

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA CÔNG NGHỆ THÔNG TIN



KHÓA LUẬN TỐT NGHIỆP

XÂY DỰNG HỆ THỐNG
CHÚ THÍCH ẢNH TỰ ĐỘNG

Người hướng dẫn: TS MAI NGỌC THẮNG
ThS VÕ HOÀNG ANH

Người thực hiện: HỒNG QUANG VINH - 51303447
NGUYỄN TRỌNG CÔNG - 51303246

Lớp: 13050303

Khóa: 17

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2017

LỜI CẢM ƠN

Sau đây nhóm xin chân thành cảm ơn sự hướng dẫn của thầy Mai Ngọc Thắng và cô Võ Hoàng Anh trong quá trình hướng dẫn làm luận văn. Những kiến thức mà thầy và cô giảng dạy đã góp phần cung cấp những kiến thức cần thiết cho chúng em vào quá trình làm luận văn. Xin chân thành cảm ơn thầy và cô.

LUẬN VĂN ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là sản phẩm luận văn tốt nghiệp của riêng chúng tôi và được sự hướng dẫn của TS Mai Ngọc Thắng và ThS Võ Hoàng Anh. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong luận văn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình. Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Hồng Quang Vinh

Nguyễn Trọng Công

TÓM TẮT

Nhận diện đối tượng (object recognition) là một bài toán quan trọng trong lĩnh vực thị giác máy tính. Đây là một hướng nghiên cứu có nhiều ứng dụng trong thực tế như giao thông, thể thao, giao tiếp người máy, an ninh...

Tự động chú thích nội dung của một hình ảnh là một bài toán trong trí tuệ nhân tạo, kết nối giữa thị giác máy tính và xử lý ngôn ngữ tự nhiên. Luận văn này trình bày một mô hình tự động chú thích dựa trên một kiến trúc Deep Recurrent, kết hợp những tiến bộ gần đây trong thị giác máy tính và dịch thuật để có thể được sử dụng để nhận diện và mô tả nội dung của hình ảnh.

Luận văn này được thực hiện với các mục đích:

- Tìm hiểu các phương pháp nhận diện và mô tả đối tượng khác nhau.
- Ứng dụng các thuật toán nhận diện và mô tả đối tượng vào thực tiễn, đặc biệt là lĩnh vực mô tả nội dung của hình ảnh.

Các nội dung chính được trình bày trong luận văn gồm:

- Rút trích đặc trưng hình ảnh bằng thuật toán Convolutional Neural Network (CNN).
- Học và mô tả hình ảnh dùng thuật toán Long Short-Term Memory (LSTM).
- Ứng dụng thực tế mô tả nội dung của ảnh.

Mục lục

1 TỔNG QUAN ĐỀ TÀI	10
1.1 Giới thiệu đề tài – Bài toán nhận diện đối tượng	10
1.2 Mục tiêu đề tài – Bài toán nhận diện, mô tả đối tượng trong ảnh .	10
1.3 Phạm vi đề tài	14
1.3.1 Giới thiệu bài toán nhận diện và mô tả nội dung hình ảnh .	14
1.3.2 Cơ sở dữ liệu sử dụng trong bài toán	14
1.4 Bố cục	15
2 CÁC CÔNG TRÌNH LIÊN QUAN	16
2.1 Rich feature hierarchies for accurate object detection and semantic segmentation [5]	16
2.1.1 Các kỹ thuật được sử dụng	16
2.1.2 Các kết quả chính	16
2.2 Deep Fragment Embeddings for Bidirectional Image Sentence Mapping [6]	17
2.2.1 Các kỹ thuật được sử dụng	17
2.2.2 Các kết quả chính	17
2.3 Deep Visual-Semantic Alignments for Generating Image Descriptions [4]	18
2.3.1 Các kỹ thuật được sử dụng	18
2.3.2 Hướng tiếp cận	18
3 CƠ SỞ LÝ THUYẾT	21
3.1 Convolutional Neural Networks (CNN)	21
3.1.1 Giới thiệu	21
3.1.2 Cấu trúc Convolutional Neural Network	22

3.2	Recurrent Neural Network (RNN)	40
3.2.1	Giới thiệu	40
3.2.2	Các dạng của RNN	41
3.2.3	Phân tích RNN	43
3.2.4	Công thức RNN	45
3.2.5	Ví dụ RNN	45
3.2.6	Huấn luyện RNN	47
3.3	Long Short-Term Memory (LSTM)	48
3.3.1	Nhắc lại Recurrent Neural Network	48
3.3.2	Vấn đề phụ thuộc quá dài (Long-Term Dependencies)	49
3.3.3	LSTM Network	52
3.3.4	Ý tưởng của LSTM	53
3.3.5	Phân tích mô hình LSTM	55
3.4	LSTM nâng cao	57
3.5	Hàm Softmax	58
4	Ý TƯỞNG GIẢI QUYẾT BÀI TOÁN	59
4.1	Mô hình huấn luyện tổng quan	59
4.1.1	Xử lý hình ảnh	60
4.1.2	Xử lý ngôn ngữ tự nhiên	64
4.2	Mô hình thử nghiệm tổng quan	67
4.2.1	Xử lý hình ảnh	68
4.2.2	Xử lý ngôn ngữ tự nhiên	71
4.3	Phương pháp đánh giá và tinh chỉnh tham số cho mô hình huấn luyện	74
4.3.1	Độ hỗn loạn thông tin - <i>Perplexity</i>	74
4.3.2	Cách tính <i>Perplexity</i>	74
4.3.3	Áp dụng vào mô hình huấn luyện	75
5	KẾT QUẢ THỰC NGHIỆM	76
5.1	Đánh giá kết quả tạo ra các câu mô tả	76
5.1.1	Tiêu chí đánh giá BLEU [10]	76
5.1.2	Phương pháp tìm kiếm Beam Search	77
5.1.3	Kết quả khi tạo mô tả cho toàn bộ tấm ảnh:	78
5.2	Kết quả thực nghiệm tạo câu chú thích cho ảnh	80

5.3 Demo	84
6 ỨNG DỤNG THỰC TẾ	85
6.1 Ứng dụng Camera Vision	85
6.1.1 Giới thiệu	85
6.1.2 Sơ đồ use case và phân tích use case	86
6.1.3 Sơ đồ lớp	88
6.1.4 Sơ đồ hoạt động	89
6.1.5 Sơ đồ tuần tự	91
6.2 Ứng dụng Web Vision	92
6.2.1 Giới thiệu	92
6.2.2 Sơ đồ use case và phân tích use case	93
6.2.3 Sơ đồ hoạt động	94
6.2.4 Sơ đồ tuần tự	95
7 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	96
7.1 Kết luận	96
7.2 Hướng phát triển	97

Danh sách hình vẽ

1.1	Ảnh dữ liệu bị mờ	12
1.2	Nội dung đoạn mô tả bị sai	12
1.3	Lượng từ vựng không phong phú	13
1.4	Mô tả một tấm ảnh trong Dataset	14
2.1	Nhận diện đối tượng và ghép thành câu hoàn chỉnh	17
2.2	Thứ tự các câu có nghĩa giống nhất	18
2.3	Kết quả hướng tiếp cận tạo chú thích cho từng vùng ảnh	19
2.4	Kết quả hướng tiếp cận tạo chú thích cho toàn bộ tấm ảnh	20
3.1	Mô hình mạng tích chập	21
3.2	Mô hình mạng tích chập	23
3.3	Ví dụ kiến trúc ConvNet	25
3.4	Hình ảnh CIFAR-10 32x32x3	26
3.5	Minh họa về bố trí không gian	28
3.6	Ví dụ bộ lọc được học bởi Krizhevsky	30
3.7	Các bước tính toán đầu ra	32
3.8	Trượt các khối từ trái qua phải, trên xuống dưới với bước nhảy là 2	33
3.9	Trượt các khối từ trái qua phải, trên xuống dưới với bước nhảy là 2	34
3.10	Trượt các khối từ trái qua phải, trên xuống dưới với bước nhảy là 2	35
3.11	Trượt các khối từ trái qua phải, trên xuống dưới với bước nhảy là 2	36
3.12	Minh họa tầng pooling	37
3.13	Mô hình RNN	40
3.14	Dạng RNN một - nhiều	41
3.15	Dạng RNN nhiều - một	42
3.16	Dạng RNN nhiều - nhiều	42
3.17	Một dạng RNN nhiều - nhiều khác	43

3.18	Phân tích RNN	43
3.19	RNN tạo các vector đầu ra	44
3.20	Công thức tổng quát của RNN	44
3.21	Công thức RNN	45
3.22	Truyền ký tự vào RNN	45
3.23	Áp dụng công thức RNN	46
3.24	Tính toán xác xuất của từ	46
3.25	Mô hình RNN có vòng lặp	48
3.26	Tách các vòng lặp từ RNN	49
3.27	Vấn đề phụ thuộc dài	49
3.28	Vấn đề phụ thuộc dài	50
3.29	Công thức LSTM	50
3.30	Áp dụng liên tiếp hàm <i>sigmoid()</i>	51
3.31	Cấu trúc RNN chuẩn	52
3.32	Cấu trúc LSTM	52
3.33	Ký hiệu sử dụng	53
3.34	Đường truyền trạng thái	54
3.35	Cổng LSTM	54
3.36	Đầu vào thông tin	55
3.37	Quyết định thông tin	55
3.38	Cập nhật vào Cell State	56
3.39	Thông tin đầu ra	56
3.40	Mô hình LSTM nâng cao	57
4.1	Mô hình huấn luyện tổng quan	59
4.2	Tiền xử lý hình ảnh	60
4.3	Áp dụng CNN	61
4.4	Mô hình VGG Net 16 Layer	62
4.5	Mô hình VGG Net áp dụng vào bài toán	63
4.6	Mô hình tiền xử lý câu mô tả	64
4.7	Áp dụng LSTM	65
4.8	Dựa vector đặc trưng hình vào mạng LSTM	65
4.9	Áp dụng LSTM	66
4.10	Tiếp tục áp dụng LSTM	66

4.11	Mô hình thử nghiệm tổng quan	67
4.12	Tiền xử lý hình ảnh	68
4.13	Áp dụng CNN	69
4.14	Mô hình VGG Net 16 Layer	69
4.15	Mô hình VGG Net áp dụng vào bài toán	70
4.16	Áp dụng LSTM	71
4.17	Truyền hình ảnh vào CNN	71
4.18	Áp dụng LSTM	72
4.19	LSTM truyền thông tin sang bước tiếp theo	72
4.20	Tiếp tục áp dụng LSTM	73
4.21	LSTM truyền thông tin sang bước tiếp theo	73
4.22	LSTM kết thúc	74
5.1	Kết quả khi tạo mô tả cho toàn bộ tấm ảnh	78
5.2	Biểu đồ đánh giá tạo câu mô tả cho toàn bộ tấm ảnh	79
5.3	Ảnh có câu chú thích phù hợp	80
5.4	Ảnh có câu chú thích phù hợp	81
5.5	Ảnh có câu chú thích chưa phù hợp	82
5.6	Ảnh có câu chú thích chưa phù hợp	83
5.7	Demo chạy với thời gian thực	84
6.1	Sơ đồ use case của ứng dụng Camera Vision	86
6.2	Sơ đồ lớp	88
6.3	Sơ đồ hoạt động UC01	89
6.4	Sơ đồ hoạt động UC02	90
6.5	Sơ đồ hoạt động UC03	90
6.6	Sơ đồ tuần tự UC01	91
6.7	Sơ đồ tuần tự UC02	91
6.8	Sơ đồ tuần tự UC03	92
6.9	Sơ đồ use case của Web Vision	93
6.10	Sơ đồ hoạt động của UC01	94
6.11	Sơ đồ tuần tự của UC01	95

Danh sách bảng

1.1	Bảng thông tin về các bộ cơ sở dữ liệu sử dụng trong bài toán	14
2.1	Tỉ lệ chính xác khi dự đoán	16
5.1	Thực nghiệm đánh giá kết quả phương pháp Beam Search trên tập MSCOCO	77
5.2	Đánh giá tạo câu mô tả cho toàn bộ tấm ảnh	79
6.1	Đặc tả actor	86
6.2	Đặc tả use case	87
6.3	Phân tích use case chụp hình	87
6.4	Phân tích use case đổi ngôn ngữ	87
6.5	Phân tích use case thoát ứng dụng	88
6.6	Đặc tả actor	93
6.7	Đặc tả use case	93
6.8	Phân tích use case gửi hình ảnh	94

Danh mục ký hiệu và chữ viết tắt

Các ký hiệu

Các chữ viết tắt

CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ConvNets	Convolutional Neural Networks
Conv	Convolutional
FC	Fully-connected

Chương 1

TỔNG QUAN ĐỀ TÀI

1.1 Giới thiệu đề tài – Bài toán nhận diện đối tượng

Nhận diện đối tượng trong thị giác máy tính là công việc tìm kiếm các vật thể trong ảnh. Đây là hướng nghiên cứu được quan tâm nhiều trong lĩnh vực công nghệ hiện nay với rất nhiều ứng dụng. Con người có thể nhận biết được các đối tượng từ ảnh một cách rất dễ dàng mặc dù sự khác nhau giữa chúng là vô cùng đa dạng. Tuy nhiên, việc này đối với hệ thống thị giác máy tính vẫn là một thách thức rất lớn. Nguyên nhân chủ yếu là do sự đa dạng về hình dáng, màu sắc của vật thể, đa dạng về diện mạo và tư thế của mỗi loại vật thể, ảnh hưởng về điều kiện ánh sáng, sự che lấp lẫn nhau giữa các vật thể, chất lượng ảnh, .v.v.

Ý tưởng chính trong các phương pháp nhận diện đối tượng là dựa vào đặc trưng cụ thể được rút ra từ những đối tượng mẫu. Đặc trưng này được sử dụng cùng với bộ phân lớp hoặc sử dụng thuật toán để nhận ra những đối tượng tương tự với các đối tượng mẫu.

Nhận diện, mô tả đối tượng trong ảnh là một nhánh của nhận diện đối tượng.

1.2 Mục tiêu đề tài – Bài toán nhận diện, mô tả đối tượng trong ảnh

Để có thể tự động chú thích nội dung ảnh bằng tiếng Anh là một nhiệm vụ khó khăn, nhưng có tác động rất to lớn như là giúp người khiêm thị hiểu rõ hơn về nội dung của hình ảnh trên web, nhận diện phân loại các đối tượng có trong ảnh... đây là các vấn đề chính của thị giác máy tính. Một câu chú thích không chỉ mô

tả đối tượng trong ảnh, mà còn phải mô tả được các mối liên hệ, sự liên kết của đối tượng đó với đối tượng khác trong cùng tấm ảnh. Hơn nữa, những kiến thức về ngữ pháp tiếng Anh cũng cần được biểu hiện, nghĩa là một mô hình ngôn ngữ cần được thêm vào để làm trực quan hơn.

Mục tiêu bài toán:

- Hiện thực các giải thuật tích hợp trên bộ cơ sở dữ liệu trên mạng.
- Xây dựng hệ thống có thể chạy được trên điện thoại và web.

Nhận diện, mô tả đối tượng trong ảnh là bài toán có nhiều ứng dụng trong thực tế. Việc phát hiện và mô tả tốt, chính xác những đối tượng, chi tiết có trong ảnh sẽ tạo tiền đề tốt cho những hướng phát triển như:

- Giúp đỡ người khiêm thị có thể nhận biết được nội dung của các tấm ảnh.
- Tìm ra những chi tiết mà mắt thường con người có thể bỏ sót do: chi tiết quá nhỏ, nhìn lướt qua, nhìn thấy nhưng không nhận biết được đối tượng là gì...
- Xây dựng được mô hình có thể chuyển sự phức tạp của một tấm hình về một câu ở dạng ngôn ngữ tự nhiên, câu mô tả được sinh ra một cách thoáng nghĩa hơn.

Những khó khăn phải đổi mới trong đề tài là:

- Các ảnh thường không đạt chuẩn, bị mờ hoặc nhòe.
- Nội dung của đoạn mô tả bị sai hoặc thiếu so với nội dung thực của tấm ảnh.
- Lượng từ vựng của máy tính không phong phú như của con người.



Hình 1.1: Ảnh dữ liệu bị mờ



Nội dung sai: A men is hugging a cat.
Nội dung đúng: A men is hugging a dog.

Hình 1.2: Nội dung đoạn mô tả bị sai



Nội dung chưa đúng: Tablet, Food, Bottle...
Nội dung đúng: A dinning table with breakfast items.

Hình 1.3: Lượng từ vựng không phong phú

1.3 Phạm vi đề tài

1.3.1 Giới thiệu bài toán nhận diện và mô tả nội dung hình ảnh

Với một số hình ảnh đầu vào, yêu cầu cho ra các câu – mô tả lại chính xác nội dung của những bức ảnh đó.

1.3.2 Cơ sở dữ liệu sử dụng trong bài toán

Để kiểm tra, đánh giá và tăng độ chính xác với kết quả bài toán, luận văn sử dụng 3 bộ cơ sở dữ liệu(database) lấy từ mạng Internet.

- Flickr8k [1], Flickr30k [2]: chứa đựng hơn 38 nghìn tấm hình thu thập được từ trang web Flickr.
- MSCOCO [3]: Tập dữ liệu nhận dạng hình ảnh, phân đoạn, và mô tả.

	Flickr8k, Flickr30k	MSCOCO
Số ảnh đánh giá	1000	40504
Số ảnh train	37875	82783
Số ảnh test	1000	40775
Tổng ảnh	39875	164062

Bảng 1.1: Bảng thông tin về các bộ cơ sở dữ liệu sử dụng trong bài toán



Hình 1.4: Mô tả một tấm ảnh trong Dataset

1.4 Bố cục

Nội dung của luận văn này được trình bày trong 7 chương, bao gồm các nội dung về các phương pháp và vấn đề về nhận diện, mô tả đối tượng. Cùng với việc áp dụng các phương pháp này vào vấn đề nhận diện, mô tả nội dung ảnh.

Chương 1 – Tổng quan về đề tài: Giới thiệu tổng quan về đề tài, bài toán, mục tiêu và phạm vi đề tài.

Chương 2 – Công trình liên quan: Giới thiệu về bài báo, công trình nghiên cứu liên quan đến đề tài đang thực hiện.

Chương 3 – Cơ sở lý thuyết: Tìm hiểu về các kỹ thuật Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM).

Chương 4 – Ý tưởng giải quyết bài toán: Mô tả ý tưởng, các bước áp dụng cơ sở lý thuyết để giải quyết bài toán.

Chương 5 – Kết quả thực nghiệm: Kết quả đạt được khi giải quyết xong bài toán.

Chương 6 – Ứng dụng thực tế: Giới thiệu ứng dụng thực tế sử dụng các thuật toán trên để chú thích ảnh và mô tả chúng bằng tiếng Anh hoặc tiếng Việt.

Chương 7 – Kết luận và hướng phát triển: Rút ra các kết luận và xác định hướng phát triển tiếp theo cho bài toán.

Chương 2

CÁC CÔNG TRÌNH LIÊN QUAN

2.1 Rich feature hierarchies for accurate object detection and semantic segmentation [5]

Luận văn tham khảo bài báo khoa học của Ross Girshick, Jeff Donahue, Trevor Darrell và Jitendra Malik công bố vào năm 2013, cơ sở dữ liệu(database) sử dụng là PASCAL VOC.

Mục tiêu bài báo là sử dụng các kỹ thuật để nhận diện ra các đối tượng trong ảnh.

2.1.1 Các kỹ thuật được sử dụng

Các kỹ thuật được sử dụng trong quá trình nhận diện là Regions with Convolutional Neural Networks (RCNN).

2.1.2 Các kết quả chính

VOC 2010 Test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	mAP
DPM v5 [14]	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	33.4
UVA [15]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	35.1
Regionlets [16]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	39.7
SegDPM [17]	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	50.2
R-CNN BB	71.8	65.8	53.0	36.8	35.9	59.7	60.0	69.9	27.9	50.6	41.4	70.0	62.0	69.0	58.1	29.5	59.4	39.3	53.7

Bảng 2.1: Tỉ lệ chính xác khi dự đoán

2.2 Deep Fragment Embeddings for Bidirectional Image Sentence Mapping [6]

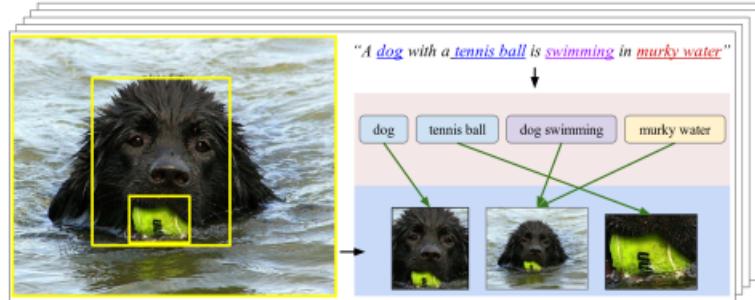
Luận văn tham khảo bài báo khoa học của Andrej Karpathy, Armand Joulin và Li Fei-Fei công bố vào năm 2014, cơ sở dữ liệu(database) sử dụng là Pascal1K, Flickr8k, và Flickr30k.

Mục tiêu bài báo là sử dụng các kỹ thuật để nhận diện ra các đối tượng trong ảnh, từ đó sử dụng tiếp các kỹ thuật để tạo ra các câu mô tả đối tượng đó.

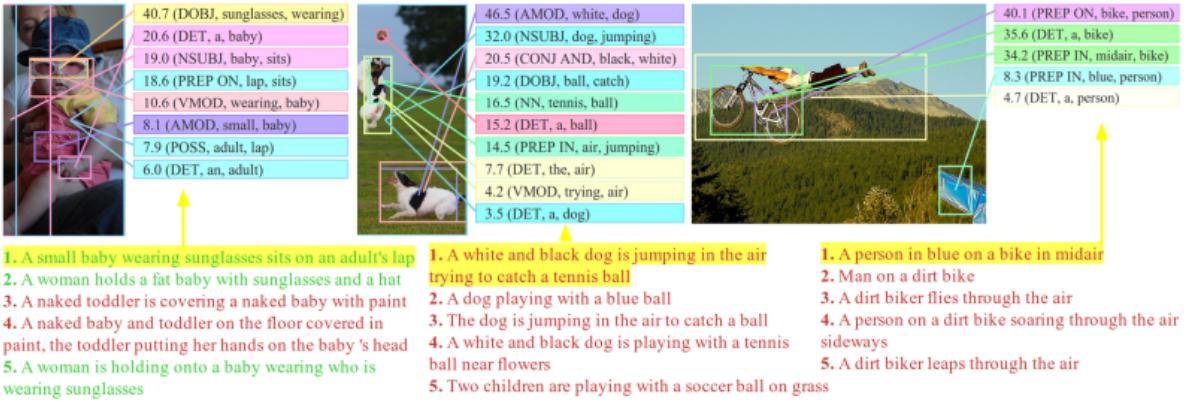
2.2.1 Các kỹ thuật được sử dụng

Các kỹ thuật được sử dụng trong quá trình nhận diện là Convolutional Neural Network (CNN). Dependency Tree Relation (DT), Bidirect Recurrent Neural Network (BRNN).

2.2.2 Các kết quả chính



Hình 2.1: Nhận diện đối tượng và ghép thành câu hoàn chỉnh



Hình 2.2: Thứ tự các câu có nghĩa giống nhất

2.3 Deep Visual-Semantic Alignments for Generating Image Descriptions [4]

Luận văn tham khảo bài báo khoa học của Andrej Karpathy và Li Fei-Fei công bố vào năm 2015, được phát triển dựa trên các kiến thức từ 2 bài báo [5] và [6] ở trên, cơ sở dữ liệu(database) sử dụng là Flickr8k, Flickr30k và MSCOCO.

Mục tiêu bài báo là sử dụng các kỹ thuật để nhận diện ra các đối tượng trong ảnh, từ đó sử dụng tiếp các kỹ thuật để tạo ra các câu mô tả đối tượng đó.

2.3.1 Các kỹ thuật được sử dụng

Các kỹ thuật được sử dụng trong quá trình nhận diện là Regions with Convolutional Neural Networks (RCNN), tính điểm số tương ứng giữa vùng ảnh và từ sử dụng Bidirectional Recurrent Neural Network (BRNN), tạo ra mô tả là Long Short-Term Memory (LSTM).

2.3.2 Hướng tiếp cận

Bài báo có 2 hướng tiếp cận chính:

- Tạo chú thích cho từng vùng ảnh.
- Tạo chú thích cho toàn bộ tấm ảnh.

2.3.2.1 * Tạo chú thích cho từng vùng ảnh

Sử dụng RCNN + BRNN + LSTM:

- RCNN: rút trích đặc trưng trong n vùng của tấm ảnh.
- BRNN: tính toán điểm số tương ứng nhằm để cân chỉnh giữ vùng ảnh và từ.
- LSTM: sau khi đã có điểm số tương ứng, tiến hành học để ghi nhớ mối liên kết giữa vùng ảnh và từ.



Hình 2.3: Kết quả hướng tiếp cận tạo chú thích cho từng vùng ảnh

2.3.2.2 * Tạo chú thích cho toàn bộ tấm ảnh

Sử dụng CNN + LSTM:

- CNN: rút trích đặc trưng của tấm ảnh.
- LSTM: học và tạo câu mô tả cho đối tượng trong hình.



Hình 2.4: Kết quả hướng tiếp cận tạo chú thích cho toàn bộ tấm ảnh

2.3.2.3 Kết luận

Luận văn chọn hướng tiếp cận "Tạo chú thích cho toàn bộ tấm ảnh" sử dụng các kỹ thuật CNN kết hợp với LSTM vì CNN đã được chứng minh có khả năng thực hiện tốt các tác vụ xử lý ảnh, nhận diện tốt đối tượng, và LSTM có thể xử lý tốt các tác vụ về ngôn ngữ tự nhiên nhanh hơn so với các kỹ thuật khác.

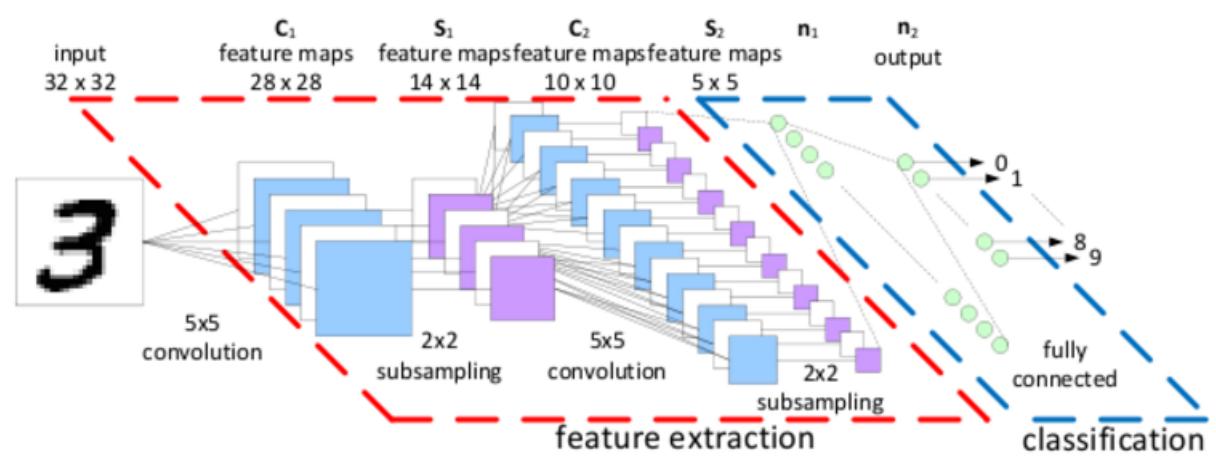
Đồng thời, hướng tiếp cận "Tạo chú thích cho toàn bộ tấm ảnh" cũng phù hợp mới mục tiêu của luận văn. Chi tiết của hướng tiếp cận và các kỹ thuật sẽ được trình bày ở chương sau.

Chương 3

CƠ SỞ LÝ THUYẾT

3.1 Convolutional Neural Networks (CNN)

3.1.1 Giới thiệu



Hình 3.1: Mô hình mạng tích chập

Mạng Neural tích chập bao gồm một hoặc nhiều tầng tích chập kết hợp với hàm phi tuyến tính và sau bởi một hoặc nhiều tầng mạng Neural nhân tạo, để tạo ra thông tin trừu tượng hơn cho các tầng tiếp theo.

Kiến trúc CNN tận dụng lợi thế của cấu trúc 2D cho hình ảnh đầu vào (hay kí hiệu lời nói) với các kết nối cục bộ và các trọng số theo sau bởi một số hình thức kết hợp từ việc chuyển đổi những đặc trưng bất biến. Thêm vào đó, CNN rất dễ dàng huấn luyện và có ít tham số hơn so với các mạng kết nối đầy đủ với cùng số lượng đơn vị lẩn.

Trong mô hình mạng Neural truyền thống, các tầng kết nối trực tiếp với nhau thông qua trọng số W . Nhưng mô hình CNN thì ngược lại các tầng liên kết với nhau thông qua cơ chế tích chập. Tầng tiếp theo là kết quả tích chập từ tầng trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Nghĩa là mỗi neural ở tầng tiếp theo sinh ra từ việc áp đặt bộ lọc lên một vùng ảnh cục bộ của neural ở tầng trước đó.

Mỗi tầng như vậy được áp đặt các bộ lọc khác nhau, thông thường có vài trăm đến vài nghìn bộ lọc như vậy. Một số tầng khác như tầng pooling/subsampling dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong suốt quá trình huấn luyện, CNN sẽ tự động học được các thông số cho các bộ lọc. Ví dụ trong tác vụ phân lớp ảnh, CNN sẽ cố gắng tìm ra thông số tối ưu cho các bộ lọc tương ứng theo thứ tự *raw pixel > edges > shapes > facial > high-level features*. Tầng cuối cùng được dùng để phân lớp ảnh.

CNN có tính bất biến và tính kết hợp cục bộ. Với cùng một đối tượng, nếu đổi tượng này được chiếu theo các góc độ khác nhau thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể. Tầng pooling cung cấp tính bất biến đối với phép dịch chuyển (*translation*), phép quay (*rotation*) và phép co giãn (*scaling*). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua quá trình tích chập từ các bộ lọc.

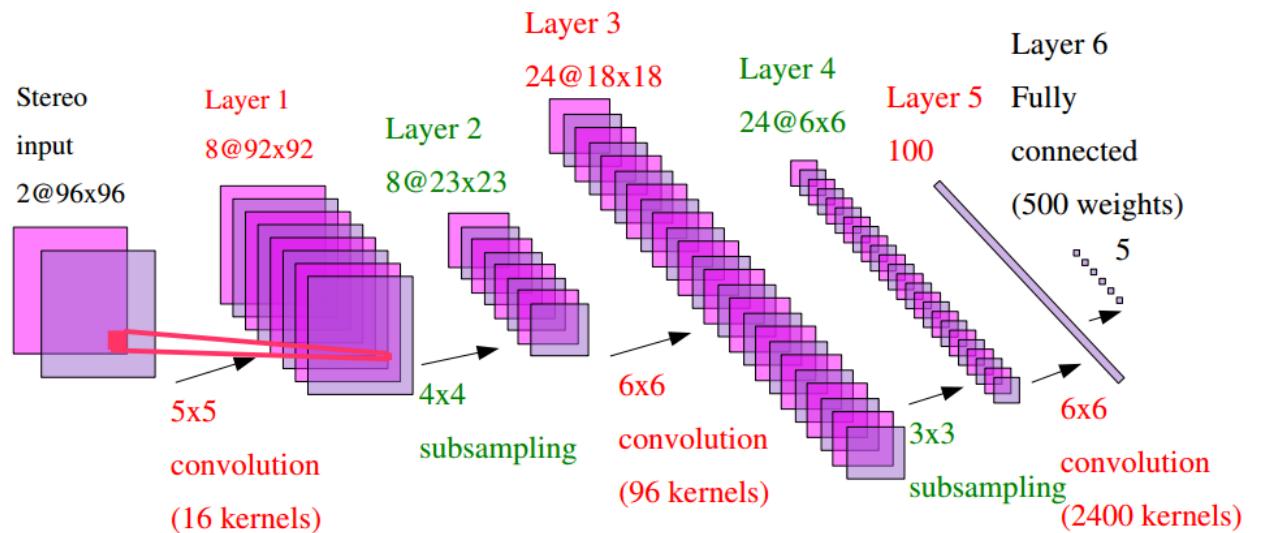
3.1.2 Cấu trúc Convolutional Neural Network

CNN bao gồm một số tầng tích chập và tầng mẫu theo sau bởi các tầng kết nối đầy đủ. Không giống như mạng Neural thông thường đầu vào của CNN là một hình ảnh có kích thước $m \times m \times r$, trong đó m vừa là chiều cao vừa là chiều rộng của hình ảnh, r là số kênh (ví dụ: một ảnh có 3 màu đỏ(red), xanh lục(green), xanh dương(blue) sẽ có $r = 3$).

Một tầng tích chập sẽ có k bộ lọc (hoặc là hạt nhân) có kích thước $m \times n \times q$ trong đó n nhỏ hơn kích thước của hình ảnh, q là số lượng kênh có thể bằng hoặc nhỏ hơn r và có thể biến đổi mỗi bộ lọc. Kích thước bộ lọc sẽ tăng khi cấu trúc kết nối cục bộ được tích chập với hình ảnh lược đồ k đặc trưng có kích thước

$m-n+1$. Mỗi lược đồ sẽ có mẫu con điển hình với sự hợp nhất trung bình hoặc tối đa thông qua miền cạnh nhau $p \times p$ với p là khoảng cách thước giữa hai hình ảnh nhỏ, thường không có nhiều hơn 5 đầu vào lớn hơn. Lược đồ bên dưới mô phỏng CNN có đầy đủ các lớp bao gồm lớp tích chập và lớp mẫu. Các đơn vị cùng màu sẽ được gắn vào một trọng số.

Thông thường kiến trúc của mạng Neural tích chập không quá 5 lớp đầu vào. Mỗi mạng Neural tích chập bao gồm tầng con tích chập và tầng mẫu.



Hình 3.2: Mô hình mạng tích chập

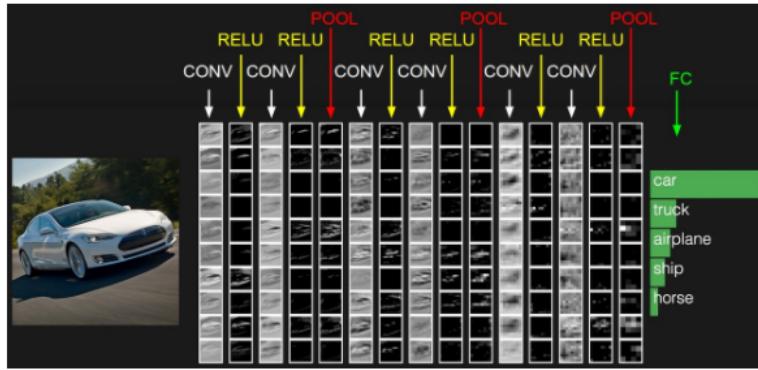
Bộ lọc được đẩy theo chiều ngang dựa vào chiều rộng và chiều cao của đầu vào sẽ tạo ra lược đồ gồm các bộ lọc có kích thước 2D. Đối với đầu vào có kích thước lớn như một hình ảnh mỗi neural trong bộ lọc sẽ kết nối với một vùng trong đầu vào. Mạng neural tích chập bao gồm nhiều 3 tầng chính: tầng tích chập (Convolution), tầng tổng hợp (Pooling), và tầng được kết nối hoàn toàn (Fully-Connected). Xếp chồng các tầng này để tạo thành kiến trúc ConvNet đầy đủ. Các tầng này có kiến trúc [INPUT - CONV - RELU - POOL - FC]. Chi tiết hơn:

- Tầng đầu vào(Input): sẽ giữ giá trị điểm ảnh thô của hình ảnh, ví dụ [32x32x3] là hình ảnh có chiều rộng 32, chiều cao 32 và với 3 kênh màu đỏ(Red), xanh lá(Green), xanh dương(Blue).
- Tầng tích chập (*Convolutional*): lớp tích chập bao gồm các neural có mạng lưới hình chữ nhật và nó đòi hỏi lớp trước đó cũng phải là các neural có mạng

lưới hình chữ nhật. Mỗi neural nhận đầu vào từ một thành phần của lớp trước đó, trọng số cho mỗi thành phần hình chữ nhật này là như nhau. Vì thế, lớp tích chập là một chập hình ảnh của lớp trước, trong đó trọng số xác định các bộ lọc chập. Ngoài ra có thể có nhiều lưới ở mỗi lớp tích chập, mỗi lưới nhận đầu vào từ các lưới ở lớp trước và sử dụng các bộ lọc có khả năng khác nhau. Tầng này sẽ tính toán đầu ra của các neural được kết nối với các vùng cục bộ của đầu vào. Điều này có thể dẫn đến độ lớn như [32x32x12] nếu quyết định sử dụng 12 bộ lọc.

- Tầng Relu: sẽ áp dụng một hàm kích hoạt, như max (0, x). Điều này làm cho kích thước của khối không thay đổi ([32x32x12]).
- Tầng Max-Pooling: Sau mỗi lớp tích chập, sẽ có một lớp tổng hợp. Các lớp tổng hợp là các khối hình chữ nhật nhỏ từ lớp tích chập và lớp mẫu để sản xuất một đầu ra duy nhất từ khối đó. Có rất nhiều cách để tổng hợp, chẳng hạn như lấy trung bình hoặc cực đại, hoặc một sự kết hợp tuyến tính đòi hỏi việc học của neural trong khối. Lớp tổng hợp sẽ luôn luôn là lớp max-pooling. Thực hiện hoạt động giảm không gian (chiều rộng, chiều cao), kết quả là độ lớn như [16x16x12].
- Tầng Full-Connected: Cuối cùng, sau một vài lớp tích chập và max-pooling, những lý luận cấp cao trong các mạng Neural được thực hiện thông qua các lớp kết nối đầy đủ. Một lớp kết nối đầy đủ nắm giữ tất cả các tế bào neural trong các lớp trước đó (có thể là đầy đủ kết nối, tổng hợp, hoặc tích chập) và kết nối nó tới mọi neural duy nhất nó có. Các lớp có đầy đủ kết nối không nằm trong không gian, do đó có thể không có các lớp tích chập sau khi một lớp có đầy đủ kết nối.

Bằng cách này, ConvNets biến đổi tầng hình ảnh ban đầu theo tầng từ giá trị điểm ảnh ban đầu đến điểm số của tầng cuối cùng. Lưu ý rằng một số tầng có chứa các tham số và khác 0. Đặc biệt, các tầng Conv / FC thực hiện các phép biến đổi không chỉ là các kích hoạt trong phần đầu vào, mà còn về các tham số (trọng số và biases của các neural). Mặt khác, các tầng Relu / Pool sẽ thực hiện một chức năng cố định. Các thông số trong các tầng Conv / FC sẽ được huấn luyện với gradient descent để điểm của tầng ConvNet tính, phù hợp với các nhãn trong tập huấn luyện cho mỗi hình ảnh.



Hình 3.3: Ví dụ kiến trúc ConvNet

Ví dụ kiến trúc ConvNet: Tầng ban đầu lưu trữ các điểm ảnh thô (trái) và tầng cuối cùng lưu trữ điểm học (bên phải). Mỗi xử lý của tầng kích hoạt được hiển thị dưới dạng một cột. Tầng cuối cùng giữ điểm số của mỗi tầng, nhưng ở đây chỉ hình dung được điểm xếp hạng trong top 5, và in các nhãn của mỗi tầng. Kiến trúc được trình bày ở đây là VGG Net.

3.1.2.1 Tầng tích chập(Convolutional)

Tầng tích chập(Conv) là khối xây dựng một mạng lưới chập xử lý hầu hết bước tính toán cốt lõi.

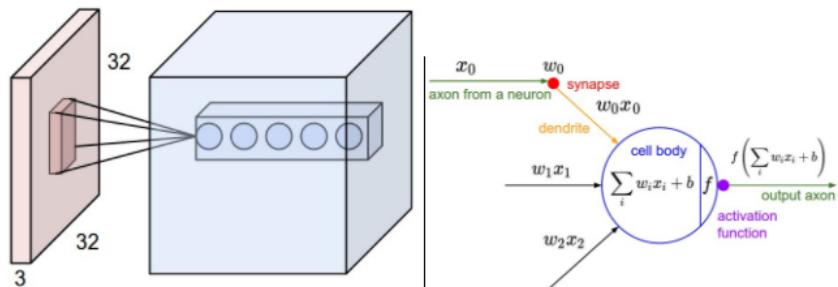
Tổng quan: Đầu tiên nói về những gì các tầng tích chập(Conv) tính mà không cần phân tích neural. Tham số tầng tích chập(Conv) bao gồm một tập hợp các bộ lọc có thể được học. Mỗi bộ lọc trong không gian nhỏ (dọc theo chiều rộng và chiều cao), nhưng mở rộng thông qua chiều sâu của đầu vào. Ví dụ, một bộ lọc thông thường trên một tầng đầu tiên của một ConvNet có thể có kích thước $5 \times 5 \times 3$ (tức là 5 điểm ảnh chiều rộng và chiều cao, và 3 vì những hình ảnh có độ sâu 3 kênh màu). Trong quá trình lan truyền thẳng, sẽ chập mỗi bộ lọc trên chiều rộng và chiều cao của khối đầu vào và tính tích vô hướng giữa các mục của các bộ lọc và các đầu vào tại vị trí bất kỳ. Khi chập bộ lọc trên chiều rộng và chiều cao của khối đầu vào, sẽ cho ra bản đồ kích hoạt 2 chiều cung cấp cho các phản ứng của bộ lọc đó ở mọi vị trí không gian. Mạng sẽ học kích hoạt bộ lọc khi nhìn thấy một số loại đặc trưng trực quan như: một cạnh hướng hoặc một đốm màu sắc trên tầng đầu tiên. Nay giờ, sẽ có một tập toàn bộ các bộ lọc trong mỗi tầng tích chập(Conv) (ví dụ: 12 bộ lọc), và mỗi bộ sẽ tạo ra một bản đồ kích hoạt 2 chiều

riêng biệt. Sắp xếp các bản đồ kích hoạt dọc theo chiều sâu và tạo các đầu ra.

Kết nối cục bộ: Khi xử lý với đầu vào có số chiều lớn như hình ảnh, đã thấy ở trên là không thực tế để kết nối các neural với tất cả các neural tập trước đó. Thay vào đó, sẽ kết nối mỗi neural với một vùng cục bộ của khối đầu vào. Trong phạm vi không gian của kết nối này là một siêu tham số được gọi là trường tiếp nhận của neural (tương đương đây là kích thước bộ lọc). Trong phạm vi của việc kết nối dọc theo trực chiều sâu là luôn luôn bằng với độ sâu của khối đầu vào. Điều quan trọng để nhấn mạnh một lần nữa sự bất đối xứng này trong cách thức xử lý chiều không gian (chiều rộng và chiều cao) và kích thước chiều sâu: các kết nối là cục bộ trong không gian (dọc theo chiều rộng và chiều cao), nhưng luôn luôn đầy đủ dọc theo toàn bộ chiều sâu của khối đầu vào.

Ví dụ 1. Ví dụ, giả sử rằng khối đầu vào có kích thước $[32 \times 32 \times 3]$, (ví dụ một hình ảnh RGB CIFAR-10). Nếu trường tiếp nhận (hoặc kích thước bộ lọc) là 5×5 , sau đó mỗi neural ở tầng tích chập(Conv) sẽ có trọng số cho $[5 \times 5 \times 3]$ vùng khôi đầu vào, tổng cộng là $5 * 5 * 3 = 75$ trọng số (và +1 tham số bias). Chú ý rằng mức độ của các kết nối dọc theo trực sâu phải là 3, vì đây là độ sâu của khôi đầu vào.

Ví dụ 2. Giả sử khôi đầu vào có kích thước $[16 \times 16 \times 20]$. Sau đó, sử dụng kích thước trường tiếp nhận là 3×3 , mỗi neural trong tầng tích chập(Conv) sẽ có tổng cộng $3 * 3 * 20 = 180$ kết nối với khôi đầu vào. Lưu ý rằng, kết nối là cục bộ trong không gian (ví dụ 3×3), nhưng đầy đủ theo chiều sâu đầu vào (20).



Hình 3.4: Hình ảnh CIFAR-10 $32 \times 32 \times 3$

Ảnh trái: Ví dụ về khôi đầu vào màu đỏ (ví dụ: hình ảnh CIFAR-10 $32 \times 32 \times 3$) và một khôi ví dụ của các neural trong tầng tích chập(Convolutonal) đầu tiên. Mỗi neural trong tầng chập chỉ được kết nối với một vùng cục bộ trong khôi không

gian đầu vào, nhưng ở độ sâu đầu đủ (tức là tất cả các kênh màu). Lưu ý, có nhiều neural (5 trong ví dụ này) đọc theo chiều sâu, tất cả đều nhìn vào cùng một khu vực của đầu vào.

Ảnh phái: Các neural từ mạng Neural vẫn không thay đổi: vẫn tính một tích vô hướng bằng trọng số của chúng với đầu vào bằng một hàm phi tuyến tính, nhưng kết nối của chúng bây giờ được giới hạn trong không gian cục bộ.

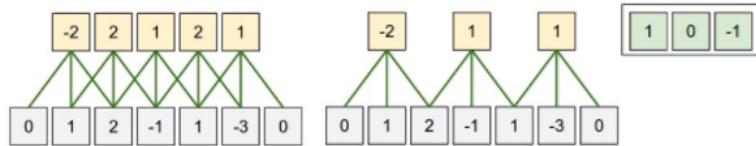
Bố trí không gian: Đã giải thích kết nối của mỗi neural trong tầng tích chập(Conv) với khối đầu vào, nhưng vẫn chưa đề cập bao nhiêu neural có trong khối đầu ra hoặc cách sắp xếp chúng. Ba siêu tham số chiều sâu, bước nhảy và 0-padding kiểm soát kích thước của khối đầu ra:

+ Thứ nhất, độ sâu của khối đầu ra là một siêu tham số: nó tương ứng với số lượng bộ lọc mà chúng ta muốn sử dụng, mỗi lần học để tìm một cái gì đó khác nhau trong đầu vào. Ví dụ, nếu tầng tích chập(conv) đầu tiên lấy hình ảnh thô làm đầu vào, thì các neural khác nhau đọc theo chiều sâu có thể kích hoạt với sự hiện diện của nhiều cạnh định hướng, hoặc các đốm màu. Đề cập đến một tập hợp các neural mà tất cả đều đang nhìn vào cùng một vùng của đầu vào như một cột độ sâu.

+ Thứ hai, phải xác định bước nhảy khi "trượt bộ lọc". Khi bước nhảy là 1 thì di chuyển các bộ lọc một điểm ảnh mỗi lần. Khi bước nhảy là 2 (hoặc phỏ biến 3 hoặc nhiều hơn, mặc dù điều này là rất hiếm trong thực tế) sau đó các bộ lọc nhảy 2 điểm ảnh tại một thời điểm khi trượt chúng xung quanh. Điều này sẽ tạo ra khối không gian nhỏ hơn.

+ Dôi khi sẽ thuận tiện để thêm khối đầu vào với số 0 xung quanh biên. Kích cỡ của 0-padding này là một siêu tham số. Các tính năng khá hay của 0-padding là nó sẽ cho phép kiểm soát kích thước không gian của khối đầu ra (phỏ biến nhất sẽ sử dụng nó để giữ chính xác kích thước không gian của khối đầu vào để chiều rộng, chiều cao đầu vào đầu ra đều giống nhau).

Có thể tính toán kích thước không gian của khối đầu ra như là một hàm của kích thước khối đầu vào (\mathbf{W}), kích thước trường tiếp nhận của neural tầng tích chập(Conv) (\mathbf{F}), bước nhảy mà chúng được áp dụng (\mathbf{S}) và lượng 0-padding được sử dụng (\mathbf{P}) trên đường biên. Có thể tự thuyết phục rằng công thức chính xác để tính số neural "phù hợp" được cho bởi $(\mathbf{W} \cdot \mathbf{F} + 2\mathbf{P}) / \mathbf{S} + 1$. Ví dụ cho một đầu vào 7×7 và một bộ lọc 3×3 với bước nhảy(stride) là 1 và padding 0, sẽ nhận được một đầu ra 5×5 . Với bước nhảy là 2 sẽ có được đầu ra 3×3 . Ví dụ khác:



Hình 3.5: Minh họa về bố trí không gian

Minh họa về bố trí không gian: chỉ có một chiều không gian (trục x), một neural với kích thước trường $F = 3$, kích thước đầu vào là $W = 5$, và 0-padding $P = 1$.

Bên trái: neural nhảy 1 bước $S = 1$, cho đầu ra có kích thước $(5 - 3 + 2) / 1 + 1 = 5$.

Bên phải: neural sử dụng bước nhảy của $S = 2$, tạo ra kích cỡ $(5 - 3 + 2) / 2 + 1 = 3$.

Chú ý rằng không thể sử dụng $S = 3$ vì nó sẽ không vừa với khối. Về phương trình, điều này có thể được xác định vì $(5 - 3 + 2) = 4$ không chia hết cho 3.

Trọng số của neural là trong ví dụ này $[1,0, -1]$ (thể hiện ở bên phải), và sai lệch của nó bằng 0. Những trọng số này được chia sẻ trên tất cả các neural màu vàng.

+ Sử dụng 0-padding. Trong ví dụ trên ở bên trái, lưu ý rằng số chiều đầu vào và số chiều đầu ra bằng nhau: bằng 5. Vì trường tiếp nhận là 3 và sử dụng 0-padding là 1. Nếu 0-padding không được sử dụng, thì khói đầu ra sẽ có kích thước không gian chỉ bằng 3, bởi vì nghĩa là có bao nhiêu neural sẽ "phù hợp" với đầu vào ban đầu. Nói chung, thiết lập 0-padding là $P = (F-1) / 2$ khi bước nhảy là $S = 1$ đảm bảo rằng khói đầu vào và khói đầu ra sẽ có cùng kích thước không gian.

+ Hạn chế về những bước nhảy: Lưu ý một lần nữa rằng các siêu tham số phân bố không gian có những ràng buộc lẫn nhau. Ví dụ, khi đầu vào có kích thước $\mathbf{W} = 10$, 0-padding được sử dụng $P = 0$, và kích thước bộ lọc là $F = 3$ thì không thể sử dụng bước nhảy $S = 2$, Vì $(W-F + 2P) / S + 1 = (10-3 + 0) / 2+1 = 4.5$, nghĩa là không phải là số nguyên, chỉ ra rằng các neural không "phù hợp" một cách cân đối và đối xứng trên đầu vào. Do đó, thiết lập này của siêu tham số được coi là không hợp lệ, và một thư viện ConvNet có thể ném một ngoại lệ hoặc đánh dấu 0 phần còn lại để làm cho nó phù hợp, hoặc cắt đầu vào để làm cho nó phù hợp,... Trong phần kiến trúc ConvNet, hãy chọn kích thước ConvNets một cách hợp lý để tất cả các kích thước phù hợp với nhau.

+ Ví dụ thực tế: Krizhevsky - Kiến trúc đã giành chiến thắng ImageNet vào năm

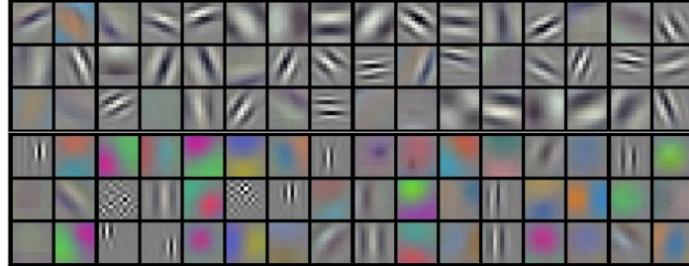
2012 đã chấp nhận hình ảnh có kích thước [227x227x3]. Trên tầng Convolutional đầu tiên, nó sử dụng neural với kích thước trường nhận $F = 11$, bước nhảy $S = 4$ và không có 0-padding $P = 0$. Vì $(227 - 11) / 4 + 1 = 55$, và vì tầng Conv có độ sâu $K = 96$, khói đầu ra Conv có kích thước [55x55x96]. Mỗi $55 * 55 * 96$ neural trong khói này được kết nối với một vùng có kích thước [11x11x3] trong khói đầu vào. Hơn nữa, tất cả 96 neural trong mỗi cột sâu đều được nối với cùng một vùng [11x11x3] của đầu vào, nhưng tất nhiên với các trọng số khác nhau. Nếu đọc được bài báo thực tế tuyên bố rằng các hình ảnh đầu vào là 224x224, điều này chắc chắn là không chính xác vì $(224-11) / 4 + 1$ khá rõ ràng không phải là một số nguyên. Điều này đã gây nhầm lẫn cho nhiều người trong lịch sử của ConvNets và ít được biết về những gì đã xảy ra. Phán đoán tốt nhất là Alex đã sử dụng số 0-padding của 3 điểm ảnh thêm mà ông không đề cập đến trong bài báo.

Chia sẻ tham số: được sử dụng trong tầng Convolutional để kiểm soát số lượng các tham số. Sử dụng ví dụ thực tế ở trên, chúng ta thấy rằng có $55 * 55 * 96 = 290,400$ neural trong tầng đầu tiên của Conv, và mỗi có $11 * 11 * 3 = 363$ trọng số và 1 bias. Cùng với nhau, điều này cho thêm lên đến $290400 * 364 = 105,705,600$ các thông số vào các tầng đầu tiên của ConvNet. Rõ ràng, con số này là rất cao.

Điều này chỉ ra rằng có thể làm giảm đáng kể số lượng các tham số bằng cách làm một giả định hợp lý: rằng nếu một đặc trưng rất hữu ích để tính toán một số vị trí không gian (x, y), thì nó cũng nên hữu ích để tính toán ở một vị trí khác nhau (x_2, y_2). Nói cách khác, biểu thị một lát 2 chiều của độ sâu gọi là depth slice (ví dụ như là một khói kích thước [55 x 55 x 96] có 96 depth slice, mỗi cái có kích thước [55 x 55]), sẽ cố định các neural trong mỗi depth slice để sử dụng cùng một trọng số và bias. Với sự chia sẻ tham số này, tầng Conv đầu tiên trong ví dụ bây giờ sẽ có chỉ 96 bộ duy nhất của khói (một cho mỗi depth slice), cho một tổng số là $96 * 11 * 11 * 3 = 34,848$ trọng số duy nhất, hoặc 34,944 tham số (+96 biases). Ngoài ra, tất cả $55 * 55$ neural trong mỗi depth slice sẽ bây giờ sử dụng các tham số tương tự. Thực tế trong quá trình lan truyền ngược(backpropagation), mỗi neural trong tập sẽ tính toán gradient cho trọng số của nó, nhưng các gradient sẽ được thêm trên mỗi depth slice và chỉ cập nhật một tập trọng số duy nhất cho mỗi depth slice.

Nhận thấy rằng nếu tất cả các neural trong một depth slice duy nhất sử dụng cùng một vector trọng số, sau đó sự lan truyền thẳng của các tầng Conv khiến mỗi depth slice được tính như một lần chập của trọng số của các neural với khói đầu vào (tầng tích chập(Convolutional)). Đây là lý do phổ biến để chỉ bộ trọng số là

một bộ lọc(filter) hoặc một hạt nhân(kernel), mà được chập (convolve) cùng với các đầu vào.



Hình 3.6: Ví dụ bộ lọc được học bởi Krizhevsky

Ví dụ bộ lọc được học bởi Krizhevsky. Mỗi bộ lọc trong số 96 bộ lọc hiển thị ở đây có kích thước $[11 \times 11 \times 3]$, và mỗi bộ lọc được chia sẻ bởi các neural $55 * 55$ ở một depth slice. Nhận thấy rằng các chia sẻ tham số giả định là tương đối hợp lý: nếu phát hiện một cạnh ngang là quan trọng tại một số điểm trong hình ảnh, thì nó cũng sẽ có lý tại một số địa điểm khác do cấu trúc chuyển đổi bất biến của hình ảnh. Do đó là không cần phải học để phát hiện một cạnh ngang tại mỗi một địa điểm $55 * 55$ khác biệt trong khối đầu ra tầng Conv.

Lưu ý rằng đôi khi các chia sẻ tham số giả định có thể không có ý nghĩa. Đặc biệt là trường hợp khi hình ảnh đầu vào cho một ConvNet có một số cấu trúc trung tâm cụ thể, ví dụ, các đặc trưng toàn khác nhau nên được học ở các khu vực hình ảnh khác nhau. Một trong những ví dụ thực tế là khi đầu vào là khuôn mặt mà đã được đặt tại trung tâm trong ảnh. Có thể mong đợi rằng các đặc trưng khác nhau của mắt hoặc tóc có thể được học ở địa điểm khác nhau của hình. Trong trường hợp đó, nó là phổ biến để chia sẻ các tham số, và thay vào đó chỉ cần gọi cho các tầng Locally-Connected.

Ví dụ Tầng tích chập(Conv): Giả sử khối đầu vào X có hình dạng \mathbf{X} có kích thước $11 \times 11 \times 4$. Giả sử thêm rằng sử dụng 0-padding 0 ($\mathbf{P} = 0$), kích thước bộ lọc là $\mathbf{F} = 5$, và bước nhảy là $\mathbf{S} = 2$. Do đó, khối đầu ra sẽ có kích thước không gian $(11-5) / 2 + 1 = 4$, cho một khối với chiều rộng và chiều cao là 4.

Để tóm tắt, tầng tích chập:

+ Chấp nhận một khối lượng kích thước $\mathbf{W1} \times \mathbf{H1} \times \mathbf{D1}$

+ Yêu cầu bốn siêu tham số:

* Số bộ lọc \mathbf{K} ,

* Phạm vi không gian \mathbf{F} ,

* Các bước nhảy \mathbf{S} ,

* Số lượng 0-padding \mathbf{P} .

+ Sản xuất một khối kích thước $\mathbf{W2} \times \mathbf{H2} \times \mathbf{D2}$ trong đó:

* $\mathbf{W2} = (\mathbf{W1}-\mathbf{F} + 2\mathbf{P}) / \mathbf{S} + 1$

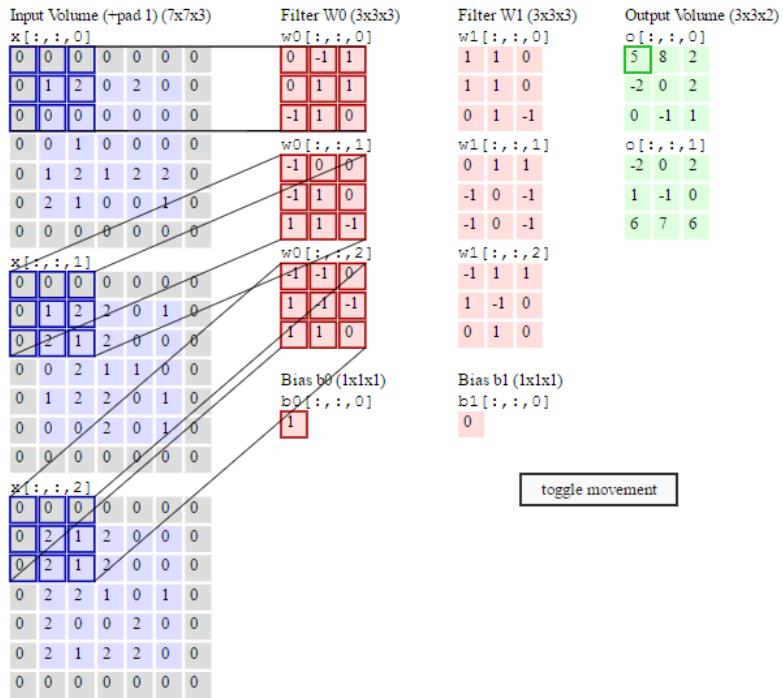
* $\mathbf{H2} = (\mathbf{H1}-\mathbf{F} + 2\mathbf{P}) / \mathbf{S} + 1$ (tức là chiều rộng và chiều cao được tính bằng nhau theo đối xứng)

* $\mathbf{D2} = \mathbf{K}$

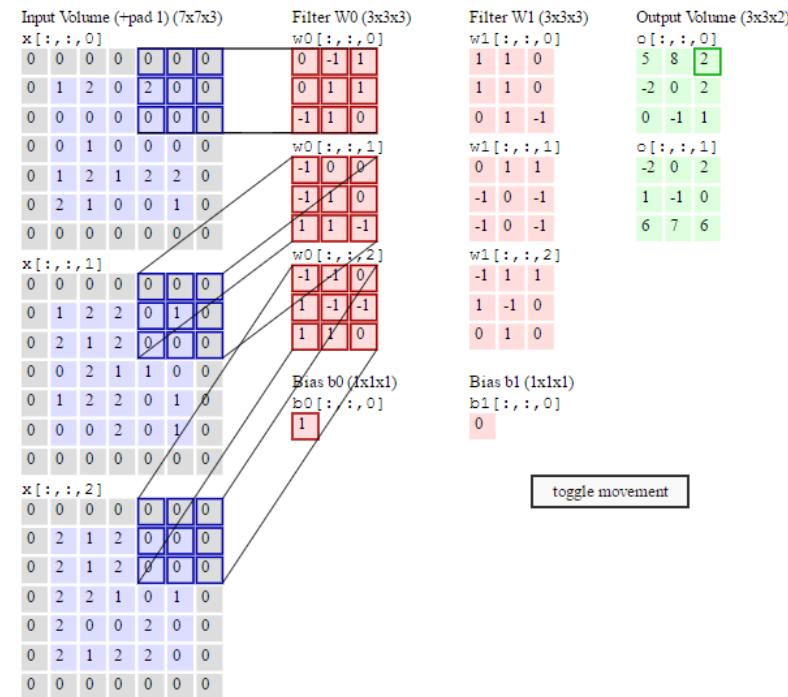
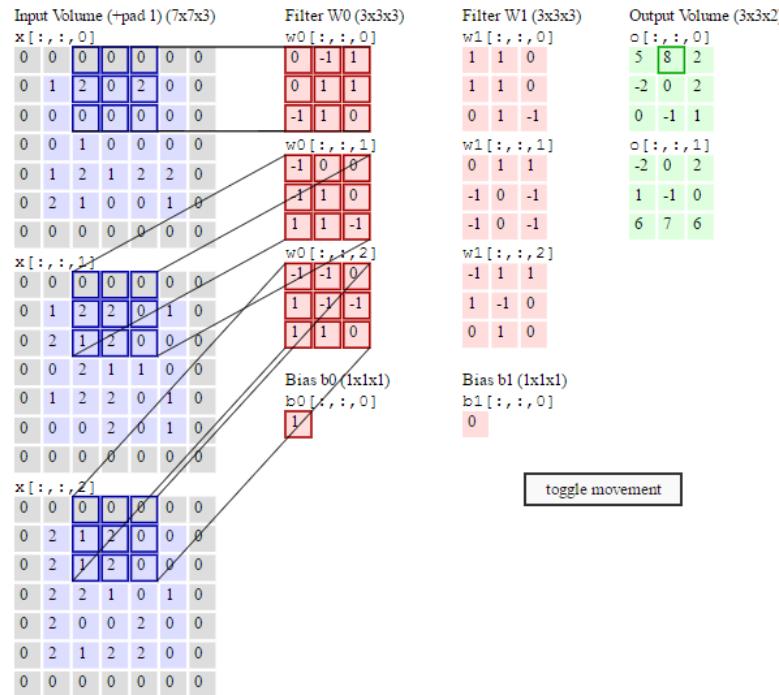
+ Trong khối đầu ra, depth slice thứ d (kích thước $\mathbf{W2} \times \mathbf{H2}$) là kết quả của việc chập của bộ lọc thứ d trên khối đầu vào với bước nhảy S, và sau đó bù đắp bằng bias thứ d.

Một thiết lập chung của siêu tham số là $\mathbf{F} = 3$, $\mathbf{S} = 1$, $\mathbf{P} = 1$. Tuy nhiên, có những quy ước chung và các quy tắc chung thúc đẩy các siêu tham số này.

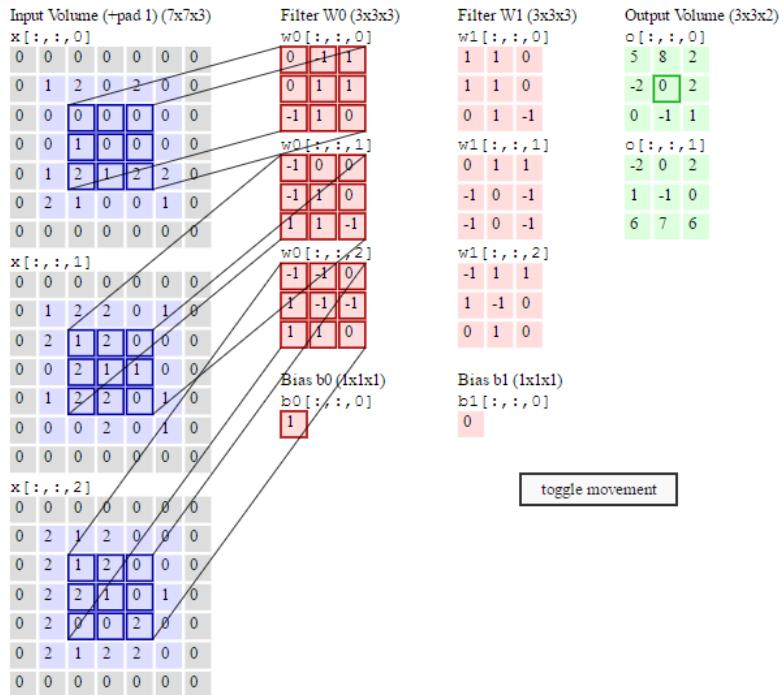
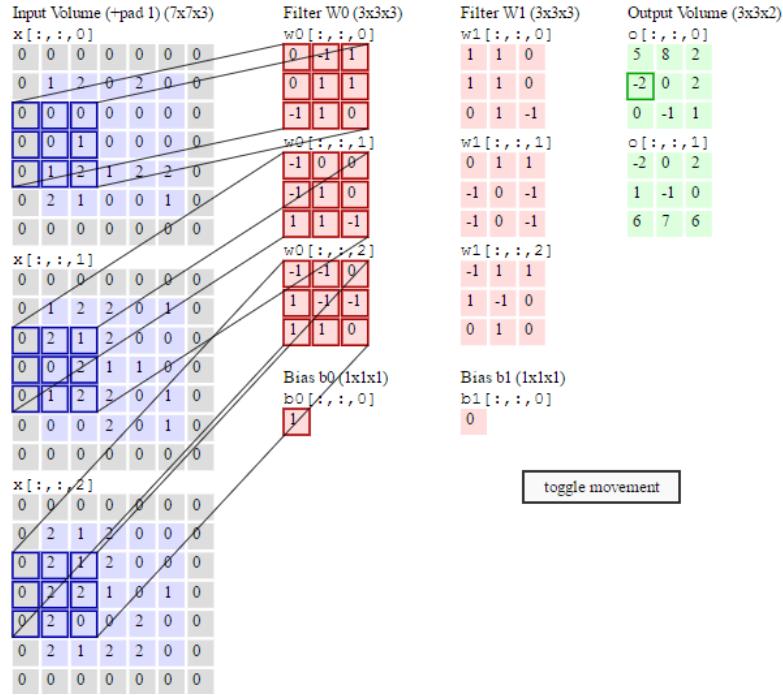
Demo tích chập: Dưới đây là bản demo của một tầng tích chập. Tất cả các khối (khối đầu vào (màu xanh), khối trọng số (màu đỏ), khối đầu ra (màu xanh lá cây) được hiển thị với mỗi lát cắt chiều xếp chồng lên nhau theo hàng. Khối đầu vào có kích thước $W_1 = 5$, $H_1 = 5$, $D_1 = 3$, và các tham số tầng Conv là $K = 2$, $F = 3$, $S = 2$, $P = 1$. Nghĩa là, có hai bộ lọc kích thước 3×3 , và chúng được áp dụng với bước nhảy 2. Vì vậy, khối đầu ra có kích thước không gian $(5 - 3 + 2) / 2 + 1 = 3$. Hơn nữa, chú ý rằng padding của $P = 1$ được áp dụng cho khối đầu vào, làm cho biên của khối đầu vào bằng 0. Hình minh họa dưới đây lặp đi lặp lại các kích hoạt đầu ra (màu xanh lá cây) và cho thấy mỗi phần tử được tính bằng cách nhân đầu vào được đánh dấu (màu xanh) với bộ lọc (màu đỏ), tổng hợp nó, và sau đó bù đắp kết quả bằng bias.



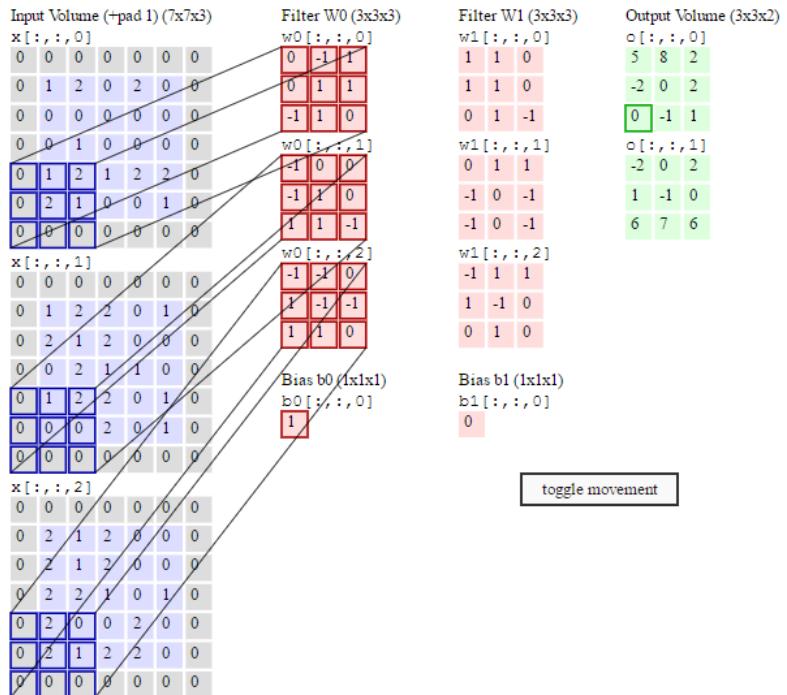
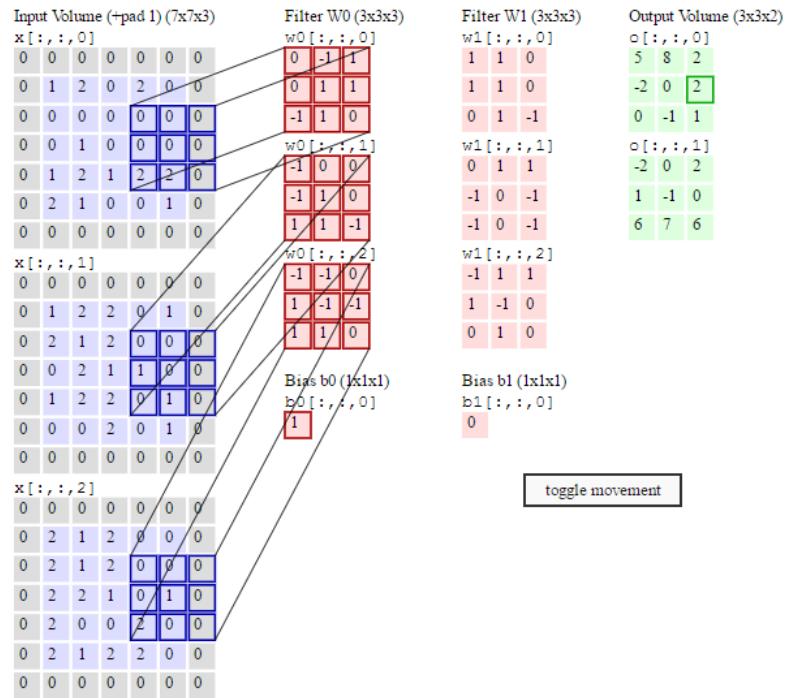
Hình 3.7: Các bước tính toán đầu ra



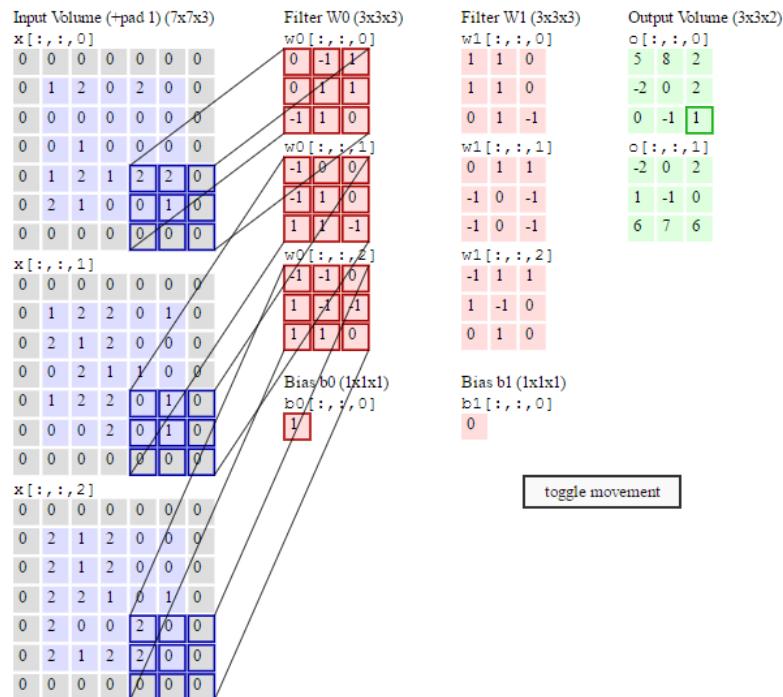
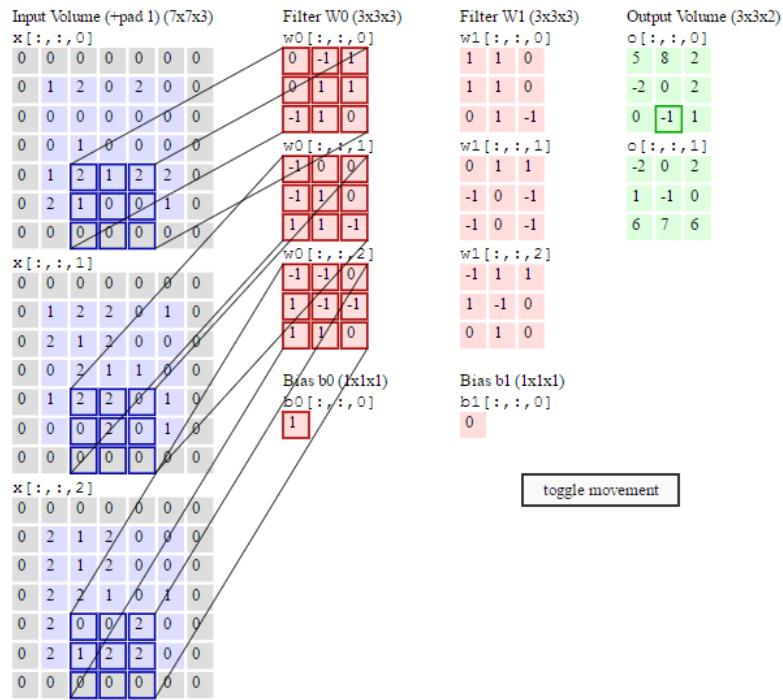
Hình 3.8: Trượt các khối từ trái qua phải, trên xuống dưới với bước nhảy là 2



Hình 3.9: Trượt các khối từ trái qua phải, trên xuống dưới với bước nhảy là 2



Hình 3.10: Trượt các khối từ trái qua phải, trên xuống dưới với bước nhảy là 2



Hình 3.11: Trượt các khối từ trái qua phải, trên xuống dưới với bước nhảy là 2

3.1.2.2 Tầng Pooling

Chức năng của nó là để giảm dần kích thước không gian khối để giảm số lượng các tham số và tính toán trong mạng Neural, và do đó cũng kiểm soát quá trình vừa khớp dữ liệu(overfitting). Tầng Pooling hoạt động độc lập trên mỗi depth slice của đầu vào và thay đổi kích thước không gian, sử dụng phương thức *Max*. Dạng phổ biến nhất là một tầng pooling với các bộ lọc kích thước 2×2 được áp dụng với một bước nhảy của 2 mẫu nhỏ mỗi depth slice của đầu vào của 2 dọc theo chiều rộng và chiều cao, loại bỏ 75% kích hoạt. Mỗi phương thức *Max* hoạt động trong trường hợp này được dùng tối đa trên 4 số (2×2 khu vực trong một số depth slice). Kích thước chiều sâu vẫn không thay đổi. Nói chung, tầng pooling:

+ Chấp nhận một khối lượng kích thước $\mathbf{W1} \times \mathbf{H1} \times \mathbf{D1}$

+ Yêu cầu hai siêu tham số:

* Phạm vi không gian \mathbf{F} của họ,

* Các bước nhảy \mathbf{S} .

+ Tạo một khối kích thước $\mathbf{W2} \times \mathbf{H2} \times \mathbf{D2}$ trong đó:

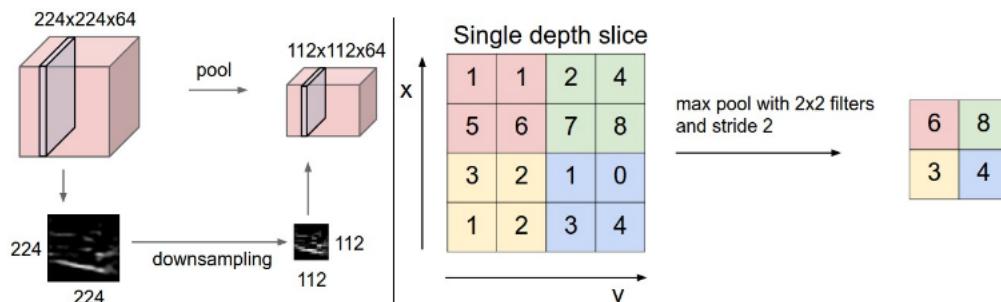
* $\mathbf{W2} = (\mathbf{W1}-\mathbf{F}) / \mathbf{S} + 1$

* $\mathbf{H2} = (\mathbf{H1}-\mathbf{F}) / \mathbf{S} + 1$

* $\mathbf{D2} = \mathbf{D1}$

+ Lưu ý rằng thường xuyên sử dụng 0-padding cho tầng pooling.

Cần lưu ý rằng chỉ có hai biến thể thường thấy của tầng tổng hợp được tìm thấy trong thực tế: Một lớp tổng hợp với $\mathbf{F} = 3$, $\mathbf{S} = 2$, và thường là $\mathbf{F} = 2$, $\mathbf{S} = 2$.



Hình 3.12: Minh họa tầng pooling

Tầng pooling giảm khói không gian, độc lập trong từng depth slice của khối đầu vào.

Trái: Trong ví dụ này, khối đầu vào có kích thước [224x224x64] được tổng hợp với kích thước bộ lọc 2, bước nhảy 2 vào khối lượng ra có kích thước [112x112x64]. Chú ý rằng độ sâu khối được giữ nguyên.

Phải: Hoạt động giảm khối không gian phổ biến nhất là làm tăng max pooling, ở đây hiển thị với bước nhảy là 2. Mỗi max được lấy qua hình vuông nhỏ 2x2).

3.1.2.3 Tầng Fully-connected

Neural trong một tầng kết nối hoàn chỉnh có kết nối đầy đủ đến tất cả các kích hoạt trong tầng trước đó, như được thấy trong mạng Neural thông thường. Các kích hoạt của chúng có thể xem là được tính bằng phép nhân ma trận theo sau là một bias.

3.1.2.4 Chuyển tầng FC sang tầng tích chập(Conv)

Chuyển tầng FC sang tầng tích chập(Conv)

Lưu ý rằng sự khác biệt duy nhất giữa các tầng FC và tích chập(Conv) là các neural trong tầng tích chập(Conv) chỉ được kết nối với một vùng cục bộ trong đầu vào và nhiều neural trong một tích chập(Conv) chia sẻ các tham số. Tuy nhiên, các neural trong cả hai tầng vẫn tính tích vô hướng, vì vậy chức năng của chúng giống hệt nhau. Do đó, chỉ ra rằng có thể chuyển đổi giữa các tầng FC và Conv:

+ Đối với bất kỳ tầng Conv có một tầng FC thực hiện cùng một hàm lan truyền. Ma trận trọng số sẽ là một ma trận lớn mà chủ yếu là số 0, ngoại trừ tại một số khói (do kết nối cục bộ), nơi trọng số trong nhiều khói là bằng nhau (do chia sẻ tham số).

+ Ngược lại, bất kỳ tầng FC nào cũng có thể được chuyển đổi thành một tầng tích chập(Conv). Ví dụ, một tầng FC với $K = 4096$ nhìn vào một số khói đầu vào có kích thước $7 \times 7 \times 512$ có thể được biểu diễn tương đương dưới dạng một tầng tích chập(Conv) với $F = 7, P = 0, S = 1, K = 4096$. Nói cách khác, đang thiết lập kích thước bộ lọc đúng kích cỡ của khói đầu vào, và do đó đầu ra chỉ đơn giản là $1 \times 1 \times 4096$ vì chỉ có một cột chiều sâu đơn "phù hợp" qua khối đầu vào, cho kết quả giống hệt như tầng FC ban đầu.

Chuyển đổi FC \Rightarrow Conv: Trong hai chuyển đổi này, khả năng chuyển đổi tầng FC sang tầng Conv đặc biệt hữu ích trong thực tế. Xem xét kiến trúc ConvNet có ảnh 224x224x3 và sử dụng hàng loạt tầng Conv và tầng POOL để giảm kích thước 7x7x512 (trong kiến trúc AlexNet, điều này được thực hiện bằng cách sử dụng 5

tập hợp các tầng mà giảm không gian đầu vào bởi một yếu tố hai lần mỗi lần, làm cho không gian cuối cùng có kích thước $224/2/2/2/2/2 = 7$). Từ đó, AlexNet sử dụng hai tầng FC kích thước 4096 và cuối cùng là các tầng FC cuối cùng với 1000 neural tính điểm tầng. Có thể chuyển đổi mỗi tầng FC này sang tầng Conv như mô tả ở trên:

- + Thay thế tầng FC đầu tiên có khối $[7 \times 7 \times 512]$ bằng một tầng tích chập(Conv) sử dụng kích thước bộ lọc $\mathbf{F} = 7$, tạo ra khối đầu ra $[1 \times 1 \times 4096]$.
- + Thay tầng FC thứ hai bằng một tầng tích chập(Conv) sử dụng kích thước bộ lọc $\mathbf{F} = 1$, tạo ra khối đầu ra $[1 \times 1 \times 4096]$
- + Thay thế tầng FC cuối cùng tương tự, với $\mathbf{F} = 1$, cho ra kết quả cuối cùng $[1 \times 1 \times 1000]$

Mỗi chuyển đổi này có thể thực tế liên quan đến thao tác (ví dụ như định dạng lại) ma trận trọng số \mathbf{W} trong mỗi tầng FC thành các tầng Conv. Hóa ra chuyển đổi này cho phép chúng ta "trượt" ConvNet ban đầu rất hiệu quả qua nhiều vị trí không gian trong một hình ảnh lớn hơn, chỉ trong một lần chuyển tiếp.

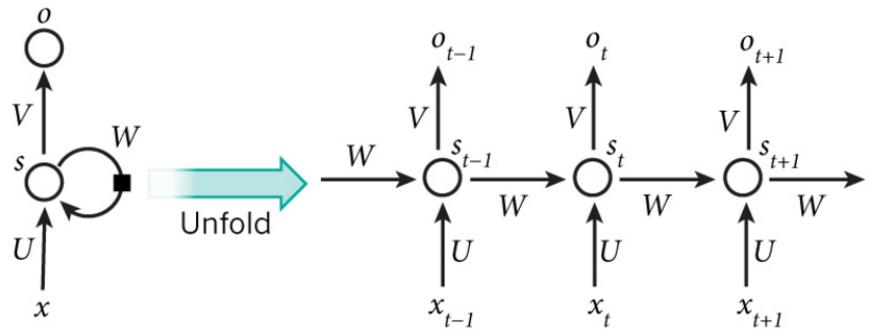
Ví dụ: nếu hình ảnh 224×224 cung cấp khối kích thước $[7 \times 7 \times 512]$ - tức là giảm xuống 32, sau đó chuyển tiếp hình ảnh có kích thước 384×384 qua kiến trúc được chuyển đổi sẽ cho khối lượng tương đương với kích thước $[12 \times 12 \times 512]$, vì $384/32 = 12$. Tiếp theo, với 3 tầng tích chập(Conv) tiếp theo mà chúng ta vừa chuyển đổi từ các tầng FC sẽ cho khối cuối cùng có kích thước $[6 \times 6 \times 1000]$, vì $(12-7) / 1 + 1 = 6$. Lưu ý rằng thay vì một vector duy nhất của điểm tầng có kích thước $[1 \times 1 \times 1000]$, hiện đang nhận được toàn bộ 6×6 mảng điểm số trên hình 384×384 .

Dương nhiên, chuyển tiếp ConvNet đã được chuyển đổi một lần duy nhất hiệu quả hơn nhiều so với việc lặp lại ConvNet gốc trên tất cả 36 vị trí, vì 36 sự đánh giá cùng chia sẻ tính toán. Thủ thuật này thường được sử dụng trong thực tế để có được hiệu năng tốt hơn, ví dụ như thay đổi kích thước một hình ảnh để làm cho nó lớn hơn, sử dụng ConvNet đã được chuyển đổi để đánh giá điểm số của tầng tại nhiều vị trí không gian và sau đó tính điểm trung bình.

Cuối cùng, áp dụng hiệu quả ConvNet ban đầu qua hình ảnh nhưng với bước nhảy nhỏ hơn 32 điểm ảnh, có thể đạt được điều này với nhiều chuyển tiếp. Ví dụ: lưu ý rằng nếu muốn sử dụng bước nhảy 16 điểm ảnh, có thể làm như vậy bằng cách kết hợp các khối nhận được bằng cách chuyển tiếp convCon đã được chuyển đổi hai lần: Trước hết qua hình ảnh gốc và thứ hai trên ảnh nhưng với hình ảnh chuyển không gian bằng 16 điểm ảnh dọc theo chiều rộng và chiều rộng.

3.2 Recurrent Neural Network (RNN)

3.2.1 Giới thiệu



Hình 3.13: Mô hình RNN

Recurrent Neural Network (*RNN*) là một trong những mô hình Deep learning được đánh giá có nhiều ưu điểm trong các tác vụ xử lý ngôn ngữ tự nhiên (*Natural Language Processing*). Tuy nhiên để nắm bắt ngay mô hình này không phải là điều đơn giản, cần có nắm vững lý thuyết về mạng Neural để có thể hiểu được cơ chế hoạt động của mô hình này.

Ý tưởng của RNN đó là thiết kế một mạng Neural sao cho có khả năng xử lý được thông tin dạng chuỗi (*sequential information*), ví dụ một câu là một chuỗi gồm nhiều từ. Recurrent có nghĩa là thực hiện lặp lại cùng một tác vụ cho mỗi thành phần trong chuỗi. Trong đó, kết quả đầu ra tại thời điểm hiện tại phụ thuộc vào kết quả tính toán của các thành phần ở những thời điểm trước đó.

Nói cách khác, RNN là một mô hình có trí nhớ (*memory*), có khả năng nhớ được thông tin đã tính toán trước đó. Không như các mô hình mạng Neural truyền thống đó là thông tin đầu vào (*input*) hoàn toàn độc lập với thông tin đầu ra (*output*). Về lý thuyết, RNN có thể nhớ được thông tin của chuỗi có chiều dài bất kì, nhưng trong thực tế mô hình này chỉ nhớ được thông tin ở vài bước trước đó.

Quan sát sơ đồ biểu diễn RNN, thấy rằng mô hình này có khả năng biểu diễn mối quan hệ phụ thuộc giữa các thành phần trong chuỗi. Ví dụ, nếu chuỗi là một câu có 5 từ thì mạng Neural này sẽ unfold (dàn ra) thành mạng Neural có 5 tầng, mỗi tầng tương ứng với mỗi từ. Dưới đây là ý nghĩa các kí hiệu toán học:

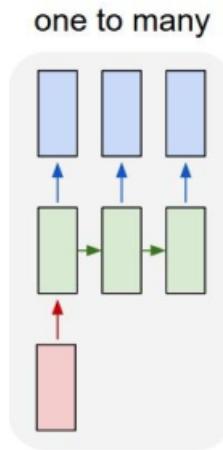
- x_t là input tại thời điểm thứ t. Ví dụ, x_1 là vector của từ thứ hai trong câu (vị trí các từ được đánh số từ 0).
- s_t là *hidden state (memory)* tại thời điểm thứ t. s_t được tính dựa trên các hidden state trước đó kết hợp với input của thời điểm hiện tại $s_t = f(Ux_1 + Ws_{t-1})$.
- Hàm f là hàm *nonlinearity* (tanh hay ReLU). s_{t-1} là hidden state được khởi tạo là một vector không.
- o_t là output tại thời điểm thứ t. o_t là một vector chứa xác suất của toàn bộ các từ trong từ điển $o_t = \text{softmax}(Vs_t)$.

Không như mạng Neural truyền thống, tại mỗi tầng phải sử dụng một tham số khác. RNN chỉ sử dụng một bộ tham số (U , V , W) cho toàn bộ các bước.

3.2.2 Các dạng của RNN

Ngoài dạng cơ bản ở trên, RNN còn có 1 số dạng đặc biệt khác, được biến hóa để phù hợp với các yêu cầu bài toán khác nhau:

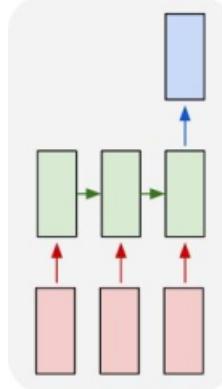
Dạng một - nhiều (One to many): thường dùng trong việc chú thích hình ảnh, với đầu vào là 1 tấm hình và đầu ra là 1 chuỗi các từ mô tả.



Hình 3.14: Dạng RNN một - nhiều

Dạng nhiều - một (Many to One): thường dùng để quyết định kết quả cuối cùng với một số dữ liệu đầu vào cho trước.

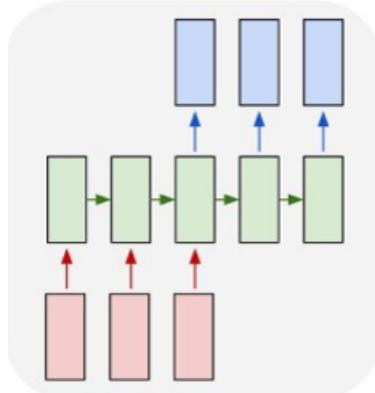
many to one



Hình 3.15: Dạng RNN nhiều - một

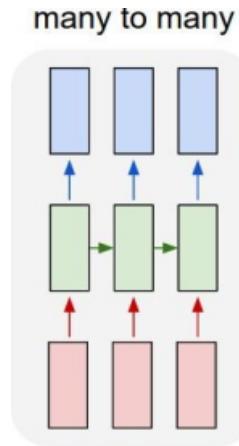
Dạng nhiều - nhiều (Many to Many): thường dùng trong việc dịch thuật, chẳng hạn như 1 đoạn văn bản từ ngôn ngữ này sang ngôn ngữ khác.

many to many



Hình 3.16: Dạng RNN nhiều - nhiều

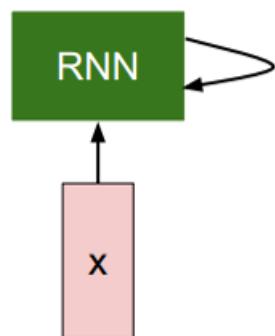
Một dạng khác của RNN nhiều - nhiều (Many to Many): thường dùng để phân loại video trên từng khung hình.



Hình 3.17: Một dạng RNN nhiều - nhiều khác

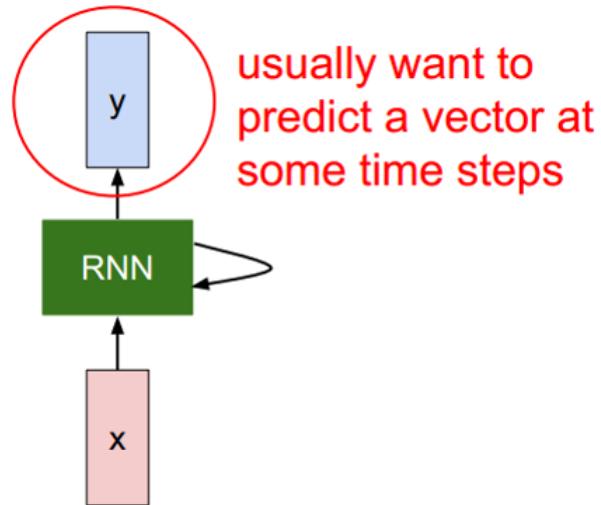
3.2.3 Phân tích RNN

Là ô xanh, nhận các vector đầu vào lại mỗi bước thời gian. Có các trọng số để xác định trạng thái của RNN tại mỗi bước thời gian, khi thay đổi các trọng số, RNN sẽ xử lý khác nhau đối với vector đầu vào.



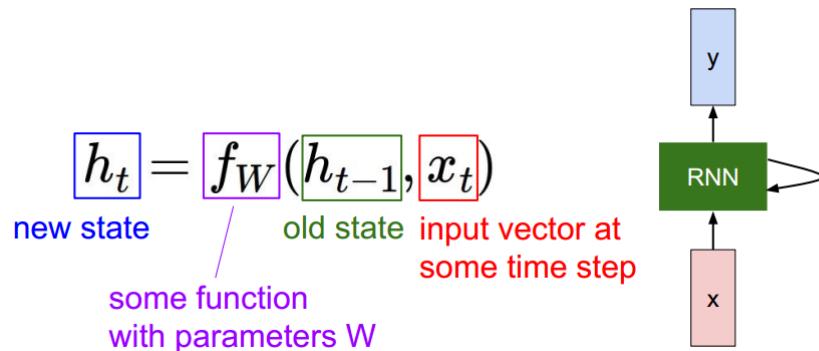
Hình 3.18: Phân tích RNN

Tiếp theo, RNN tạo ra các vector đầu ra dựa trên trạng thái của RNN.



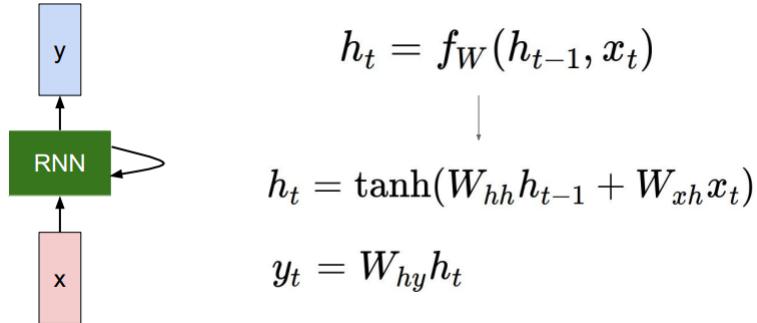
Hình 3.19: RNN tạo các vector đầu ra

Xử lý một chuỗi các vector X , áp dụng công thức RNN tại mỗi bước thời gian:



Hình 3.20: Công thức tổng quát của RNN

3.2.4 Công thức RNN



Hình 3.21: Công thức RNN

Chỉ có 1 trạng thái ẩn h duy nhất. Công thức RNN cho biết làm thế nào cập nhật trạng thái ẩn h như 1 giá trị của tầng ẩn trong bước thời gian trước và đầu vào hiện tại X_t .

Các ma trận trọng số W_{hh} , W_{xh} sẽ phỏng đại các giá trị tầng ẩn của bước thời gian trước và đầu vào hiện tại X_t , sau đó cộng 2 giá trị lại với nhau và tính $\tanh()$ của giá trị đó. Sau đó cập nhật với trạng thái ẩn tại bước thời gian t.

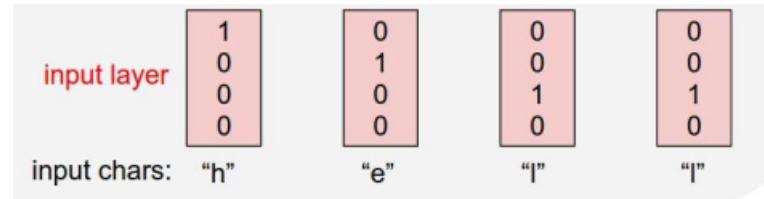
Sau đó sẽ dự đoán đầu ra dựa vào h_t và ma trận trọng số W_{hy} .

3.2.5 Ví dụ RNN

Cho bộ từ vựng [h, e, l, o].

Chuỗi từ huấn luyện: “hello”

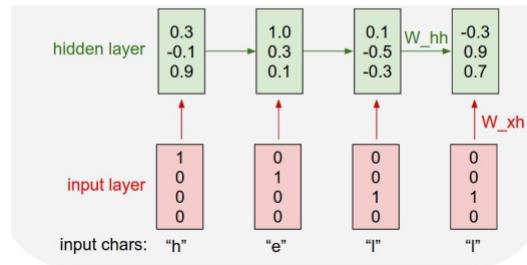
1) Truyền cùng lúc các ký tự vào RNN



Hình 3.22: Truyền ký tự vào RNN

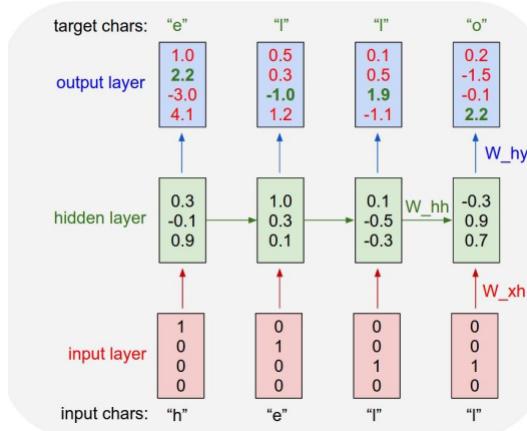
2) Bắt đầu bằng ký tự "h", áp dụng công thức để tính trạng thái RNN tại mỗi bước thời gian. Giả sử chỉ có 3 số trong trạng thái ẩn, sẽ biến chúng thành các vector đại diện, tại mỗi bước thời gian, tổng hợp các ký tự từ trước đến lúc đó.

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$



Hình 3.23: Áp dụng công thức RNN

3) Áp dụng RNN để huấn luyện tại mỗi bước thời gian, ký tự gì sẽ xuất hiện tiếp theo. Có 4 ký tự trong bộ từ vựng, vì vậy sẽ huấn luyện 4 số tại mỗi bước thời gian. VD: đã biết "e" sẽ đứng sau "h" nên 2.2 sẽ là số đúng, tương tự -1.0 cho "l", 1.9 cho "l" và 2.2 cho "o". Vì đã hết từ huấn luyện nên sau đó RNN sẽ gắn 1 "End token" cho ký tự cuối cùng là "o".



Hình 3.24: Tính toán xác suất của từ

3.2.6 Huấn luyện RNN

Huấn luyện RNN tương tự như huấn luyện mạng Neural truyền thống. Cũng sử dụng đến thuật toán lan truyền ngược(backpropagation) nhưng có một chút tinh chỉnh. Gradient tại mỗi output không chỉ phụ thuộc vào kết quả tính toán của bước hiện tại mà còn phụ thuộc vào kết quả tính toán của các bước trước đó.

Ví dụ, để tính gradient tại thời điểm $t = 4$, cần lan truyền ngược 3 bước trước đó và cộng dồn các gradient này lại với nhau. Kỹ thuật này gọi là Backpropagation Through Time (*BPTT*). Điểm hạn chế ở đây đó là tầng ẩn không có trí nhớ dài hạn. Vấn đề này còn gọi là *vanishing/exploding gradient problem* và LSTM được sinh ra để giải quyết vấn đề này.

3.3 Long Short-Term Memory (LSTM)

Deep learning là một kĩ thuật Machine Learning mạnh mẽ đang được nhiều người trong ngành biết đến và nghiên cứu. Kĩ thuật này nổi trội là do chúng thực hiện được hai việc cùng lúc: biểu diễn thông tin (represent problem/feature engineering) và học (learning). Do đó, kĩ thuật này còn được gọi là representation learning.

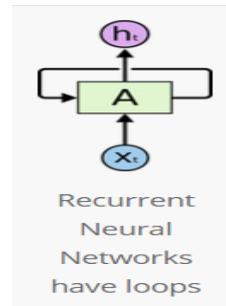
Bên cạnh các lĩnh vực đã gặt hái được nhiều thành công như xử lý ảnh số và video số, hay xử lý tiếng nói, Deep Learning cũng được áp dụng vào xử lý ngôn ngữ tự nhiên. Cụ thể trong đề tài này, Long short-term memory (LSTM) là mô hình cải tiến từ RNN cũng thuộc họ Deep Learning mà ta cần quan tâm.

3.3.1 Nhắc lại Recurrent Neural Network

Để đọc hiểu một câu hay một đoạn văn, con người chúng ta không quên hết những thông tin trước đó mà sẽ xâu chuỗi lại các dữ liệu đã đọc để làm rõ ý nghĩa cho thông tin hiện tại. Tương tự như việc ôn bài, việc làm này giúp các liên kết bên trong não được khắc sâu và tổ chức lại thông tin cho việc truy suất và xử lý thông tin trong tương lai được rõ ràng và nhanh chóng hơn.

Các mô hình mạng Neural truyền thống không thể làm được điều này. Ví dụ, trong bài toán phân lớp sự kiện cho một đoạn video theo từng giây. Thật khó để một mô hình như vậy có thể dựa vào thông tin trước đó để phân lớp.

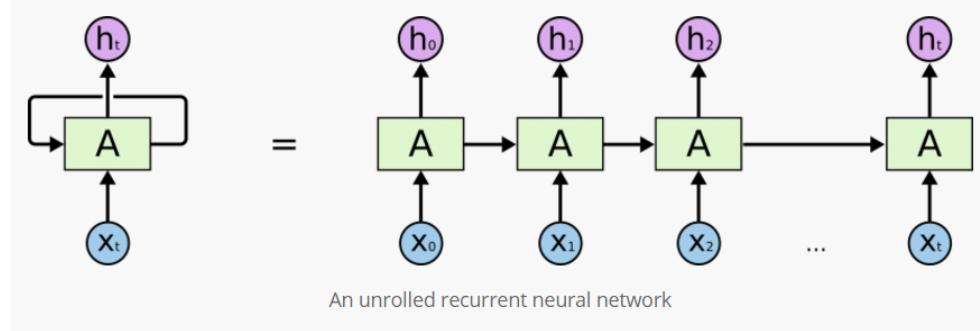
RNN giải quyết vấn đề này bằng cách tạo ra các mạng vòng lặp bên trong chúng, cho phép thông tin được lưu trữ lại cho các lần phân tích tiếp theo.



Hình 3.25: Mô hình RNN có vòng lặp

Trong biểu đồ trên, A nhận thông tin của x_t tại thời điểm t và phản hồi lại tương ứng kết quả đầu ra h_t tại thời điểm t . Có thể rã vòng lặp trên thành một

tiến trình để dễ hình dung hơn.

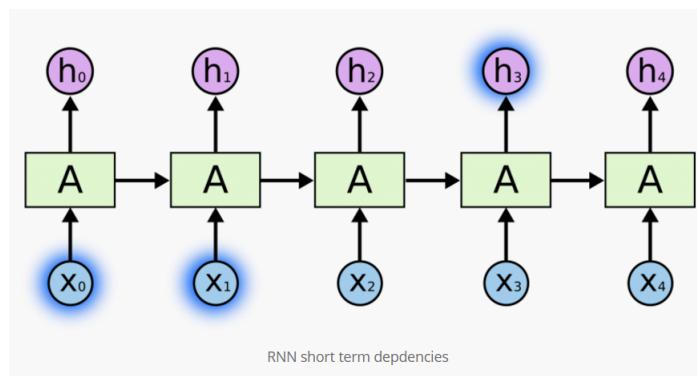


Hình 3.26: Tách các vòng lặp từ RNN

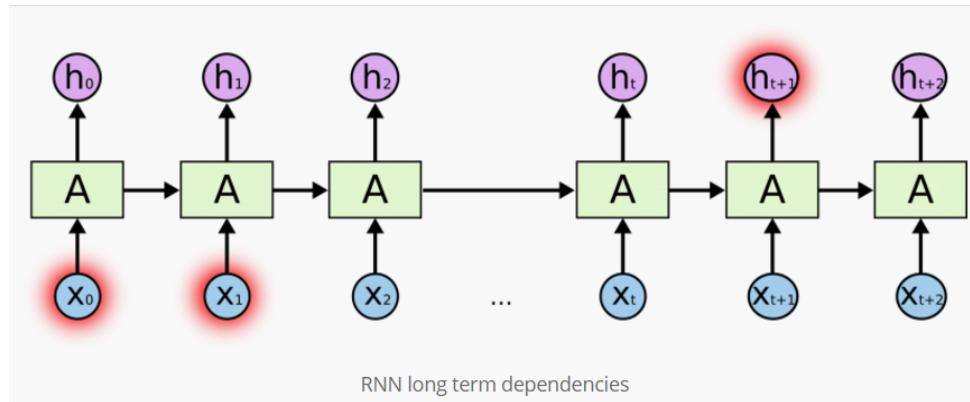
Thay vì chỉ nhận một đầu vào là x như các mạng Neural truyền thống, RNN hình thành nên một chuỗi các đầu vào A cùng với x_t tại thời điểm t .

3.3.2 Vấn đề phụ thuộc quá dài (Long-Term Dependencies)

Một trong những ý tưởng khởi thuỷ của RNN là kết nối những thông tin trước đó nhằm hỗ trợ cho các xử lý hiện tại. Nhưng đôi khi, chỉ cần dựa vào một số thông tin gần nhất để thực hiện tác vụ hiện tại. Ví dụ, khi cố gắng dự đoán từ tiếp theo dựa vào các từ trước đó. Nếu chúng ta dự đoán từ cuối cùng trong câu “đám mây bay trên bầu trời”, thì chúng ta không cần truy tìm quá nhiều từ trước đó, ta có thể đoán ngay từ tiếp theo sẽ là “bầu trời”. Trong trường hợp này, khoảng cách tới thông tin liên quan được rút ngắn lại.



Hình 3.27: Vấn đề phụ thuộc dài



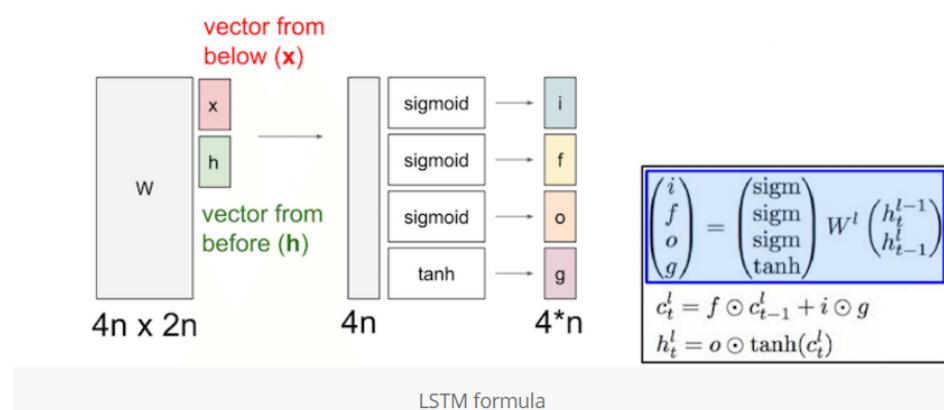
Hình 3.28: Vấn đề phụ thuộc dài

Về lý thuyết, RNN hoàn toàn có khả năng xử lý “long-term dependencies”, nghĩa là thông tin hiện tại có được là nhờ vào chuỗi thông tin trước đó. Thật không may, trong thực tế, RNN thường như không có khả năng này. Vấn đề này đã được “Hochreiter (1991) [German] and Bengio, et al. (1994)” đưa ra như một thách thức cho mô hình RNN.

Trong những năm 1990, RNN phải đối diện với hai thách thức lớn đó là **Vanishing** và **Exploding Gradients** ảnh hưởng lớn đến hiệu suất của mô hình. Vấn đề này phát sinh trong quá trình huấn luyện.

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]



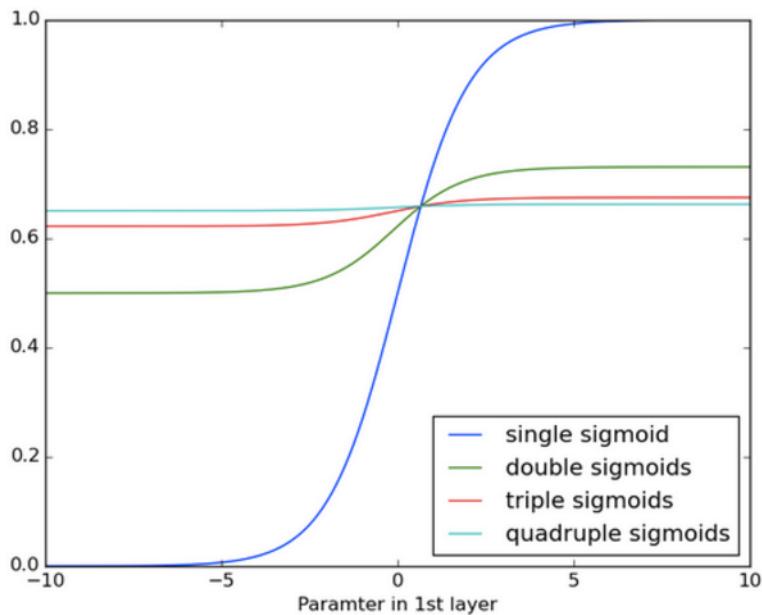
Hình 3.29: Công thức LSTM

Nếu trọng số của ma trận W nhỏ (trị riêng trọng số của ma trận nhỏ hơn 1.0), điều này sẽ dẫn đến trường hợp được gọi là **Vanishing gradients** khi gradient signal ngày càng nhỏ/tan biến theo quá trình huấn luyện, khiến cho quá trình tối thiểu hoá hàm lỗi hội tụ chậm hoặc dừng hẳn.

Ngược lại, nếu trọng số của ma trận W lớn (trị riêng trọng số của ma trận lớn hơn 1.0), điều này sẽ dẫn đến trường hợp được gọi là **Exploding gradients** khi gradient signal ngày càng bị phân tán trong quá trình huấn luyện, khi đó quá trình tối thiểu hoá hàm lỗi không hội tụ.

Mạng recurrent tìm kiếm và hình thành mối liên kết giữa kết quả cuối cùng(final output) và các sự kiện đầu vào(input event) thông qua nhiều bước trước khi kết thúc quá trình huấn luyện. Vẫn đề đặt ra là các thông tin đầu vào(input) trước đó cần đặt trọng số tương ứng là bao nhiêu. Do vậy, đối với câu huấn luyện càng dài thì thông tin trước đó ngày càng bị nhiễu hoặc bị che lấp. Khi thực hiện quá nhiều phép toán nhân ma trận liên tục xuyên suốt chiều dài của chuỗi thì hiệu ứng **Vanishing/Exploding** sẽ xuất hiện.

Hình minh họa dưới đây cho thấy hiệu ứng khi áp dụng liên tiếp hàm $sigmoid()$. Dữ liệu thu được ngày càng tan biến dần cho đến lúc không còn nhận ra được nữa. Tương tự như **gradient vanishing** khi ta truyền dữ liệu qua nhiều tầng.

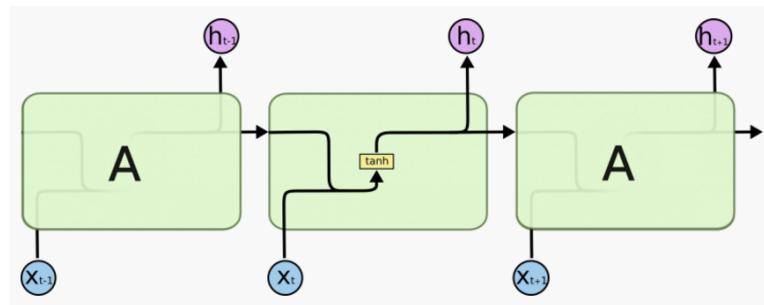


Hình 3.30: Áp dụng liên tiếp hàm $sigmoid()$

3.3.3 LSTM Network

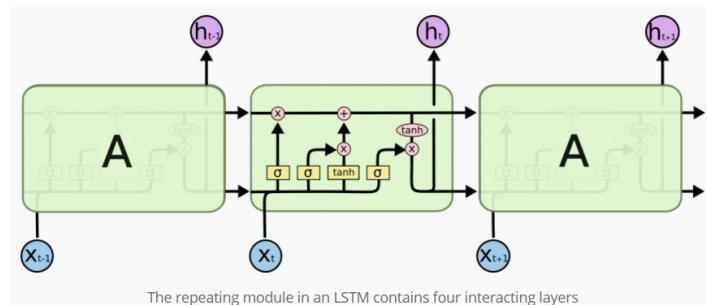
Mạng Long Short Term Memory – thường được gọi là “LSTM”, là trường hợp đặc biệt của RNN, có khả năng học long-term dependencies. Mô hình này được giới thiệu bởi Hochreiter & Schmidhuber (1997), và được cải tiến lại. Sau đó, mô hình này dần trở nên phổ biến nhờ vào các công trình nghiên cứu gần đây. Mô hình này có khả năng tương thích với nhiều bài toán nên được sử dụng rộng rãi ở các ngành liên quan.

LSTM được thiết kế nhằm loại bỏ vấn đề long-term dependency. Trước khi đi vào LSTM, cần quan sát lại mô hình RNN bên dưới, các tầng đều mắc nối với nhau thành các thành phần trong mạng Neural. Trong RNN chuẩn, thành phần lặp lại(repeating module) này có cấu trúc rất đơn giản chỉ gồm một hàm tanh.



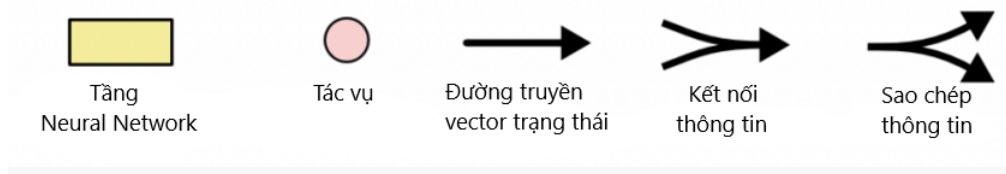
Hình 3.31: Cấu trúc RNN chuẩn

LSTM cũng có cấu trúc mắc xích tương tự, nhưng các thành phần lặp lại(repeating module) có cấu trúc khác hẳn. Thay vì chỉ có một tầng mạng Neural, có tối bốn tầng, tương tác với nhau theo một cấu trúc cụ thể.



Hình 3.32: Cấu trúc LSTM

Trước tiên, ta hãy làm quen với ký hiệu được sử dụng.



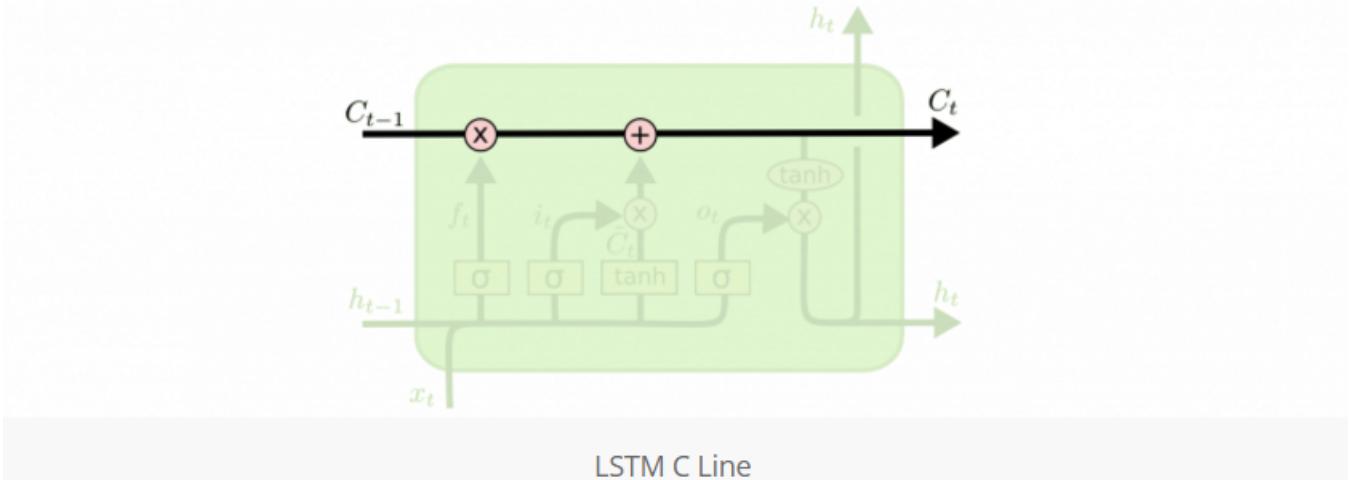
Hình 3.33: Ký hiệu sử dụng

Ở biểu đồ trên, hình tròn nền hồng biểu diễn tác vụ sẽ thực hiện, ví dụ cộng vector. Hình hộp nền vàng là các tầng mạng Neural được huấn luyện. Đường kẻ giao nhau biểu thị kết nối các thông tin, trong khi đó đường rẽ nhánh biểu thị cho sự sao chép từ vị trí này sang vị trí khác

3.3.4 Ý tưởng của LSTM

Có lẽ sau khi quan sát mô hình thiết kế của LSTM, sẽ nhận ra ngay, đây là một bảng mạch số, gồm các mạch logic và các phép toán logic trên đó. Thông tin, hay nói khác hơn là tần số của dòng điện di chuyển trong mạch sẽ được lưu trữ, lan truyền theo cách mà chúng ta thiết kế bảng mạch.

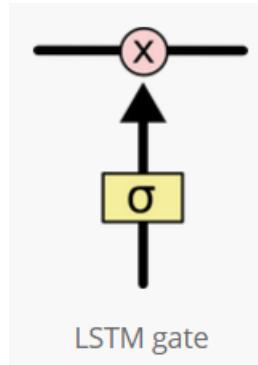
Mấu chốt của LSTM là cell state (đường trạng thái), đường kẻ ngang chạy dọc ở trên top diagram. Cell state giống như băng chuyên. Nó chạy xuyên thảng toàn bộ mắc xích, chỉ một vài tương tác nhỏ tuyến tính (minor linear interaction) được thực hiện. Điều này giúp cho thông tin ít bị thay đổi xuyên suốt quá trình lan truyền.



Hình 3.34: Đường truyền trạng thái

LSTM có khả năng thêm hoặc bớt thông tin vào cell state, được quy định một cách cẩn thận bởi các cấu trúc gọi là cổng (gate).

Các cổng này là một cách (tuỳ chọn) để định nghĩa thông tin băng qua. Chúng được tạo bởi một hàm *sigmoid()*.

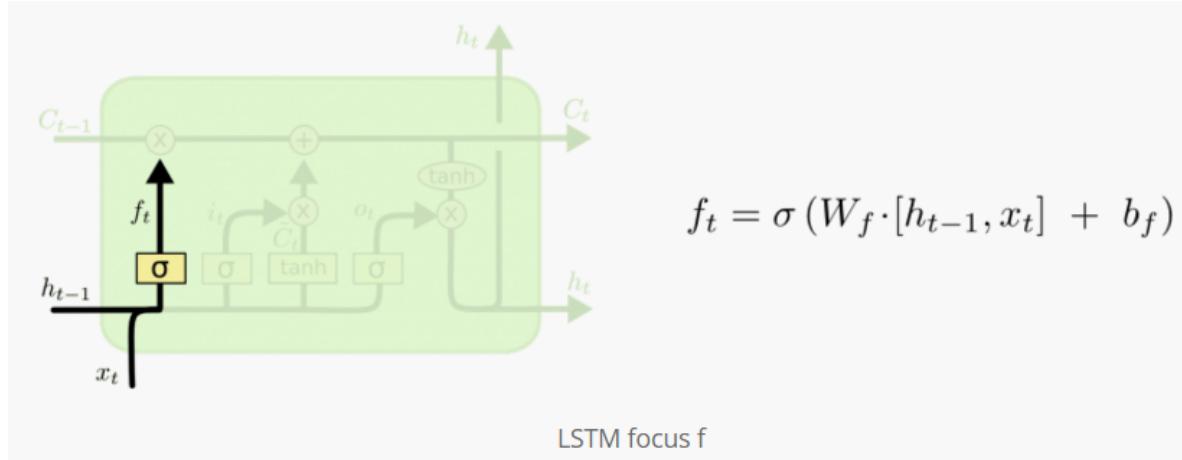


Hình 3.35: Cổng LSTM

Hàm *sigmoid()* có giá trị trong khoảng từ 0 đến 1, mô tả độ lớn thông tin được phép truyền qua. Nếu thu được 0, điều này có nghĩa là “không cho bất kỳ thông tin gì đi qua”, ngược lại nếu thu được giá trị là 1 thì có nghĩa là “cho phép mọi thông tin đi qua”. Một LSTM có ba cổng như vậy để bảo vệ và điều khiển cell state.

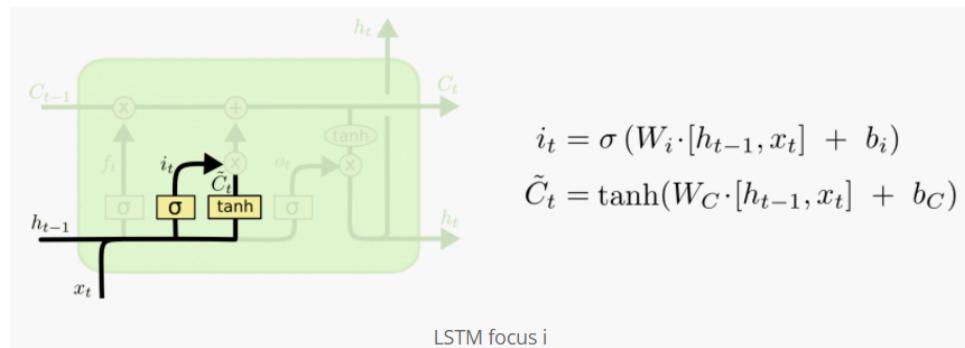
3.3.5 Phân tích mô hình LSTM

Bước đầu tiên của mô hình LSTM là quyết định xem thông tin nào chúng ta cần loại bỏ khỏi cell state. Tiến trình này được thực hiện thông qua một hàm *sigmoid()* gọi là “forget gate layer” – cống quên. Đầu vào là h_{t-1} và x_t , đầu ra là một giá trị nằm trong khoảng $[0, 1]$ cho cell state. 1 tương đương với “giữ lại thông tin”, 0 tương đương với “loại bỏ thông tin”.



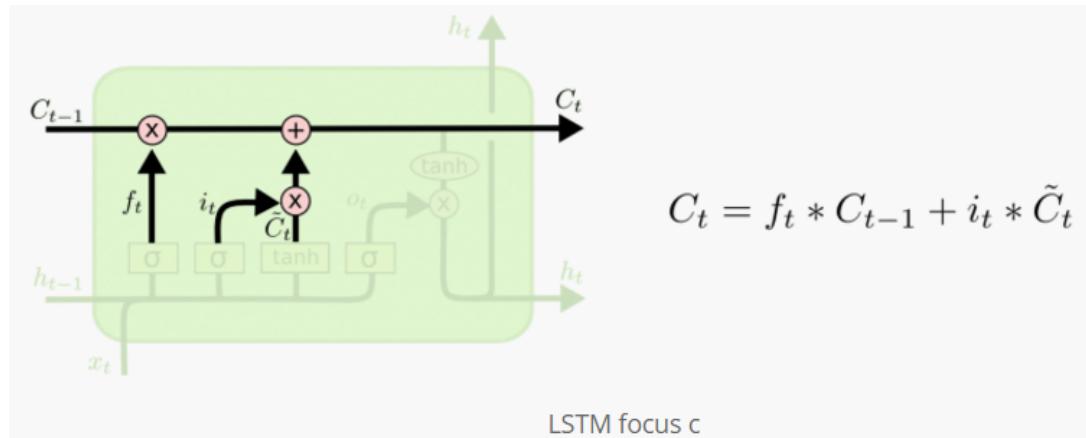
Hình 3.36: Đầu vào thông tin

Bước tiếp theo, ta cần quyết định thông tin nào cần được lưu lại tại cell state. Ta có hai phần. Một, hàm *sigmoid()* được gọi là “input gate layer” quyết định các giá trị chúng ta sẽ cập nhật. Tiếp theo, một hàm *tanh* tạo ra một vector ứng viên mới, \tilde{C}_t được thêm vào trong cell state.



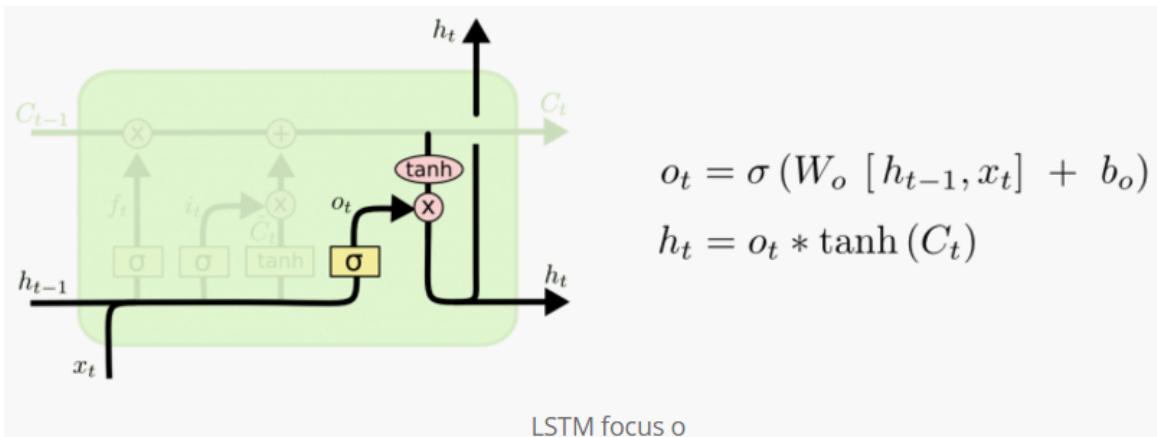
Hình 3.37: Quyết định thông tin

Ở bước tiếp theo, ta sẽ kết hợp hai thành phần này lại để cập nhật vào cell state. Lúc cập nhật vào cell state cũ, C_{t-1} , vào cell state mới C_t . Ta sẽ đưa state cũ hàm f_t , để quên đi những gì trước đó. Sau đó, ta sẽ thêm $(i_t * C_t)$. Đây là giá trị ứng viên mới, co giãn (scale) số lượng giá trị mà ta muốn cập nhật cho mỗi state.



Hình 3.38: Cập nhật vào Cell State

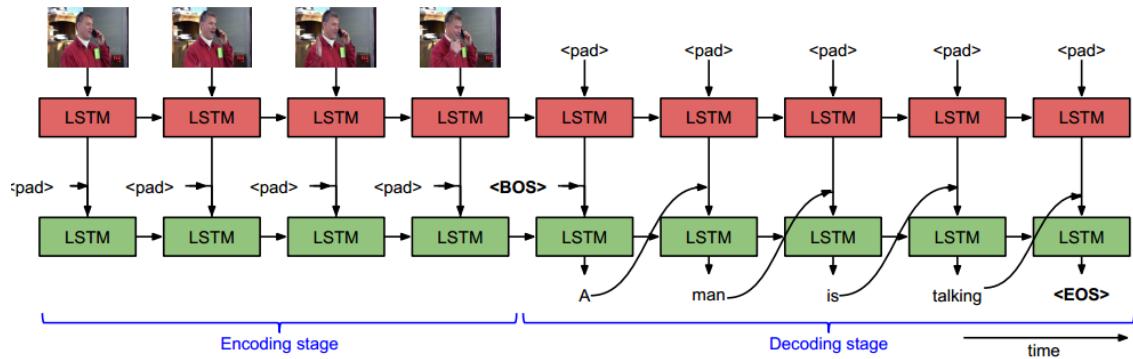
Cuối cùng, cần quyết định xem thông tin output là gì. Output này cần dựa trên cell state, nhưng sẽ được lọc bớt thông tin. Đầu tiên, áp dụng hàm *sigmoid()* để quyết định xem phần nào của cell state dự định sẽ đưa vào output. Sau đó, đẩy cell state qua *tanh* (đẩy giá trị vào khoảng -1 và 1) và nhân với một hàm *sigmoid()*. Output, để giữ lại những phần ta muốn xuất ra ngoài.



Hình 3.39: Thông tin đầu ra

3.4 LSTM nâng cao

Ý tưởng chính là đề xuất một mô hình, trực tiếp lập một LSTM xếp chồng lên nhau, lần đầu tiên đọc vào hình ảnh và sau đó tạo ra các từ.



Hình 3.40: Mô hình LSTM nâng cao

Một ngăn xếp (stack) của hai LSTM, học một đại diện của một hình ảnh để giải mã nó thành một câu mô tả sự kiện. Các lớp LSTM hàng đầu (màu đỏ) mô hình hóa đặc trưng đầu vào hình ảnh. Lớp LSTM thứ hai (màu xanh lá cây) mô hình hóa ngôn ngữ cho đầu vào văn bản và các đại diện ẩn của hình ảnh. Sử dụng **<BOS>** (Begin of sentence) để chỉ ra sự bắt đầu của câu và **<EOS>** (End of sentence) cho thẻ kết thúc câu. Các số 0 được sử dụng như một **<pad>** khi không có dữ liệu đầu vào ở bước tiếp theo.

Cách tiếp cận, được mô tả trong hình trên. Trước tiên mã hóa đầu vào đến một vector chiều dài cố định sử dụng một LSTM và sau đó sử dụng một LSTM khác để ánh xạ Vector vào một chuỗi các kết quả đầu ra, dựa vào một LSTM đơn cho cả giai đoạn mã hóa và giải mã. Điều này cho phép chia sẻ thông số giữa giai đoạn mã hóa và giải mã.

Lớp LSTM đầu nhận vào hình ảnh và mã hóa chúng trong khi LSTM thứ hai lớp nhận được biểu diễn ẩn (h_t) và nối nó với các từ đầu vào 0 (zeros), mà nó sau đó mã hóa. Không có chi phí hao tổn khi truyền thông tin giữa 2 tầng LSTM. Sau đó, lớp LSTM thứ hai được đưa vào thẻ (**<BOS>**), sẽ nhắc nó bắt đầu giải mã thông tin thành một chuỗi các từ.

3.5 Hàm Softmax

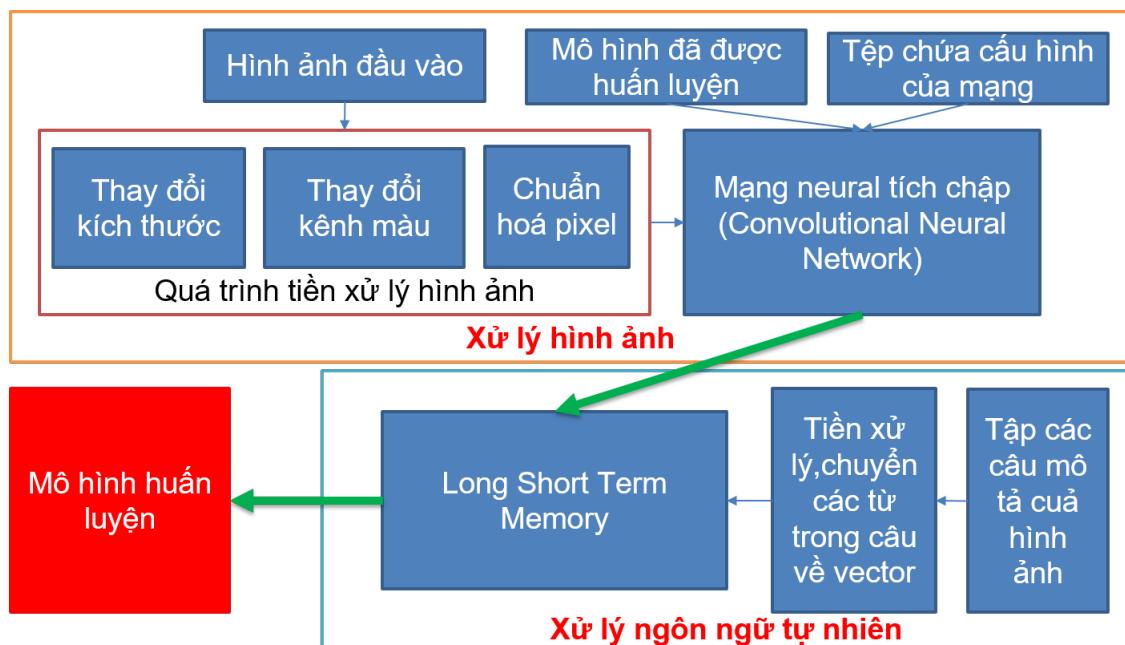
Có một vài thủ thuật cần thiết để có được huấn luyện RNN/LSTM hiệu quả. Thấy rằng sự phân bố của những từ trong bộ từ vựng rất không đồng đều. Do đó, mô hình sử dụng một vài lần lặp đầu tiên để học các bias cho bộ phân loại Softmax sao cho nó dự đoán mọi từ một cách ngẫu nhiên với số liệu tần số thích hợp. Có thể thu được kết quả nhanh hơn trong giai đoạn huấn luyện bằng cách khởi tạo rõ ràng các bias của tất cả các từ trong bộ từ vựng (trong bộ phân loại Softmax) để ghi lại xác suất sự xuất hiện của chúng trong dữ liệu huấn luyện.

Vì vậy, với trọng số nhỏ và thiết lập bias một cách thích hợp, mô hình dự đoán ngay từ ngẫu nhiên theo phân bố xác suất của chúng. Sau khi thực hiện các thí nghiệm bổ sung với việc so sánh một RNN với một LSTM và thấy rằng LSTM luôn mang lại kết quả tốt hơn, nhưng huấn luyện lâu hơn.

Chương 4

Ý TƯỞNG GIẢI QUYẾT BÀI TOÁN

4.1 Mô hình huấn luyện tổng quan



Hình 4.1: Mô hình huấn luyện tổng quan

Mô hình tổng quan của quá trình huấn luyện được chia làm 2 thành phần chính:

* Quá trình xử lý hình ảnh:

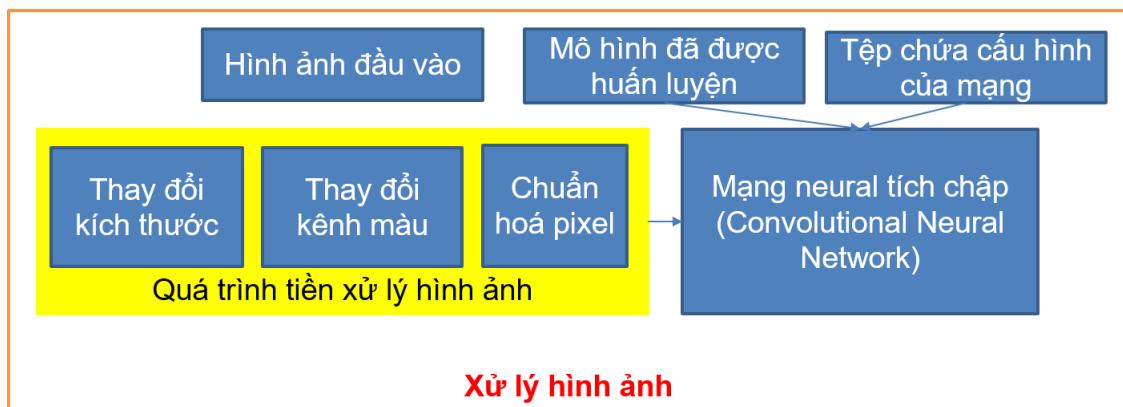
- Quá trình tiền xử lý.
- Dưa vào mạng Convolutional Neural Network để rút trích đặc trưng ảnh.

* Quá trình xử lý ngôn ngữ tự nhiên:

- Quá trình tiền xử lý các câu mô tả và tạo bộ từ điển riêng cho mô hình.
- Chuẩn hóa các từ trong câu mô tả và vector đặc trưng hình ảnh về cùng một kích thước là 256 đặc trưng.
- Dưa vào mô hình Long Short-Term Memory để thực hiện quá trình học câu mô tả cho hình ảnh.

4.1.1 Xử lý hình ảnh

4.1.1.1 Quá trình tiền xử lý hình ảnh



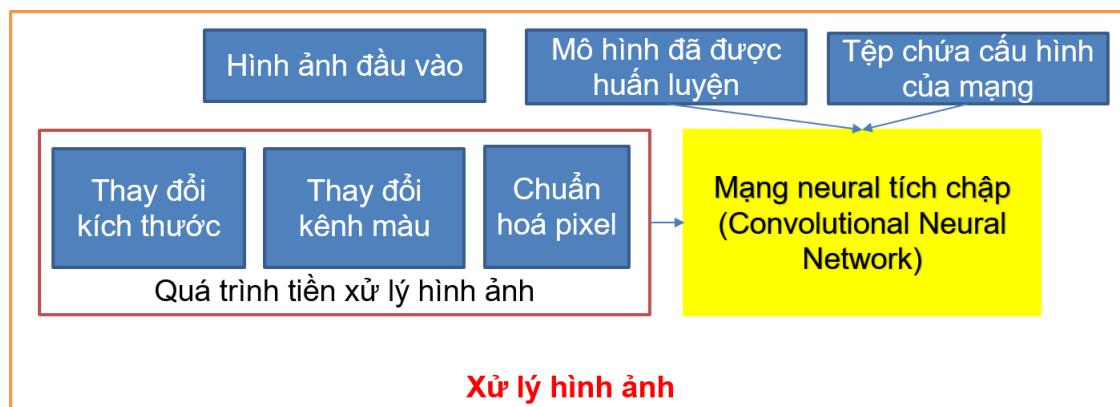
Hình 4.2: Tiền xử lý hình ảnh

Dầu vào của bước này là tấm ảnh và trải qua 3 quá trình:

- Chuyển tất cả hình ảnh huấn luyện về cùng một kích thước 224x224 phù hợp với yêu cầu đầu vào của Convolutional Neural Network trong mô hình.

- Sử dụng phương pháp Mean Subtraction để chuẩn hóa điểm ảnh trên 3 kênh màu. Đây là một trong những cách phổ biến để xử lý hình ảnh đầu vào. Tham số trung bình của 3 kênh màu được tính toán dựa trên tập huấn luyện trong mô hình đã được huấn luyện trước đó.
- Thực hiện đổi hình từ kênh màu RGB sang BGR, kênh màu BGR là một trong những kênh màu phù hợp được hỗ trợ nhiều thư viện hơn.

4.1.1.2 Áp dụng Convolutional Neural Network (CNN)



Hình 4.3: Áp dụng CNN

Tiếp tục quá trình, thực hiện giải thuật Convolutional Neural Network trên hình ảnh đã được tiền xử lý. Kết hợp với một mô hình đã được huấn luyện từ trước (Pretrained model). Mô hình đã được huấn luyện từ trước này sẽ thúc đẩy quá trình học một cách nhanh chóng và chính xác hơn. Nhóm chọn mô hình VGG-Net 16 layer [12]. Đây là mô hình đã tham gia cuộc thi ImageNet ILSVRC 2014 [13]. Mô hình khi tham gia cuộc thi này thì đã được huấn luyện qua 1 triệu 2 trăm nghìn tấm hình với 1000 đối tượng phân lớp. Chi tiết mô hình này được trình bày ở hình sau:

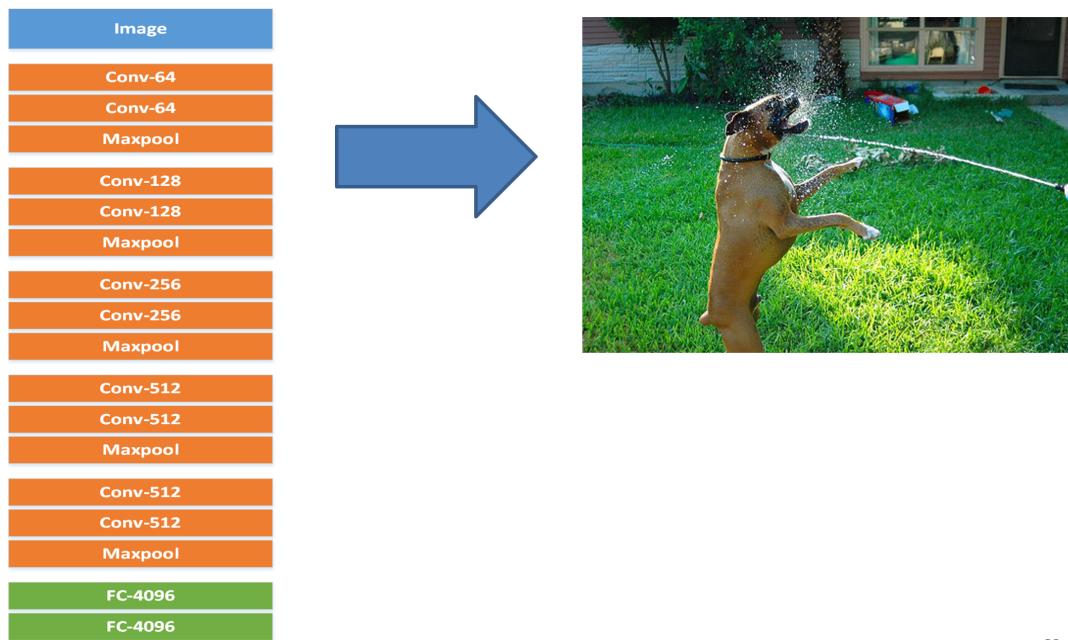


Hình 4.4: Mô hình VGG Net 16 Layer

Sau đây là những lý do nhóm chọn mô hình VGG-Net cho mô hình rút trích CNN:

- Mô hình phù hợp với nhiều loại bài toán.
- Mô hình này đạt kết quả đúng nhất xác định vị trí đối tượng, đứng thứ hai về xử lý phân lớp.
- Đạt tỷ lệ lỗi phân lớp lấy 5 lớp cao nhất (7.5%).

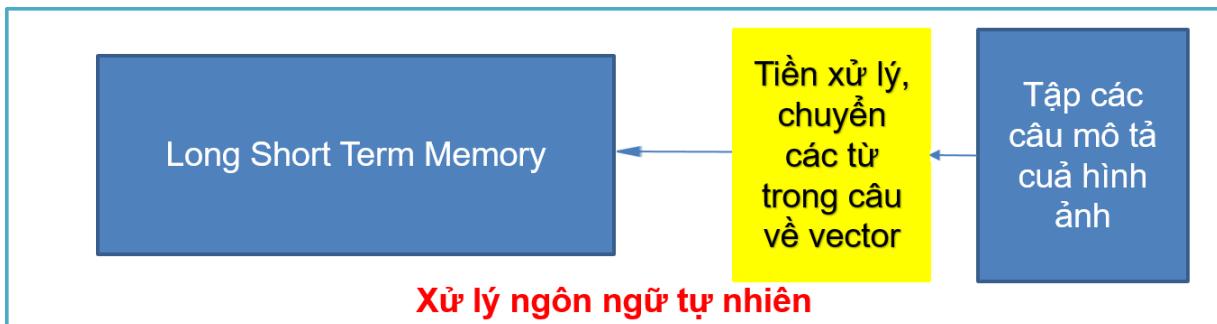
Do quá trình xử lý hình ảnh nhóm tập trung vào việc rút trích đặc trưng nên sẽ bỏ qua 2 phần cuối cùng của mạng, để phù hợp với mô hình bài toán.



Hình 4.5: Mô hình VGG Net áp dụng vào bài toán

4.1.2 Xử lý ngôn ngữ tự nhiên

4.1.2.1 Quá trình tiền xử lý câu mô tả



Hình 4.6: Mô hình tiền xử lý câu mô tả

Tiền xử lý câu mô tả:

- Lọc tất cả các từ của tất cả các câu.
- Chọn ra bộ từ vựng riêng.

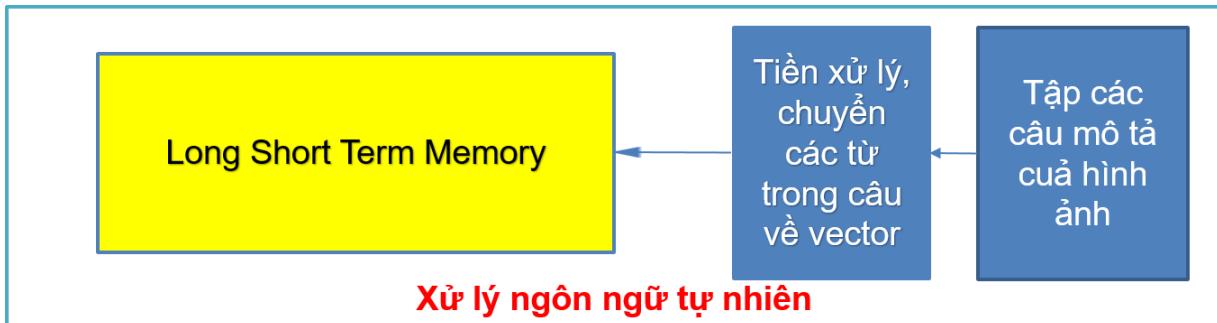
→ Rút ngắn quá trình học.

Chuyển đổi mỗi từ trong bộ từ vựng về một vector đặc trưng có kích thước 256 dựa trên một hàm tuyến tính.

Chuyển đổi kích thước đặc trưng hình ảnh từ 4096 về 256 dựa trên một hàm tuyến tính.

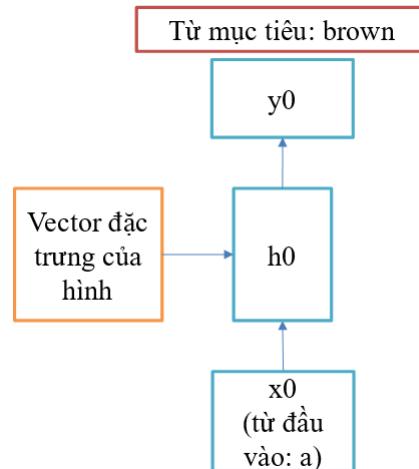
Câu mô tả ảnh “a brown dog is sprayed with water” sẽ được chuyển đổi thành vector 2 chiều [7x256].

4.1.2.2 Áp dụng LSTM



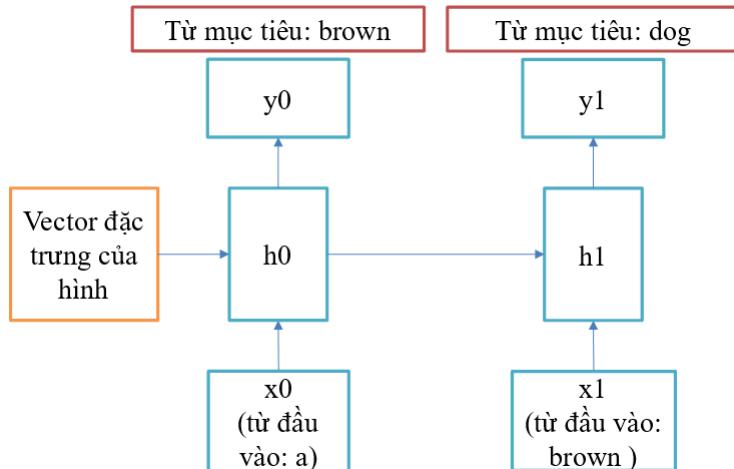
Hình 4.7: Áp dụng LSTM

- 1) Trong tầng đầu khi chạy LSTM, x_0 là vector đặc trưng của từ “a”, vector đặc trưng của hình sẽ được truyền vào h_0 để tính toán, y_0 là vector đầu ra sẽ được tính toán tiếp từ h_0 để xác định cho từ mục tiêu kế tiếp là từ “brown”.



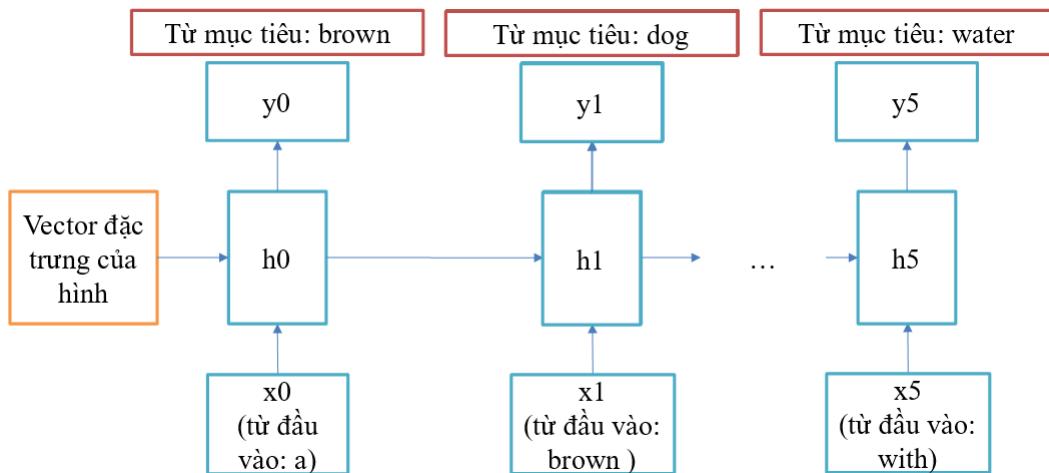
Hình 4.8: Dưa vector đặc trưng hình vào mạng LSTM

2) Kế tiếp, x_1 là vector đặc trưng của từ “brown”, h_0 sẽ được truyền qua h_1 để tính toán, y_1 sẽ được tính toán tiếp từ h_1 để xác định cho từ mục tiêu kế tiếp là từ “dog”.



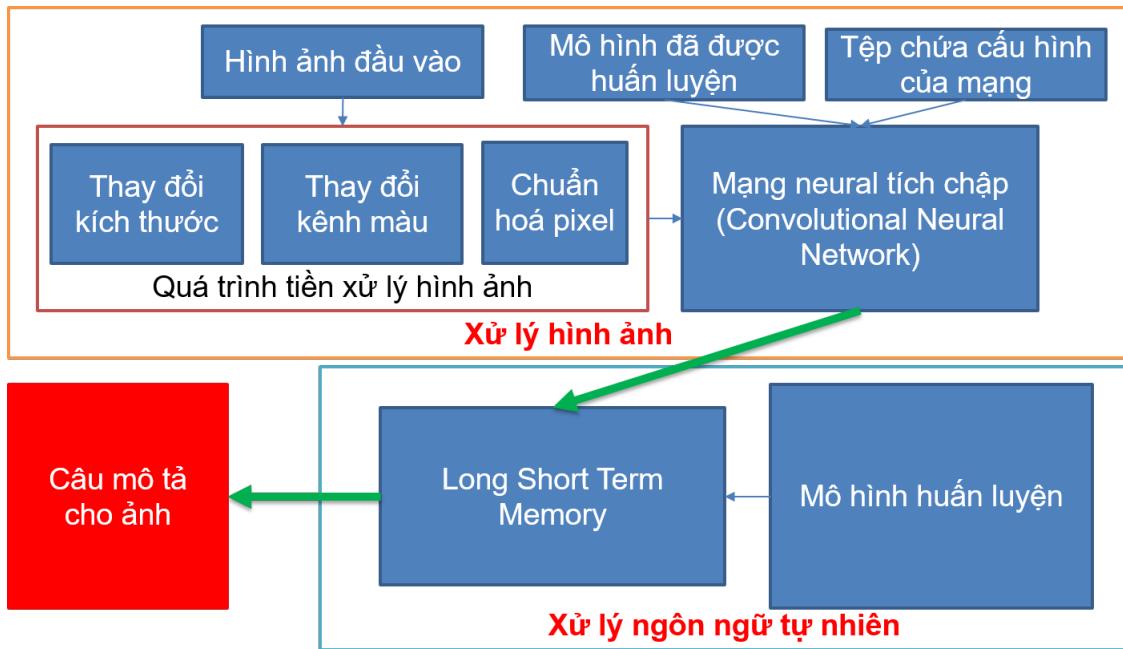
Hình 4.9: Áp dụng LSTM

3) Các từ sẽ được tính toán qua lần lượt các tầng. Đến từ cuối cùng “water” thì sẽ gắn token “End” vào cuối.



Hình 4.10: Tiếp tục áp dụng LSTM

4.2 Mô hình thử nghiệm tổng quan



Hình 4.11: Mô hình thử nghiệm tổng quan

Mô hình tổng quan của quá trình thử nghiệm được chia làm 2 thành phần chính:

* Quá trình xử lý hình ảnh:

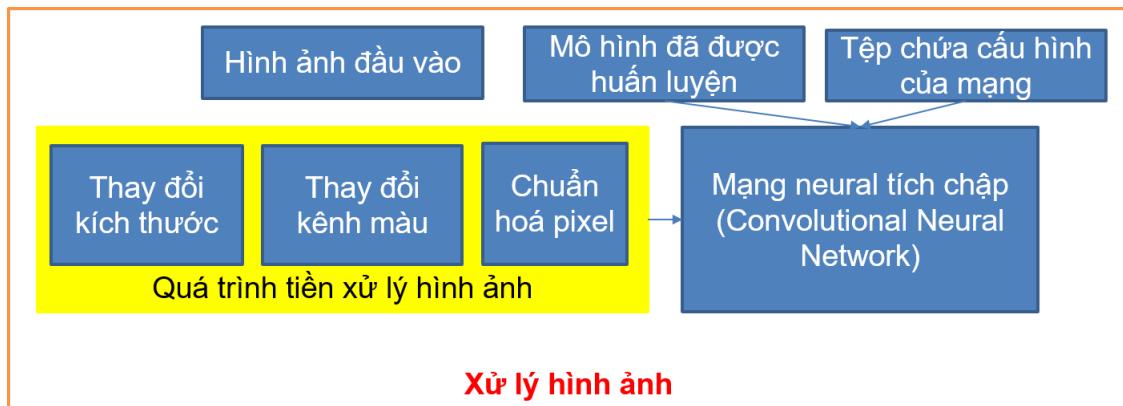
- Quá trình tiền xử lý.
- Đưa vào mạng Convolutional Neural Network để rút trích đặc trưng ảnh.

* Quá trình xử lý ngôn ngữ tự nhiên:

- Kết hợp giữa mô hình đã được huấn luyện với CNN, đưa vào mô hình Long Short-Term Memory để thực hiện quá trình tạo câu mô tả cho hình ảnh.

4.2.1 Xử lý hình ảnh

4.2.1.1 Quá trình tiền xử lý hình ảnh

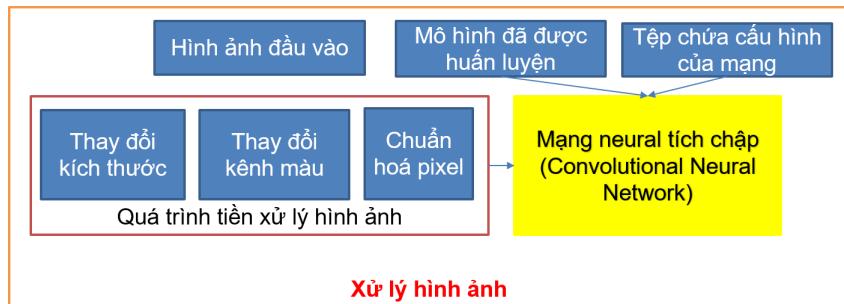


Hình 4.12: Tiền xử lý hình ảnh

Đầu vào của bước này là tấm ảnh và trải qua 3 quá trình:

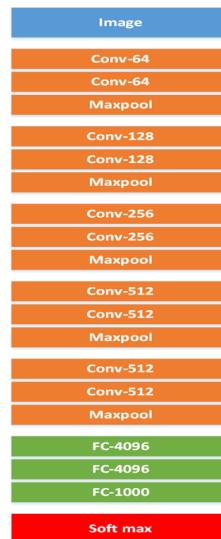
- Chuyển tất cả hình ảnh huấn luyện về cùng một kích thước 224x224 phù hợp với yêu cầu đầu vào của Convolutional Neural Network trong mô hình.
- Sử dụng phương pháp Mean Subtraction để chuẩn hóa điểm ảnh trên 3 kênh màu. Đây là một trong những cách phổ biến để xử lý hình ảnh đầu vào. Tham số trung bình của 3 kênh màu được tính toán dựa trên tập huấn luyện trong mô hình đã được huấn luyện trước đó.
- Thực hiện đổi hình từ kênh màu RGB sang BGR, kênh màu BGR là một trong những kênh màu phù hợp được hỗ trợ nhiều thư viện hơn.

4.2.1.2 Áp dụng Convolutional Neural Network (CNN)



Hình 4.13: Áp dụng CNN

Tiếp tục quá trình, thực hiện giải thuật Convolutional Neural Network trên hình ảnh đã được tiền xử lý. Kết hợp với một mô hình đã được huấn luyện từ trước (Pretrained model). Mô hình đã được huấn luyện từ trước này sẽ thúc đẩy quá trình học một cách nhanh chóng và chính xác hơn. Nhóm chọn mô hình VGG-Net 16 layer [12]. Đây là mô hình đã tham gia cuộc thi ImageNet ILSVRC 2014 [13]. Mô hình khi tham gia cuộc thi này thì đã được huấn luyện qua 1 triệu 2 trăm nghìn tấm hình với 1000 đối tượng phân lớp. Chi tiết mô hình này được trình bày ở hình sau:

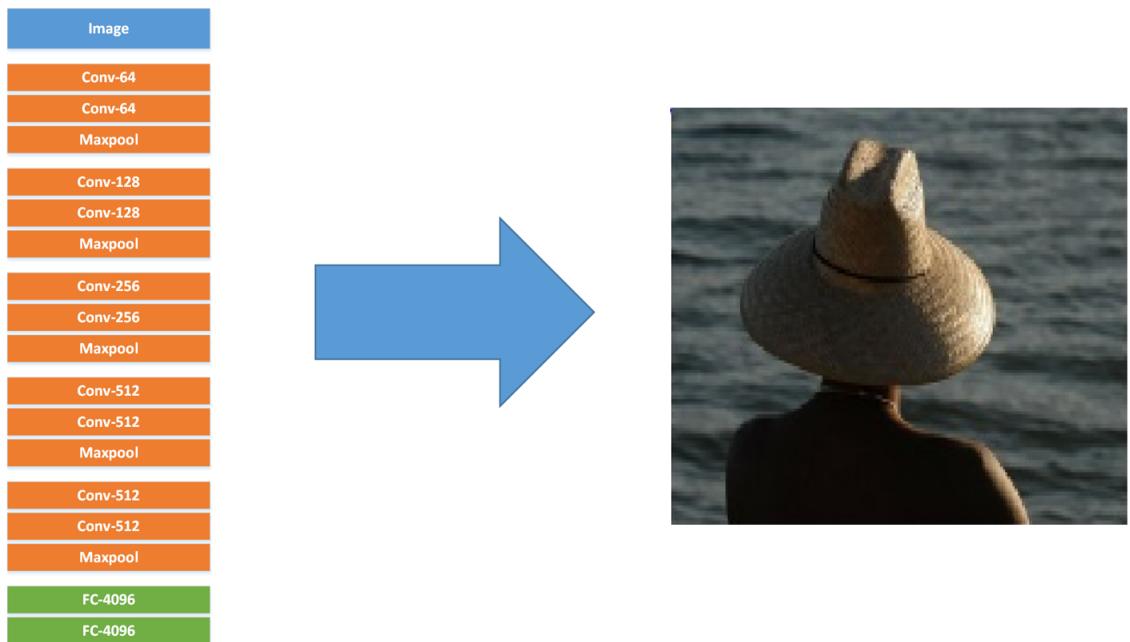


Hình 4.14: Mô hình VGG Net 16 Layer

Sau đây là những lý do nhóm chọn mô hình VGG-Net cho mô hình rút trích CNN:

- Mô hình phù hợp với nhiều loại bài toán.
- Mô hình này đạt kết quả đúng nhất xác định vị trí đối tượng, đứng thứ hai về xử lý phân lớp.
- Đạt tỷ lệ lỗi phân lớp lấy 5 lớp cao nhất (7.5%).

Do quá trình xử lý hình ảnh nhóm tập trung vào việc rút trích đặc trưng nên sẽ bỏ qua 2 phần cuối cùng của mạng, để phù hợp với mô hình bài toán.



Hình 4.15: Mô hình VGG Net áp dụng vào bài toán

4.2.2 Xử lý ngôn ngữ tự nhiên

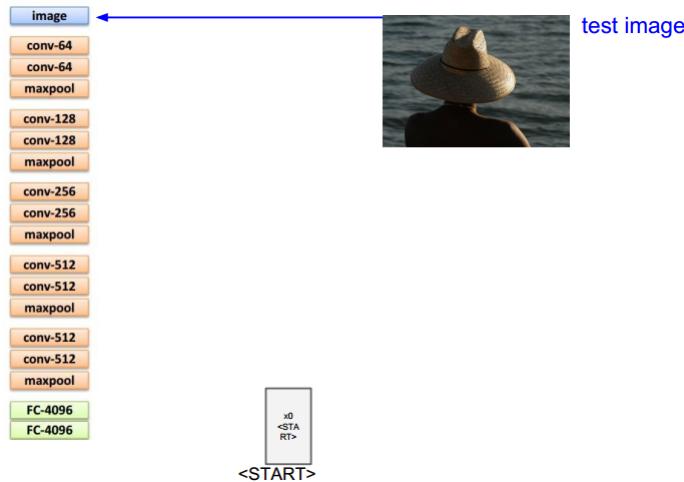
4.2.2.1 Áp dụng LSTM



Hình 4.16: Áp dụng LSTM

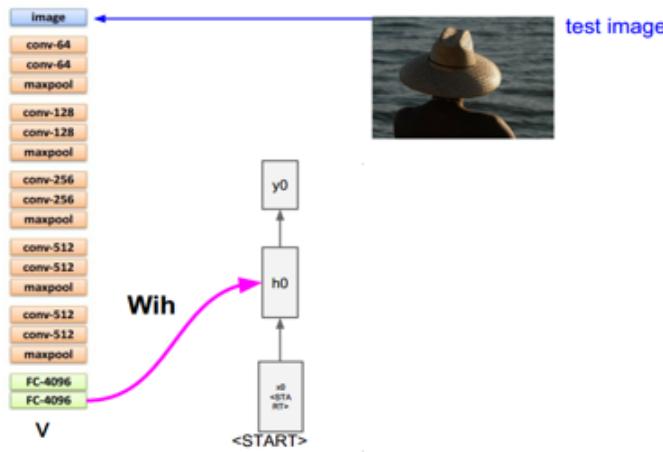
Áp dụng mô hình đã được huấn luyện vào LSTM để tạo ra câu mô tả cho hình ảnh.

1) Truyền hình ảnh vào CNN, xử lý qua hết các tầng và chuyển vào không gian LSTM với vector START.



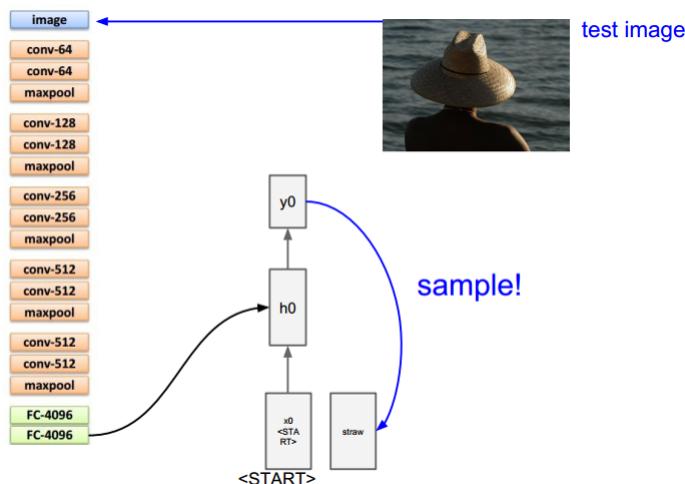
Hình 4.17: Truyền hình ảnh vào CNN

2) Áp dụng LSTM với y_0 là từ đầu tiên "Straw" (cái nón). W_{ih} là điều kiện biến trạng thái ẩn thành trạng thái cụ thể khi xác suất của từ "Straw" có thể cao hơn 1 chút.

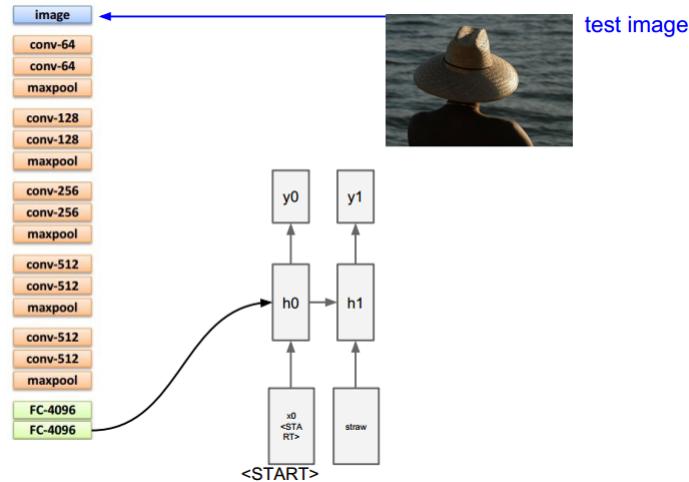


Hình 4.18: Áp dụng LSTM

3) Huấn luyện dự đoán từ tiếp theo và lưu nhớ thông tin của bước trước. Lấy từ "Straw" để làm đầu vào của LSTM tiếp theo. "Straw" sẽ có mối quan hệ với không gian vector, được học với mỗi từ trong bộ từ vựng. Từ đó tính toán xác suất từ tiếp theo.

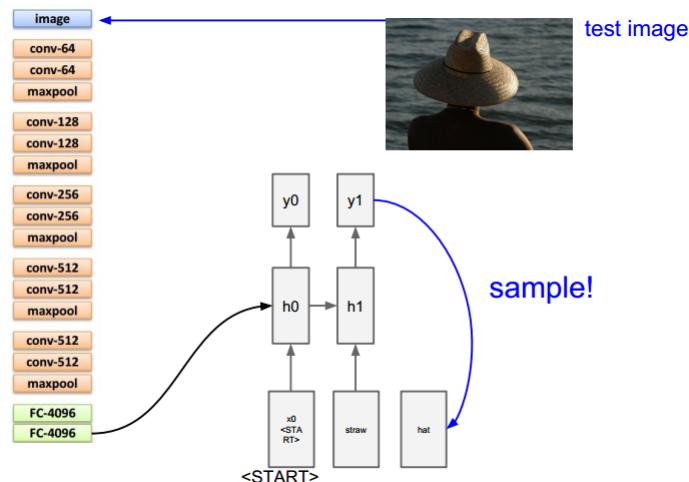


Hình 4.19: LSTM truyền thông tin sang bước tiếp theo



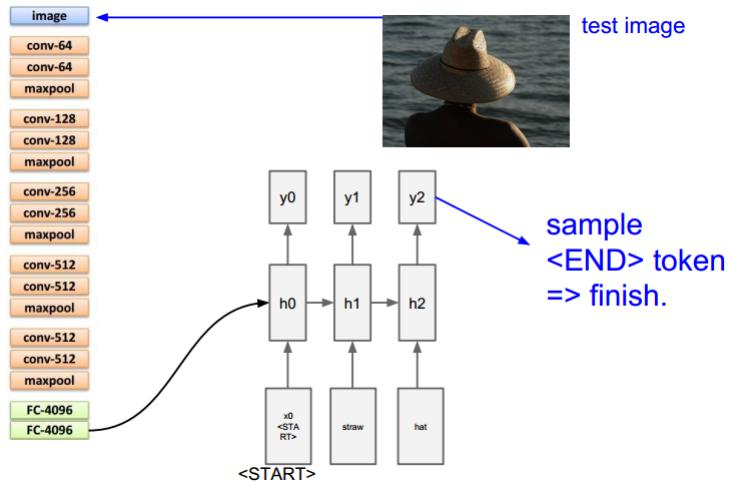
Hình 4.20: Tiếp tục áp dụng LSTM

4) Giả sử rằng "Hat" được tìm thấy tiếp theo và tiếp tục tính toán



Hình 4.21: LSTM truyền thông tin sang bước tiếp theo

5) Cho đến khi gặp thẻ <END> Token thì dừng lại, báo hiệu kết thúc câu (Giả sử trong tập huấn luyện, cấu trúc câu có dạng "Từ 1 + Từ 2 + End"). LSTM kết thúc và caption cho hình ảnh là "Straw hat."



Hình 4.22: LSTM kết thúc

4.3 Phương pháp đánh giá và tinh chỉnh tham số cho mô hình huấn luyện

4.3.1 Độ hỗn loạn thông tin - *Perplexity*

Độ hỗn loạn thông tin của một từ là thể hiện số từ có thể đi sau từ đó. Ví dụ độ hỗn loạn thông tin của từ "nón" là thể hiện số từ có thể đi sau từ "nón". Độ hỗn loạn thông tin của mô hình ngôn ngữ, có thể hiểu đơn giản là số lựa chọn trung bình mà mô hình ngôn ngữ phải đưa ra quyết định. Như vậy, độ hỗn loạn càng thấp thì chứng tỏ mô hình càng có sự ổn định, độ chính xác trong mô hình ngôn ngữ càng cao.

4.3.2 Cách tính *Perplexity*

Giả sử ta có tập các câu để test (không nằm trong tập huấn luyện) $x^{(1)}, x^{(2)}, \dots, x^{(m)}$. Mỗi câu $x^{(i)}$ có $i \in \{1 \dots m\}$ là chuỗi các từ $x_1^{(i)}, \dots, x_{n_i}^{(i)}$ trong đó n_i là độ dài của câu thứ i .

Tính xác suất $p(x^{(i)})$ cho từng $x^{(i)}$ thông qua mô hình ngôn ngữ vừa huấn luyện xong. Khi đó, chất lượng của mô hình ngôn ngữ này sẽ được tính như sau: $\prod_{i=1}^m p^{(i)}$

Giá trị thu được từ phép tính trên càng cao thì chất lượng của mô hình càng tốt với dữ liệu chưa từng có trong tập huấn luyện.

Perplexity được định nghĩa là: $Perplexity = 2^{-l}$

$$\text{Trong đó: } \begin{cases} M = \sum_{i=1}^m n_i \\ l = \frac{1}{M} \sum_{i=1}^m \log_2 p(x^{(i)}) \end{cases}$$

Theo công thức trên, nếu giá trị của *Perplexity* càng nhỏ, thì mô hình ngôn ngữ xây dựng được càng tốt.

4.3.3 Áp dụng vào mô hình huấn luyện

Sau quá trình huấn luyện dữ liệu thì kết quả đạt được là nhiều mô hình huấn luyện. Tiến hành quá trình validation-set dùng để đánh giá các mô hình huấn luyện. Sử dụng 49000 ảnh cho việc huấn luyện, 1000 ảnh cho việc đánh giá. Quá trình validation-set này được sử dụng như quá trình thử nghiệm ảo nhằm để tinh chỉnh các thông số cho mô hình. Khoảng chia thường được sử dụng trong khoảng 50-90% cho tập huấn luyện và còn lại cho tập đánh giá. Tuy nhiên điều này còn phụ thuộc nhiều yếu tố. Chẳng hạn như nếu mô hình có nhiều thông số, nên sử dụng bộ đánh giá có số lượng lớn hơn.

Để đánh giá mô hình huấn luyện này có tốt hay không khi so với các mô hình huấn luyện khác, tiến hành so sánh các độ hỗn loạn thông tin giữa các mô hình. Mô hình có độ hỗn loạn thông tin nhỏ nhất sẽ được chọn.

Chương 5

KẾT QUẢ THỰC NGHIỆM

5.1 Đánh giá kết quả tạo ra các câu mô tả

5.1.1 Tiêu chí đánh giá BLEU [10]

Là tiêu chí xác định xem trong số các từ mô tả được tạo ra, có bao nhiêu phần trăm từ mô tả đúng với nội dung của các câu cho mô tả khi huấn luyện.

Công thức: $B = \frac{m}{w} = \frac{2}{7}$

Candidate: the the the the the the the.

Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

Trong đó:

- B (BLEU): là tỉ lệ chính xác.
- m: số từ mô tả đúng.
- w: số từ mô tả trong câu huấn luyện.

5.1.2 Phương pháp tìm kiếm Beam Search

Là phương pháp lặp đi lặp lại việc xem xét giữ k số câu tốt nhất đã tạo ra trong quá trình dự đoán thay vì chỉ một như trước đây.

Phương pháp này nhằm làm giảm nguy cơ quá trình dự đoán tạo ra một câu mô tả không tối ưu.

k	1	5	10	15	20
B-1	65.1	64.7	63.9	63.5	65.3
B-2	46.4	46.7	45.8	45.3	45.1
B-3	32.1	33.8	33.0	32.5	32.3
B-4	22.4	24.9	24.2	23.9	23.7

Bảng 5.1: Thực nghiệm đánh giá kết quả phương pháp Beam Search trên tập MSCOCO

Với $k = 5$ cho thấy kết quả tốt nhất trong các điểm BLEU.

5.1.3 Kết quả khi tạo mô tả cho toàn bộ tấm ảnh:

Bây giờ đánh giá khả năng của mô hình LSTM để mô tả hình ảnh và vùng. Đầu tiên huấn luyện LSTM để tạo ra các câu trên hình ảnh đầy đủ với mục đích xác minh rằng mô hình này đủ phong phú để hỗ trợ ánh xạ từ dữ liệu hình ảnh đến chuỗi các từ. Từ các thực nghiệm hình ảnh đầy đủ này, sử dụng framework VGGNet mạnh hơn. BLEU được tính bằng code coco-caption. Phương pháp đánh giá “một câu ứng viên” bằng cách đo sự phù hợp của chúng với một bộ “năm câu tham khảo” được viết bởi con người.



Hình 5.1: Kết quả khi tạo mô tả cho toàn bộ tấm ảnh

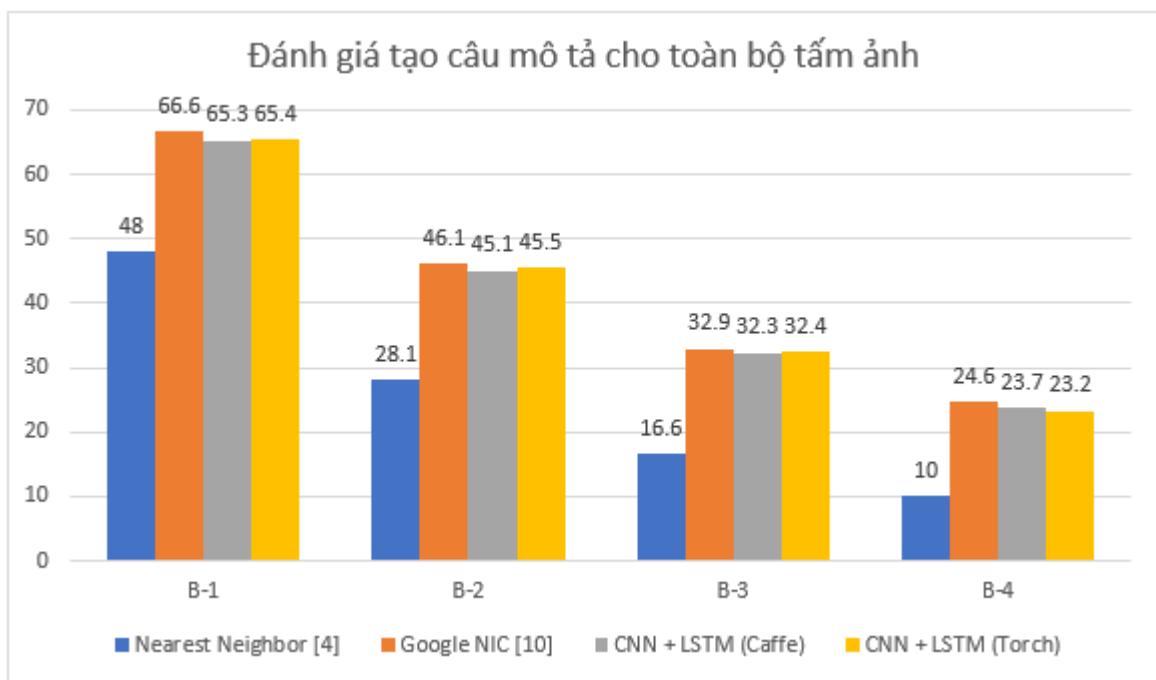
Mô hình tạo ra các mô tả hình ảnh hợp lý. Dự đoán đầu tiên "man in black shirt is playing a guitar" không xuất hiện trong tập huấn luyện. Tuy nhiên, có 20 lần xuất hiện của "man in black shirt" và 60 lần xuất hiện của "is paying guitar", mà mô hình có thể đã gộp để mô tả hình ảnh đầu tiên. Nói chung, một phần tương đối lớn các câu được tạo ra (60%) có thể được tìm thấy trong dữ liệu huấn luyện.

LSTM hoạt động tốt hơn Nearest Neighbor [4]. Ở đây, chú thích mỗi hình ảnh thử nghiệm với một câu hình ảnh huấn luyện tương tự nhất được xác định bởi chỉ tiêu L2 qua tính năng VGGNet fc7. Bảng sau cho thấy LSTM tốt hơn phương pháp truy xuất này. Thậm chí với 113.000 bức ảnh trong bộ MSCOCO phương pháp Nearest Neighbor là không đầy đủ. Ngoài ra, LSTM chỉ mất một phần giây để đánh giá mỗi hình ảnh.

Bảng sau đánh giá dự đoán hình ảnh đầy đủ trên 1.000 hình ảnh thử nghiệm. B-n là điểm BLEU sử dụng đến n-gram. Kết quả càng cao càng tốt.

	Flickr8k				Flickr30k				MSCOCO			
	B-1	B-2	B-3	B-4	B-1	B-2	B-3	B-4	B-1	B-2	B-3	B-4
Nearest Neighbor [4]	-	-	-	-	-	-	-	-	48.0	28.1	16.6	10.0
Mao [8]	58	28	23	-	55	24	20	-	-	-	-	-
Google NIC [10]	63	41	24	-	66.3	42.3	27.7	18.3	66.6	46.1	32.9	24.6
CNN + LSTM (Caffe)	55.3	35.2	21.7	11.7	59.3	37.8	24.1	15.1	65.3	45.1	32.3	23.7
CNN + LSTM (Torch)	-	-	-	-	-	-	-	-	65.4	45.5	32.4	23.2

Bảng 5.2: Đánh giá tạo câu mô tả cho toàn bộ tấm ảnh



Hình 5.2: Biểu đồ đánh giá tạo câu mô tả cho toàn bộ tấm ảnh

5.2 Kết quả thực nghiệm tạo câu chú thích cho ảnh



DESCRIPTION

a group of people walking down a street



DESCRIPTION

a man is talking on a cell phone

Hình 5.3: Ảnh có câu chú thích phù hợp



DESCRIPTION

a group of people standing on top of a sandy beach



DESCRIPTION

a man sitting in front of a laptop computer

Hình 5.4: Ảnh có câu chú thích phù hợp



DESCRIPTION

a man riding a wave on top of a surfboard



DESCRIPTION

a man sitting on a bench in front of a building

Hình 5.5: Ảnh có câu chú thích chưa phù hợp



DESCRIPTION

a large clock tower towering over a city



DESCRIPTION

a group of people riding bikes down a street

Hình 5.6: Ảnh có câu chú thích chưa phù hợp

5.3 Demo

Demo chạy với thời gian thực, có thể mô tả lại sự vật hiện tượng ngay tại thời điểm quay.



Hình 5.7: Demo chạy với thời gian thực

Chương 6

ỨNG DỤNG THỰC TẾ

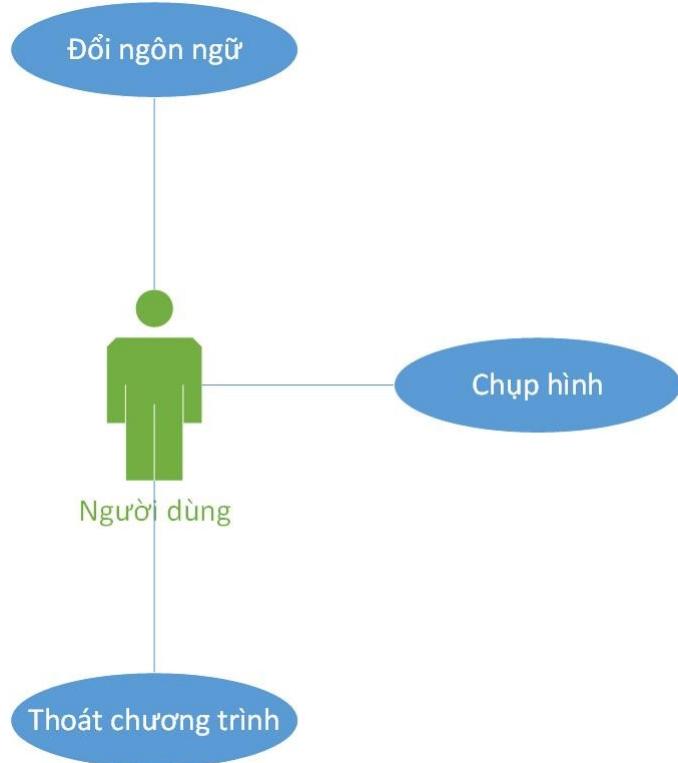
6.1 Ứng dụng Camera Vision

6.1.1 Giới thiệu

Với tốc độ phát triển hiện nay của khoa học máy tính, một trong những điều cấp thiết nhất là đưa nghiên cứu trở thành ứng dụng phục vụ vào đời sống con người. Trong đề tài nghiên cứu ở trên, nhóm đã thực nghiệm và đạt được kết quả khả quan. Vì vậy nhóm ứng dụng đề tài nghiên cứu vào hệ điều hành android để phát triển ứng dụng Camera Vision. Đây là ứng dụng hỗ trợ người khiếm thị nhận biết được những sự vật, con người ở phía trước trong thời gian thực, ứng dụng dựa trên camera của điện thoại để thu hình ảnh và đưa về web server để xử lý, sau khi xử lý xong thì ứng dụng sẽ đọc câu mô tả hình ảnh vừa mới gửi cho người dùng biết. Ứng dụng hỗ trợ thao tác qua nút bấm trên thiết bị đính kèm theo điện thoại và hỗ trợ hai ngôn ngữ Anh và Việt.

6.1.2 Sơ đồ use case và phân tích use case

6.1.2.1 Sơ đồ use case



Hình 6.1: Sơ đồ use case của ứng dụng Camera Vision

6.1.2.2 ĐẶC TẢ ACTOR

STT	Mã	Tên	Mô tả
1	Actor1	Người dùng	Là người sử dụng chương trình

Bảng 6.1: Đặc tả actor

6.1.2.3 Đặc tả use case

STT	Mã	Tên	Mô tả
1	UC01	Chụp hình	Cho phép actor chụp ảnh phía trước
2	UC02	Đổi ngôn ngữ	Cho phép actor đổi ngôn ngữ
3	UC03	Thoát chương trình	Cho phép actor thoát ứng dụng

Bảng 6.2: Đặc tả use case

6.1.2.4 Phân tích use case

UC01 - Chụp hình

Use case	Chụp hình
Actor	Người dùng
Brief description	Người dùng dùng nút trên kính để chụp hình vào smart phone
Main flow	<ol style="list-style-type: none"> 1. Người dùng nhấn nút trên kính. 2. Ứng dụng sẽ gửi hình ảnh lên server. 3. Ứng dụng sẽ đọc câu mô tả ảnh.
Alternative flow	Nếu ứng dụng truy vấn lỗi sẽ thông báo sai cho người dùng.
Pre-condition	
Special requirements	

Bảng 6.3: Phân tích use case chụp hình

UC02 - Đổi ngôn ngữ

Use case	Đổi ngôn ngữ
Actor	Người dùng
Brief description	Người dùng dùng 2 lần liên tiếp nút trên kính để thay đổi ngôn ngữ.
Main flow	<ol style="list-style-type: none"> 1. Người dùng nhấn 2 lần nút trên kính. 2. Ứng dụng sẽ đọc thông báo về ngôn ngữ hiện tại.
Alternative flow	
Pre-condition	
Special requirements	

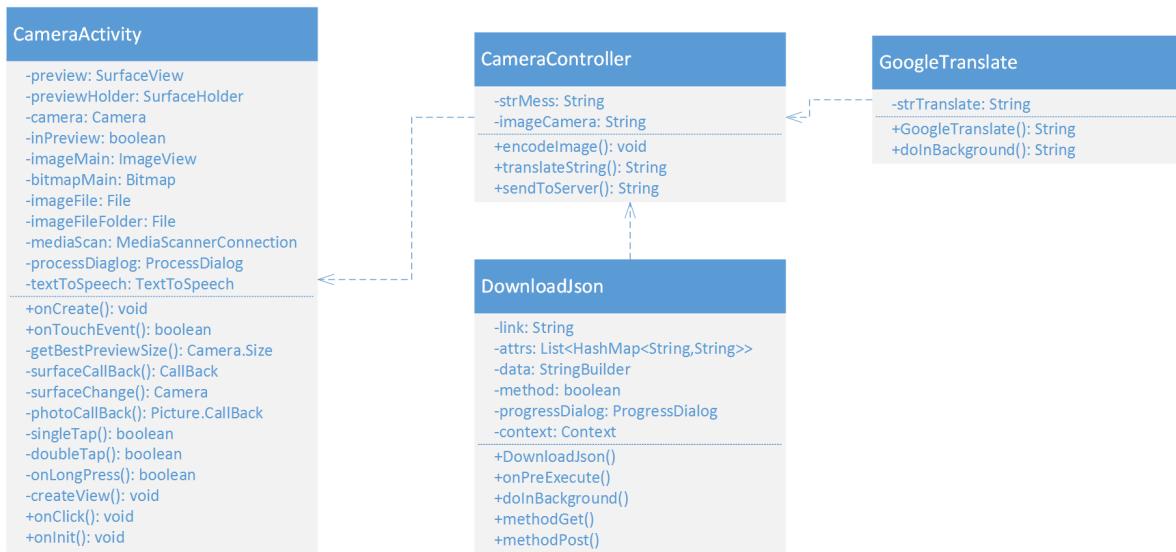
Bảng 6.4: Phân tích use case đổi ngôn ngữ

UC03 - Thoát ứng dụng

Use case	Thoát ứng dụng
Actor	Người dùng
Brief description	Người dùng nhấn giữ nút trên kính.
Main flow	<ol style="list-style-type: none"> 1. Người dùng nhấn giữ nút trên kính. 2. Ứng dụng sẽ đọc thông báo và thoát ứng dụng.
Alternative flow	
Pre-condition	
Special requirements	

Bảng 6.5: Phân tích use case thoát ứng dụng

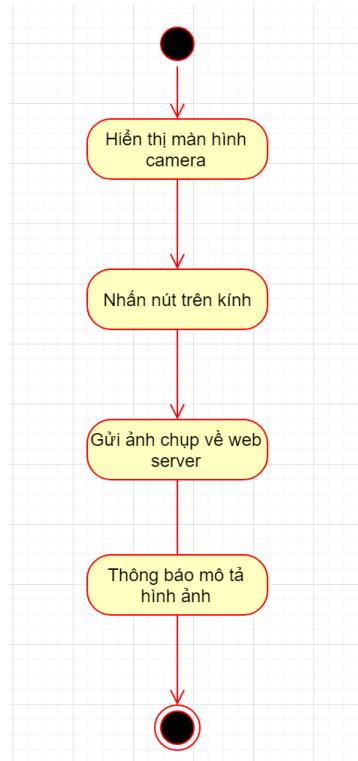
6.1.3 Sơ đồ lớp



Hình 6.2: Sơ đồ lớp

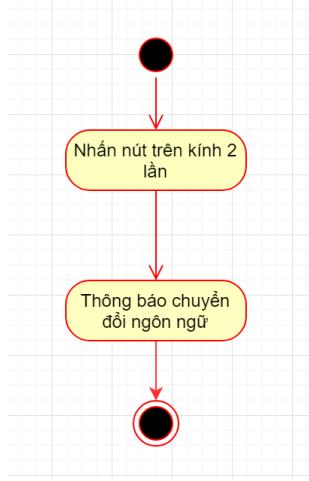
6.1.4 Sơ đồ hoạt động

UC01 - Chụp hình



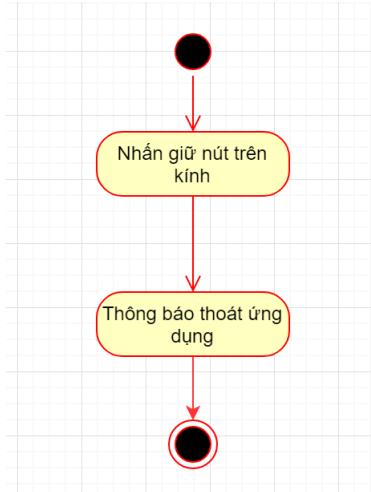
Hình 6.3: Sơ đồ hoạt động UC01

UC02 - Chuyển đổi ngôn ngữ



Hình 6.4: Sơ đồ hoạt động UC02

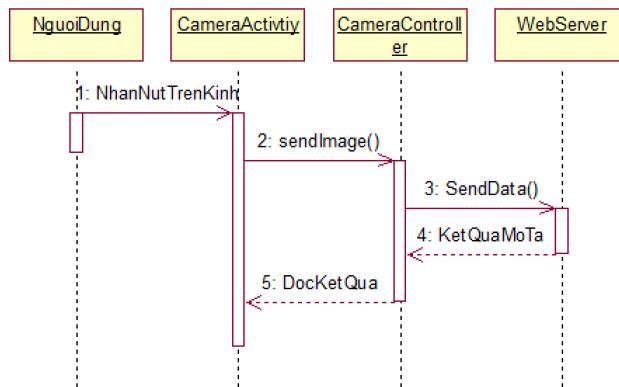
UC03 - Thoát ứng dụng



Hình 6.5: Sơ đồ hoạt động UC03

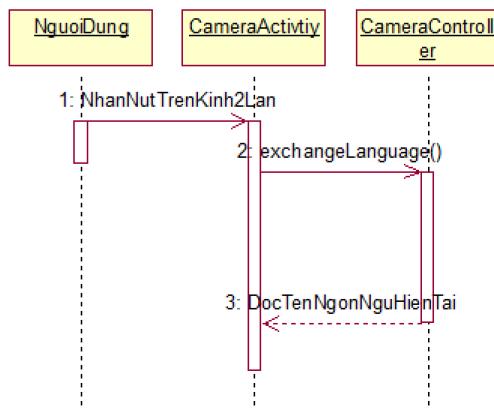
6.1.5 Sơ đồ tuần tự

UC01 - Chụp hình



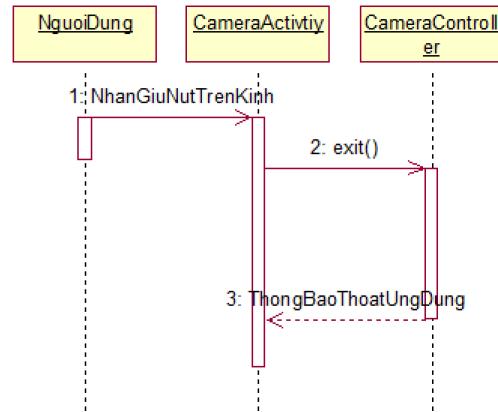
Hình 6.6: Sơ đồ tuần tự UC01

UC02 - Thay đổi ngôn ngữ



Hình 6.7: Sơ đồ tuần tự UC02

UC03 - Thoát ứng dụng



Hình 6.8: Sơ đồ tuần tự UC03

6.2 Ứng dụng Web Vision

6.2.1 Giới thiệu

Với tốc độ phát triển hiện nay của khoa học máy tính, một trong những điều cấp thiết nhất là đưa nghiên cứu trở thành ứng dụng phục vụ vào đời sống con người. Trong đề tài nghiên cứu ở trên, nhóm đã thực nghiệm và đạt được kết quả khả quan. Vì vậy nhóm ứng dụng đề tài nghiên cứu vào web để phát triển ứng dụng Web Vision. Đây là trang web mang tính học thuật cho trẻ em và tính đóng góp dữ liệu để phát triển thêm cơ sở dữ liệu sau này. Trang web sẽ nhận hình ảnh được đăng tải lên hoặc chụp từ camera điện thoại để tạo ra câu mô tả cho hình ảnh.

6.2.2 Sơ đồ use case và phân tích use case

6.2.2.1 Sơ đồ use case



Hình 6.9: Sơ đồ use case của Web Vision

6.2.2.2 ĐẶC TẢ ACTOR

STT	Mã	Tên	Mô tả
1	Actor1	Người dùng	Là người sử dụng chương trình

Bảng 6.6: Đặc tả actor

6.2.2.3 ĐẶC TẢ USE CASE

STT	Mã	Tên	Mô tả
1	UC01	Gửi hình ảnh	Cho phép actor gửi hình ảnh

Bảng 6.7: Đặc tả use case

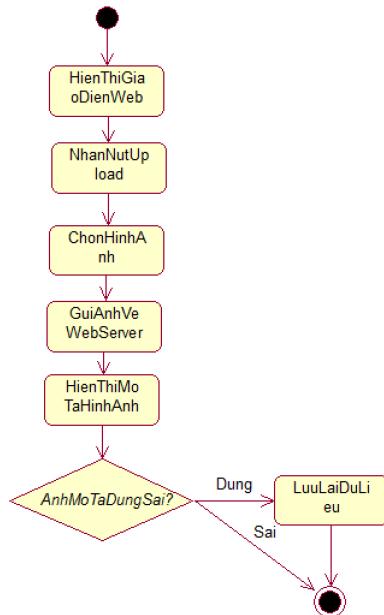
6.2.2.4 Phân tích use case

UC01 - Gửi hình ảnh

Use case	Gửi hình ảnh
Actor	Người dùng
Brief description	Người dùng gửi hình ảnh từ máy hoặc camera điện thoại để tạo mô tả hình ảnh
Main flow	<ol style="list-style-type: none"> 1. Người dùng ấn nút upload. 2. Ứng dụng sẽ gửi hình ảnh lên server. 3. Ứng dụng sẽ hiển thị câu mô tả của hình ảnh.
Alternative flow	
Pre-condition	
Special requirements	

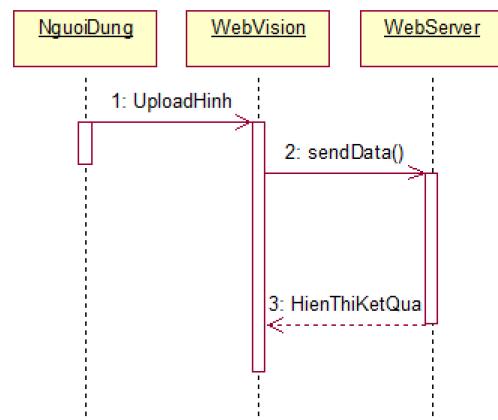
Bảng 6.8: Phân tích use case gửi hình ảnh

6.2.3 Sơ đồ hoạt động



Hình 6.10: Sơ đồ hoạt động của UC01

6.2.4 Sơ đồ tuần tự



Hình 6.11: Sơ đồ tuần tự của UC01

Chương 7

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1 Kết luận

Sau thời gian tìm hiểu đề tài “XÂY DỰNG HỆ THỐNG CHÚ THÍCH ẢNH TỰ ĐỘNG”, với mục tiêu huấn luyện được hệ thống thị giác máy tính mô tả nội dung ảnh, luận văn đã thực hiện đề tài với kết quả đã xây dựng ứng dụng có thể xuất ra đoạn mô tả cho 1 tấm ảnh đầu vào. Luận văn đã đóng góp:

- Ứng dụng có thể mô tả nội dung ảnh trong thời gian thực, có thể xuất ra giọng nói ở 2 loại ngôn ngữ là Tiếng Việt hoặc Tiếng Anh.
- Tìm hiểu nắm được ý tưởng cũng như cách rút trích đặc trưng đổi tượng cơ bản của thuật toán CNN.
- Tìm hiểu nắm được ý tưởng và cách xử lý ngôn ngữ tự nhiên với thuật toán RNN, LSTM.
- Phương pháp nhận diện mô tả có độ chính xác khá cao, bằng cách áp dụng giải thuật rút trích hình ảnh, xử lý ngôn ngữ như CNN, LSTM.

Bên cạnh đó, đề tài còn nhiều hạn chế:

- Vì thuật toán chưa được tối ưu và cải tiến nên kết quả thu được bên cạnh những trường hợp chính xác vẫn còn nhiều trường hợp sai sót, như nhận diện mô tả sai hoặc thiếu sót, tỉ lệ sai vẫn còn khá cao.
- Để cho mô hình được chính xác và hiệu quả thì phụ thuộc rất nhiều vào tập dữ liệu huấn luyện. Tập dữ liệu này phải thu thập đủ lớn và khách quan, lượng từ vựng cũng như các sự vật/con người/hiện tượng phải đa dạng, phong phú. Ngoài ra, việc nhận diện biển báo còn bị ảnh hưởng rất nhiều bởi chất lượng ảnh, vì vậy cần phải hạn chế tối đa độ nhòe/mờ để có thể thu được kết quả ít sai sót nhất.

7.2 Hướng phát triển

Hướng phát triển của đề tài này là phải khắc phục được các hạn chế nêu ở trên, cụ thể như sau:

- Huấn luyện với một tập mẫu lớn hơn để nâng cao khả năng nhận dạng mô tả đối tượng của chương trình, tăng tỉ lệ chính xác.
- Khắc phục được sai sót do ảnh hưởng từ điều kiện vốn từ và chất lượng ảnh kém.
- Đề xuất thêm một thành phần dịch máy để có thể chuyển đổi qua nhiều ngôn ngữ.

TÀI LIỆU THAM KHẢO

Cơ sở dữ liệu

1. Flickr8k - <http://nlp.cs.illinois.edu/HockenmaierGroup>
2. Flickr30k - <http://shannon.cs.illinois.edu/DenotationGraph>
3. MSCOCO – <http://mscoco.org>

Tiếng Anh

4. “Deep Visual-Semantic Alignments for Generating Image Descriptions” by Andrej Karpathy and Li Fei-Fei (2015).
5. “Rich feature hierarchies for accurate object detection and semantic segmentation” by Ross Girshick, Jeff Donahue, Trevor Darrell and Jitendra Malik (2014).
6. “Deep Fragment Embeddings for Bidirectional Image Sentence Mapping” by Andrej Karpathy, Li Fei-Fei and Armand Joulin (2014).
7. “Unifying visual-semantic embeddings with multimodal neural language models” by R. Kiros, R. Salakhutdinov and R. S. Zemel (2014).
8. “Explain images with multimodal recurrent neural networks” by J. Mao, W. Xu, Y. Yang, J. Wang and A. L. Yuille (2014).
9. “Evaluation Metrics” by Jaime Arguello (2013).
10. “Show and tell: A neural image caption generator” by O. Vinyals, A. Toshev, S. Bengio, and D. Erhan (2014).
11. "Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge" by Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2016).
12. VGG-Net 16 layer - http://www.robots.ox.ac.uk/~vgg/research/very_deep/
13. ImageNet ILSVRC 2014 - <http://www.image-net.org/challenges/LSVRC/2014/>

14. "Discriminatively trained deformable part models" release 5 by R. Girshick, P. Felzenszwalb, and D. McAllester (2008).
15. "Selective search for object recognition" by J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. S (2013).
16. "Regionlets for generic object detection" by X. Wang, M. Yang, S. Zhu, and Y. Lin (2013).
17. "Bottom-up segmentation for top-down detection" by S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun (2013).