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

BÁO CÁO GIỮA KỲ
HỌC SÂU

Người hướng dẫn: **PGS. LÊ ANH CƯỜNG**

Người thực hiện: **LƯU HỮU TRÍ – 52200167**

NGUYỄN QUANG TRUNG – 52200193

HỒ THU YẾN NGỌC – 52200149

Lớp **: 22050301**

Khoá **: 26**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành đến PGS. Lê Anh Cường vì đã giúp đỡ chúng em trong quá trình làm bài đồ án vừa qua. Nhờ sự hướng dẫn và giải đáp của thầy, chúng em đã có thể hoàn thành bài đồ án của mình một cách tốt nhất có thể.

Thầy đã dành thời gian và tâm huyết để hướng dẫn chúng em từng bước, giải thích những khái niệm khó hiểu, và chỉ ra những lỗi sai mà chúng tôi mắc phải trong quá trình viết bài. Những lời khuyên và sự hỗ trợ của thầy đã giúp chúng em tự tin hơn trong việc hoàn thiện bài đồ án của mình.

Chúng em rất biết ơn sự giúp đỡ của thầy và mong rằng sẽ có nhiều học trò khác được thầy hướng dẫn và giúp đỡ như chúng em. Chúng em hy vọng sẽ có cơ hội được học hỏi thêm từ thầy trong tương lai.

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng chúng tôi và được sự hướng dẫn khoa học của TS. Trần Lương Quốc Đại. 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 Dự án 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 tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 7 tháng 3 năm 2025

Tác giả

(Ký tên và ghi rõ họ tên)

Lưu Hữu Trí

Nguyễn Quang Trung

Hồ Thu Yến Ngọc

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

Phần xác nhận của GV hướng dẫn

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

Phần đánh giá của GV chấm bài

_____

_____

_____

_____

_____

_____

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Chương 1: Cơ sở lý thuyết, chương này trình bày các khái niệm nền tảng về Visual Question Answering (VQA), bao gồm CNN để trích xuất đặc trưng từ ảnh và LSTM để xử lý câu hỏi. Báo cáo cũng đề cập đến cơ chế Attention, giúp mô hình tập trung vào vùng ảnh quan trọng, và kỹ thuật Multimodal, kết hợp CNN và LSTM để giải quyết bài toán VQA.

Chương 2: Thực nghiệm, báo cáo tiến hành thực nghiệm trên một bộ dữ liệu VQA với bốn mô hình: MobileNet_v2 + LSTM và CNN tổng quát + LSTM, mỗi mô hình được thử nghiệm với và không có Attention. Kết quả được so sánh để đánh giá ảnh hưởng của Attention và lựa chọn mô hình tối ưu.

MỤC LỤC

DANH MỤC HÌNH ẢNH

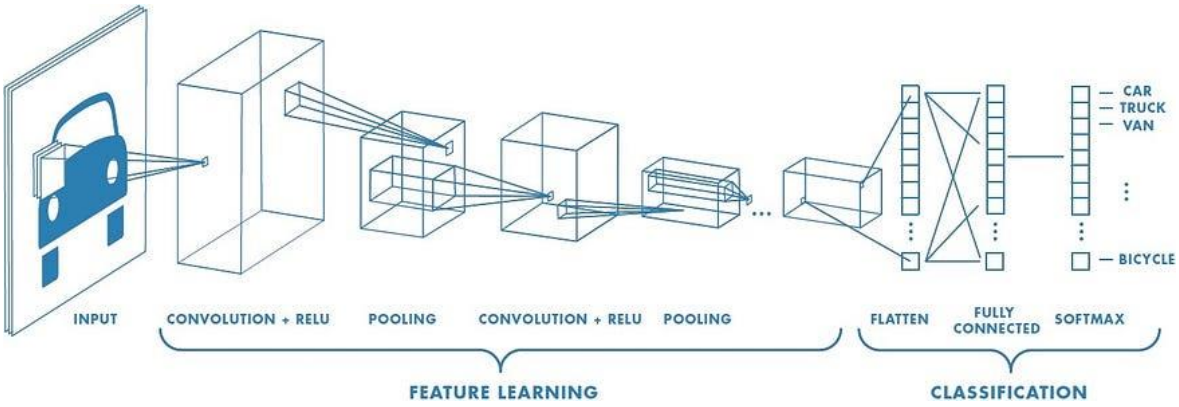
DANH MỤC BẢNG

CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

1.1 Convolutional Neural Network (CNN)

1.1.1 Giới thiệu chung

Convolutional Neural Network (CNN), hay mạng nơ-ron tích chập, là một kiến trúc mạng nơ-ron chuyên dụng trong xử lý dữ liệu dạng lưới, điển hình là hình ảnh và video. CNN lấy cảm hứng từ cơ chế hoạt động của thị giác con người, nơi các tế bào thần kinh trong não xử lý thông tin thị giác theo từng khu vực nhỏ, sau đó tổng hợp để nhận diện toàn bộ đối tượng.



Hình 1.1 Ví dụ về một kiến trúc của mạng thần kinh tích chập

CNN được cấu trúc từ ba loại lớp chính: lớp tích chập (convolutional layer), lớp kích hoạt (activation layer) và lớp gộp (pooling layer). Lớp tích chập đóng vai trò chính, thực hiện việc trích xuất các đặc trưng từ dữ liệu đầu vào bằng cách áp dụng các bộ lọc (filters hay kernels) trên từng phần nhỏ của dữ liệu. Mỗi bộ lọc có khả năng phát hiện các đặc trưng cụ thể, chẳng hạn như cạnh, góc, hoặc các hoa văn phức tạp hơn trong ảnh. Lớp kích hoạt, phổ biến nhất là hàm ReLU (Rectified Linear Unit), giúp tăng tính phi tuyến cho mạng, đảm bảo mô hình có thể học các mối quan hệ phức tạp. Lớp gộp có nhiệm vụ giảm kích thước dữ liệu, làm giảm số lượng tham số cần học, từ đó tăng hiệu quả tính toán và giảm nguy cơ quá khớp (overfitting).

CNN thường kết thúc với một hoặc nhiều lớp kết nối đầy đủ (fully connected layer), nơi toàn bộ thông tin đặc trưng đã trích xuất được hợp nhất để thực hiện các tác vụ cuối cùng như phân loại hình ảnh hoặc dự đoán.

Ưu điểm lớn nhất của CNN nằm ở khả năng tự động trích xuất các đặc trưng từ dữ liệu thô. Nhờ khả năng này, CNN đã trở thành lựa chọn hàng đầu trong nhiều bài toán xử lý hình ảnh như nhận diện khuôn mặt, phát hiện vật thể, và xử lý ảnh y tế. Ngoài ra, CNN cũng được áp dụng trong xử lý ngôn ngữ tự nhiên (NLP), phân tích chuỗi thời gian, và các lĩnh vực khác nơi dữ liệu có thể được biểu diễn dưới dạng lưới.

Tuy nhiên, thách thức lớn nhất là yêu cầu lượng dữ liệu lớn và sức mạnh tính toán cao để đạt hiệu quả tốt. Dù vậy, với sự hỗ trợ của phần cứng mạnh mẽ và các thư viện phần mềm hiện đại, CNN đã trở thành công cụ không thể thiếu trong các ứng dụng AI hiện nay.

1.1.2 Các lớp trong CNN

1.1.2.1 Lớp tích chập (Convolutional layer)

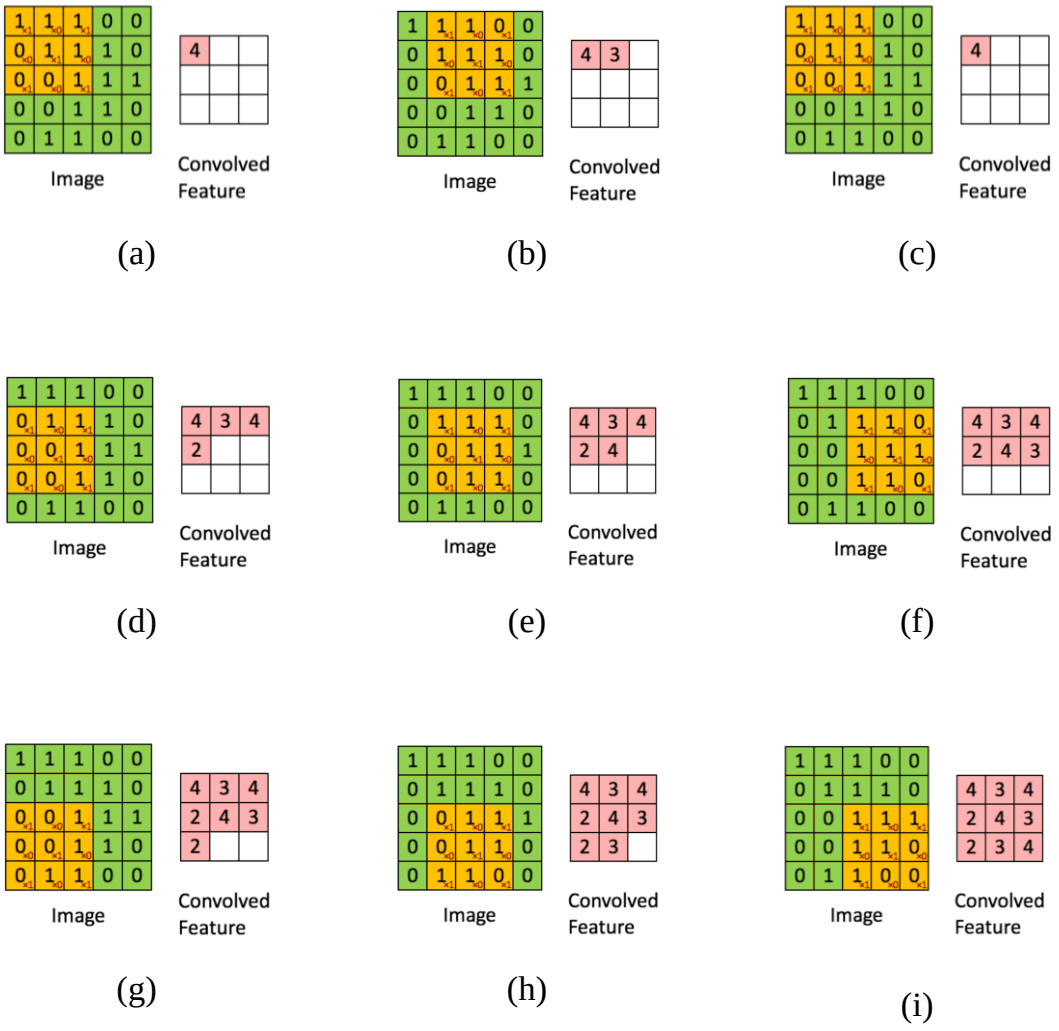
Phép tích chập (Convolution)

Tích chập (convolution) là một phép toán toán học cốt lõi trong các mạng nơ-ron tích chập (Convolutional Neural Networks - CNN), giúp trích xuất các đặc trưng từ dữ liệu đầu vào, chẳng hạn như ảnh. Nó hoạt động bằng cách áp dụng các bộ lọc (filters) hay kernels trên dữ liệu trượt qua ảnh để phát hiện các đặc điểm như cạnh, góc hoặc hoa văn phức tạp.

Quá trình tích chập diễn ra bằng cách trượt kernel (thường có kích thước 3x3 hoặc 5x5) qua dữ liệu đầu vào. Tại mỗi vị trí, bộ lọc thực hiện phép nhân từng phần tử (element-wise multiplication) giữa các giá trị trong bộ lọc và vùng tương ứng của đầu vào, sau đó cộng các giá trị đó lại để tạo thành một giá trị duy nhất. Quá trình này được lặp lại cho toàn bộ dữ liệu đầu vào, và kết quả cuối cùng là một bản đồ đặc trưng (feature map).

Giả sử cho một ma trận ảnh xám đầu vào X với kích thước $H \times W$, ta định nghĩa kernel W:

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$



Hình 1.2 Hoạt động của phép tích chập

Trong ví dụ trên, kernel W được áp lên một vùng con của ma trận đầu vào như hình 1.2a. Gọi F là bản đồ đặc trưng của kết quả tích chập, kết quả tích chập tại F_{11} được tính như sau:

$$F_{11} = (1 \times X_{11}) + (0 \times X_{12}) + (1 \times X_{13}) + (0 \times X_{21}) + (1 \times X_{22}) + (1 \times X_{23}) + (1 \times X_{31}) + (0 \times X_{32}) + (1 \times X_{33})$$

$$= 1.1 + 0.1 + 1.1 + 0.0 + 1.1 + 1.1 + 1.0 + 0.0 + 1.1 = 4.$$

Kernel tiếp tục trượt trên ma trận đầu vào, đến khi feature map được tính toán xong (hình 1.2i). Quá trình này lặp lại cho từng vùng con khác nhau trong ma trận đầu vào. Tại mỗi bước, Kernel sẽ thực hiện phép nhân từng phần tử tương ứng với

vùng con hiện tại, sau đó cộng tất cả các kết quả lại để thu được một giá trị duy nhất. Giá trị này sẽ được ghi vào vị trí tương ứng trên bản đồ đặc trưng.

Padding (đệm)

Khi thực hiện phép tích chập trên một ma trận, bộ lọc chỉ có thể áp dụng trên các vùng nằm hoàn toàn bên trong ma trận đầu vào. Điều này có nghĩa là các giá trị ở viền hoặc góc của ma trận sẽ ít được tham chiếu hơn trong các phép tính, dẫn đến việc mất mát thông tin. Trong ví dụ, bộ lọc 3×3 được áp dụng trên ma trận 5×5, vì không thể vượt ra khỏi giới hạn ma trận để xử lí các phần tử biên, nên kích thước của ma trận đặc trưng bị giảm xuống còn 3×3 và một phần thông tin đã bị bỏ qua.

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

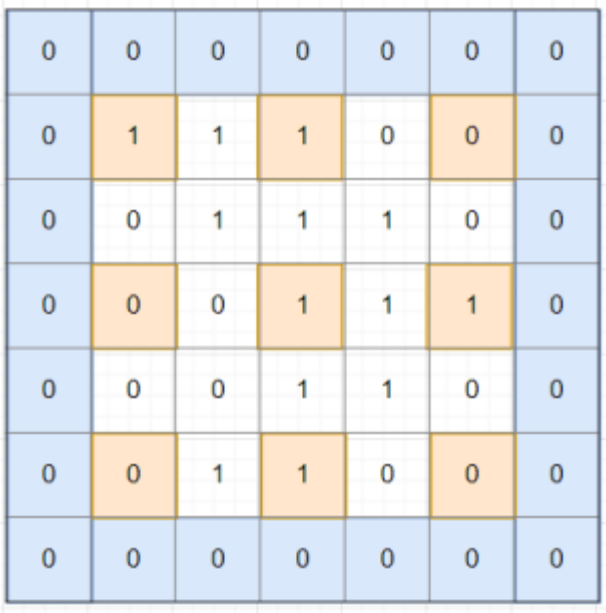
Hình 1.3 Ma trận được mở rộng bằng lớp đệm có giá trị 0.

Để giải quyết vấn đề này, ta thêm một viền giá trị xung quanh ma trận gốc. Lớp viền này thường có giá trị là 0, đóng vai trò như một vùng đệm, cho phép áp dụng bộ lọc áp dụng tích chập vào biên của ma trận gốc. Nhờ đó, dữ liệu từ ma trận gốc được bảo toàn.

Stride (bước nhảy)

Khi thực hiện tích chập trên các ma trận có kích thước lớn, việc áp dụng bộ lọc qua từ phần tử trong ma trận, số lượng phép toán cần thực hiện là rất lớn. Để giảm

số lượng phép toán, ta cho bộ lọc nhảy qua một khoảng cách lớn hơn, thay vì di chuyển qua từng phần tử.



Hình 1.4 Các bước nhảy trên ma trận với $stride = 2$ và $padding = 1$.

Số dòng và sốt cột di chuyển qua mỗi lần nhảy được gọi là *stride (sải bước)*. Khi bộ lọc di chuyển qua từng phần tử, giá trị của $stride = 1$. Tương tư, khi $stride = 2$, bộ lọc sẽ di chuyển qua 2 phần từ mỗi trong mỗi hàng và di chuyển qua 2 hàng trong ma trận gốc.

Lớp tích chập

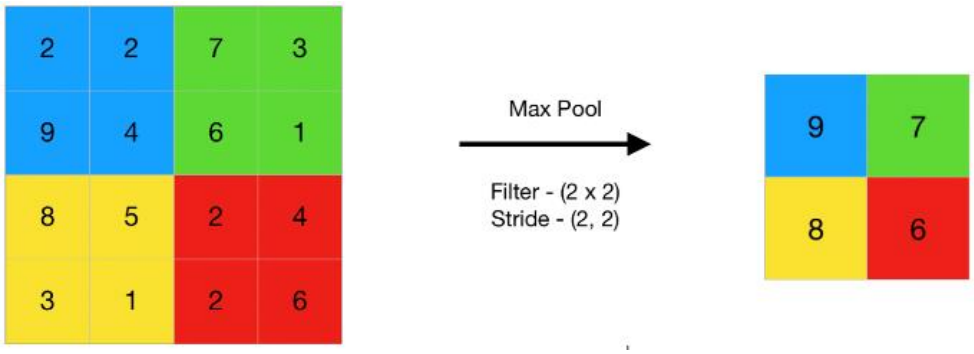
Lớp tích chập, là thành phần chính trong kiến trúc của CNN, có nhiệm vụ trích xuất các đặc trưng quan trọng từ dữ liệu đầu vào. Các đặc trưng này có thể là cạnh, kết cấu của dữ liệu đầu vào. Mỗi phần đặc trưng mà bộ lọc phát hiện được được lưu trữ trong *feature map (bản đồ đặc trưng)*. Đây cũng là kết quả của lớp tích chập và được truyền cho lớp tiếp theo. Kết quả của feature map phụ thuộc vào padding, stride, kích thước và số lượng của bộ lọc được sử dụng trong lớp này.

1.1.2.2 Lớp gộp (Pooling layer)

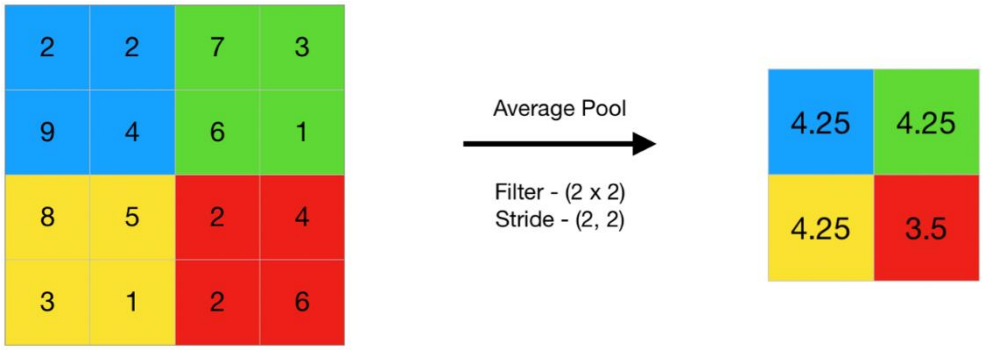
Lớp gộp là một thành phần quan trọng trong kiến trúc mạng nơ-ron tích chập, giúp giảm kích thước của dữ liệu sau mỗi bước tích chập, làm cho mô hình hiệu quả hơn trong việc tính toán và dễ dàng học được các đặc trưng từ dữ liệu đầu vào mà

không bị overfitting. Hơn nữa, lớp gộp còn giúp giảm số chiều của dữ liệu đầu vào, giúp mô hình tập trung vào các đặc trưng quan trọng thay vì các chi tiết nhỏ. Đồng thời, cải thiện hiệu suất tính toán của mạng.

Tại lớp này, ta chia dữ liệu đầu vào thành các vùng nhỏ bằng cách sử dụng các bộ lọc kết hợp với stride. Từ các vùng này, ta thực hiện các phép tính toán để xây dựng một bản đồ đặc trưng mới.



Hình 1.5 Minh họa Max Pooling.



Hình 1.6 Minh hoạt Average Pooling

Trong các mạng tích chập, có hai phương pháp pooling phổ biến là max pooling và average pooling. Max pooling là phương pháp được sử dụng rộng rãi nhất, trong đó tại mỗi vùng con của bản đồ đặc trưng, ta chỉ lấy giá trị lớn nhất trong vùng đó. Điều này giúp giữ lại thông tin mạnh mẽ nhất từ mỗi vùng nhỏ, làm cho mô hình trở nên bất biến với các biến đổi nhỏ trong dữ liệu, chẳng hạn như sự thay đổi vị trí của đối tượng trong hình ảnh. Average pooling ngược lại, tính giá trị trung bình của

tất cả các phần tử trong mỗi vùng, giúp mô hình tập trung vào các đặc trưng tổng quát hơn và ít bị ảnh hưởng bởi các thay đổi lớn trong dữ liệu.

1.1.2.3 Lớp kích hoạt (Activation layer)

Trong mạng tích chập, lớp kích hoạt giúp thêm tính phi tuyến tính cho mô hình. Các hàm kích hoạt có nhiệm vụ chuyển đổi đầu ra của mỗi nơ-ron thành một giá trị phi tuyến. Khi một nơ-ron nhận tín hiệu đầu vào, tín hiệu này sẽ được xử lý qua hàm kích hoạt trước khi được gửi đến các nơ-ron tiếp theo. Việc sử dụng hàm kích hoạt phi tuyến giúp mạng học được các đặc trưng phức tạp và không giới hạn trong việc học các mô hình tuyến tính.

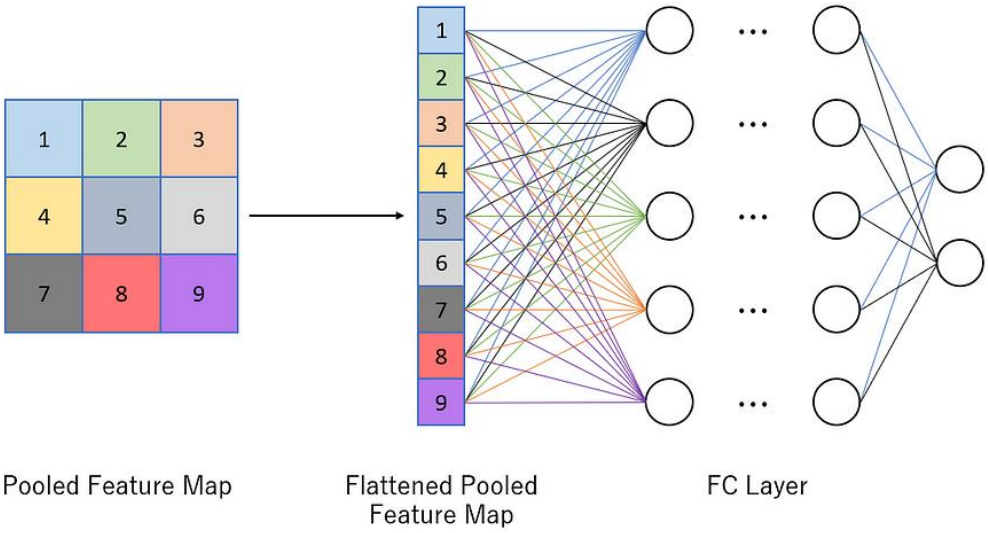
Một trong những hàm kích hoạt phổ biến nhất là ReLU (Rectified Linear Unit), với công thức đơn giản là $f(x) = \max(0, x)$, trong đó mọi giá trị âm sẽ được chuyển thành 0, trong khi giá trị dương không thay đổi. ReLU giúp mạng huấn luyện nhanh hơn và giảm hiện tượng biến mất gradient (gradient vanishing).

Ngoài ra còn có nhiều hàm kích hoạt khác như Sigmoid, Tanh và Softmax. Sigmoid chuyển đổi đầu vào về giá trị trong khoảng $[0,1]$, rất hữu ích trong phân loại nhị phân, nhưng gặp vấn đề gradient vanishing khi giá trị đầu vào quá lớn hoặc quá nhỏ, làm chậm quá trình huấn luyện. Tanh là một biến thể của sigmoid, có đầu ra trong khoảng $[-1,1]$, giúp giảm bớt vấn đề gradient vanishing nhưng vẫn có thể gặp khó khăn khi đầu vào quá lớn. Softmax là hàm kích hoạt được sử dụng trong lớp đầu ra của các bài toán phân loại đa lớp. Nó chuyển đổi các giá trị đầu ra thành xác suất cho mỗi lớp, sao cho tổng xác suất của tất cả các lớp bằng 1, giúp mô hình chọn lớp có xác suất cao nhất.

1.1.2.4 Lớp kết nối đầy đủ (Fully connected layer)

Lớp kết nối đầy đủ, thường được gọi là "fully connected layer" hoặc "dense layer" trong mạng nơ-ron tích chập (CNN), là một thành phần quan trọng và phổ biến trong kiến trúc mạng nơ-ron sâu. Đây là lớp mà mỗi nơ-ron trong lớp hiện tại được kết nối với mọi nơ-ron trong lớp tiếp theo. Nhiệm vụ chính của lớp kết nối đầy đủ là tổng hợp các đặc trưng đã được trích xuất từ các lớp trước đó (chẳng hạn như lớp tích

chập hoặc lớp gộp) và chuyển chúng thành một dạng đầu ra phù hợp với bài toán cụ thể, chẳng hạn như phân loại, hồi quy hoặc xác suất của từng lớp.



Hình 1.7 Lớp kết nối đầy đủ.

Lớp kết nối đầy đủ hoạt động dựa trên phép toán ma trận, trong đó đầu vào là một vector chứa các giá trị đặc trưng. Vector này được nhân với một ma trận trọng số và sau đó cộng thêm một vector bias. Kết quả thu được từ phép tính này sẽ được chuyển qua một hàm kích hoạt phi tuyến, như ReLU, sigmoid hoặc softmax, để tạo ra đầu ra mong muốn. Trong quá trình huấn luyện, các trọng số và bias của lớp được tối ưu hóa thông qua thuật toán lan truyền ngược (backpropagation), nhằm giảm giá trị của hàm mất mát và cải thiện hiệu suất của mô hình.

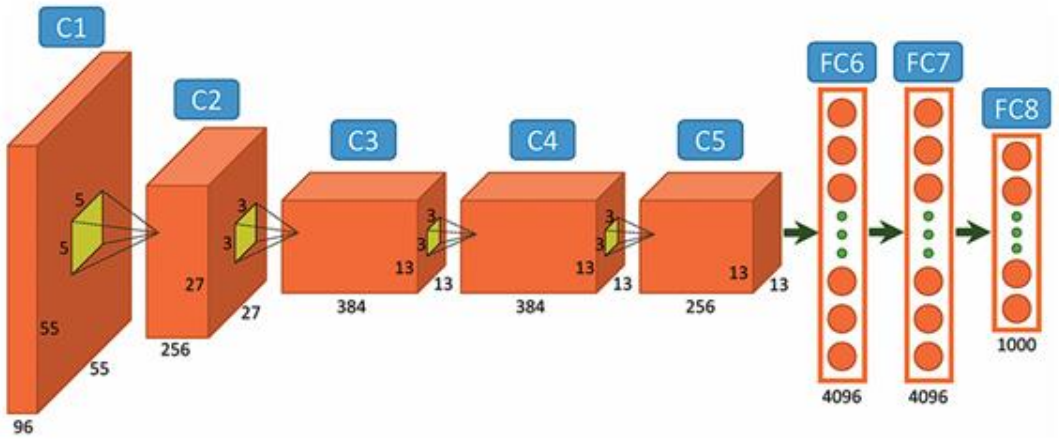
Lớp kết nối đầy đủ thường được sử dụng ở phần cuối của một mạng CNN, đặc biệt là trong các bài toán phân loại. Sau khi các lớp tích chập đã trích xuất được các đặc trưng quan trọng từ dữ liệu, lớp fully connected sẽ sử dụng những đặc trưng này để đưa ra quyết định cuối cùng. Ví dụ, trong một bài toán phân loại hình ảnh, lớp kết nối đầy đủ cuối cùng thường có số nơ-ron bằng số lượng lớp (classes) cần phân loại, và hàm softmax được sử dụng để chuyển đổi đầu ra thành xác suất cho từng lớp.

1.1.3 Một số kiến trúc CNN nổi bật

1.1.3.1 AlexNet

AlexNet được giới thiệu vào năm 2012 bởi Alex Krizhevsky, Ilya Sutskever, và Geoffrey Hinton. Đây là một trong những mạng nơ-ron tích chập (CNN) đầu tiên đạt được bước tiến lớn trong thị giác máy tính, đặc biệt khi giành chiến thắng tại cuộc thi ImageNet Large Scale Visual Recognition Challenge (ILSVRC) năm 2012.

AlexNet (hình 1.8) gồm 8 tầng, trong đó có 5 tầng tích chập và 3 tầng kết nối đầy đủ, cùng với việc sử dụng hàm kích hoạt ReLU (Rectified Linear Unit) thay thế cho Sigmoid hoặc Tanh, giúp tăng tốc quá trình huấn luyện. Một trong những điểm nổi bật của AlexNet là áp dụng kĩ thuật tăng cường dữ liệu (Data Augmentation) để giảm overfitting, như dịch chuyển hoặc lật ảnh. Ngoài ra, Dropout được sử dụng tại các tầng fully connected để hạn chế overfitting. AlexNet cũng đánh dấu bước ngoặt trong việc tận dụng GPU để tăng tốc độ huấn luyện, và nó trở thành nền tảng cho sự phát triển của các mạng CNN sau này.



Hình 1.8 Kiến trúc của AlexNet.

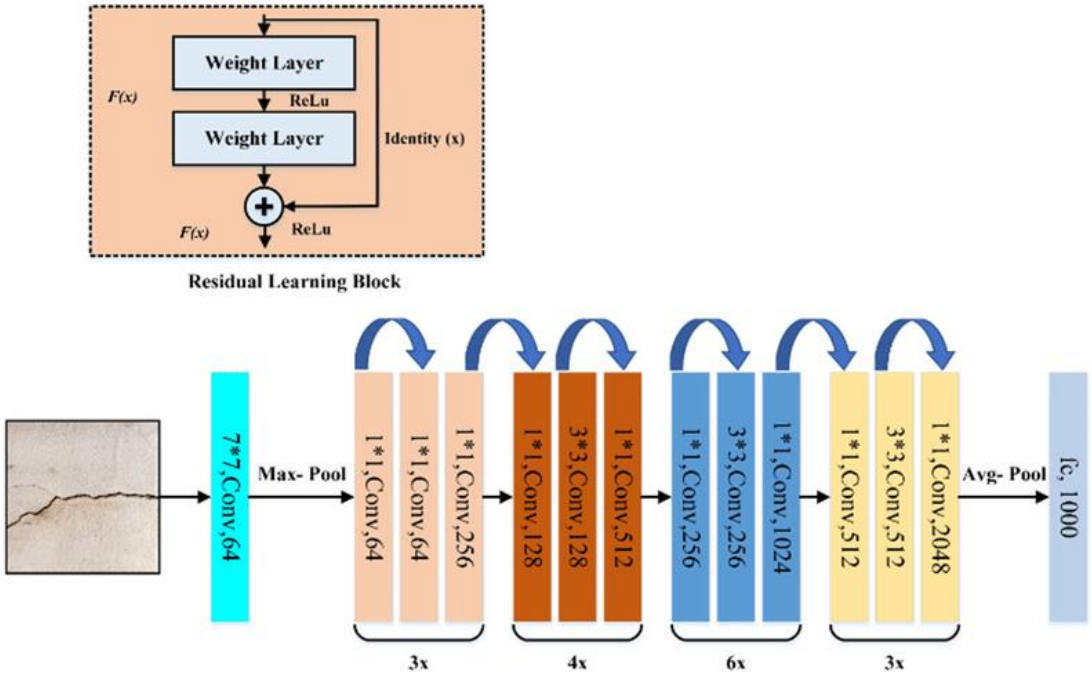
1.1.3.2 ResNet

ResNet (Residual Network) được phát triển vào năm 2015 bởi Kaiming He và các cộng sự. ResNet đã giải quyết được một trong những thách thức lớn nhất của các mạng sâu là hiện tượng mất mát gradient (gradient vanishing).

Kiến trúc của ResNet sử dụng các khối residual, trong đó có các đường tắt (skip connections) để truyền trực tiếp đầu vào qua các tầng, giúp tránh mất mát thông tin khi mạng sâu hơn. Công thức chính của khối residual là $H(x) = F(x) + x$, trong đó

H(x) là đầu ra của khối, F(x) là tín hiệu được biến đổi, và x là tín hiệu ban đầu được thêm vào thông qua đường tắt. ResNet có thể đạt độ sâu cực lớn, với các phiên bản từ 18 tầng đến hơn 152 tầng, và vẫn đảm bảo hiệu suất tốt. Phiên bản ResNet 152 đạt tỉ lệ lỗi top-5 chỉ 3.57% trên bộ dữ liệu ImageNet, một con số vượt trội tại thời điểm đó. ResNet hiện được ứng dụng rộng rãi trong nhận diện hình ảnh, xử lý video, và các lĩnh vực khác nhờ khả năng xử lý tốt của các mạng cực kỳ sâu.

Hình trên mô tả kiến trúc mạng ResNet với các khối học dư thừa (Residual Learning Block). Mỗi khối gồm hai lớp tích chập kèm hàm kích hoạt ReLU và đường truyền tắt (skip connection) giúp truyền trực tiếp đầu vào qua khối, giải quyết vấn đề gradient biến mất trong mạng sâu. Dữ liệu đầu vào được xử lý qua lớp tích chập 7x7 và Max Pooling trước khi đi qua các khối residual với kích thước filter tăng dần từ 64 đến 2048. Lớp tích chập 1x1 điều chỉnh chiều sâu kênh, còn lớp 3x3 trích xuất đặc trưng không gian. Cuối cùng, Average Pooling giảm kích thước không gian trước khi dữ liệu được đưa vào lớp Fully Connected với 1000 đầu ra, phù hợp cho bài toán phân loại hình ảnh như ImageNet.



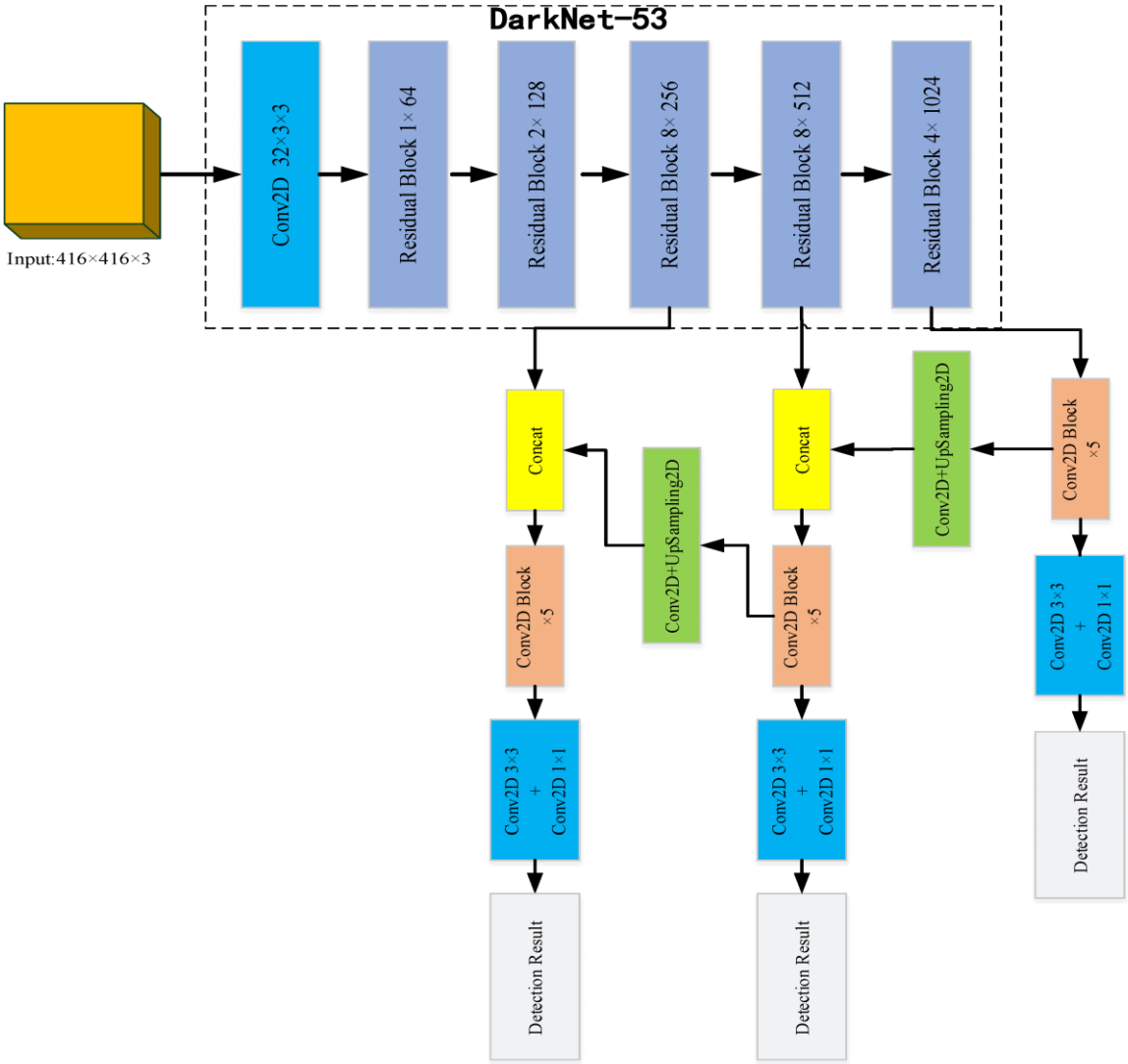
Hình 1.9 Kiến trúc của một mô hình ResNet.

1.1.3.3 YOLO (You Only Look Once)

YOLO (You Only Look Once) được phát triển vào năm 2016 bởi Joseph Redmon và nhanh chóng trở thành một trong những phương pháp nhận diện đối tượng theo thời gian thực hiệu quả nhất. Khác với các phương pháp trước đó như R-CNN hay Faster R-CNN, YOLO chỉ cần xử lý toàn bộ ảnh trong một lần tính toán duy nhất, giúp tăng đáng kể tốc độ. Phương pháp này chia ảnh thành các lưới (grid), sau đó dự đoán bounding box (hộp chứa đối tượng) và nhãn của các đối tượng trong mỗi lưới. Với tốc độ xử lý lên đến 45 khung hình mỗi giây (FPS), YOLO phù hợp cho các ứng dụng thời gian thực như xe tự hành, giám sát an ninh, và nhận diện đối tượng trong video. Qua các phiên bản cải tiến, YOLO không chỉ cải thiện về tốc độ mà còn nâng cao độ chính xác, khiến nó trở thành lựa chọn hàng đầu cho các bài toán nhận diện đối tượng.

Hình trên mô tả kiến trúc của mô hình YOLOv3, với DarkNet-53 là backbone chính đảm nhận vai trò trích xuất đặc trưng từ ảnh đầu vào. Ảnh đầu vào có kích thước 416x416x3, được đưa qua một lớp convolution ban đầu với 32 bộ lọc kích thước 3x3. Sau đó, mô hình sử dụng 5 nhóm Residual Blocks với số lượng tăng dần các bộ lọc (64, 128, 256, 512 và 1024), giúp trích xuất các đặc trưng ở nhiều mức khác nhau.

Từ tầng đặc trưng sâu nhất của DarkNet-53, các nhánh được tách ra để thực hiện phát hiện vật thể ở các tỉ lệ khác nhau. Tầng đặc trưng này được đưa qua các khối convolution và upsampling (tăng kích thước), sau đó kết hợp (concat) với các đặc trưng ở tầng trung gian từ backbone. Quá trình này diễn ra hai lần để tạo ra ba đầu ra với độ phân giải khác nhau, mỗi đầu ra đi qua một khối convolution 3x3 và 1x1 trước khi tạo ra kết quả dự đoán cuối cùng. Việc kết hợp các đặc trưng từ nhiều tầng giúp YOLOv3 có khả năng phát hiện vật thể ở các kích thước lớn, trung bình và nhỏ một cách hiệu quả.



Hình 1.10 Kiến trúc mô hình YOLOv3.

1.1.3.4 Một số kiến trúc khác

Ngoài một số kiến trúc nổi bật nói trên, còn có một số kiến trúc CNN khác đã tạo nên dấu ấn quan trọng trong lĩnh vực thị giác máy tính. Chẳng hạn như LeNet, được giới thiệu bởi Yann LeCun vào năm 1989, là một trong những kiến trúc CNN đầu tiên, chuyên dùng để nhận dạng chữ viết tay và số, với thiết kế gồm các lớp tích chập và pooling đơn giản nhưng hiệu quả. Năm 2014, VGGNet thu hút sự chú ý nhờ thiết kế sử dụng các lớp tích chập nhỏ (3x3) chồng lên nhau, cho phép tăng độ sâu mạng và cải thiện khả năng trích xuất đặc trưng, mặc dù yêu cầu tài nguyên tính toán lớn. Cùng năm, GoogleNet (Inception) được ra mắt với kiến trúc Inception module,

tập trung vào việc tối ưu hóa số lượng tham số và cải thiện hiệu năng bằng cách kết hợp nhiều kích thước kernel trong cùng một lớp. Những kiến trúc này không chỉ thúc đẩy sự phát triển của các ứng dụng thị giác máy tính mà còn đặt nền móng cho nhiều cải tiến sâu rộng trong các lĩnh vực liên quan đến học sâu.

1.2 Long-Short Term Memory (LSTM)

1.2.1 Giới thiệu

Trong các bài toán xử lý dữ liệu chuỗi như xử lý ngôn ngữ tự nhiên (NLP), dự đoán chuỗi thời gian, dịch máy và nhận diện giọng nói, Recurrent Neural Network – RNN (mạng nơ-ron hồi quy) thường được sử dụng để mô hình hóa dữ liệu có tính liên kết theo thời gian. Tuy nhiên, RNN truyền thống gặp khó khăn trong việc ghi nhớ thông tin dài hạn do hiện tượng *gradient vanishing* và *gradient exploding*. Gradient vanishing xảy ra trong quá trình lan truyền ngược, gradient bị giảm dần qua từng lớp. Khi gradient trở nên quá nhỏ, các tham số của mô hình gần như không được cập nhật, khiến mô hình khó học được mối quan hệ dài hạn trong dữ liệu. Ngược lại, gradient exploding làm cho gradient không ngừng tăng lên qua từng lớp, nó sẽ trở nên quá lớn, làm cho các tham số thay đổi một cách không kiểm soát được.

Để giải quyết vấn đề này, Hochreiter & Schmidhuber (1997) đã đề xuất Long Short-Term Memory (LSTM), một loại mạng nơ-ron hồi quy RNN nhưng có khả năng ghi nhớ thông tin dài hạn tốt hơn RNN truyền thống nhờ vào trạng thái bộ nhớ (cell state) và ba cổng điều khiển chính (forget gate, input gate, output gate).

Mỗi bước trong chuỗi thời gian, LSTM sẽ quyết định:

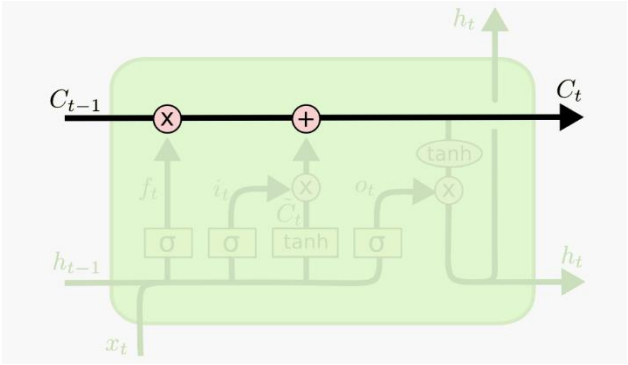
- Thông tin nào cần quên đi?
- Thông tin nào cần thêm vào bộ nhớ?
- Thông tin nào cần xuất ra làm đầu ra của bước hiện tại?

1.2.2 Hoạt động của LSTM

LSTM có thể giữ lại thông tin quan trọng trong thời gian dài mà không gặp phải vấn đề gradient vanishing như RNN truyền thống. Cơ chế này giúp nó vượt trội

trong các bài toán như xử lý ngôn ngữ tự nhiên, dịch máy, nhận diện giọng nói, và phân tích chuỗi thời gian.

Trạng thái bộ nhớ (cell state) là thành phần quan trọng trong LSTM, giúp lưu trữ và truyền tải thông tin trong suốt chuỗi thời gian. Ta có thể hình dung cell state như một băng chuyền chạy xuyên suốt qua các nút mạng, cho phép thông tin được duy trì mà không bị biến đổi quá nhiều. Nhờ cơ chế này, LSTM có thể tránh được vấn đề gradient vanishing mà RNN truyền thống gặp phải. Tuy nhiên, không phải lúc nào tất cả thông tin trong cell state cũng cần được giữ lại. LSTM có khả năng chọn lọc thông tin cần thiết bằng cách sử dụng các cổng điều khiển (gate). Các cổng này hoạt động như bộ lọc thông minh, quyết định giữ lại, loại bỏ hoặc cập nhật thông tin mới vào cell state. Quá trình này được thực hiện thông qua sự kết hợp của các tầng mạng sigmoid và các phép toán nhân, giúp mô hình chọn lọc và điều chỉnh thông tin một cách hợp lý, đảm bảo chỉ những dữ liệu quan trọng mới được giữ lại.



Hình 1.11 Cell state

Để quản lý việc cập nhật và quên thông tin, LSTM sử dụng ba cổng chính: Forget gate (Cổng quên) để quyết định phần nào của bộ nhớ cũ C_{t-1} sẽ bị loại bỏ. Input gate (Cổng nhập) xác định thông tin mới nào sẽ được thêm vào bộ nhớ C_t. Output gate (Cổng đầu ra) chọn lọc thông tin từ C_t để tạo đầu ra h_t.

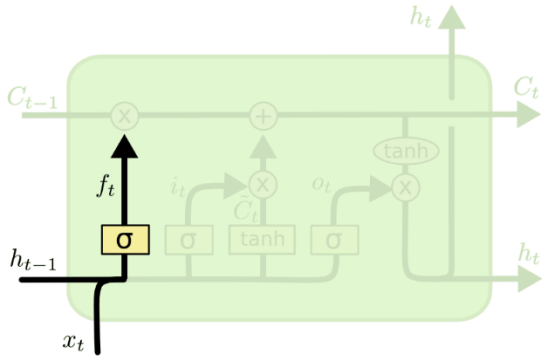
Forget gate (cổng quên)

Forget gate có nhiệm vụ quyết định xem mô hình có nên quên một phần thông tin từ quá khứ hay không. Nghĩa là khi một thông tin mới xuất hiện, LSTM sẽ so sánh thông tin đó với những gì đã lưu trữ trước đó và quyết định có nên giữ lại hay không.

Ở mỗi bước thời gian t, LSTM nhận hai đầu vào:

- x_t: Đầu vào tại bước hiện tại (ví dụ: một từ trong câu).
- h_{t-1}: Trạng thái ẩn từ bước trước đó (thông tin tóm tắt của những gì đã xảy ra).

Hai giá trị này được đưa qua một mạng nơ-ron nhân tạo với ma trận trọng số W_f và cộng thêm một số hằng số điều chỉnh b_f. Sau đó, kết quả được đưa vào hàm sigmoid σ, để chuyển đổi thành giá trị trong khoảng 0 đến 1 nhằm quyết định giữ lại hay loại bỏ thông tin từ trạng thái bộ nhớ trước đó C_{t-1}.



Hình 1.12 Forget gate

Trong đó:

- W_f và b_f là trọng số và bias của cổng quên.
- σ là hàm sigmoid, đưa giá trị đầu ra về khoảng (0,1).
    - Nếu giá trị gần 0, thông tin sẽ bị quên đi.
    - Nếu giá trị gần 1, thông tin sẽ được giữ lại.

Ví dụ: Nếu f_t = 0.6, điều đó có nghĩa là 40% thông tin cũ bị quên đi và 60% được giữ lại.

Sau khi tính được f_t, trạng thái bộ nhớ được cập nhật bằng phép nhân từng phần tử (element-wise multiplication):

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Input Gate – Cổng nhập

Sau khi quyết định loại bỏ một phần thông tin cũ bằng cổng quên f_t, LSTM cần xác định thông tin mới nào sẽ được thêm vào bộ nhớ (C_t). Quá trình này gồm hai phần chính:

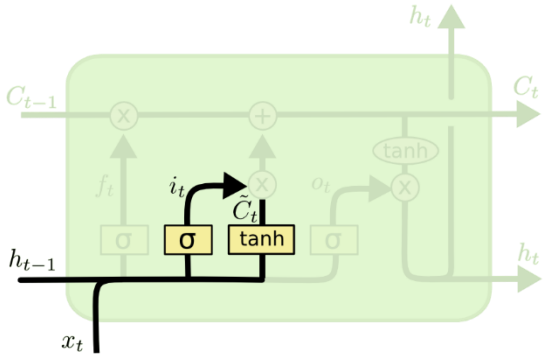
Đầu tiên là xác định thông tin nào sẽ được cập nhật vào bộ nhớ. Ta có lớp sigmoid (cổng nhập - Input Gate) dùng để xác định phần nào của thông tin mới sẽ được cập nhật vào bộ nhớ. Cổng này hoạt động giống như một bộ lọc, chỉ cho phép thông tin quan trọng đi vào.

$$i_t = \sigma(W_i . [h_{t-1}, x_t] + b_i$$

- Nếu i_t gần 1, thông tin đó sẽ được lưu vào bộ nhớ.
- Nếu i_t gần 0, thông tin sẽ bị bỏ qua.

Lớp tanh dùng để tạo ra một vector chứa các giá trị ứng viên mới $\tilde{C}_t$, là những thông tin tiềm năng có thể được thêm vào bộ nhớ. Vì nếu chỉ dùng sigmoid, giá trị đầu ra sẽ nằm trong khoảng (0,1) làm cho thông tin bị giới hạn, đồng thời giảm đi khả năng biểu diễn của mô hình. Do đó, hàm tanh giúp mô hình có thể ghi nhớ được giá trị cả âm và dương, giúp lưu trữ được nhiều thông tin hơn.

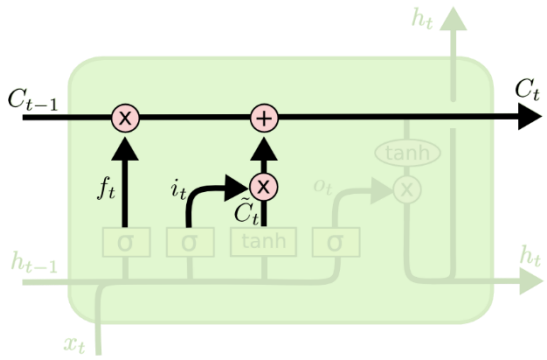
$$\tilde{C}_t = tanh(W_C . [h_{t-1}, x_t] + b_C$$



Hình 1.13 Input gate - Xác định thông tin được cập nhật vào bộ nhớ

Sau khi tính toán hai giá trị trên, LSTM kết hợp chúng để thực hiện cập nhật trạng thái bộ nhớ từ bước trước đó C_{t-1} thành trạng thái mới C_t. Các quyết định về việc quên và cập nhật thông tin đã được xác định từ trước, sau đó, cập nhật cell state theo công thức sau:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$



Hình 1.14 Input gate - Cập nhật trạng thái bộ nhớ

($f_t \odot C_{t-1}$): Giữ lại một phần trạng thái cũ, hoạt động của nó là nhân trạng thái bộ nhớ cũ C_{t-1} với giá trị của cổng quên f_t. Điều này giúp loại bỏ thông tin không còn cần thiết nhưng vẫn giữ lại những phần quan trọng.

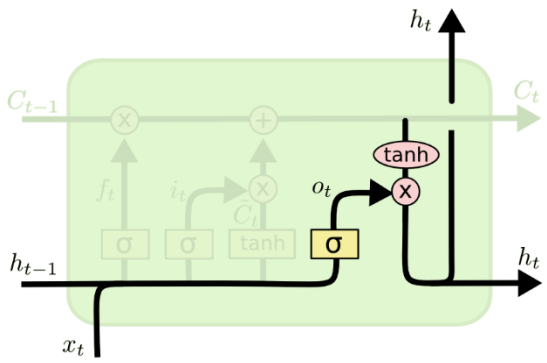
- Nếu f_t gần 0, thông tin trong bộ nhớ cũ C_{t-1} sẽ bị loại bỏ.
- Nếu f_t gần 1, thông tin trong C_{t-1} sẽ được giữ lại.

($i_t \odot \tilde{C}_t$): Thêm thông tin mới vào bộ nhớ, nghĩa là nhân giá trị ứng viên cập nhật $\tilde{C}_t$ với kết quả của cổng nhập i_t. Điều này đảm bảo rằng chỉ những thông tin thực sự cần thiết mới được thêm vào bộ nhớ.

Output gate – Cổng đầu ra

Sau khi bộ nhớ đã được cập nhật, LSTM cần quyết định thông tin nào sẽ được xuất ra để sử dụng cho bước tiếp theo hoặc để làm đầu ra cho mô hình. Trạng thái bộ nhớ C_t chứa rất nhiều thông tin, nhưng không phải lúc nào cũng cần sử dụng tất cả, vì vậy cổng đầu ra đóng vai trò chọn lọc thông tin phù hợp.

Quá trình của cổng đầu ra diễn ra qua hai bước chính:



Hình 1.15 Output gate

Bước thứ nhất là xác định phần nào của trạng thái bộ nhớ sẽ được sử dụng để tạo đầu ra. Ta có lớp sigmoid o_t sẽ xác định mức độ quan trọng của từng phần trong trạng thái bộ nhớ C_t. Nếu giá trị gần 1, thông tin tương ứng sẽ được đưa vào đầu ra. Nếu gần 0, thông tin đó sẽ bị loại bỏ.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o$$

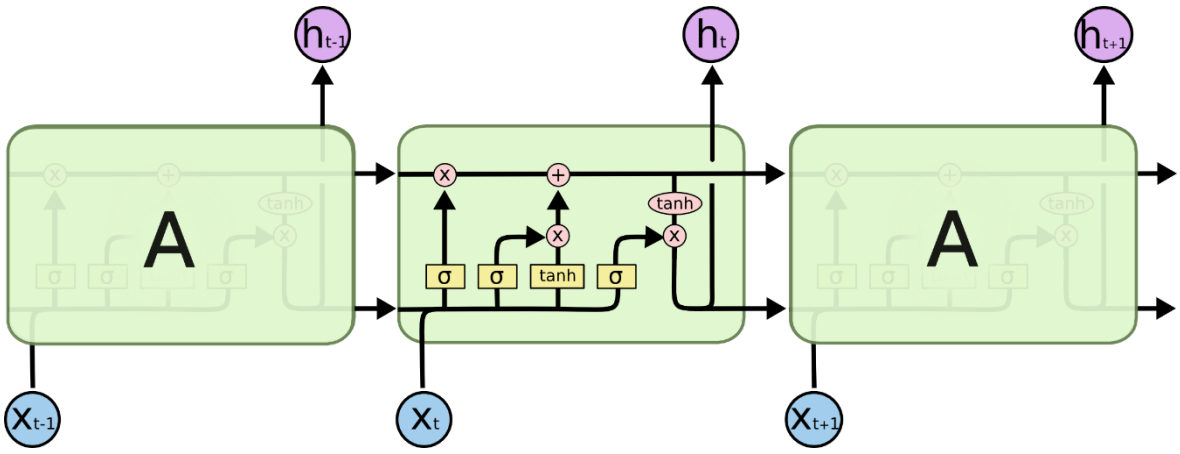
Trong đó:

W_o và b_o là trọng số và độ lệch của cổng đầu ra, giúp điều chỉnh quá trình chọn thông tin.

Bước thứ hai là tạo đầu ra cuối cùng từ trạng thái bộ nhớ. Sau khi xác định được phần thông tin cần sử dụng, LSTM cần điều chỉnh giá trị đó sao cho phù hợp. Để đảm bảo rằng đầu ra có giá trị trong khoảng $(-1,1)$, ta sử dụng hàm tanh để chuẩn hóa C_t. Sau đó, ta nhân kết quả này với giá trị của cổng đầu ra o_t để đảm bảo chỉ những thông tin quan trọng mới xuất hiện ở đầu ra.

$$h_t = o_t \odot \tanh(C_t)$$

h_t: là đầu ra cuối cùng tại thời điểm t, được dùng để dự đoán hoặc làm đầu vào cho bước tiếp theo.



Hình 1.16 Các đơn vị LSTM liên kết với nhau

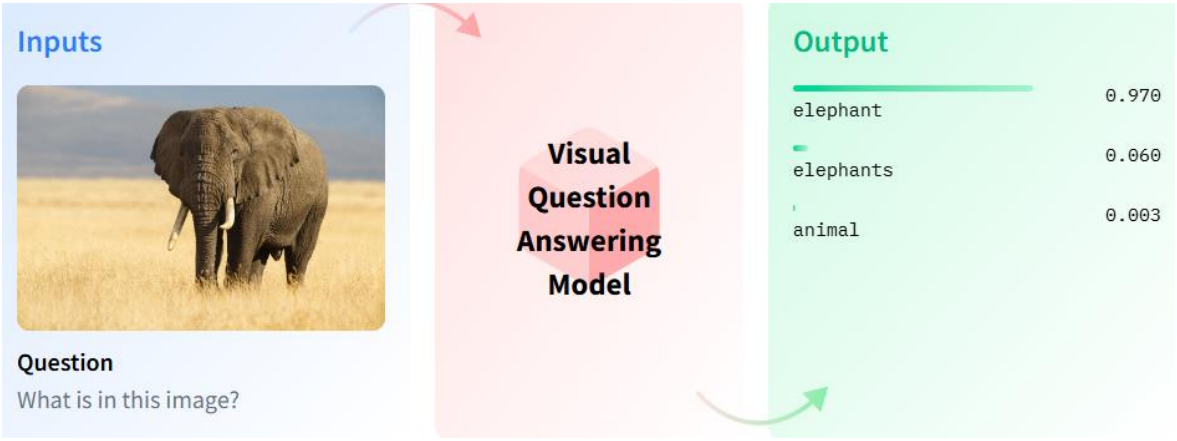
Hình ảnh 1.16 minh họa cách các đơn vị LSTM được kết nối theo chuỗi thời gian. Mỗi khối LSTM đại diện cho một bước trong chuỗi dữ liệu và có đầu vào từ bước trước (X_{t-1}, h_{t-1}) và đầu ra truyền đến bước kế tiếp X_{t+1}, h_{t+1}.

Trạng thái bộ nhớ (C_t) chạy xuyên suốt các bước và được điều chỉnh thông qua ba cổng chính: cổng quên, cổng nhập và cổng đầu ra. Cổng quên quyết định phần nào của trạng thái bộ nhớ từ bước trước được giữ lại. Cổng nhập chọn lọc thông tin mới để thêm vào bộ nhớ, còn cổng đầu ra quyết định thông tin nào sẽ được truyền đi tiếp.

Sự kết nối này giúp LSTM có thể lưu giữ thông tin dài hạn trong chuỗi dữ liệu, tránh tình trạng mất mát thông tin như trong các mạng nơ-ron truyền thống.

1.3 Visual Question Answering (VQA)

Visual Question Answering (VQA) là một lĩnh vực trong trí tuệ nhân tạo kết hợp giữa xử lý ngôn ngữ tự nhiên và thị giác máy tính, nhằm mục tiêu phát triển các hệ thống có khả năng trả lời các câu hỏi dựa trên nội dung của hình ảnh. Nhiệm vụ này đòi hỏi hệ thống không chỉ hiểu và phân tích hình ảnh mà còn phải diễn giải câu hỏi bằng ngôn ngữ tự nhiên để đưa ra câu trả lời chính xác.



Hình 1.17 Ảnh minh hoạ Visual Question Answering

VQA có nhiều ứng dụng thực tiễn, bao gồm hỗ trợ người khiếm thị bằng cách cung cấp mô tả chi tiết về nội dung hình ảnh dựa trên câu hỏi của người dùng. Ngoài ra, nó còn được sử dụng trong tìm kiếm hình ảnh thông minh, cho phép người dùng tìm kiếm hình ảnh dựa trên câu hỏi tự nhiên, cũng như trong tương tác người-máy để nâng cao khả năng giao tiếp giữa con người và máy tính thông qua việc hiểu và phản hồi dựa trên nội dung hình ảnh.

Một hệ thống VQA điển hình bao gồm nhiều thành phần quan trọng. Đầu tiên, hệ thống cần trích xuất đặc trưng hình ảnh bằng cách sử dụng các mô hình thị giác máy tính, như mạng nơ-ron tích chập (CNN), để phân tích và rút trích các đặc trưng quan trọng từ hình ảnh đầu vào. Tiếp theo, câu hỏi đầu vào được biểu diễn thông qua các kĩ thuật xử lý ngôn ngữ tự nhiên, chẳng hạn như mạng nơ-ron hồi tiếp dài ngắn hạn (LSTM) hoặc các mô hình Transformer, nhằm mã hóa câu hỏi thành vector đặc trưng. Sau khi có thông tin từ cả hình ảnh và câu hỏi, hệ thống sẽ tích hợp hai nguồn thông tin này để tạo ra một biểu diễn chung, thường sử dụng các phương pháp như nhân chập, nhân tử hoặc cơ chế chú ý (attention). Cuối cùng, hệ thống sử dụng một lớp phân loại hoặc mô hình sinh để dự đoán câu trả lời dựa trên biểu diễn kết hợp của hình ảnh và câu hỏi.

Quá trình tính toán trong VQA thường bao gồm các bước chính. Trước tiên, hình ảnh được đưa qua mô hình CNN để trích xuất đặc trưng và chuyển đổi thành tensor đặc trưng. Câu hỏi sau đó được mã hóa bằng LSTM hoặc Transformer để biểu diễn dưới dạng vector đặc trưng. Sau đó, Attention được sử dụng để xác định các vùng quan trọng trong hình ảnh có liên quan đến nội dung của câu hỏi, từ đó kết hợp thông tin giữa hình ảnh và câu hỏi để tạo ra một biểu diễn chung. Dựa trên biểu diễn này, hệ thống sẽ dự đoán câu trả lời thông qua một lớp phân loại.

Để triển khai một hệ thống VQA, trước tiên cần chuẩn bị dữ liệu bằng cách sử dụng các bộ dữ liệu VQA có sẵn hoặc tự tạo bộ dữ liệu mới bằng cách thu thập hình ảnh và xây dựng cặp câu hỏi - câu trả lời tương ứng. Tiếp theo, cần thiết kế mô hình với các thành phần chính như trích xuất đặc trưng hình ảnh, mã hóa câu hỏi và kết hợp thông tin. Sau khi mô hình được xây dựng, quá trình huấn luyện được thực hiện trên dữ liệu huấn luyện để tối ưu hóa các tham số. Cuối cùng, hệ thống sẽ được đánh giá trên tập dữ liệu kiểm tra để xác định hiệu suất và thực hiện tinh chỉnh nhằm cải thiện độ chính xác của mô hình.

1.4 Cơ chế Attention

Trong bài toán Visual Question Answering, mô hình thường sử dụng CNN để trích xuất đặc trưng từ hình ảnh và LSTM để sinh ra chuỗi câu trả lời dựa trên các

đặc trưng đó. Trong quá trình LSTM sinh chuỗi, Attention đóng vai trò quan trọng, giúp mô hình tập trung vào những vùng liên quan của hình ảnh tại từng bước sinh câu trả lời, thay vì xử lý toàn bộ ảnh một cách đồng đều.

Việc sử dụng hay không sử dụng attention trong quá trình LSTM sinh chuỗi sẽ ảnh hưởng lớn đến khả năng của mô hình trong việc chọn lọc thông tin và đưa ra câu trả lời chính xác. Nếu không có attention, LSTM chỉ sử dụng đặc trưng ảnh cố định ở mọi bước, trong khi với attention, mô hình có thể lựa chọn thông tin từ các vùng ảnh phù hợp tại từng thời điểm, từ đó cải thiện hiệu quả và độ chính xác trong việc trả lời câu hỏi.

Cụ thể, trong mô hình không sử dụng attention, sau khi CNN trích xuất đặc trưng từ toàn bộ hình ảnh, các đặc trưng này được nén lại thành một vector cố định, sau đó được ghép với vector biểu diễn câu hỏi qua LSTM để làm đầu vào cho một LSTM khác chuyên dùng để sinh ra chuỗi câu trả lời. Tuy nhiên, khi LSTM sinh ra từng từ trong câu trả lời, nó dùng cùng một vector đặc trưng hình ảnh lặp đi lặp lại tại mỗi bước thời gian, làm cho LSTM không phân biệt được phần nào của hình ảnh quan trọng hơn phần khác. Điều này khiến mô hình không thể tập trung vào các phần quan trọng trong ảnh và dễ bỏ qua thông tin có liên quan đến câu hỏi. Việc mô hình xử lý toàn bộ ảnh một cách đồng đều khiến cho hiệu quả giảm ở những câu hỏi đòi hỏi sự tập trung vào chi tiết.

Khác với cách trên, ở mô hình sử dụng attention, tại mỗi bước thời gian khi LSTM sinh ra từng từ trong câu trả lời, nó không dùng toàn bộ đặc trưng hình ảnh một cách đồng đều mà sẽ chọn lọc thông tin từ các vùng ảnh liên quan nhất. Điều này được thực hiện nhờ vào cơ chế attention, khi đó mô hình sẽ tính attention weights (trọng số chú ý), là mức độ quan trọng của từng vùng ảnh đối với câu hỏi. Sau đó, mô hình sẽ tổng hợp thông tin từ các vùng ảnh dựa trên attention weights, vùng nào liên quan đến câu hỏi hơn thì sẽ được ưu tiên và ảnh hưởng hơn đến kết quả. Các thông tin đã chọn lọc này sẽ được kết hợp với trạng thái ẩn hiện tại của LSTM để sinh ra từ tiếp theo trong câu trả lời. Quá trình này gồm các bước:

- Tính trọng số chú ý giữa các vùng ảnh và trạng thái ẩn hiện tại của LSTM.
- Tổng hợp thông tin từ các vùng ảnh theo trọng số này (vùng quan trọng được ưu tiên hơn).
- Kết hợp thông tin đã chọn lọc này với trạng thái hiện tại của LSTM để sinh ra từ tiếp theo trong câu trả lời.

Hình 1.18 Ví dụ attention trong image description

Việc sử dụng attention giúp mô hình linh hoạt và thông minh hơn vì nó biết tập trung vào những phần hình ảnh thực sự liên quan, giống như con người khi nhìn vào ảnh và trả lời câu hỏi. Nhờ đó, việc sử dụng attention không chỉ cải thiện độ chính xác mà còn nâng cao khả năng suy luận và hiểu ngữ cảnh của mô hình VQA. Do đó, cơ chế attention giúp mô hình tập trung vào thông tin phù hợp và cần thiết tại từng thời điểm, làm cho quá trình sinh chuỗi câu trả lời bằng LSTM trở nên hiệu quả hơn trong bài toán VQA.

1.5 Kết hợp CNN và LSTM – Kĩ thuật Multimodal Fusion

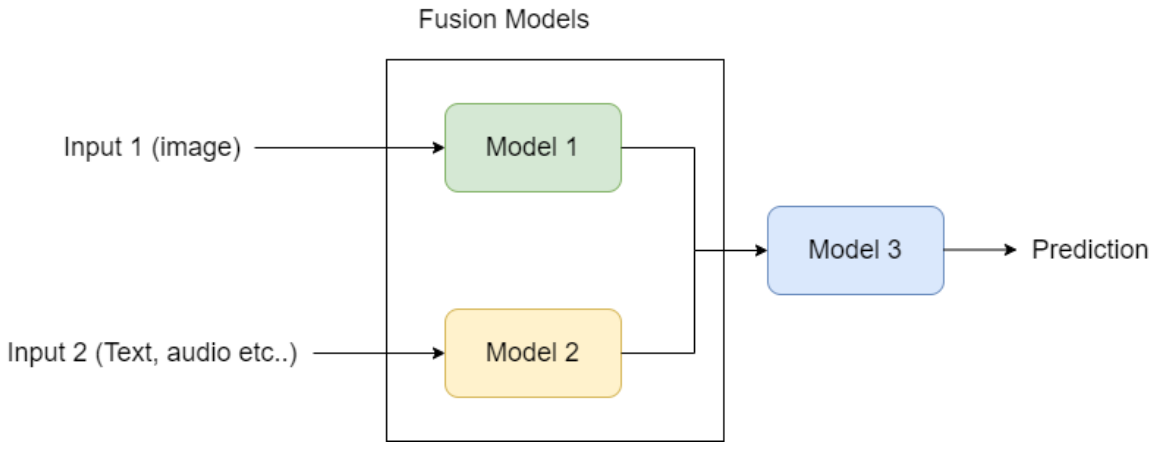
Trong bài toán Visual Question Answering (VQA), hệ thống cần xử lý hai loại dữ liệu khác nhau: hình ảnh và câu hỏi dạng văn bản. Để giải quyết yêu cầu này, các mô hình học sâu thường kết hợp giữa Convolutional Neural Network (CNN) và Long

Short-Term Memory (LSTM) nhằm tận dụng ưu điểm của từng mô hình trong việc xử lý dữ liệu đặc thù.

CNN được sử dụng để xử lý hình ảnh đầu vào. Mạng CNN có khả năng trích xuất các đặc trưng quan trọng từ hình ảnh như màu sắc, hình dạng, kết cấu, và bố cục tổng thể. Thông qua các lớp tích chập và gộp, CNN biến đổi ảnh gốc thành một vector đặc trưng có kích thước cố định, đại diện cho toàn bộ thông tin quan trọng trong hình ảnh. Vector này sau đó sẽ được sử dụng làm nguồn thông tin thị giác cho hệ thống.

LSTM được sử dụng để xử lý câu hỏi dưới dạng chuỗi từ. Nhờ khả năng ghi nhớ thông tin theo thứ tự thời gian, LSTM có thể nắm bắt ngữ nghĩa của câu hỏi bằng cách mã hóa từng từ, từng cụm từ và quan hệ giữa chúng thành một vector đặc trưng ngôn ngữ. Vector này thể hiện nội dung và mục đích của câu hỏi, giúp hệ thống hiểu được người dùng muốn hỏi điều gì về hình ảnh.

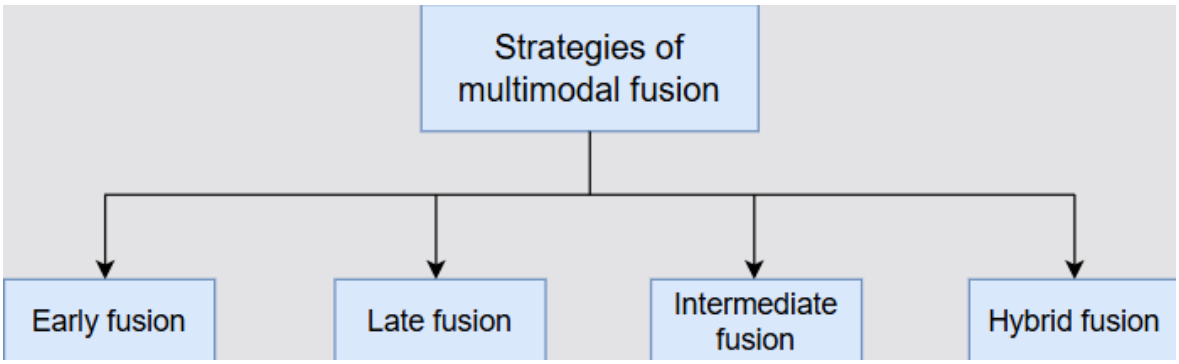
Sau khi có được hai vector đặc trưng, một vector từ CNN (ảnh) và một vector từ LSTM (câu hỏi) thì hệ thống sẽ tiến hành kết hợp thông tin từ hai nguồn này để dự đoán câu trả lời phù hợp, sự kết hợp này được gọi là **Multimodal Fusion** (kết hợp đa phương thức). Kĩ thuật fusion nhằm tạo ra một biểu diễn chung (joint representation), giúp mô hình hiểu được mối liên hệ giữa ảnh và câu hỏi để đưa ra câu trả lời chính xác.



Hình 1.19 Multimodal fusion hoạt động

Có nhiều cách tiếp cận khác nhau đối với multimodal fusion, được phân loại theo thời điểm và cách thức kết hợp giữa các modality - là dạng dữ liệu khác nhau

mà mô hình cần xử lý và kết hợp, bao gồm Early fusion, Intermediate fusion, Late fusion và Hybrid fusion.



Hình 1.20 Nhiều cách tiếp cận của multimodal fusion

Early Fusion (kết hợp sớm) là phương pháp kết hợp thông tin giữa các nguồn dữ liệu ngay sau khi đặc trưng từ từng nguồn được trích xuất xong. Ví dụ cụ thể trong bài toán VQA, hai vector ảnh và câu hỏi được nối lại với nhau thành một vector lớn hơn, gọi là vector kết hợp. Vector này sẽ được đưa vào các lớp fully connected để tiếp tục xử lý và đưa ra dự đoán câu trả lời. Phương pháp này đơn giản, dễ triển khai vì không cần thêm cơ chế tính toán phức tạp nào để điều phối thông tin. Tuy nhiên, do quá trình kết hợp diễn ra ngay từ đầu, thông tin từ ảnh và câu hỏi còn khá "thô" nên mô hình khó học được mối liên hệ sâu sắc giữa nội dung hình ảnh và ý nghĩa câu hỏi. Điều này dẫn đến việc early fusion thường hiệu quả thấp hơn so với các phương pháp fusion khác, đặc biệt là trong những bài toán yêu cầu suy luận chi tiết như VQA.

Trong quá trình **Intermediate Fusion (kết hợp trung gian)**, để kết hợp thông tin từ hai nguồn dữ liệu là hình ảnh và câu hỏi, mô hình thường sử dụng các phép toán kết hợp đặc trưng. Những phép toán này giúp mô hình học được mối liên hệ giữa các vector đặc trưng trích xuất từ ảnh (qua CNN) và từ câu hỏi (qua LSTM), từ đó tạo ra một vector chung phục vụ cho việc dự đoán.

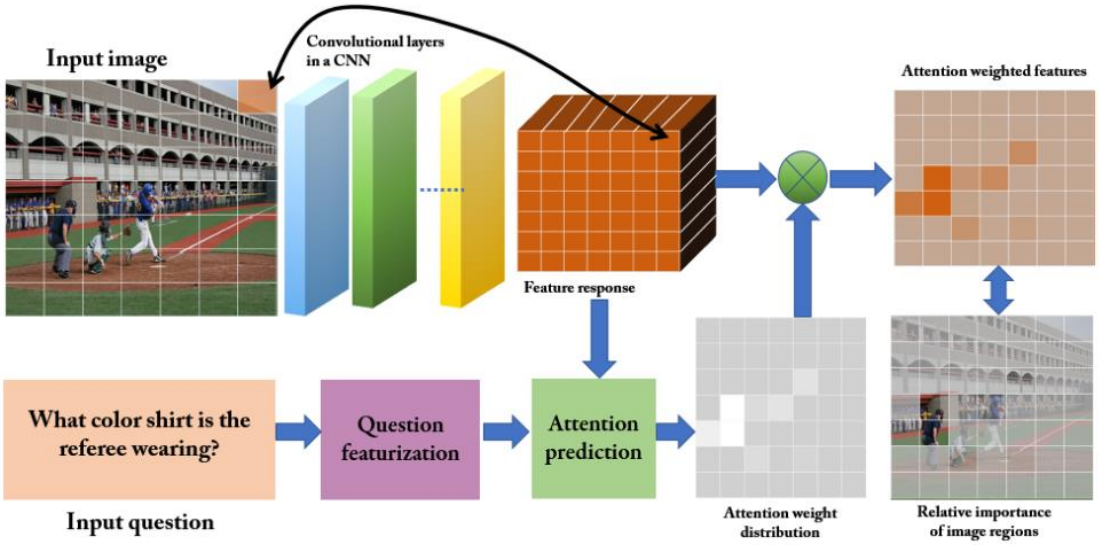
Một phép toán phổ biến là nhân từng phần tử (element-wise multiplication), trong đó hai vector có cùng kích thước được nhân với nhau tại từng vị trí tương ứng. Phép nhân này giúp làm nổi bật các đặc trưng mà cả hai nguồn dữ liệu đều cho là quan trọng. Ngoài ra, mô hình cũng có thể sử dụng cộng từng phần tử (element-wise addition) để tổng hợp thông tin nhanh chóng, tuy nhiên phép cộng thường không thể

hiện rõ mối tương tác giữa các nguồn như phép nhân. Một cách kết hợp đơn giản khác là phép nối (concatenation), tức là nối hai vector lại với nhau thành một vector lớn hơn, rồi đưa vào các lớp tiếp theo. Phép nối dễ triển khai nhưng có thể làm tăng đáng kể kích thước vector, từ đó tăng chi phí tính toán.

Ngoài ra, các mô hình VQA hiện đại còn sử dụng bilinear pooling, là một kĩ thuật tạo ra ma trận tương tác giữa mọi chiều của hai vector. Bilinear pooling giúp mô hình khai thác tương tác sâu sắc giữa các đặc trưng từ ảnh và câu hỏi. Một số biến thể tối ưu của kĩ thuật này là Multimodal Compact Bilinear Pooling (MCB) hoặc Mutan.

Trong **Late Fusion (kết hợp muộn)**, mỗi modality được xử lý riêng biệt. Sau khi có kết quả dự đoán riêng từ từng mô hình, hệ thống mới kết hợp kết quả để đưa ra quyết định cuối cùng, bằng cách trung bình hóa, gán trọng số ưu tiên, hoặc bỏ phiếu đa số. Ví dụ, trong VQA với câu hỏi “What color is the car?”, mô hình xử lý ảnh dự đoán “red”, mô hình xử lý câu hỏi dự đoán “blue”. Hệ thống có thể kết hợp kết quả bằng cách chọn đáp án có mức tin tưởng cao hơn, hoặc nếu có nhiều mô hình, chọn đáp án được nhiều mô hình dự đoán nhất. Trong trường hợp này, nếu “red” được hai mô hình chọn, còn “blue” chỉ được một mô hình chọn, hệ thống sẽ trả lời “red”. Phương pháp này đơn giản, phù hợp khi hệ thống xử lý dữ liệu độc lập. Tuy nhiên, do không kết hợp thông tin giữa ảnh và câu hỏi trong quá trình xử lý nên thường không tận dụng tốt mối liên kết giữa hai nguồn dữ liệu, dẫn đến hiệu quả thấp hơn trong bài toán VQA.

Hybrid Fusion (kết hợp lai) là sự pha trộn giữa hai hoặc nhiều phương pháp trên. Ví dụ, mô hình có thể thực hiện early fusion để tạo biểu diễn ban đầu, sau đó tiếp tục sử dụng attention (thuộc intermediate fusion), và cuối cùng áp dụng late fusion để tổng hợp kết quả. Cách kết hợp linh hoạt này cho phép mô hình tận dụng tốt ưu điểm của từng phương pháp.



Hình 1.21 Ảnh minh hoạ cách kết hợp attention trong hệ thống VQA

Trong thực tế, nhiều hệ thống VQA hiện đại sử dụng attention để kết hợp thông tin từ CNN và LSTM một cách hiệu quả hơn. Như hình 1.20 minh họa CNN trích xuất các đặc trưng từ nhiều vùng khác nhau trong ảnh, còn câu hỏi được biểu diễn thành vector đặc trưng thông qua bước featurization. Attention sẽ tính toán mức độ liên quan giữa từng vùng ảnh và nội dung câu hỏi, từ đó gán trọng số cao hơn cho các vùng ảnh quan trọng. Thông tin ảnh sau khi đã được chọn lọc sẽ kết hợp với đặc trưng câu hỏi để dự đoán câu trả lời, thông qua mô hình sinh chuỗi như LSTM.

Về bản chất, attention trong hệ thống này là một phần của quá trình kết hợp thông tin (fusion) giữa ảnh và câu hỏi. Vì attention xảy ra sau khi đã có đặc trưng ảnh và câu hỏi nên nó được xếp vào Intermediate Fusion. Hoặc nếu attention được sử dụng cùng với các cách kết hợp khác như nối vector hoặc pooling, mô hình có thể thuộc Hybrid Fusion.

Như vậy, trong bài toán Visual Question Answering, việc xử lý riêng biệt hình ảnh bằng CNN và câu hỏi bằng LSTM là chưa đủ để mô hình đưa ra câu trả lời chính xác. Hệ thống cần có một cách kết hợp hợp lý giữa hai nguồn thông tin này để hiểu được mối liên hệ giữa nội dung hình ảnh và ý nghĩa câu hỏi. Các kĩ thuật multimodal fusion chính là giải pháp để thực hiện điều đó, trong đó các phương pháp như early, intermediate, late và hybrid fusion được lựa chọn tùy theo yêu cầu bài toán. Đặc biệt,

việc ứng dụng attention trong quá trình kết hợp đã giúp mô hình tập trung vào thông tin quan trọng và cải thiện đáng kể độ chính xác khi trả lời. Do đó, việc kết hợp hiệu quả giữa CNN và LSTM thông qua các kĩ thuật fusion phù hợp đóng vai trò rất quan trọng trong hiệu suất hoạt động của hệ thống VQA.

CHƯƠNG 2. THỰC NGHIỆM

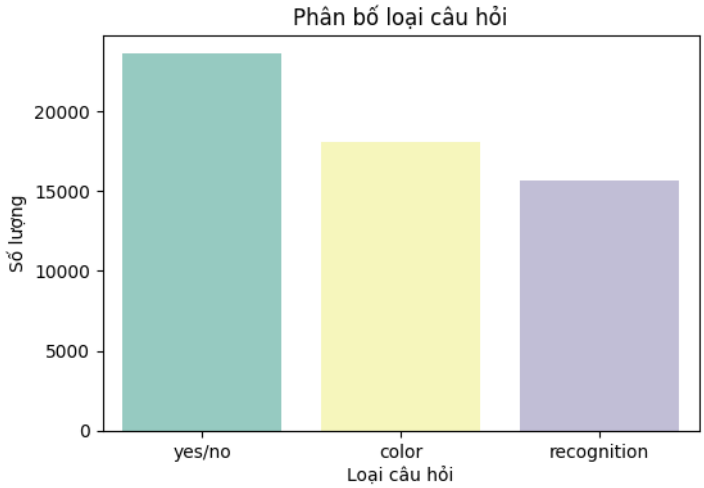
2.1 Dataset

Bộ dữ liệu có tổng cộng **65,331 mẫu** được chia thành các tập gồm: tập Train chứa 41,613 mẫu, tập Validation chứa 4,839 mẫu và tập Test chứa 10,879 mẫu. Mỗi mẫu gồm một cặp câu hỏi, câu trả lời và liên kết với hình ảnh cụ thể, nhằm mục tiêu huấn luyện và đánh giá mô hình VQA.

Mỗi mẫu dữ liệu gồm các trường:

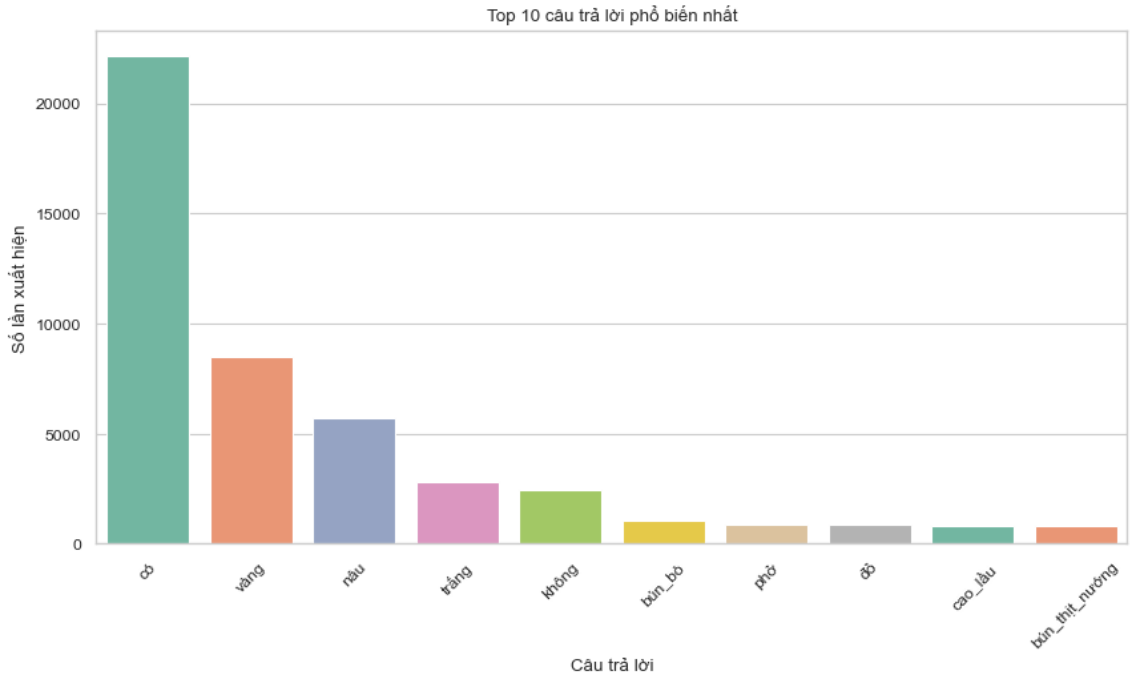
- question: Câu hỏi tự nhiên về hình ảnh (tiếng Việt).
- answer: Câu trả lời.
- question_type: Phân loại câu hỏi (ví dụ: recognition, yes/no, color).
- answer_type: Loại câu trả lời (text hoặc boolean).
- image_path: Đường dẫn tới hình ảnh tương ứng.

Chúng tôi phân loại câu hỏi thành ba nhóm chính: **recognition** (là nhận diện, ví dụ như “Món này là món gì?”), **yes/no** (ví dụ như “Bánh bèo có nước chấm không?”), và **color** (là màu sắc, ví dụ như “Món ăn có màu gì?”). Trong đó, câu hỏi dạng yes/no chiếm tỷ lệ cao nhất, tiếp theo đến color và cuối cùng là recognition. Từ tỷ lệ trên cho thấy bộ dữ liệu tập trung chủ yếu vào khả năng nhận diện món ăn, cùng với yêu cầu phân tích đặc điểm thị giác như màu sắc, thành phần.



Hình 2.1 Phân bố câu hỏi trong dataset

Về phần câu trả lời, các câu trả lời phần lớn có tính ngắn gọn. Câu trả lời chỉ gồm 1 từ chiếm đa số 89%, các câu trả lời boolean (“có”, “không”) chiếm khoảng 40%, đồng thời những câu trả lời phổ biến nhất là những từ như “có”, “vàng”, “nâu”, “trắng”, “không” cho thấy sự ngắn gọn và tập trung của câu trả lời giúp việc đánh giá mô hình dễ dàng hơn, yêu cầu mô hình phải hiểu đúng ý câu hỏi và hình ảnh để đưa ra câu trả lời chính xác.



Hình 2.2 Câu trả lời phổ biến trong dataset

Mỗi hình ảnh trong dataset có thể liên kết với nhiều câu hỏi, khai thác các khía cạnh khác nhau của cùng một nội dung hình ảnh, từ nhận diện tên món ăn, đến phân tích màu sắc và các đặc điểm liên quan.

2.2 Kiến trúc mô hình

2.2.1 MobileNet_v2 và LSTM không attention

Đầu tiên, ảnh đầu vào ký hiệu là $I \in R^{C\times H\times W}$, được đưa vào mạng CNN MobileNetV2 để trích xuất đặc trưng. Sau khi trích xuất, đầu ra được đưa qua lớp Pooling nhằm giảm chiều và thu được vector đặc trưng ảnh có kích thước 1280 chiều, ký hiệu là $\widehat{F_I}$. Để đưa đặc trưng này về không gian thấp hơn, mô hình sử dụng một lớp ánh xạ tuyến tính kết hợp với hàm kích hoạt ReLU, thu được vector đặc trưng ảnh đã được mã hóa, ký hiệu là $F'_I \in R^{256}$. Quá trình này được mô tả bằng công thức:

$$F_I = \text{MobileNetV2}(I) \rightarrow \text{Pool}(\hat{F}_I), \quad \hat{F}_I \in \mathbb{R}^{1280}$$

$$F'_I = \sigma(W_I \hat{F}_I + b_I) \in \mathbb{R}^{256}$$

Câu hỏi đầu vào được biểu diễn dưới dạng chuỗi các từ $Q = (q_1, q_2, \dots, q_T)$, với T là số từ trong câu hỏi. Mỗi từ q_t được ánh xạ sang một vector nhúng có kích thước 256 chiều thông qua lớp nhúng từ. Sau đó, toàn bộ chuỗi vector nhúng được đưa vào mạng BiLSTM (LSTM hai chiều), từ đó thu được các vector trạng thái ẩn tại mỗi thời điểm. Mô hình chỉ sử dụng trạng thái ẩn cuối cùng, ký hiệu là $H_t \in R^{512}$, để đại diện cho toàn bộ câu hỏi. Trạng thái ẩn này tiếp tục được ánh xạ qua một lớp tuyến tính và ReLU để thu được đặc trưng câu hỏi đã được mã hóa, ký hiệu là $F'_Q \in R^{256}$.

$$E_t = \text{Embedding}(q_t) \in \mathbb{R}^{256}$$

$$H_t = \text{BiLSTM}(E_{1:T}) \in \mathbb{R}^{2h}, \quad h = 256 \Rightarrow 2h = 512$$

$$F'_Q = \sigma(W_Q H_T + b_Q) \in \mathbb{R}^{256}$$

Sau khi có hai vector đặc trưng ảnh F'_I và đặc trưng câu hỏi F'_Q mô hình tiến hành kết hợp thông tin từ hai nguồn này bằng cách thực hiện nhân từng phần tử

(element-wise) giữa chúng, tạo ra vector đặc trưng kết hợp $F \in R^{256}$. Vector này được ánh xạ tiếp qua một lớp tuyến tính và ReLU, cho ra vector $F' \in R^{256}$. Cuối cùng F' được đưa qua hàm softmax để để tính xác suất cho từng lớp và đưa ra dự đoán cuối cùng, ký hiệu $\hat{y} \in R^{num\_classes}$. Các bước này được biểu diễn qua công thức:

$$F = F'_I \odot F'_Q \in \mathbb{R}^{256}$$

$$F' = \sigma(W_F F + b_F) \in \mathbb{R}^{256}$$

$$\hat{y} = \text{Softmax}(W_C F' + b_C) \in \mathbb{R}^{\text{num\_classes}}$$

Trong đó, σ là hàm kích hoạt ReLU, $\odot$ là phép nhân từng phần tử, W_I, W_Q, W_F, W_C là các ma trận trọng số học được trong quá trình huấn luyện, và b_I, b_Q, b_F, b_C là các vector bias tương ứng.

2.2.2 MobileNet_v2 và LSTM attention

Image Encoder

Ảnh đầu vào $I \in R^{C \times H \times W}$ được đưa vào mạng CNN MobileNetV2 để trích xuất đặc trưng, sau đó được làm phẳng qua phép pooling. Kết quả là một vector đặc trưng $\hat{F}_I \in R^{1280}$. Vector này tiếp tục được ánh xạ qua một lớp tuyến tính kết hợp hàm kích hoạt ReLU để thu được đặc trưng ảnh $F'_I \in R^{256}$.

$$F_I = \text{MobileNetV2}(I) \xrightarrow{\text{Pool}} \hat{F}_I \in \mathbb{R}^{1280}$$

$$F'_I = \sigma(W_I \hat{F}_I + b_I) \in \mathbb{R}^{256}$$

Text Encoder

Câu hỏi đầu vào được biểu diễn dưới dạng chuỗi các từ $Q = (q_1, q_2, \ldots, q_T)$. Trong đó, mỗi từ q_t được ánh xạ thành vector qua lớp nhúng (embedding). Chuỗi vector này được đưa qua một mạng LSTM, và chỉ lấy trạng thái ẩn cuối cùng $H_t \in R^{256}$. Sau đó, trạng thái này tiếp tục được ánh xạ tuyến tính và kích hoạt để tạo ra đặc trưng câu hỏi $F'_Q \in R^{256}$.

$$E_t = \text{Embedding}(q_t) \in R^{256}$$

$$H_t = \text{LSTM}(E_{1:T}) \in R^{256}$$

$$F'_Q = \sigma\left(W_Q H_T + b_Q\right) \in R^{256}$$

Attention Fusion

Đặc trưng ảnh F'_I được ánh xạ qua một lớp attention để tạo thành vector mới $\widetilde{F_I} \in R^{256}$. Sau đó, hai vector $\widetilde{F_I}$ và F'_Q được nhân từng phần tử (element-wise) để tạo thành vector tương tác $F_{attn} \in R^{256}$.

$$\widetilde{F_I} = \sigma\left(W_{attn} F'_I\right) \in R^{256}$$

$$F_{attn} = \widetilde{F_I} \odot F'_Q \in R^{256}$$

Fusion và Classfication

Hai vector F_{attn} và F'_Q được nối lại để tạo thành vector hợp nhất $F_{fusion} \in R^{512}$. Vector này được đưa qua một lớp ánh xạ và kích hoạt để giảm chiều còn 256, thu được $F' \in R^{256}$. Cuối cùng F' được đưa qua hàm softmax để dự đoán xác suất cho từng lớp, tạo ra kết quả $\hat{y} \in R^{num\_classes}$.

$$F_{fusion} = [F_{attn}; F'_Q] \in R^{512}$$

$$F' = \sigma\left(W_F F_{fusion} + b_F\right) \in R^{256}$$

$$\hat{y} = \mathrm{Softmax}(W_C F' + b_C) \in R^{num\_classes}$$

Trong đó, σ là hàm kích hoạt ReLU giúp tăng tính phi tuyến, $\odot$ phép nhân từng phần tử giữa hai vector cùng chiều, các trọng số W và bias b là tham số được mô hình học trong quá trình huấn luyện.

2.2.3 CNN và LSTM không attention

Image Encoder

Ảnh đầu vào $I \in R^{3\times H\times W}$ với 3 là số kênh màu (RGB), H và W là chiều cao và chiều rộng ảnh. Ảnh này được đưa vào chuỗi các lớp CNN để trích xuất đặc trưng, sau đó qua lớp trung bình cộng (Average Pooling) để thu được vector đặc trưng ảnh, ký hiệu là $\widehat{F_I} \in R^{512}$. Đặc trưng này tiếp tục được ánh xạ qua một lớp tuyến tính kết hợp với hàm kích hoạt ReLU nhằm giảm chiều và tạo vector đặc trưng ảnh cuối cùng $F'_I \in R^{256}$.

$$F_I = \mathrm{CNN}(I) \rightarrow \mathrm{AvgPool}, \qquad \widehat{F_I} \in R^{512}$$

$$F'_I = \sigma\left(W_I \widehat{F_I} + b_I\right) \in R^{256}$$

Text Encoder

Câu hỏi đầu vào được biểu diễn dưới dạng chuỗi các từ $Q = (q_1, q_2, \dots, q_T)$. Trong đó, mỗi từ q_t được ánh xạ thành vector qua lớp nhúng (embedding), thu được $E_t \in R^{256}$. Chuỗi vector này được đưa qua một mạng BiLSTM (LSTM hai chiều), kết quả là trạng thái ẩn tại mỗi thời điểm $H_t \in R^{1024}$. Mô hình sử dụng trạng thái ẩn cuối cùng H_t để đại diện cho toàn câu hỏi. Tiếp đó, H_t được ánh xạ qua lớp tuyến tính và ReLU, thu được đặc trưng câu hỏi $F_Q \in R^{256}$, sau đó được chuẩn hóa bằng Batch Normalization để thu được F'_Q.

$$E_t = \text{Embedding}(q_t) \in R^{256}$$

$$H_t = \text{BiLSTM}\,(E_{1:T}) \in R^{1024}, H_t \in R^{1024}$$

$$F_Q = \sigma\left(W_Q H_T + b_Q\right) \in R^{256}$$

$$F'_Q = \text{BatchNorm}(F_Q)$$

Fusion và Classfication

Hai vector đặc trưng ảnh F'_I và đặc trưng câu hỏi F'_Q được kết hợp bằng cách nối (concatenation) để tạo vector $F \in R^{512}$. Vector này được ánh xạ qua một lớp tuyến tính và ReLU, tiếp theo là chuẩn hóa bằng Batch Normalization và dropout nhằm tránh overfitting, thu được $F' \in R^{256}$. Cuối cùng $\hat{F}$ được đưa vào lớp Softmax để dự đoán phân phối xác suất trên các lớp, ký hiệu là $\hat{y} \in R^{\text{num\_classes}}$.

$$F_{\text{fusion}} = [F'_I; F'_Q] \in R^{512}$$

$$F' = \sigma\,(W_F F + b_F) \to BN, Dropout, \quad \hat{F} \in R^{256}$$

$$\hat{y} = \text{Softmax}\left(W_C \hat{F} + b_C\right) \in R^{\text{num\_classes}}$$

Trong đó, σ là hàm kích hoạt ReLU giúp tăng tính phi tuyến, các trọng số W và bias b là tham số được mô hình học trong quá trình huấn luyện, BN là Batch Normalization.

2.2.4 CNN và LSTM attention

Mô hình VQAClassifier được thiết kế để kết hợp thông tin từ hình ảnh và câu hỏi dạng văn bản nhằm đưa ra dự đoán cho bài toán Visual Question Answering. Mô hình gồm ba thành phần chính: Image Encoder (Bộ mã hóa ảnh), Text Encoder (Bộ mã hóa văn bản), và Fusion & Classification (Khối hợp nhất và phân loại).

Image Encoder

Ảnh đầu vào được biểu diễn dưới dạng tensor ba chiều $I \in R^{C\times H\times W}$. Trong đó C là số kênh màu (thường là 3 đối với ảnh RGB), H là chiều cao ảnh, và W là chiều rộng ảnh. Ảnh này được đưa qua một mạng CNN gồm nhiều lớp tích chập (convolution), chuẩn hóa (batch normalization), và phi tuyến (ReLU), kết thúc bằng lớp gộp trung bình toàn cục (AdaptiveAvgPool2d) để rút gọn kích thước đặc trưng.

$$\mathrm{F_I} = \mathrm{CNN(I)} \in \mathrm{R^{512}}$$

Sau đó, ta dùng một lớp Fully Connected để chiếu xuống không gian đặc trưng 256 chiều, kết quả là vector đặc trưng ảnh $\mathrm{F'_I}$ được tính như sau:

$$\mathrm{F'_I} = \sigma\,(\mathrm{W_I F_I} + \mathrm{b_I}) \qquad (1)$$

Trong đó σ là hàm kích hoạt (ReLU), $\mathrm{W_I}$ và $\mathrm{b_I}$ là trọng số và bias của lớp tuyến tính.

Text Encoder

Câu hỏi đầu vào được biểu diễn dưới dạng chuỗi các từ $\mathrm{Q} = (\mathrm{q_1}, q_2, \dots, q_T)$, trong đó T là số từ trong câu. Mỗi từ q_t được ánh xạ thành vector qua lớp nhúng (embedding):

$$\mathrm{E_t} = \mathrm{Embedding(q_t)}.$$

Sau đó, đưa embedding qua một LSTM hai chiều (bidirectional LSTM) để khai thác thông tin ngữ cảnh. Tại mỗi thời điểm t, LSTM tạo ra một vector ẩn $\mathrm{H_t} \in \mathrm{R^{2h}}$ (vì là LSTM 2 chiều):

$$\mathrm{H_t} = \mathrm{LSTM}\,(\mathrm{E_t}, \mathrm{H_{t-1}}) \in \mathrm{R^{2h}} \qquad (2)$$

Nếu sử dụng attention, mô hình sẽ học trọng số α_t cho từng thời điểm t để tập trung vào các từ quan trọng. Cụ thể:

$$\alpha_t = \mathrm{softmax}(\mathrm{H_t^T w_\alpha}), \quad \mathrm{F_Q} = \sum \alpha_t \mathrm{H_t} \quad (3)$$

Trong đó w_α là vector tham số học được, α_t là trọng số attention tại thời điểm t, và F_Q là vector tổng hợp toàn câu. Sau đó, dùng một lớp fully connected để giảm chiều về 256:

$$F'_Q = \sigma(W_Q F_Q + b_Q) \in R^{256} \quad (4)$$

Fusion và Classification

Hai đặc trưng ảnh F'_I và văn bản F'_Q được ghép lại thành một vector:

$$F = [F'_I; F'_Q] \in R^{512}$$

Sau đó, vector F được đưa qua fully connected để giảm còn 256 chiều:

$$F' = \sigma\,(W_F F + b_F) \in R^{256} \quad (5)$$

Cuối cùng, F′ được đưa vào lớp phân loại (classifier) để dự đoán nhãn đầu ra:

$$\hat{y} = \text{Softmax}(W_C F' + b_C) \quad (6)$$

Trong đó, ŷ là vector xác suất dự đoán trên các lớp đầu ra.

2.3 So sánh

Model	Loss	Accuracy
MobileNet_v2 + LSTM Attention	0.4503	0.8317
MobileNet_v2 + LSTM no Attention	0.4682	0.8288
CNN + LSTM Attention	0.5589	0.8032
CNN + LSTM no Attention	0.6600	0.7703

Bảng 2.1 So sánh 4 mô hình

Dựa trên kết quả thu được, MobileNet cho thấy hiệu suất vượt trội hơn so với CNN trong cả hai trường hợp có và không có attention. Cụ thể, MobileNet có độ chính xác cao nhất (83.17%) và mức loss thấp nhất (0.4503), trong khi phiên bản không có attention vẫn duy trì hiệu suất tốt với accuracy 82.88% và loss 0.4682. Ngược lại, CNN có độ chính xác thấp hơn đáng kể, đặc biệt khi không sử dụng attention, chỉ đạt accuracy 77.03% và loss cao nhất (0.6600). Khi bổ sung attention, CNN có sự cải thiện đáng kể, với accuracy tăng lên 80.32% và loss giảm còn 0.5589.

Kết quả này cho thấy attention giúp cải thiện hiệu suất của cả hai mô hình bằng cách tập trung vào các đặc trưng quan trọng trong ảnh, giảm nhiễu và nâng cao khả năng phân loại. Tuy nhiên, MobileNet chỉ có sự gia tăng nhẹ về độ chính xác khi thêm attention, trong khi CNN hưởng lợi rõ rệt hơn.

Tóm lại, MobileNet với attention là mô hình tốt nhất trong các thử nghiệm, đạt độ chính xác tối ưu và tổn thất thấp nhất. Nếu cần lựa chọn mô hình để triển khai thực tế, đây sẽ là phương án hợp lý nhất, trong khi CNN không có attention tỏ ra kém hiệu quả nhất.

TÀI LIỆU THAM KHẢO

Agrawal, A., Lu, J., Antol, S., Mitchell, M., Zitnick, C. L., Batra, D., & Parikh, D. (2016). *VQA: Visual Question Answering* (No. arXiv:1505.00468). arXiv. https://doi.org/10.48550/arXiv.1505.00468

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., & Zhang, L. (2018). *Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering* (No. arXiv:1707.07998). arXiv. https://doi.org/10.48550/arXiv.1707.07998

Brownlee, J. (2017, Tháng Sáu 29). Attention in Long Short-Term Memory Recurrent Neural Networks. *MachineLearningMastery.Com*. https://www.machinelearningmastery.com/attention-long-short-term-memory-recurrent-neural-networks/

Understanding LSTM Networks—Colah's blog. (không ngày). Truy vấn 23 Tháng Ba 2025, từ https://colah.github.io/posts/2015-08-Understanding-LSTMs/

What is LSTM - Long Short Term Memory? (19:24:13+00:00). GeeksforGeeks. https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

Anderson, P., et al. (2018). *Bottom-Up and Top-Down Attention for Image Captioning and VQA*. CVPR.

Lu, J., et al. (2016). *Hierarchical Question-Image Co-Attention for Visual Question Answering*. NeurIPS. arXiv:1606.00061

Baltrusaitis, T., Ahuja, C., & Morency, L. P. (2019). *Multimodal Machine Learning: A Survey and Taxonomy*. IEEE TPAMI, 41(2), 423-443.

Yu, Z., et al. (2017). *Multi-modal Factorized Bilinear Pooling with Co-Attention Learning for VQA*. ICCV. arXiv:1708.01471

https://how.dev/answers/what-is-multimodal-fusion

Visual Question Answering: Datasets, Algorithms, and Future Challenges https://arxiv.org/pdf/1610.01465

A Systematic Review of Intermediate Fusion in Multimodal Deep Learning for Biomedical Applications https://arxiv.org/pdf/2408.02686

BẢNG PHÂN CÔNG CÔNG VIỆC

Họ và tên	MSSV	Đánh giá (%)
Lưu Hữu Trí	52200167	100
Hồ Thu Yến Ngọc	52200149	100
Nguyễn Quang Trung	52200193	100