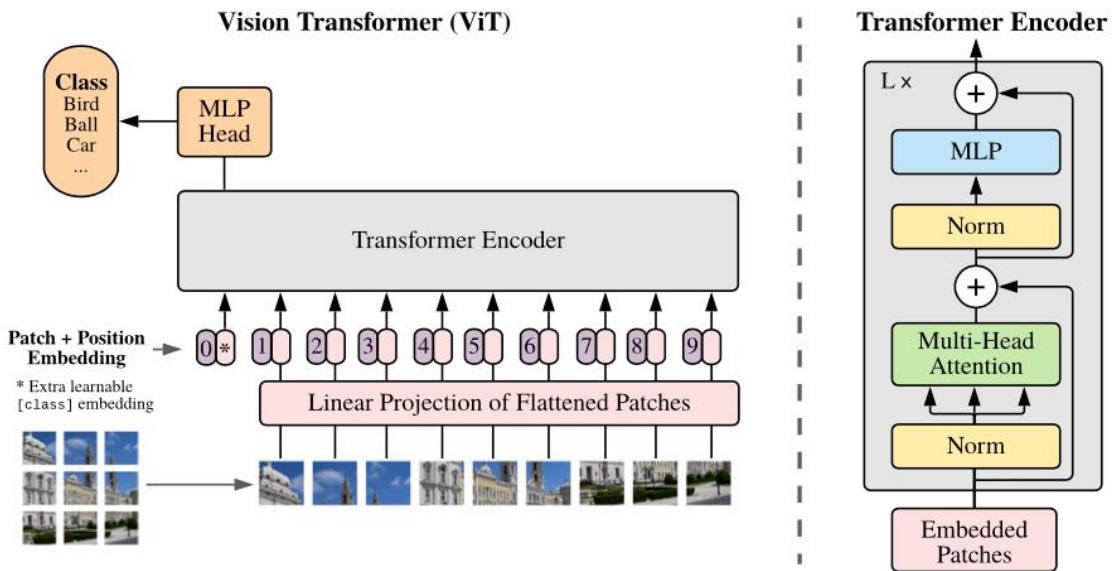


Figure 2.25: Kiến trúc của mô hình ViT

Các bước tính toán của ViT có thể được mô tả như sau:

- B1: Chia ảnh thành các mảng (patch nhỏ) với kích thước cố định.
- B2: Làm phẳng các hình ảnh.
- B3: Tạo các feature embedding có chiều thấp hơn từ các mảng hình ảnh đã được làm phẳng này
- B4: Nhúng thêm các vị trí về thứ tự hình ảnh.
- B5: Chuỗi các feature embedding được làm đầu vào cho Transformer Encoder.

Ở bước biến các hình ảnh đầu vào thành các vector embedding thì hình ảnh sẽ được chia thành các mảng con có kích thước bằng nhau. Ta có đầu vào của ảnh là ảnh 2D $x \in \mathbb{R}^{H \times W \times C}$ trong đó H, W, C lần lượt là chiều cao, chiều rộng và số chiều của ảnh.

Sau bước trên, ảnh trở thành các patch $x_p \in R^{Nx(P^2.C)}$, trong đó (P, P) là kích thước cố định của từng ảnh con và $N = \frac{HW}{P^2}$. Vì Encoder của Transformer chỉ nhận đầu vào với kích thước cố định D nên các ảnh con sẽ được duỗi thẳng và đi qua một lớp Linear có thể học được để chuyển sang một vector D chiều.

Cũng tương tự như bên NLP, thứ tự của các từ rất quan trọng nên thứ tự của các mảng ảnh trong ViT cũng cần được chú ý. Do đó một lớp Positional Embedding được sử dụng để lưu các thông tin về vị trí của các ảnh con và tạo thành được một Embedded Patches. Tiếp đến Embedded Patches sẽ được nối thêm một class embedding vào đầu. Sau đó, các Embedded Patch sẽ được đi qua khối Encoder của Transformer (vẫn được giữ nguyên y như bên NLP), tức là các patch này sẽ được đi qua lớp chuẩn hóa (Norm), lớp Multi-head Attention và lớp MLP. Sau cùng thì từ đầu ra của khối Encoder, giờ đầu ra tương ứng với class embedding sẽ đi qua một lớp MLP Head có số chiều là K bằng với số lớp trong tập dữ liệu.

Và cũng tương tự như cơ chế chú ý bên NLP, ViT cũng sẽ dựa vào từng mảng ảnh con và tập trung chú ý vào ảnh con có liên quan nhất đối với ảnh đang xét. Hình ảnh bên dưới sẽ thể hiện rõ nhất về cơ chế này

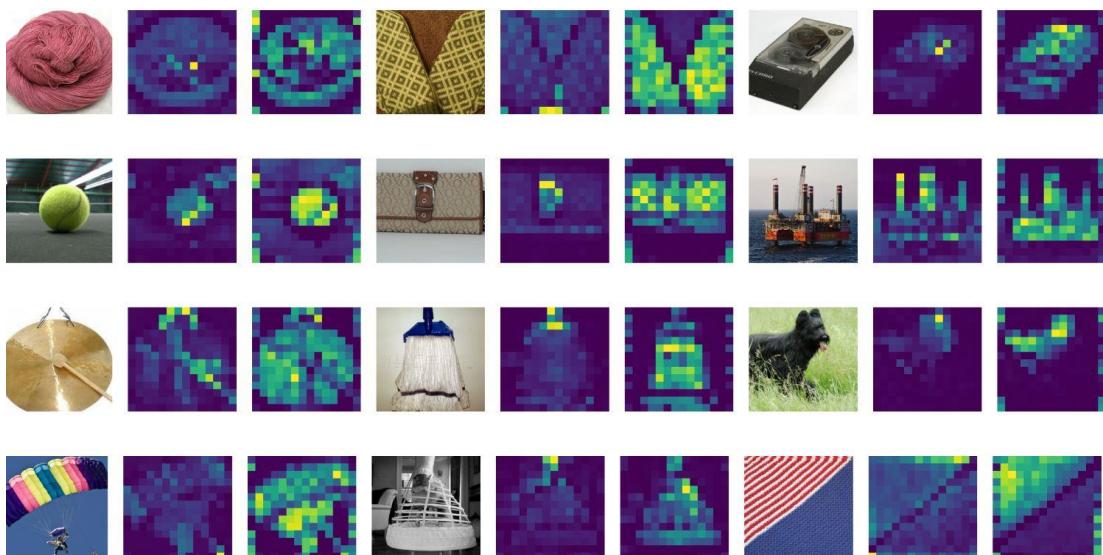


Figure 2.26: Cơ chế tập trung của mô hình ViT

Với ảnh trên, có thể thấy rằng sau khi được học, attention của ViT đã tập trung gần như hoàn toàn vào các đối tượng cần được chú ý, từ đó có thể nâng cao hiệu suất của mô hình.

2.3 Các phương pháp trên bộ dữ liệu VinText [6]

2.3.1 ABCNet [7]

Có rất nhiều phương pháp end-to-end được đưa ra để xử lý bài toán phát hiện văn bản, chủ yếu sử dụng hướng hai hướng tiếp cận segmentation-based hoặc character-based. Tuy nhiên các phương thức này làm cho việc duy trì pipeline trở nên phức tạp hơn hoặc cần yêu cầu một lượng lớn sự chú ý vào mức kí tự. Hơn nữa, hầu hết các phương pháp trên đều chậm lúc inference, cản trở khó khăn trong quá trình áp dụng vào ứng dụng.

ABCNet đưa ra một tiếp cận mới giải quyết rất tốt với dữ liệu cong, xiên,... và đặc biệt nó đem lại tốc độ tính toán nhanh giúp mô hình có thể chạy thời gian thực. ABCNet cho phép phát hiện văn bản có hình dạng cong, với một đường cong Bezier thích nghi hiệu quả. Ngoài ra, bài báo đưa ra một lớp căn chỉnh tính năng mới - BezierAlign - để tính toán chính xác các đặc điểm tích tụ của các thẻ hiện văn bản ở dạng cong và do đó có thể đạt được độ chính xác dạng cao với chi phí tính toán giàn như không đáng kể.

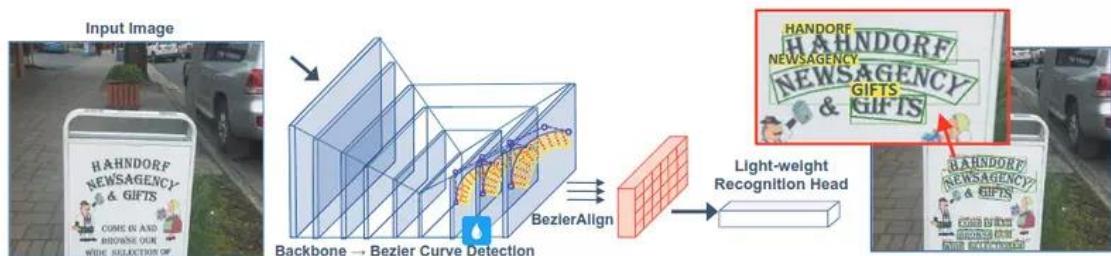
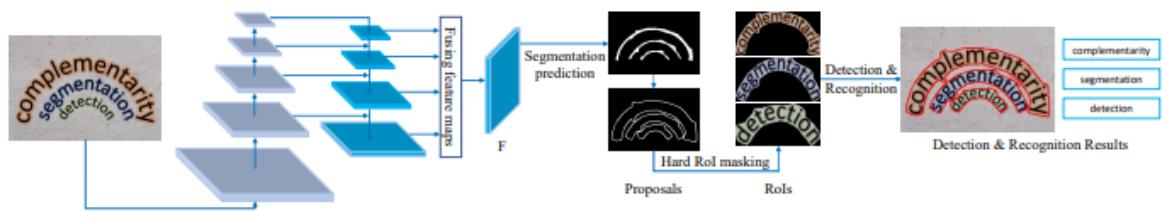


Figure 2.27: Kiến trúc của ABCNet

2.3.2 Mask TextSpotter v3 [8]

Mask TextSpotter v3 là một phương pháp huấn luyện nâng cao end-to-end cho bài toán Scene Text Spotting, được thiết kế để giải quyết các thách thức trong việc xử lý các phiên bản văn bản có tỷ lệ khung hình cực cao hoặc hình dạng không đều. Phương pháp này tích hợp các quy trình phát hiện và nhận dạng, đồng thời giới thiệu việc sử dụng Segmentation Proposal Network (SPN) thay vì Region Proposal Network (RPN).



2.3.3 Dictionary-guided Scene Text Recognition [6]

Các phương pháp nhận dạng văn bản ngữ cảnh hiện tại thường sử dụng từ điển để cải thiện hiệu suất nhận dạng, bằng cách chuyển đầu ra thành một từ điển dựa trên edit distance. Tuy nhiên chúng có rất nhiều hạn chế.

Tác giả giới thiệu một phương pháp mới để tích hợp từ điển trong cả quá trình huấn luyện và suy luận của một hệ thống nhận dạng văn bản ngữ cảnh, giúp xử lý những trường hợp mơ hồ và cải thiện hiệu suất tổng thể của các mô hình nhận dạng văn bản ngữ cảnh tiên tiến như ABCNet và Mask TextSpotter v3.

Cụ thể, thay vì buộc kết quả dự đoán phải là một từ trong từ điển, họ sử dụng từ điển để tạo ra một danh sách ứng viên. Các từ trong danh sách này sau đó sẽ được đưa vào một mô-đun đánh giá để tìm ra kết quả cuối cùng phù hợp nhất với đặc điểm xuất hiện trên hình ảnh.

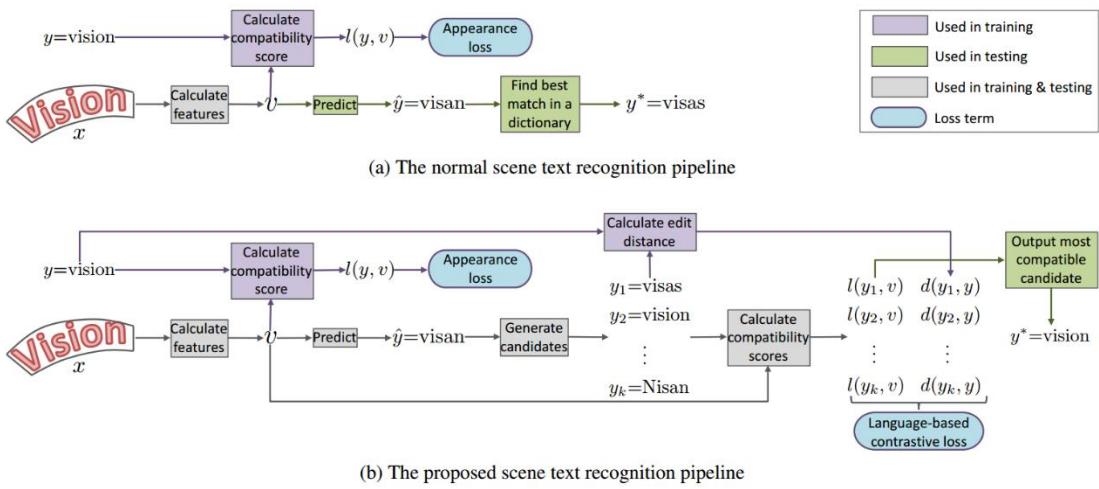


Figure 2.28: Kiến trúc của khôi nhận. Hình trên là kiến trúc thông thường. Hình dưới là kiến trúc do tác giả đề xuất

Ngoài ra, họ cũng đóng góp bộ dữ liệu VinText về văn bản ngẫu cảnh tiếng Việt.

CHƯƠNG 3: BỘ DỮ LIỆU VINTEXT

Vì đây là chủ đề nóng và được rất nhiều các tác giả quan tâm nên có rất nhiều bộ dữ liệu và đa dạng về các ngôn ngữ có trong các bộ dữ liệu.

3.1 Khảo sát dữ liệu

Đây là bài toán nhận được sự quan tâm đặc biệt của cộng đồng các nhà nghiên cứu kể cả trong lĩnh vực xử lý ảnh và xử lý ngôn ngữ tự nhiên. Các bộ dữ liệu đã có thể được liệt kê lại như sau:

Nhóm dữ liệu ICDAR: [2003, 2011, 2013, 2015] đây đều là những bộ dữ liệu trong bài toán Phát hiện và Nhận diện văn bản trong ảnh ngoại cảnh và được tăng dần về số lượng ảnh, độ khó theo từng năm.

CTW-1500: Đây là bộ dữ liệu chụp bằng điện thoại trên văn bản tiếng Trung và Tiếng Anh.

UAIC2021: Đây là bộ dữ liệu dành riêng cho cuộc thi UIT-AI Challenge cho chủ đề nhận diện chữ nghệ thuật trong ngoại cảnh.

Ngoài ra vẫn còn rất nhiều bộ dữ liệu khác với cùng chủ đề.

3.2 Bộ dữ liệu Vin-Text [6]

Bộ dữ liệu chúng tôi lựa chọn để giải quyết bài toán này là bộ dữ liệu VinText của VinAI được cung cấp dành riêng cho cuộc thi AI Challenge 2021. Bộ dữ liệu này được đánh giá là với rất nhiều thách thức lớn, chứa đựng rất nhiều hình ảnh đa dạng với nhiều biến cửa hàng, bảng quảng cáo, biển tuyên truyền. Do đó, bộ dữ liệu này sẽ được kỳ vọng như một thước đo thách thức để đánh giá tính ứng dụng và tính ổn định của các thuật toán phát hiện và nhận diện văn bản cảnh.

Tập dữ liệu VinText bao gồm 2000 hình ảnh với khoảng 56,000 trường văn bản được chia ngẫu nhiên thành ba bộ train/val/test với số mẫu lần lượt là 1200 ảnh, 300 ảnh, 500 ảnh.



Figure 3.1: Một số mẫu dữ liệu trong VinText

CHƯƠNG 4: PHƯƠNG PHÁP ĐÁNH GIÁ

4.1 Đánh giá riêng cho detection

Đối với tác vụ Phát hiện văn bản, chúng tôi sẽ sử dụng chỉ số HmeanIOUMetric đến từ thư viện mmocr [9]. HmeanIOUMetric là một phương pháp được sử dụng rộng rãi để đánh giá trên tác vụ Phát hiện văn bản. Phương pháp này dựa trên IoU (Intersection over Union) giữa 2 polygon: $IoU = \frac{\text{Diện tích của phần giao}}{\text{Diện tích của phần hợp}}$.

Chúng tôi có thể mô tả quá trình sử dụng HmeanIOUMetric để đánh giá hiệu suất trên giai đoạn phát hiện văn bản như sau:

Bước 1: Lọc bỏ những dự đoán không quan tâm

- Lọc ra những dự đoán có confident score (chỉ số tự tin) $< \text{predict score threshold}$ (Đây là một siêu tham số). Giá trị threshold được xác định trong quá trình đánh giá trên tập validation sau đó được sử dụng trên tập test.

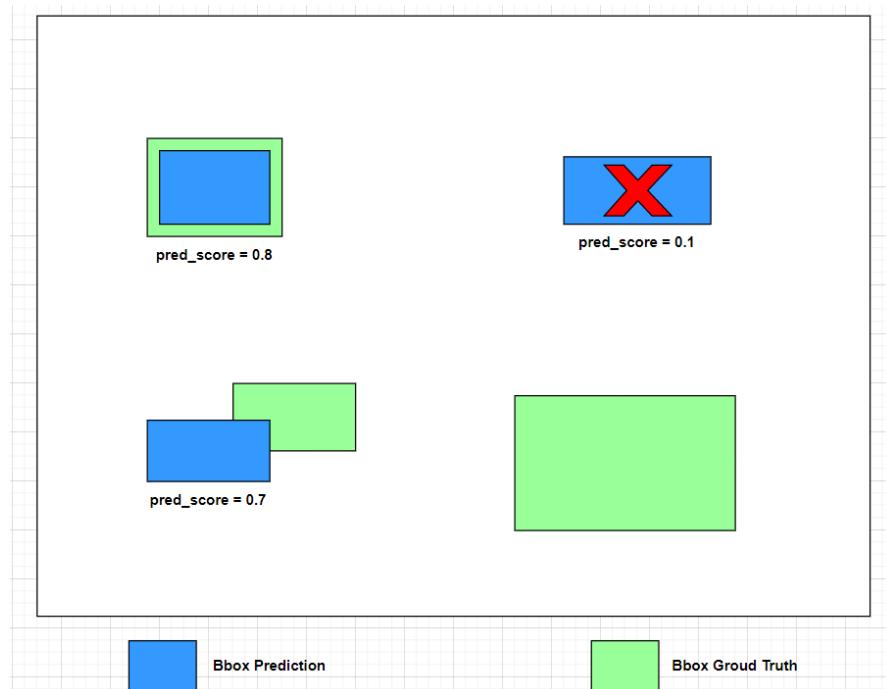


Figure 4.1: Các polygon có pred_score < 0.5 bị lọc bỏ

- Lọc bỏ các dự đoán chòng chéo với những ground truth bị bỏ qua với tỷ lệ chòng chéo cao hơn ignore precision threshold = 0.5. Trong VinText những ground truth bị bỏ qua là những polygon có nhãn văn bản là ### biểu thị cho những vùng văn bản khó đọc. Điều này có nghĩa là nếu polygon dự đoán mà overlap với các ground truth này ở một ngưỡng nào đó thì sẽ được lọc bỏ và không tính trong đánh giá.

2013,273,2040,273,2041,283,2015,284,CÔNG
2044,271,2078,271,2081,282,2043,283,TRƯỜNG
2083,272,2107,272,2107,282,2083,282,ĐANG
2110,272,2122,273,2122,282,2109,282,###
2125,273,2148,272,2149,281,2123,283,CONG
2015,311,2038,309,2038,318,2017,319,###

Figure 4.2: Các Bbox trong GT có nhãn là ###

Bước 2: Tính số lượng polygon dự đoán khớp với ground truth.

Sau bước 1, ta sẽ chỉ còn những polygon dự đoán và ground truth cần quan tâm. Từ đây ta sẽ tính số lượng dự đoán phù hợp dựa trên chúng. Đối với các polygon dự đoán có IoU so với ground truth lớn hơn hoặc bằng ngưỡng 0.5 sẽ được coi là khớp. Trong trường hợp có nhiều polygon thỏa mãn ngưỡng, polygon có IoU lớn nhất sẽ được chọn, các polygon còn lại sẽ được tính là false positive.

Bước 3: Tính evaluation score.

- Recall: Cho biết khả năng của hệ thống phát hiện văn bản ngoại cảnh trong việc tìm ra tất cả các vùng chứa văn bản có trong hình ảnh.

$$\begin{aligned}
 \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \\
 &= \frac{\text{TP}}{\text{Số lượng ground truth quan tâm}}
 \end{aligned}$$

- Precision: Cho biết khả năng của hệ thống phát hiện văn bản ngoại cảnh trong việc xác định chính xác các vùng chứa văn bản.

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ &= \frac{\text{TP}}{\text{Số lượng dự đoán quan tâm}} \end{aligned}$$

Trong đó:

- TP (*True Positive*): Số lượng dự đoán khớp với ground truth
- FP (*False Positive*): Số lượng dự đoán sai
- FN (*False Negative*): Số lượng ground truth mà mô hình không có dự đoán nào khớp với chúng.
- HmeanIOUMetric là trung bình điều hoà giữa Precision và Recall và được tính theo công thức sau:

$$Hmean = 2 \times \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

4.2 Đánh giá riêng cho Recognition

Accuracy và OneMinusNED là 2 thang đo phổ biến giúp đánh giá mô hình nhận dạng văn bản trong ngữ cảnh.

Accuracy sẽ đánh giá nghiêm ngặt vì nó yêu cầu văn bản dự đoán với ground truth phải giống nhau hoàn toàn mới được xem là đúng. Accuracy được tính theo công thức:

$$\text{Accuracy} = \frac{\text{Số dự đoán đúng}}{N}$$

OneMinusNED sẽ ít nghiêm ngặt hơn vì nó dựa trên khoảng cách Levenshtein giữa văn bản dự đoán và ground truth. OneMinusNED này được tính như sau:

$$\text{OneMinusNED} = 1 - \frac{1}{N} \sum_{i=1}^N \frac{D(\mathbf{s}_i, \hat{\mathbf{s}}_i)}{\max(l_i, \hat{l}_i)}$$

Trong đó: \mathbf{s}_i , $\hat{\mathbf{s}}_i$ lần lượt là ground truth, văn bản dự đoán; l_i , \hat{l}_i lần lượt là chiều dài của ground truth và văn bản dự đoán; $D(\mathbf{s}_i, \hat{\mathbf{s}}_i)$ là khoảng cách Levenshtein($\mathbf{s}_i, \hat{\mathbf{s}}_i$) - Số thao tác ít nhất cần thực hiện để từ **s_i thành ŝ_i**. Các thao tác bao gồm: insert (thêm một ký tự), delete (xoá một ký tự), replace (đổi một ký tự thành ký tự khác).

4.3 Đánh giá cho mô hình end to end

Sử dụng thang đo giống với VinText. Thang đo này kết hợp đánh giá khả năng phát hiện và nhận dạng văn bản của mô hình trên tất cả các văn bản được quan tâm. Những văn bản không được quan tâm sẽ là “###”- thể hiện văn bản khó nhìn, không thể đánh nhãn; các văn bản chứa ký tự chữ và số, và có độ dài ít hơn 3 ký tự cũng sẽ không được quan tâm. Việc phát hiện vùng văn bản sẽ được đánh giá tương tự ở mục 4.1. Sau đó, kết quả nhận dạng của vùng văn bản được phát hiện đúng sẽ được so sánh với văn bản ground truth. Hai chuỗi văn bản sẽ được viết hoa và loại bỏ những dấu câu ở vị trí đầu và cuối trước khi được so sánh có khớp hoàn toàn hay không. Việc tính precision, recall, hmean tương tự mục 4.1.

CHƯƠNG 5: HƯỚNG TIẾP CẬN

5.1 Hướng tiếp cận chung

Có hai hướng tiếp cận chủ yếu cho bài toán này. Hướng thứ nhất là trực tiếp xây dựng ánh xạ từ ảnh đầu vào thành tập các văn bản trong một framework thống nhất. Hướng thứ 2 là sử dụng hai module độc lập được huấn luyện riêng biệt để giải quyết 2 bài toán nhỏ, một cho text detection, một cho text recognition sau đó kết nối chúng lại để giải quyết bài toán lớn. Theo hướng thứ 2, ta sẽ phát biểu 2 bài toán nhỏ như sau:

Phát hiện văn bản (Text Detection): Trong giai đoạn này, đầu vào sẽ là ảnh chứa nội dung văn bản bằng Tiếng Việt, với mỗi hình ảnh có thể chứa một hoặc nhiều đối tượng văn bản và với các biến dạng khác nhau như kích thước, màu sắc, ... và đầu ra của khâu này sẽ là các vùng văn bản hay các hộp giới hạn được cắt ra từ ảnh đầu vào, mỗi hình sẽ chứa các văn bản cần được phát hiện.

Nhận diện văn bản (Text Recognition): Trong giai đoạn này, đầu vào sẽ chính là đầu ra của giai đoạn Text Recognition. Ảnh đầu vào sẽ chỉ là ảnh chứa duy nhất một đối tượng văn bản. Đầu ra sẽ nhận diện được nội dung văn bản từ ảnh đầu vào nói trên.

5.2 Hướng tiếp cận của nhóm

Tuy VinText là bộ dữ liệu được sử dụng cho cuộc thi HCM AIC 2021, giải pháp của các đội lại không được công bố. Các giải pháp nhóm tìm thấy được sử dụng trên bộ dữ liệu này chỉ có trên paper [6] và hầu hết là các phương pháp theo hướng thứ nhất. Vì vậy, nhóm quyết định sẽ thử sử dụng một phương pháp theo hướng thứ hai để so sánh với các phương pháp đã biết.

Được lấy cảm hứng từ giải pháp của top 1 UAIC 2022 (UIT AI Challange 2022 - Nhận dạng văn bản tiếng Việt trong cảnh) (cũng là nhóm chúng em), một giải pháp

theo hướng tiếp cận thứ 2, sử dụng những mô hình được công bố trong 2 năm trở lại đây (phù hợp với mục tiêu của nhóm), nhóm quyết định sử dụng DBNetpp [10] cho giai đoạn detection và PARSeq [11] cho giai đoạn recognition.

5.2.1 DBNetpp (Detection)

DBNetpp được công bố trong [10], là một phương pháp detection dựa trên segmentation. Những phương pháp dựa trên segmentation trước DBNetpp đều gặp phải giới hạn bởi giải thuật hậu xử lý phức tạp và sự mạnh mẽ tỷ lệ (scale robustness) của những mô hình segmentation. Cụ thể, quá trình hậu xử lý bị tách biệt với quá trình tối ưu mô hình và tốn thời gian, còn scale robustness thì luôn được tăng cường bằng cách dung hợp các feature map đa tỷ lệ (multi-scale) một cách trực tiếp. Để giải quyết vấn đề này, DBNetpp kết hợp quá trình nhị phân hóa (binarization), một trong những bước quan trọng của quá trình hậu xử lý, vào segmentation network. Ngoài ra, mô hình còn kết hợp các feature map đa tỷ lệ một cách thích nghi thay vì chỉ đơn giản là concat hay trung bình cộng. Với những cải tiến này, DBNetpp có được độ chính xác cao và tốc độ nhanh nhất so với các phương pháp trước đó.

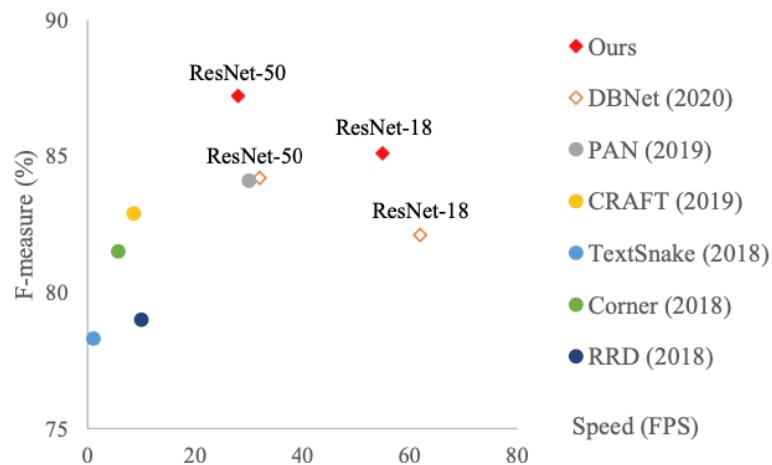


Figure 5.1: Biểu đồ so sánh DBNetpp với một vài phương pháp scene text detection trên dữ liệu MRSA-TD500 với batch size = 1 và một 1080ti GPU. Ours là DBNetpp. DBNetpp có sự trade-off lý tưởng giữa tốc độ và độ chính xác

5.2.1.1 Kiến trúc mô hình

Kiến trúc của mô hình như hình Figure 5.2. Đầu tiên, ảnh đầu vào (RGB) được đưa vào feature-pyramid backbone. Thứ hai, những đặc trưng kim tự tháp được up-sample lên cùng tỷ lệ và đưa vào ASF module để sản xuất đặc trưng ngữ cảnh F. Sau đó, F được sử dụng để dự đoán probability map P và threshold map T. Tiếp đến, binary map xấp xỉ \hat{B} được tính từ B, T. Trong giai đoạn huấn luyện, sự giám sát sẽ được áp dụng trên probability map, threshold map và binary map xấp xỉ, trong đó, probability map và binary map xấp xỉ chia sẻ cùng sự giám sát. Trong giai đoạn suy luận (inference), các bounding box có thể được tạo ra từ binary map xấp xỉ hoặc probability map qua quá trình hình thành box (box formation)

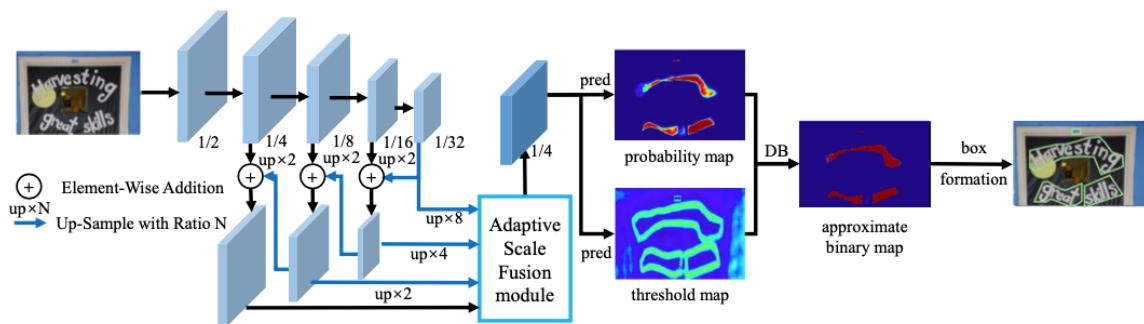


Figure 5.2: Kiến trúc mô hình DBNetpp

5.2.1.1.1 Feature Pyramid Backbone (Backbone kim tự tháp đặc trưng)

Mô hình sử dụng deformable ResNet-18 hoặc deformable ResNet-50 để trích xuất đặc trưng với nhiều tỷ lệ khác nhau. Các đặc trưng với tỷ lệ khác nhau mang trường nhận thức khác nhau nên chúng giúp mô tả các đối tượng văn bản ở các tỷ lệ khác nhau. Ví dụ, những đặc trưng nông hay kích thước lớn có thể nhận thức được những chi tiết của đối tượng văn bản nhỏ nhưng không thể nắm bắt được cái nhìn toàn cục của những đối tượng văn bản lớn. Trong khi những đặc trưng sâu hay kích thước nhỏ lại ngược lại. Ngoài ra, mô hình sử dụng deformable convolution thay vì convolution bình thường là vì deformable convolution cung cấp trường nhận thức

linh hoạt cho mô hình, thứ đặc biệt lợi ích cho những văn bản có tỷ lệ khung hình cực cao.

5.2.1.1.2 Adaptive Scale Fusion (ASF - Bộ dung hợp tỷ lệ thích nghi)

Các đặc trưng với tỷ lệ khác nhau được up-sample lên cùng kích thước (với tỷ lệ cạnh là $\frac{1}{4}$ so với ảnh đầu vào) như hình Figure 5.2 rồi đưa vào ASF module. ASF module nhận đầu vào là $N = 4$ feature map, mỗi feature map $\in R^{CxHxW}$, trong đó, C là số channel, H là chiều cao, W là chiều rộng. Đầu tiên, ta sẽ concat N feature map theo chiều channel rồi đưa qua lớp tích chập 3×3 để nhận được đặc trưng gian $S \in R^{CxHxW}$. Thứ hai, attention weights $A \in R^{NxHxW}$ có thể được tính bằng cách áp dụng spatial attetion module (Figure 5.3) lên đặc trưng S. Thứ 3, attetion weights A được chia thành N phần dọc theo chiều channel và nhân có trọng số với các đặc trưng tương ứng để nhận được đặc trưng dung hợp $F \in R^{NxCxHxW}$.

Kỹ thuật spatial attention trong ASF giúp attetion weights linh hoạt hơn trên chiều không gian.

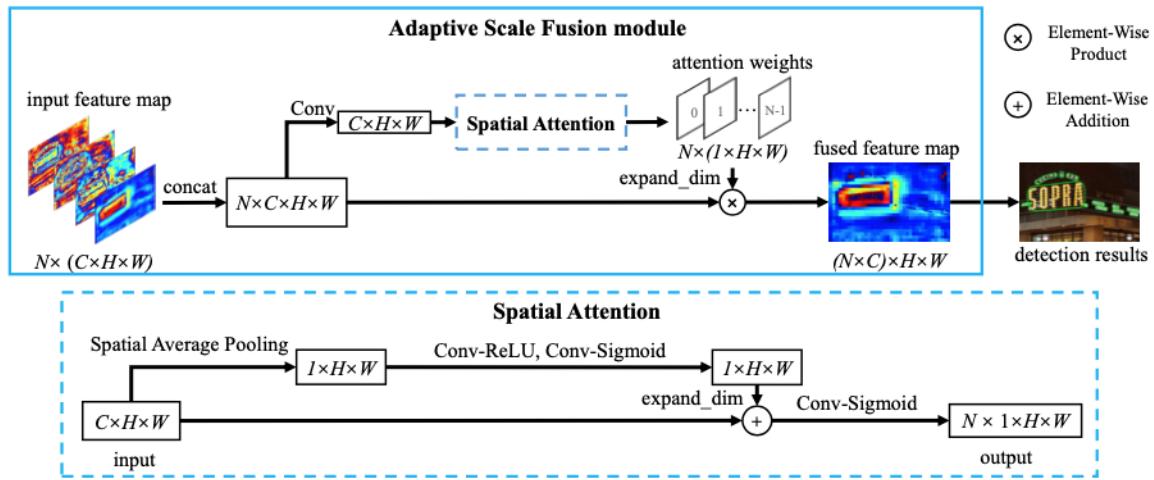


Figure 5.3: Kiến trúc của ASF Module

5.2.1.1.3 Pred block

Úng với probability map và threshold map sẽ có 2 pred block. Pred block gồm một lớp convolution sau bởi 2 lớp transposed convolution với activation function cuối block là sigmoid. Block này có tác dụng khai thác các đặc trưng cục bộ; mở rộng kích thước của đặc trưng ngữ cảnh F thành kích thước đầu vào ban đầu của mạng; dự đoán probability map hay threshold map

5.2.1.1.4 Differentiable Binarization (DB - Nhị phân hóa khả vi)

Xét standard binarization (SB): Cho một probability map $P \in R^{HxW}$ được tạo bởi segmentation network, với H, W lần lượt là chiều cao và chiều rộng của map, việc chuyển đổi nó thành binary map $B \in R^{HxW}$, trong đó pixel có giá trị 1 được cho là vùng văn bản, là thiết yếu. Quá trình nhị phân này luôn có thể được miêu tả như sau:

$$B_{i,j} = \begin{cases} 1 & \text{nếu } P_{i,j} \geq t, \\ 0 & \text{còn lại} \end{cases} \quad (1)$$

Trong đó, t là threshold được định nghĩa trước và (i, j) chỉ vị trí pixel trên map.

Xét differentiable binarization: SB được mô tả theo Eq1 là không khả vi. Vì vậy, nó không thể được tối ưu cùng với segmentation network trong giai đoạn huấn luyện. Để giải quyết vấn đề này, DBNetpp sử dụng DB với hàm xấp xỉ như sau:

$$\hat{B}_{i,j} = \frac{1}{1 + e^{-k(P_{i,j} - T_{i,j})}} \quad (2)$$

Trong đó, $\hat{B}_{i,j}$ là binary map xấp xỉ, T là threshold map thích ứng được dự đoán bởi network, $k = 50$ là nhân tố khuếch đại. Hàm xấp xỉ hoạt động tương tự với SB (Nhìn

Figure 5.4) nhưng có thể cùng tối ưu với segmentation network trong giai đoạn huấn luyện. DB với threshold thích ứng không những giúp mô hình phân biệt được giữa văn bản và nền mà còn tách biệt những văn bản liên kết chặt chẽ với nhau.

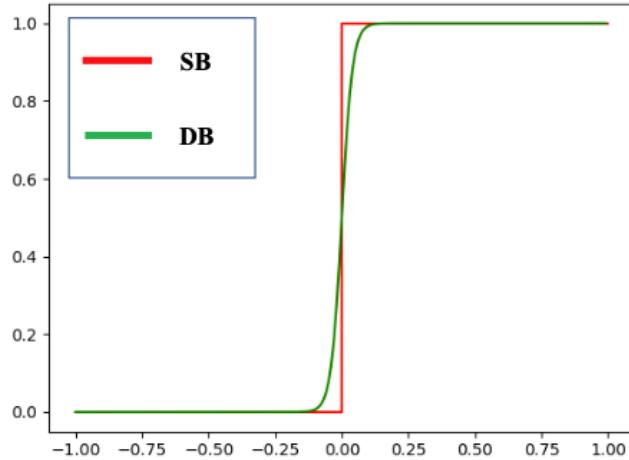


Figure 5.4: Đồ thị của SB và DB

Việc sử dụng DB giúp tăng hiệu suất của mô hình là bởi nó cho gradient tốt khi back propagation. Binary cross-entropy loss được biểu diễn như sau:

$$L_{bce} = -\frac{1}{N} \sum_{i=1}^H \sum_{j=1}^W (\hat{y}_{i,j} \log(y_{i,j}) + (1 - \hat{y}_{i,j}) \log(1 - y_{i,j})) \quad (3)$$

Trong đó, $y_{i,j} \in [0,1]$ và $\hat{y}_{i,j} \in \{0,1\}$ lần lượt là giá trị dự đoán và giá trị mục tiêu. Vì vậy, trong nhiệm vụ segmentation, hàm loss l^+ cho nhãn positive (label = 1) và hàm loss l^- cho nhãn negative (label = 0) là:

$$l^+ = -\log(y_{i,j})$$

$$l^- = -\log(1 - y_{i,j})$$

(4)

Nếu không quan tâm đến activation function, đạo hàm của segmentation loss có thể được tính với chain rule:

$$\frac{\partial l^+}{\partial y_{i,j}} = -\frac{1}{y_{i,j}}$$

$$\frac{\partial l^-}{\partial y_{i,j}} = \frac{1}{1 - y_{i,j}}$$

(5)

Đặt $x_{i,j} = P_{i,j} - T_{i,j}$. Hàm DB có thể được viết thành $f(x) = \frac{1}{1 + e^{-kx_{i,j}}}$. Tương tự, hàm loss l_b^+ cho nhãn positive và l_b^- cho nhãn negative là:

$$l_b^+ = -\log\left(\frac{1}{1 + e^{-kx_{i,j}}}\right)$$

$$l_b^- = -\log\left(1 - \frac{1}{1 + e^{-kx_{i,j}}}\right)$$

(6)

Đạo hàm của hàm loss với hàm DB:

$$\frac{\partial l_b^+}{\partial x_{i,j}} = \frac{-ke^{-kx_{i,j}}}{1 + e^{-kx_{i,j}}}$$

$$\frac{\partial l_b^-}{\partial x_{i,j}} = \frac{k}{1 + e^{-kx_{i,j}}}$$

(7)

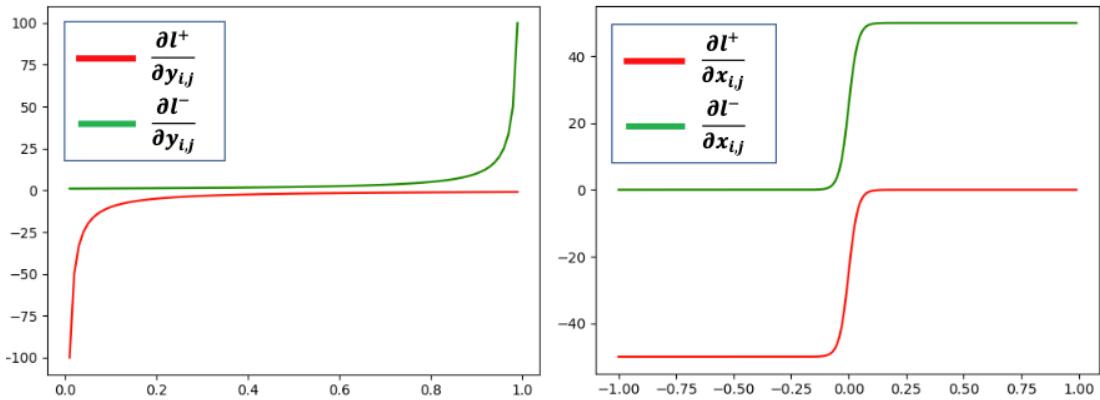


Figure 5.5: Đạo hàm của Eq5 (Trái) và Eq7 (phải)

Khi vẽ đồ thị của Eq5 và Eq7 ta sẽ có được đồ thị như Figure 5.5. Từ đồ thị, ta có thể nhận ra được 2 điều sau:

1: Xét độ lớn của đạo hàm xung quanh giá trị biên. Đối với SB binary cross-entropy loss, độ lớn của $\frac{\partial l^+}{\partial y_{i,j}}$ và $\frac{\partial l^-}{\partial y_{i,j}}$ khá nhỏ xung quanh giá trị biên (0.5) giữa giá trị positive (>0.5) và giá trị negative (< 0.5). Hệ quả là backpropagation có thể không đáng kể khi giá trị dự đoán nằm gần giá trị biên như 0.4 và 0.6. Với DB binary cross-entropy loss, độ lớn của $\frac{\partial l_b^+}{\partial x_{i,j}}$, $\frac{\partial l_b^-}{\partial x_{i,j}}$ lớn xung quanh giá trị biên (< 0), nơi được tăng cường bởi nhân tố khuếch đại k . Vì vậy, DB giúp tạo ra các dự đoán phân biệt xung quanh giá trị biên.

2: Xét giới hạn trên nhỏ nhất và giới hạn dưới lớn nhất. với SB binary cross-entropy, không có giới hạn dưới lớn nhất cho $\frac{\partial l^+}{\partial y_{i,j}}$ và không có giới hạn trên nhỏ nhất cho $\frac{\partial l^-}{\partial y_{i,j}}$.

Còn với DB binary cross-entropy, giới hạn dưới lớn nhất của $\frac{\partial l_b^+}{\partial x_{i,j}}$ và giới hạn trên nhỏ nhất của $\frac{\partial l_b^-}{\partial x_{i,j}}$ được xác định bởi nhân tố khuếch đại k . Vì vậy, DB có xu hướng bên vững hơn với một vài giá trị cực lớn hoặc cực nhỏ.

Threshold map có xu hướng nhán mạnh biên của vùng văn bản ngay cả khi không có sự giám sát cho threshold map. Điều này chỉ ra rằng threshold map giống biên vùng văn bản sẽ có ích với kết quả cuối. Vì vậy, Áp dụng sự giám sát lên threshold map sẽ định hướng cho mô hình học tốt hơn.

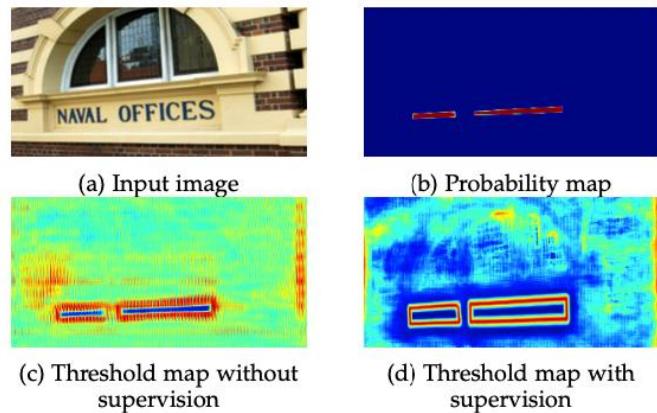


Figure 5.6: Threshold map có hoặc không có sự giám sát

5.2.1.2 Khởi tạo nhán

Khởi tạo nhán cho probability map và binary map xáp xỉ. Với một ảnh văn bản, mỗi polygon của vùng văn bản được mô tả bằng tập các phân đoạn:

$$\mathbf{G} = \{\mathbf{S}_k\}_{k=1}^n$$

n là số định, có thể khác nhau tuỳ vào từng bộ dữ liệu, ví dụ như 4 cho VinText, 16 cho CTW1500. Sau đó, vùng positive được khởi tạo bằng cách làm co polygon G thành G_s sử dụng thuật toán Vatti clipping [12]. Độ dời \mathbf{D} của quá trình làm co được tính từ chu vi \mathbf{L} , diện tích \mathbf{A} của polygon gốc và hệ số co $r = 0.4$:

$$\mathbf{D} = \frac{\mathbf{A}(1 - r^2)}{\mathbf{L}}$$

Khởi tạo nhán cho threshold map. Với thủ tục tương tự trên, ta có thể tạo nhán cho threshold map. Đầu tiên polygon \mathbf{G} được kéo dãn với độ dời \mathbf{D} thành \mathbf{G}_d . Chúng ta

xem khoảng cách giữa \mathbf{G}_s và \mathbf{G}_d như là biên của vùng văn bản, ở đó, nhãn của threshold map có thể được khởi tạo bằng cách tính khoảng cách tới phân đoạn gần nhất của \mathbf{G} .

5.2.1.3 Hàm mất mát

Hàm loss L được tính:

$$L = L_s + \alpha x L_b + \beta x L_t$$

Trong đó, L_s, L_b, L_t lần lượt là hàm loss của probability map, binary map xấp xỉ và threshold map; α và β được đặt bằng 1 và 10 một cách tương ứng.

DBNetpp sử dụng binary cross-entropy (BCE) loss cho cả L_s và L_b . Để vượt qua sự mất cân bằng giữa số lượng nhãn positive và negative, ta lấy mẫu cho tập S_t sao cho tỉ lệ giữa positive và negative là 1:3. S_t bao gồm tất cả các mẫu positive và top-k negative (được sắp xếp bằng xác suất dự đoán), với k bằng 3 lần số lượng mẫu positive. Như vậy:

$$L_s = L_b = \sum_{i \in S_t} (y_i \log(x_i) + (1 - y_i) \log(1 - x_i))$$

L_t được tính như là tổng của các khoảng cách L1 giữa các dự đoán và nhãn bên trong polygon được kéo dãn \mathbf{G}_d :

$$L_t = \sum_{i \in R_d} |y_i^* - x_i^*|$$

Trong đó, R_d là tập các pixel nằm bên trong polygon được kéo dãn \mathbf{G}_d , y^* là nhãn cho threshold map.

5.2.1.4 Sự hình thành polygon (box formation)

Trong giai đoạn suy luận, chúng ta có thể sử dụng hoặc probability map hoặc binary map xấp xỉ để tạo polygon cho văn bản, vì cả 2 đều cho kết quả hầu như giống nhau. Để hiệu quả tốt hơn, chúng tôi sử dụng probability map để nhánh threshold map được loại bỏ. Quá trình hình thành box bao gồm 3 bước: Thứ nhất, probability map hoặc binary map xấp xỉ được nhị phân hóa với hằng số threshold 0.2, để nhận được binary map; Thứ 2, những vùng liên thông (vùng văn bản bị co) được nhận lấy từ binary map; Thứ 3, Các vùng văn bản bị co được kéo dãn với độ dài D' theo thuật toán Vatti clipping. D' được tính theo chu vi L' và diện tích A' của polygon bị co và hằng số $r' = 1.5$:

$$D' = \frac{A' \times r'}{L'}$$

5.2.2 PARSeq (Recognition)

5.2.2.1 Ý tưởng

Có ba cách giải mã chuỗi ký tự phổ biến:

- Autoregressive (AR) Decoding: Mô hình AR dự đoán mỗi ký tự dựa trên các ký tự đã được dự đoán trước đó và các đặc trưng trích xuất từ ảnh đầu vào. Nói cách khác, mỗi ký tự chỉ được dự đoán sau khi ký tự trước đó đã được xác định.
- Non-autoregressive (NAR) Decoding: Mô hình NAR, ngược lại, dự đoán tất cả các ký tự trong chuỗi cùng một lúc. Điều này dựa trên ảnh và một token đặc biệt, ví dụ như [B] (Token bắt đầu).
- Iterative Refinement (IR) Decoding: Mô hình IR dự đoán mỗi ký tự dựa trên tất cả các ký tự ở vị trí khác trong chuỗi. Điều này giống như việc sửa lỗi chuỗi đã dự đoán, với mỗi dự đoán mới là một bước cải thiện đối với dự đoán trước đó.

Đối với AR, mô hình chỉ có thể học sự phụ thuộc của các token theo 1 hướng thường là trái sang phải (L2R) hoặc phải sang trái (R2L). Ngoài ra, trong quá trình infer mô hình chỉ có thể dự đoán các token theo tuần tự đã được huấn luyện.

Mô hình ABINet [13] kết hợp NAR và IR, cho phép dự đoán tất cả các ký tự trong chuỗi, sau đó dùng external language model để sửa lỗi dự đoán. Tuy nhiên sự độc lập giữa language model và ảnh đầu vào dẫn đến khả năng điều chỉnh sai những dự đoán đúng nếu sai chính tả hoặc có từ tương đồng với xác suất cao hơn.

PARSeq được giới thiệu để giải quyết những hạn chế trên. Có thể xem PARSeq như một tập hợp các mô hình AR và học nhiều hoán vị của chuỗi nhãn thay vì chỉ là L2R hoặc R2L như các mô hình AR truyền thống. Việc này giúp mô hình học được cả AR, NAR, và IR. Nó cho phép mô hình xem xét nhiều khả năng sắp xếp của chuỗi đầu ra, giúp cải thiện khả năng dự đoán và chú ý đến các mô hình giải mã khác nhau.

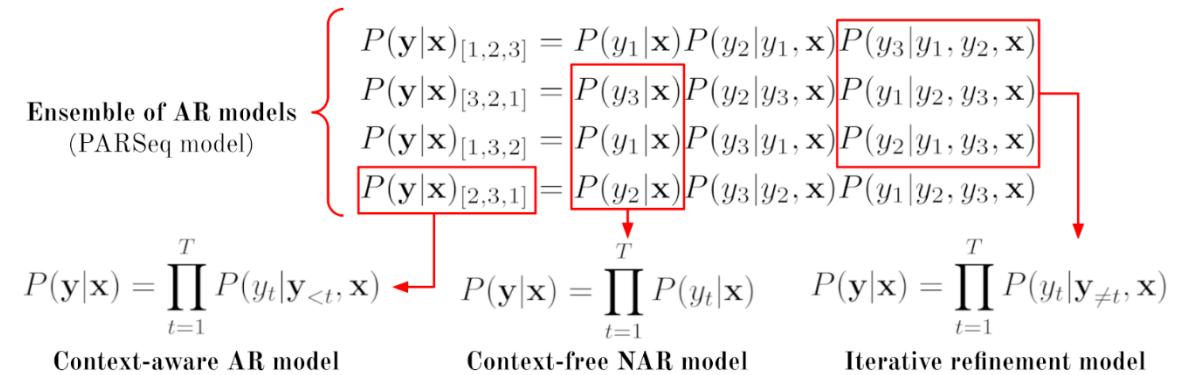


Figure 5.7: Minh họa của mô hình NAR và IR (cloze) trong mối quan hệ với sự kết hợp của nhiều mô hình AR cho hình x với nhãn y gồm 3 thành phần.

5.2.2.2 Kiến trúc mô hình

Kiến trúc mô hình PARSeq được chia thành 2 khối chính: Vision Transformer (ViT) Encoder và Visio-lingual decoder (Bộ giải mã ngôn ngữ trực quan).

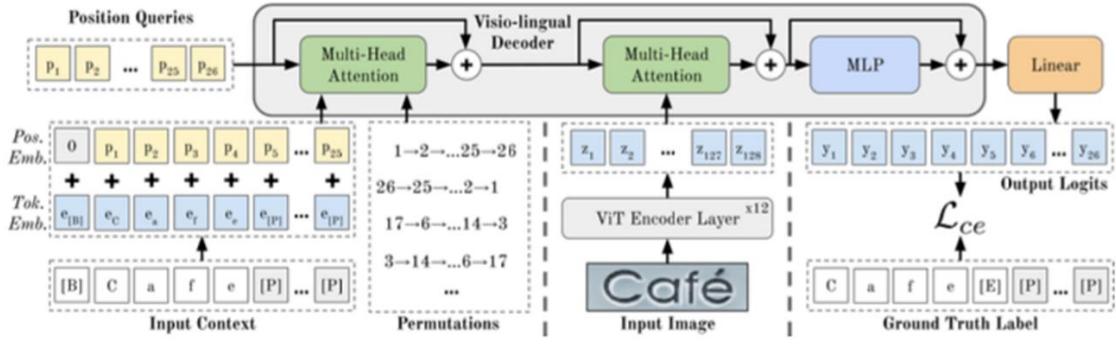


Figure 5.8: Kiến trúc mô hình PARSeq

Khối ViT Encoder gồm 12 lớp ViT layers có chung kiến trúc với Transformer Encoder, gồm 1 mô-đun Multi-head Attention (MHA) được sử dụng cho self-attention. Các query vector (q), key vector (k) và value vector (v) đều bằng nhau. Không có đầu phân loại (classification head) và không có token [CLS] trong mô hình.

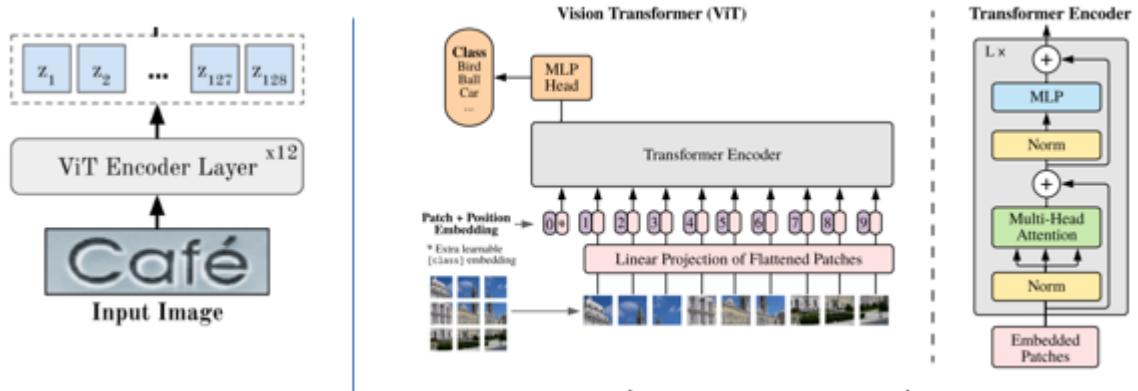


Figure 5.9: Kiến trúc ViT Encoder. Trái là encoder trong PARSeq. Phải là encoder trong paper ViT gốc.

Khác với mô hình ViT tiêu chuẩn, tất cả các token đầu ra z của khối ViT encoder được sử dụng làm đầu vào cho mô-đun giải mã (khối Visio-lingual decoder).

$$z = \text{Enc}(x) \in R^{\left(\frac{W \cdot H}{p_w \cdot p_h}\right) \times d_{\text{model}}}$$

Khối Visual-lingual decoder của mô hình này được xây dựng theo cấu trúc của Transformer với một số sửa đổi, đặc biệt là việc sử dụng preLayerNorm và số lượng attention heads gấp đôi (n head = $d_{model}/32$) so với Transformer tiêu chuẩn. Decoder nhận ba đầu vào bắt buộc là position (vị trí), context (ngữ cảnh), và image tokens (token ảnh), cùng với một attention mask tùy chọn. Output của decoder là logits (giá trị đầu ra trước khi áp dụng hàm softmax) có kích thước là $(T+1) \times (S+1)$, trong đó T là độ dài của chuỗi đầu ra và S là kích thước của bộ ký tự (charset) được sử dụng để huấn luyện mô hình.

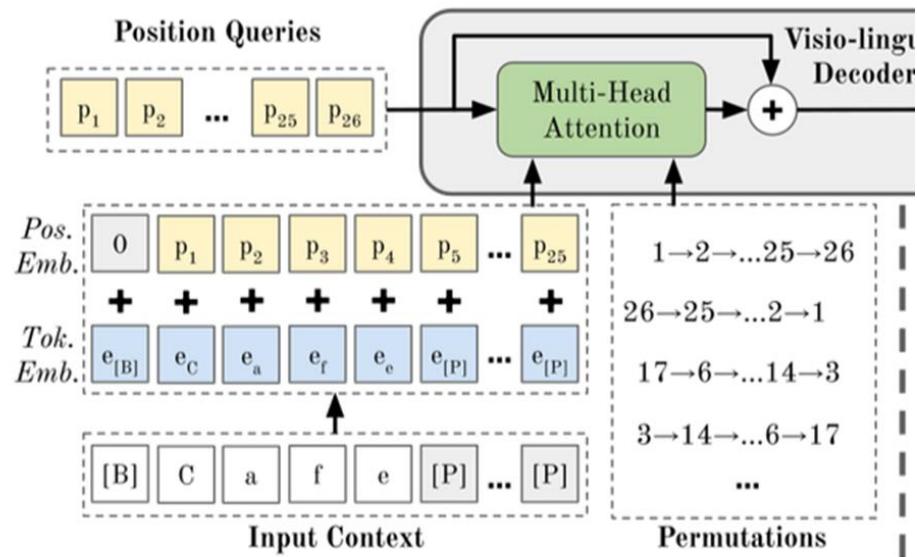


Figure 5.10: Khối MHA đầu tiên trong Visio-lingual decoder

Mô hình sử dụng một Multi-Head Attention (MHA) module cho attention giữa context và position theo công thức:

$$h_c = p + \text{MHA}(p, c, c, m) \in R^{(T+1) \times d_{model}}$$

Với p là position query, c là context embedding, m là mask tùy theo AR, NAR, IR sẽ là 1 ma trận khác nhau. T là context length ($T+1$ do sử dụng các token đặc biệt như [B] hoặc [E]).

Nếu không có mã thông báo vị trí, tức là nếu mã thông báo ngữ cảnh được sử dụng làm truy vấn giống như trong Transformers tiêu chuẩn, mô hình sẽ không học được bất kỳ điều gì có ý nghĩa từ PLM và sẽ chỉ hoạt động giống như mô hình AR tiêu chuẩn.

Trong quá trình huấn luyện, mask được tạo từ các hoán vị ngẫu nhiên. Trong quá trình dự đoán (inference), mask có thể là một standard left-to-right lookahead mask (AR decoding), cloze mask (iterative refinement), hoặc không mask (NAR decoding). Quyết định về mask được sử dụng phụ thuộc vào cách mô hình được sử dụng.

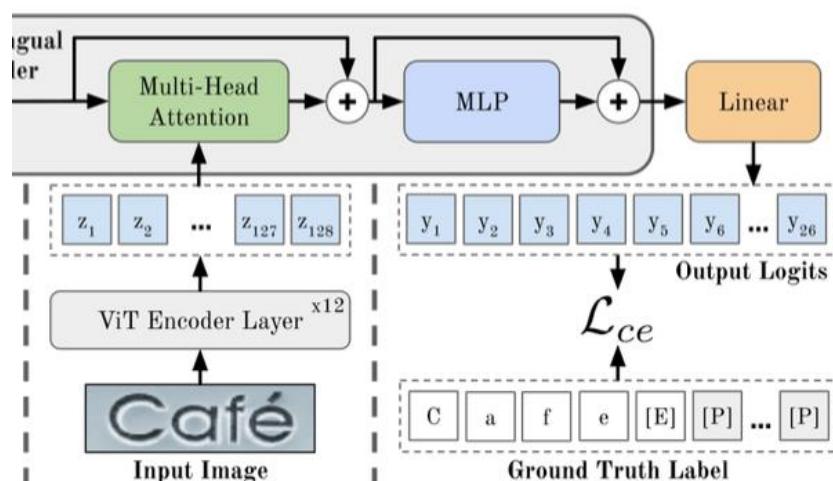


Figure 5.11: Khối MHA thứ hai và khối MLP của Visio-lingual decoder

MHA thứ hai được sử dụng để attention giữa ảnh và vị trí. Công thức cho phần này là:

$$h_i = h_c + \text{MHA}(h_c, z, z) \in R^{(T+1) \times d_{model}}$$

Hidden state cuối cùng của decoder là đầu ra của một Multi-Layer Perceptron (MLP), và logits đầu ra được tính dựa trên:

$$y = \text{Linear}(h_{\text{dec}}) \in R^{(T+1) \times (S+1)}$$

Trong đó h_{dec} là hidden state cuối của decoder:

$$h_{\text{dec}} = h_i + \text{MLP}(h_i) \in R^{(T+1) \times d_{\text{model}}}$$

Tóm lại, với một attention mask m , khôi decoder của mô hình là một hàm:

$$y = \text{Dec}(z, p, c, m) \in R^{(T+1) \times (S+1)}$$

Mục tiêu của mô hình PARSeq là tối đa hóa xác suất chuỗi văn bản $y = [y_1, y_2, \dots, y_T]$ dưới tập hợp các tham số θ cho ảnh đầu vào x . Trong mô hình AR tiêu chuẩn, xác suất được phân rã bằng chain rule theo thứ tự $[1, 2, \dots, T]$ và công thức log xác suất của mô hình là:

$$\log p(y | x) = \sum_{t=1}^T \log p_\theta(y_t | y_{<t}, x)$$

Nhưng trong mô hình Transformer, tất cả các token được xử lý đồng thời, cho phép các token đầu ra truy cập tất cả các token đầu vào, điều này không phù hợp với tính chất AR. Vì vậy Permutation Language Modeling [14] (PLM) ra đời để giải quyết vấn đề này bằng cách đào tạo mô hình trên tất cả các sắp xếp có thể của chuỗi nhãn, thay vì chỉ theo trình tự chuẩn theo thứ tự $[1, 2, \dots, T]$. Cụ thể, mỗi sắp xếp tạo ra một attention mask khác nhau để đảm bảo rằng mô hình không thể phụ thuộc vào thông tin từ các token tương lai khi dự đoán token hiện tại.

$$\log p(y|x) = E_{z \sim Z_T} \left[\sum_{t=1}^T \log p_\theta(y_{z_t} | y_{z < t}, x) \right]$$

Trong đó, Z_T là tập tất cả hoán vị có thể của chuỗi vị trí $[1, 2, \dots, T]$; z_t và z_{t+1} lần lượt là phần tử ở vị trí thứ t và $t - 1$ phần tử đầu của hoán vị $z \in Z_T$. Để cài đặt PLM trong Transformers, ta không cần thiết phải thực sự hoán vị nhãn văn bản y , mà thay vào đó, ta sẽ tạo attention mask để bắt buộc thứ tự được xác định bởi z .

Thực tế, không thẻ huấn luyện trên tất cả các sắp xếp có thẻ (T!), thay vào đó chỉ sử dụng K sắp xếp khả thi, được chọn một cách cụ thể để giữ cho quá trình huấn luyện ổn định hơn.

Hàm mất mát là trung bình của các mất mát cross-entropy đối với mỗi attention mask tương ứng với một sắp xếp, không tính vào mất mát các padding token. Công thức của hàm mất mát là:

$$L = \frac{1}{K} \sum_{k=1}^K L_{ce}(y_k, \hat{y}) \quad \text{với } y_k = \text{Dec}(z, p, c, m_k)$$

5.2.2.3 Hiệu suất mô hình

Mô hình PARSeq có độ chính xác cao và có chi phí tính toán thấp hơn khi so sánh với các mô hình khác.

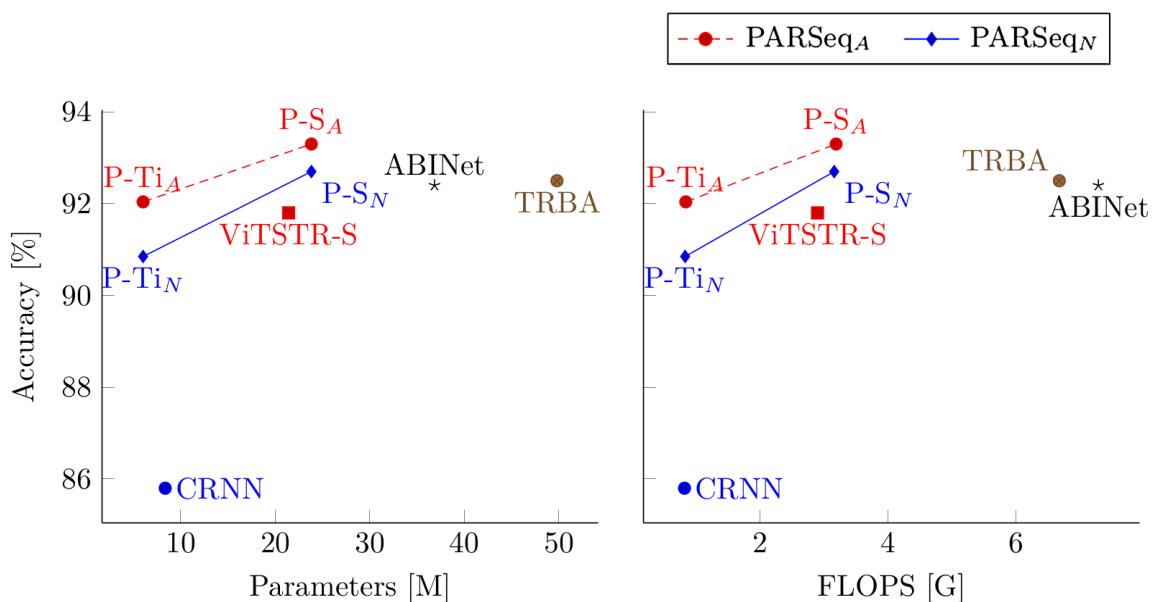


Figure 5.12: Biểu đồ so sánh kích thước, độ chính xác và FLOPS giữa PARSeq với một số mô hình khác.

PARSeq có hai loại cài đặt là PARSeq_A (AR + 1 Iterative refinement) và PARSeq_N (NAR + 2 Iterative refinement). Cùng với hai phiên bản PARSeq-S

(phiên bản gốc) và PARSeq-Ti (phiên bản nhỏ hơn với số tham số bằng $\frac{1}{2}$ so với bản gốc).

Ta thấy PARSeq-S đạt được độ chính xác từ trung bình cao nhất, ít tham số và FLOPS hơn đáng kể trên cả 2 loại. PARSeq-Ti cũng đạt được độ chính xác từ cao hơn nhiều so với CRNN [15].

CHƯƠNG 6: THỰC NGHIỆM

6.1 Cài đặt DBNetpp

6.1.1 Thông số mô hình

6.1.1.1 Feature Pyramid Backbone

Sử dụng kiến trúc của ResNet-50 như hình Figure 6.1, tuy nhiên chỉ sử dụng feature map đầu ra c_2, c_3, c_4, c_5 của conv2_x, conv3_x, conv4_x, conv5_x tương ứng với số channels là 256, 512, 1024, 2048. Ngoài ra, thay các lớp convolution với kernel size 3x3 thành deformable convolution với kernel size 3x3.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
				3×3 max pool, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1			average pool, 1000-d fc, softmax		
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 6.1: Các kiến trúc của ResNet

$c_i (2 \leq i \leq 5)$ được đi qua một lớp convolution tương ứng, với kernel size 1x1, out channels 256 trước khi đi qua lớp upsample với scale factor 2, chế độ nearest (c_2 không được upscale x2).

Quan sát Figure 6.2, với nhánh nằm ngang ứng với c_2 , ngay trước khi đi vào ASF module sẽ có một lớp convolution với kernel size 3x3, out channels 64, padding 1. Với nhánh nằm ngang ứng với c_3 , ngay trước khi đi vào ASF sẽ có lớp convolution với kernel size 3x3, out channels 64, padding 1, theo sau bởi lớp upsample với scale factor 2 và chế độ nearest. Với nhánh nằm ngang ứng với c_4 , ngay trước khi đi vào

ASF sẽ có lớp convolution với kernel size 3x3, out channels 64, padding 1, theo sau bởi lớp upsample với scale factor 4 và chế độ nearest. Với nhánh nằm ngang ứng với c_5 , ngay trước khi đi vào ASF sẽ có lớp convolution với kernel size 3x3, out channels 64, padding 1, theo sau bởi lớp upsample với scale factor 8 và chế độ nearest

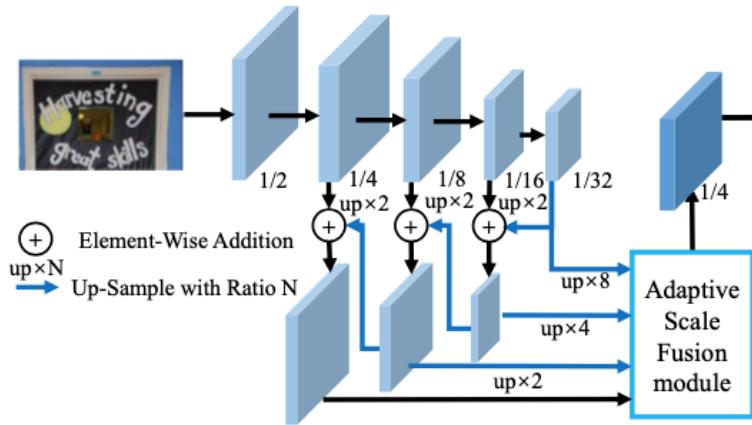


Figure 6.2: Feature Pyramid Backbone

6.1.1.2 Adaptive Scale Fusion

ASF module bao bắt đầu bằng việc concat 4 feature maps được trích xuất từ FPN theo chiều channel. Tiếp đến đi qua một lớp convolution với kernel size 3x3, out channel 64, padding 1, stride 1. Sau đó đi qua khối spatial attention để tạo ra attention weights. Attention weights nhân với đầu vào được concat ban đầu để tạo ra được fused feature map.

Khối spatial attention chứa 3 lớp convolution với các thông số lần lượt là: kernel size 3x3, out channels 1, padding 1, stride 1; kernel size 1x1, out channels 1, padding 0, stride 1; kernel size 1x1, out channels 4, padding 0, stride 1.

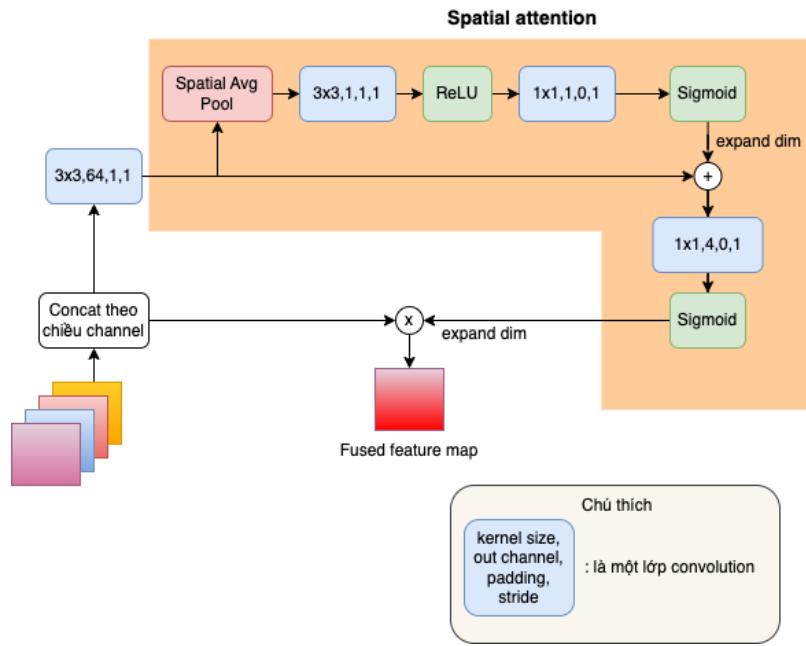


Figure 6.3: Mô tả chi tiết của khối ASF

6.1.1.3 Pred Block

Threshold map và probability map sẽ có một pred block riêng. Mỗi pred block sẽ có kiến trúc giống nhau như Figure 6.4. Pred block bao gồm một lớp convolution với kernel size 3x3, out channel 64, padding 1, stride 1 có tác dụng tích chập các thông tin cục bộ, được sau bởi 2 lớp transposed convolution với cùng thông số kernel size 2x2, padding 0, stride 2, riêng out channel lần lượt là 64 và 1. Hai lớp transposed convolution này có tác dụng tích chập và mở rộng độ dài mỗi cạnh của feature map gấp 4 lần để trở về kích thước đầu vào ban đầu của mô hình.

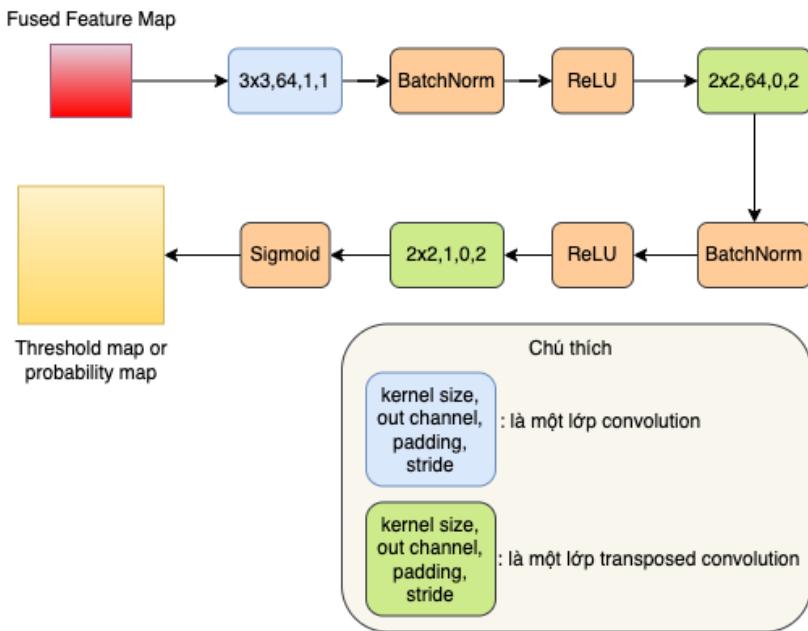


Figure 6.4: Mô tả chi tiết của khối Pred

6.1.1.4 Differentiable Binarization

Hàm DB sử dụng nhân tố khuếch đại $k = 50$

6.1.2 Tăng cường dữ liệu

Flip left right với xác suất 50%, xoay ảnh với góc $\in [-10^\circ; 10^\circ]$, Random crop với tỷ lệ cạnh nhỏ nhất 0.1.

6.1.3 Môi trường huấn luyện và các siêu tham số khác

Môi trường: Colab pro plus, GPU V100, VRAM 51 GB

Ảnh đầu vào:

- Kích thước khi huấn luyện (w x h) là 640 x 640
- Kích thước khi test/val (w x h) là 4068 x 1024
- Số kênh 3

Số epoch 600

Batch size cho train và val/test lần lượt là 8, 4

Optimizer SGD; learning rate 0.007; momentum 0.0001

Learning rate scheduler: Poly learning rate

6.2 Cài đặt PARSeq

6.2.1 Thông số mô hình

Embedding dimension là 384

Vision encoder: 6 attention heads; MLP ratio 4; Độ sâu 12 khối vision transformer

Visual-lingual decoder: 12 attention heads; MLP ratio 4; Độ sâu 1 khối

Dropout rate 0.1

Chế độ decode là auto regressive với 1 lần điều chỉnh.

6.2.2 Tăng cường dữ liệu

Sử dụng RandAugment với số layer bằng 3 và magnitude bằng 5, xác suất sử dụng bằng 0.5 và tập các thao tác xử lý ảnh gồm 15 thao tác. Nghĩa là ta sẽ chọn lần lượt 3 thao tác xử lý ảnh trong tập thao tác với tần suất như nhau. Sau khi chọn xong, ứng với mỗi layer, ta sẽ có xác suất để sử dụng thao tác đã chọn là 50%. Với magnitude là 5, thông số của 15 thao tác sẽ như sau:

Thao tác	Thông số
AutoContrast	-
Equalize	-
Invert	-
Rotate	degress $\in [-15^\circ; 15^\circ]$

Posterize	# bit = 2
SolarizeAdd	threshold = 128, add = 55
Color	factor = 1
Brightness	factor = 1
Contrast	factor = 1
ShearX	factor = 0.15
ShearY	factor = 0.15
TranslateXRel	pct = 0.05
TranslateYRel	pct = 0.05
GaussianBlur	radius = 2
PoissionNoise	lambda = 21

Table 6.1: Thông số của các thao tác tăng cường dữ liệu cho PARSeq

6.2.3 Môi trường huấn luyện và các siêu tham số khác

Môi trường: Colab thường, GPU Tesla T4, VRAM 16 GB

Ảnh đầu vào:

- Kích thước (w x h) là 128 x 32
- Số channel 3
- Patch size (w x h) là 8 x 4

Từ điển ký tự gồm 229 ký tự bao gồm các ký tự trong bảng chữ cái tiếng việt (kết hợp với các dấu thanh), tiếng anh kẽ cả hoa thường, các chữ số từ 0 – 9, các dấu câu, khoảng cách và một số ký tự đặc biệt.

Kích thước nhãn tối đa mà mô hình dự đoán là 25

Số hoán vị chuỗi là 6

Số epoch 20

Batch size cho cả train, val, test là 200

Optimizer Adam; learning rate 0.0007

6.3 Kết quả thực nghiệm

6.3.1 Chọn threshold cho DBNetpp

Kết quả dự đoán của DBNetpp gồm polygons và confidence scores, polygon có score càng thấp thể hiện mức độ tự tin của mô hình về dự đoán càng thấp. Việc lọc bỏ những polygon có score nhỏ hơn hoặc bằng giá trị threshold có thể giúp kết quả đánh giá tốt hơn cho cả detection và end to end. Nếu threshold nhỏ thì recall cao nhưng precision thấp, còn nếu threshold lớn thì precision cao nhưng recall thấp. Để chọn ra được threshold tốt nhất, nhóm đã tiến hành grid search threshold trên **[0.1; 0.9]** với bước nhảy là **0.1** trên tập validation. Đối với detection, Figure 6.5 cho thấy threshold = 0.7 cho kết quả đánh giá tốt nhất với recall = 0.8014, precision = 0.9332, hmean = 0.8623. Còn đối với end to end, quan sát Figure 6.6 ta thấy được threshold = 0.7 vẫn cho kết quả tốt nhất với precision = 0.7778, recall = 0.7132, hmean = 0.7441. Như vậy, ta sẽ chọn threshold = 0.7 để đánh giá trên tập test cho cả detection và end to end.

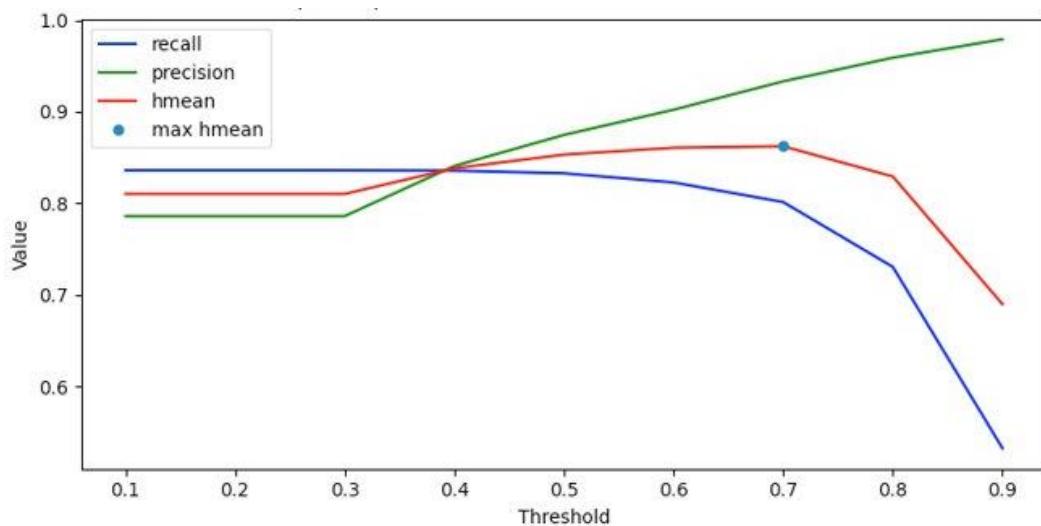


Figure 6.5: Hmean, precision, recall của detection trên tập val với threshold từ 0.1 đến 0.9

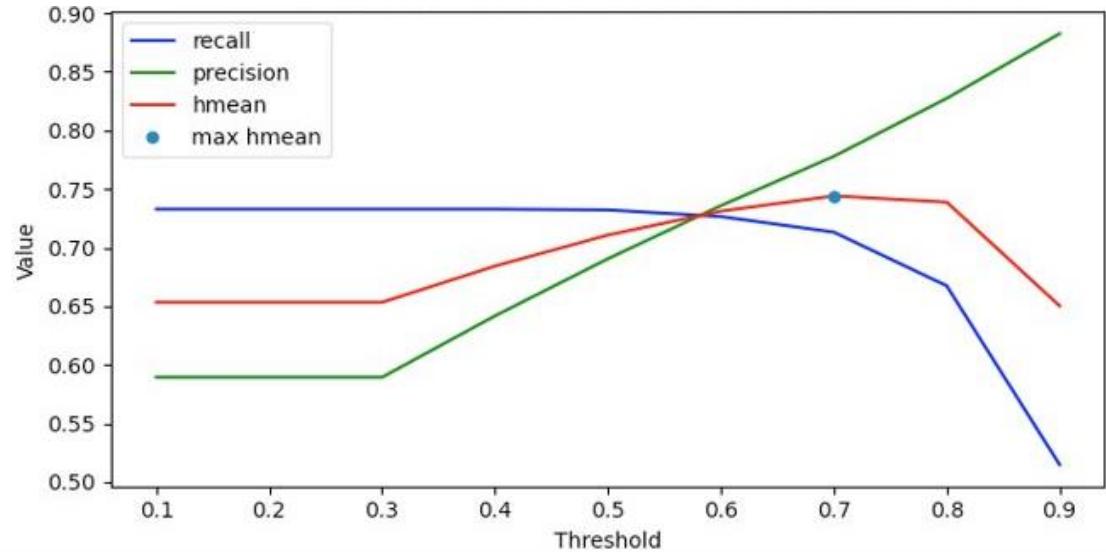


Figure 6.6: Hmean, precision, recall của end to end trên tập val với threshold từ 0.1 đến 0.9

6.3.2 Kết quả đánh giá

Phương pháp detection cho kết quả khá tốt với Hmean là 0.8839. Phương pháp recognition cũng tương tự, với Accuracy, OneMinusNED lần lượt là 0.8703 và 0.9446. Ngoài ra, phương pháp end to end kết hợp DBNetpp với PARSeq cho kết quả tốt nhất với Hmean = 0.785 khi so sánh với các phương pháp khác trên VinText được đề cập trong paper (Table 6.1).

Detection			Recognition			End2End		
Precision	Recall	Hmean	Accuracy	OneMinusNED	Precision	Recall	Hmean	
0.9332	0.8396	0.8839	0.8703	0.9446	0.8073	0.7649	0.785	

Table 6.2: Kết quả đánh giá trên tập test (threshold = 0.7 cho detectoin và end to end)

Phương pháp	Hmean
ABCNet + dictionary	58.5

ABCNet+D + dictionary	63.6
Mask Textspotterv3+D + dictionary	70.3
DBNetpp + PARSeq (Ours)	78.5

Table 6.3: Hmean của một số phương pháp trên tập VinText. Những phương pháp +D sử dụng kỹ thuật dictionary guided, + dictionary sử dụng bộ tự vựng trong quá trình suy luận

6.4 Phân tích lỗi sai

Tuy kết quả đánh giá trên tập VinText khá tốt, nhưng phương pháp vẫn có những hạn chế cần phải kể đến. Việc phân tích những trường hợp dự đoán sai sẽ giúp nhóm biết được những hạn chế này đồng thời đưa ra những định hướng phát triển trong tương lai.

6.4.1 Detection

Trong phần này, các vùng văn bản ground truth sẽ được tô màu đỏ, còn những vùng văn bản được dự đoán sẽ có đường viền màu lục. Các ground truth không được dự đoán có thể do mô hình bỏ qua hoặc được dự đoán nhưng confidence score quá thấp nên bị lọc bỏ trong quá trình hậu xử lý. Ta sẽ xem 2 trường hợp này là do mô hình bỏ qua.





Figure 6.7: Văn bản nhỏ, mờ bị mô hình bỏ qua trong lúc dự đoán



Figure 6.8: Số điện thoại bị dự đoán rời rạc thành hai hay nhiều polygon khác nhau





Figure 6.9: Nhầm biếu tượng với văn bản



Figure 6.10: Gộp các vùng văn bản với nhau thành một





Figure 6.11: Khó nhận dạng các văn bản có phông chữ đặc biệt hoặc chữ viết tay



Figure 6.12: Vùng văn bản nằm trên nền phức tạp như có nhiều đường gạch, màu nền
gắn giống màu văn bản, ...



Figure 6.13: Thiếu dấu

6.4.2 Recognition

Ảnh	Dự đoán	Ground truth
	lamon	lemon
	COMFANY	COMPANY
	TOH:DO	10h:00

Table 6.4: Hình ảnh kích thước nhỏ, mờ, nhòe

Ảnh	Dự đoán	Ground truth
	CHÀO	CHÂU
	Hanoi,	Hanoi

	VÀO	VAO
--	-----	-----

Table 6.5: Nền phông tạp

Ảnh	Dự đoán	Ground truth
	la	Cá
	laVie	LaVie
	hungoy	hungoy

Table 6.6: Phông chữ đặc biệt, chữ viết tay

Ảnh	Dự đoán	Ground truth
	LLOX	xanh
	UNI	LINH

	Chuyang	shipping
--	---------	----------

Table 6.7: Văn bản mà người còn khó đọc

Ảnh	Dự đoán	Ground truth
	T3	M

Table 6.8: Nhầm văn bản bình thường thành văn bản nghệ thuật

Ảnh	Dự đoán	Ground truth
	cafe	caFe
	Wine	WINe
	CO	cɔ̄

Table 6.9: Nhầm lẫn ký tự hoá – thường

Ảnh	Dự đoán	Ground truth
	Multimeia	Multimedia
	0 (số)	O (chữ cái)
	I	/

Table 6.10: Nhầm lẫn giữa các ký tự gần giống nhau (do bản chất gần giống nhau hoặc do môi trường làm biến đổi ký tự trở nên gần giống với ký tự khác)

Ảnh	Dự đoán	Ground truth
	109 05'	109'05'0
	87 38	87'38
	Nội	Nội,

Table 6.11: Không nhận dạng được các ký tự đặc biệt

Ảnh	Dự đoán	Ground truth
	0703 824 786	0703824786
	Cholime	Cholimex
	0918.387.817	0918.387.81

Table 6.12: Dữ liệu bị gán nhãn sai

6.5 Nhận xét

Phương pháp cho kết quả tốt nhất so với các phương pháp được đề cập trong bài báo công bố bộ dữ liệu VinText. Tuy nhiên mô hình vẫn còn một số hạn chế khi dự đoán sai trong các trường hợp được phân tích trên và bộ dữ liệu vẫn có một số mẫu bị đánh nhãn sai.

CHƯƠNG 7: XÂY DỰNG DEMO

7.1 Các công cụ chính hỗ trợ

7.1.1 FastAPI

FastAPI là một web framework Python hiện đại, rất hiệu quả trong việc xây dựng API, code đơn giản nhưng hỗ trợ tốt cho việc làm ra sản phẩm.



Figure 7.1: FastAPI

Các tính năng của FastAPI:

- Nhanh: Hiệu năng rất cao khi so sánh với NodeJS và Go. Một trong những Python framework nhanh nhất, giúp hệ thống truy vấn phản hồi nhanh hơn.
- Code nhanh: Tăng tốc độ phát triển tính năng từ 200% tới 300%
- Ít lỗi hơn: Do đơn giản nên giảm khoảng 40% những lỗi được sinh ra bởi developer.
- Trực giác tốt hơn: Được các trình soạn thảo hỗ trợ tuyệt vời. Completion mọi nơi. Ít thời gian gỡ lỗi.
- Dễ dàng: Được thiết kế để dễ dàng học và sử dụng. Ít thời gian đọc tài liệu.
- Ngắn: Tối thiểu việc lặp code. Các tham số truyền vào có nhiều tính năng. Ít bugs.
- Mạnh mẽ: hiệu năng mạnh mẽ, có thể tương tác API qua docs.

7.1.2 ReactJS

ReactJS là một trong những công cụ phát triển front-end hot nhất hiện nay. Nó là một opensource được phát triển bởi Facebook, ra mắt vào năm 2013, bản thân nó là một thư viện Javascript được dùng để để xây dựng các tương tác với các thành phần trên website.



Figure 7.2: ReactJS

Một trong những điểm nổi bật nhất của ReactJS đó là việc render dữ liệu không chỉ thực hiện được trên tầng Server mà còn ở dưới Client. ReactJS giúp xây dựng giao diện nhanh, hạn chế lỗi trong khi code; Tách ứng dụng thành các Component nhỏ hơn, dễ quản lý, tái sử dụng và mở rộng ứng dụng; Debug dễ dàng với 1 Chrome extension do Facebook phát triển.

7.2 Phân tích thiết kế hệ thống

Hệ thống sẽ có hai chức năng chính:

- Chức năng hiển thị thông tin sau khi xử lý ảnh bằng pipeline End to End
- Chức năng tải file JSON (chứa thông tin output từ pipeline End to End của từng ảnh) về máy local

Sơ đồ hệ thống cho chức năng đầu tiên như sau:

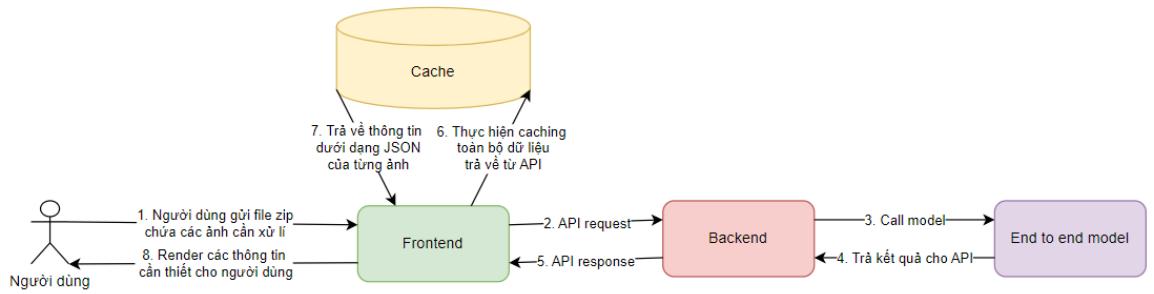


Figure 7.3: Sơ đồ hệ thống cho chức năng 1

Sơ đồ hệ thống cho chức năng thứ hai như sau:

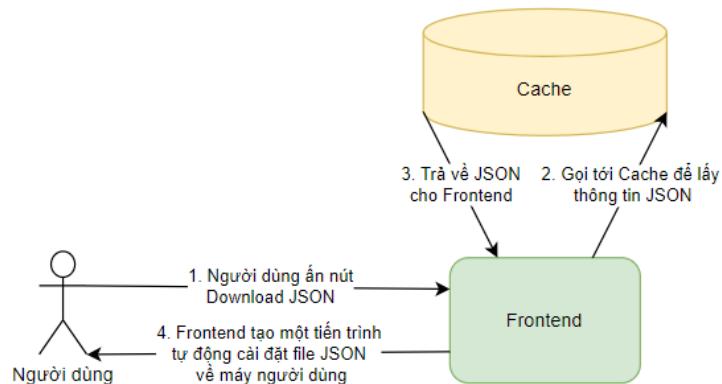


Figure 7.4: Sơ đồ hệ thống cho chức năng 2

7.3 Thiết kế giao diện

Web sẽ có hai phần chính: bên trái (vùng màu cam) là **Gallery**, bên phải là **Info**. Để thử nghiệm, người dùng cần ấn nút Upload Image và gửi file images.zip, file này chứa các ảnh mà người dùng cần demo.

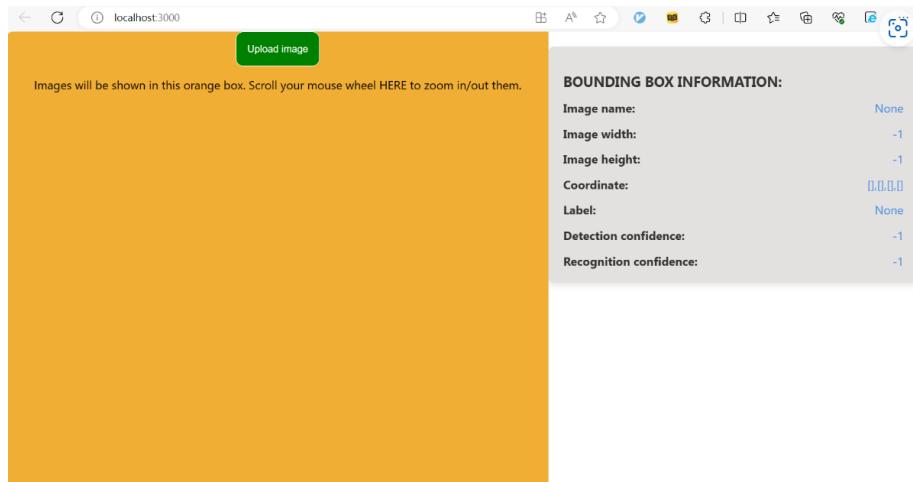


Figure 7.5: Giao diện

7.4 Hướng dẫn sử dụng

Đây là giao diện sau khi người dùng gửi file .zip lên hệ thống (để xem thông tin của bounding box, người dùng có thể click vào box tương ứng trên hình):

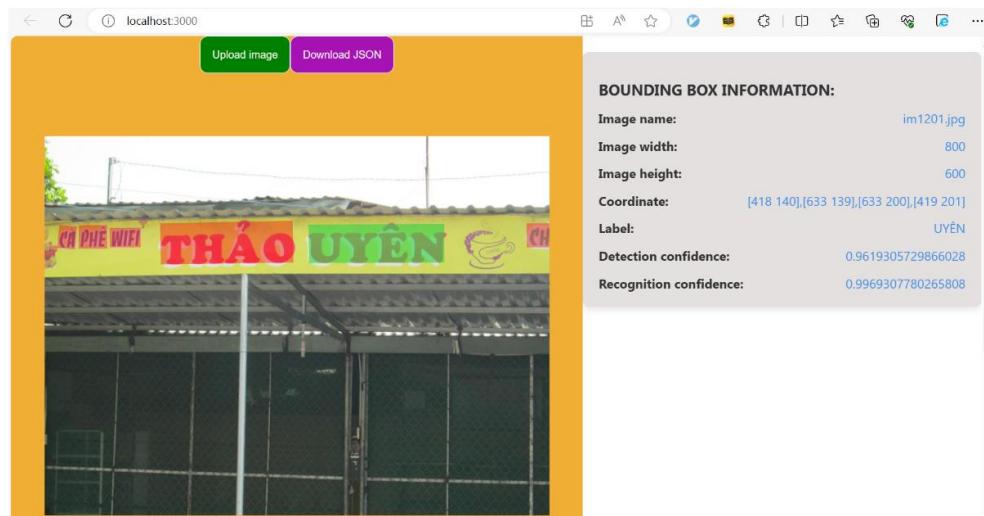


Figure 7.6: Giao diện sau khi gửi file zip

Nếu người dùng không nhìn rõ chi tiết trong ảnh thì có thể lăn chuột để phóng to:

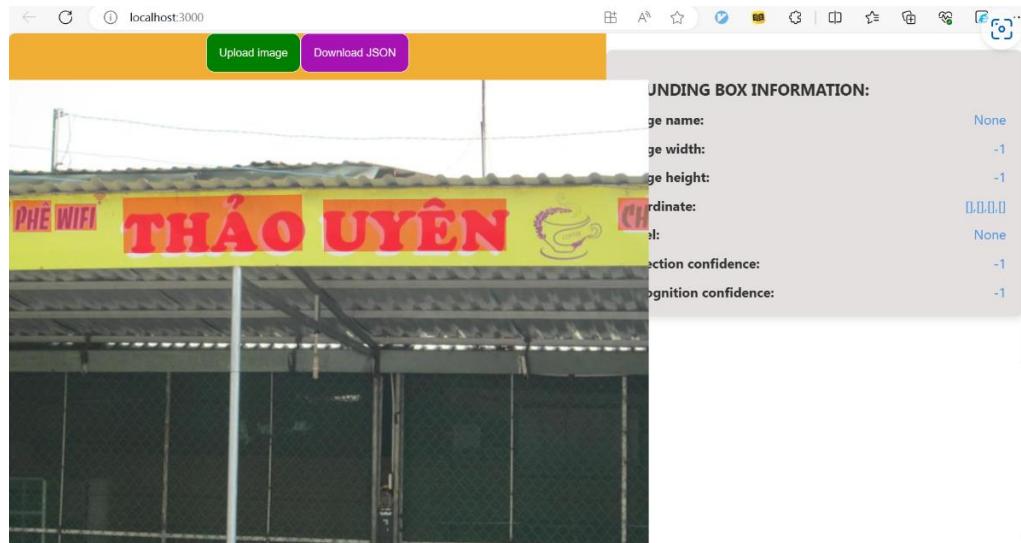


Figure 7.7: Giao diện sử dụng chức năng thay đổi kích thước ảnh

Lưu ý, người dùng chỉ có thể cuộn trang lên/xuống bằng cách lăn chuột ở phần **Info**, bởi vì sự kiện lăn chuột ở phần **Gallery** được chuyển thành phóng to/thu nhỏ ảnh.

Cuối cùng, nếu người dùng cần kết quả xử lí của toàn bộ ảnh trong file .zip, bấm vào nút Download JSON để tải file .json về máy của mình.

7.5 Môi trường cài đặt

Hệ thống được cài đặt trên cấu hình môi trường như sau:

- Frontend: React (18.2.0), SASS (1.69.5)
- Backend: FastAPI (0.104.1), uvicorn (0.24.0.post1)

KẾT LUẬN

Sau khi tìm hiểu và thực hiện đồ án “Phát hiện và nhận dạng văn bản tiếng Việt trong ngoại cảnh” chúng tôi đã hoàn thành đề tài và đạt được các kết quả như sau:

- Tìm hiểu tổng quan về bài toán phát hiện và nhận dạng văn bản trong ngoại cảnh, quy trình xây dựng ứng dụng web để demo
- Nắm được các kiến thức bao gồm kiến trúc và nhiệm vụ của các thành phần, thuật toán và các kỹ thuật sử dụng trong mô hình phát hiện và nhận dạng văn bản theo phương pháp học sâu:
 - CNN, transposed convolution, deformable convolution, multi-head self attention, ...
 - Các thao tác xử lý ảnh, RandAugment giúp tăng cường dữ liệu
 - Áp dụng 2 mô hình DBNetpp, PARSeq cho tiếng Việt.
- Nghiên cứu và ứng dụng các công cụ như Fastapi, ReactJS. Huấn luyện các mô hình trên tập dataset VinText
- Xây dựng một ứng dụng web trực quan, dễ dàng sử dụng, cho phép người dùng gửi ảnh và trả về tập các văn bản được phát hiện và nhận dạng.

Định hướng nghiên cứu tiếp theo:

- Tìm hiểu và sử dụng các mô hình detection, recognition tốt hơn hoặc một mô hình end to end gồm 2 module detection và recognition được huấn luyện cùng nhau.
- Áp dụng kỹ thuật dict-guided cho PARSeq cũng như các mô hình recognition khác để tăng khả năng nhận dạng.

- Sử dụng các dataset có sẵn khác, tìm hiểu một số phương pháp tăng cường dữ liệu giúp mô hình học tốt hơn.
- Ứng dụng:
 - Cải tiến pipeline hiện tại hoặc sử dụng các phương pháp khác tốt hơn.
 - Sử dụng cpu, gpu mạnh để xử lý yêu cầu nhanh hơn.
 - Mở rộng sang mobile app, kết hợp với mô hình dịch máy để tạo ra ứng dụng dịch ảnh tiện lợi với người dùng.

TÀI LIỆU THAM KHẢO

- [1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition,” trong *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016.
- [2] Jonathan Long, Evan Shelhamer, Trevor Darrell, “Fully convolutional networks for semantic segmentation,” trong *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, 2015.
- [3] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, Yichen Wei, “Deformable Convolutional Networks,” trong *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017.
- [4] Ashish Vaswani, et al., “Attention Is All You Need,” trong *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*, California, USA, 2017.
- [5] Alexey Dosovitskiy, et al., “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” trong *The Ninth International Conference on Learning Representations (ICLR 2021)*, Vienna, Austria, 2021.
- [6] Nguyen Nguyen, Thu Nguyen, Vinh Tran , Minh-Triet Tran, Thanh Duc Ngo, Thien Huu Nguyen, Minh Hoai, “Dictionary-guided Scene Text Recognition,” trong *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Nashville, TN, USA, 2021.
- [7] Yuliang Liu, Hao Chen, Chunhua Shen, Tong He, Lianwen Jin, Liangwei Wang, “ABCNet: Real-Time Scene Text Spotting With Adaptive Bezier-Curve Network,” trong *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Seattle, WA, USA, 2020.
- [8] Minghui Liao, Guan Pang, Jing Huang, Tal Hassner, Xiang Bai, “Mask TextSpotter v3: Segmentation Proposal Network for Robust Scene Text Spotting,” trong *ECCV 2020*, Glasgow, Scotland, 2020.
- [9] Kuang, et al., “MMOCR: A Comprehensive Toolbox for Text Detection, Recognition and Understanding,” mmocr, 6 4 2023. [Trực tuyến]. Available: https://mmocr.readthedocs.io/en/latest/basic_concepts/evaluation.html#metrics. [Đã truy cập 16/11/2023].
- [10] Minghui Liao, Zhisheng Zou, Zhaoyi Wan, Cong Yao, Xiang Bai, “Real-Time Scene Text Detection with Differentiable Binarization and Adaptive Scale Fusion,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, tập 45, số 1, pp. 919 - 931, 2022.
- [11] Darwin Bautista, Rowel Atienza, “Scene Text Recognition with Permuted Autoregressive Sequence Models,” trong *the 17th European Conference on Computer Vision (ECCV 2022)*, Tel-Aviv, 2022.
- [12] B. R. Vatti, “A generic solution to polygon clipping,” *Communications of the ACM*, tập 35, số 7, p. 56–63, 1992.
- [13] Fang, S., Xie, H., Wang, Y., Mao, Z., Zhang, Y., “Read like humans: Autonomous, bidirectional and iterative language modeling for scene text recognition,” trong *2021*

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR),
Nashville, TN, USA, 2021.

- [14] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R.R., Le, Q.V., “Xlnet: Generalized autoregressive pretraining for language understanding,” trong *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, Vancouver, BC, Canada, 2019.
- [15] Baoguang Shi, Xiang Bai, Cong Yao, “An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, tập 39, số 11, pp. 2298 - 2304, 2017.