TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ MÔN

HỌC SÂU

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

Người thực hiện: **NGUYỄN QUỐC DUY – 52200196**

NGUYỄN HOÀNG ÂN – 52200183

NGUYỄN NHẬT TRƯỜNG – 52200192

Lớp **: 22050301**

Khoá **: 26**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CUỐI KÌ MÔN

HỌC SÂU

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

Người thực hiện: **NGUYỄN QUỐC DUY - 52200196**

NGUYỄN HOÀNG ÂN – 52200183

NGUYỄN NHẬT TRƯỜNG – 52200192

Lớp **: 22050301**

Khoá **: 26**

THÀNH PHỐ HỒ CHÍ MINH NĂM 2024

LỜI CẢM ƠN

Trước tiên, chúng em xin gửi lời cảm ơn chân thành và lòng biết ơn sâu sắc đến ***PGS.TS. Lê Anh Cường***. Thầy là người đã luôn hỗ trợ và hướng dẫn tận tình cho chúng em trong suốt quá trình nghiên cứu và hoàn thành bài báo cáo của môn "***Học Sâu***".

Tiếp theo, chúng em xin gửi lời cảm ơn đến **Khoa Công nghệ thông tin - Trường Đại học Tôn Đức Thắng** vì đã tạo điều kiện cho chúng em được học tập và nghiên cứu môn học này. Khoa đã luôn sẵn sàng chia sẻ các kiến thức bổ ích cũng như chia sẻ các kinh nghiệm tham khảo tài liệu, giúp ích không chỉ cho việc thực hiện và hoàn thành đề tài nghiên cứu mà còn giúp ích cho việc học tập và rèn luyện trong quá trình thực hành tại **trường Đại học Tôn Đức Thắng**.

Cuối cùng, sau khoảng thời gian học tập trên lớp chúng em đã hoàn tất bài báo cáo nhờ vào sự hướng dẫn, giúp đỡ và những kiến thức học hỏi được từ Quý thầy cô. Do giới hạn về mặt kiến thức và khả năng lý luận nên tôi vẫn còn nhiều thiếu sót và hạn chế, kính mong sự chỉ dẫn và đóng góp của Quý thầy cô giáo để bài báo cáo của chúng em được hoàn thiện hơn. Hơn nữa, nhờ những góp ý từ thầy cô và các bạn hữu, chúng em sẽ hoàn thành tốt hơn ở những bài nghiên cứu, tiểu luận trong tương lai.

Chúng em xin chân thành cảm ơn!

ĐỒ ÁN ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng em xin cam đoan đây là sản phẩm đồ án của riêng chúng em và được sự hướng dẫn 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 đồ 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 chúng em xin hoàn toàn chịu trách nhiệm về nội dung đồ 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 09 tháng 03 năm 2025

Tác giả

(ký tên và ghi rõ họ tên)

Nguyễn Quốc Duy

Nguyễn Hoàng Ân

Nguyễn Nhật Trường

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)

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH ẢNH

CHƯƠNG 1

CHƯƠNG 2

CHƯƠNG 5

DANH MỤC BẢNG

CHƯƠNG 4

CHƯƠNG 5

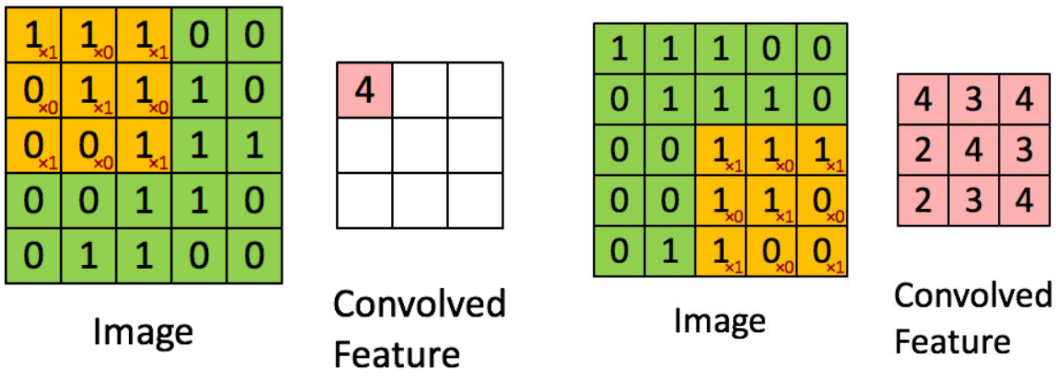
CHƯƠNG 1 – CƠ SỞ LÝ THUYẾT

1.1 Tổng quan về Convolutional Neural Networks (CNN) và Long Short-Term Memory(LSTM)

1.1.1 Giới thiệu về Convolutional Neural Networks(CNN)

1.1.1.1 Convolutional là gì?

Là một cửa sổ trượt (Sliding Windows) trên một ma trận.



Hình 1. 1 Hoạt động của Convolutional

Convolutional là một loại cửa sổ dạng trượt nằm trên một ma trận. Các convolutional layer sẽ chứa các parameter có khả năng tự học, qua đó sẽ điều chỉnh và tìm ra cách lấy những thông tin chính xác nhất trong khi không cần chọn feature. Lúc này, convolution hay tích chập đóng vai trò là nhân các phần tử thuộc ma trận. Sliding Window, hay được gọi là kernel, filter hoặc feature detect, là loại ma trận có kích thước nhỏ.

Đây chính là lớp đóng vai trò mấu chốt của CNN, khi layer này đảm nhiệm việc thực hiện mọi tính toán. Stride, padding, filter map, feature map là những yếu tố quan trọng nhất của convolutional layer.

Lớp Convolution sử dụng bộ lọc W để trích xuất đặc trưng:

$$X' = X * W + b$$

Trong đó:

- X là ảnh đầu vào
- W là bộ lọc (kernel)
- b là bias

Quá trình trượt các bộ lọc thường có các giá trị được quy định bao gồm:

- padding: quy định bộ đệm của bộ lọc hay chính là phần màu xám được thêm vào ảnh
- stride: quy định bước nhảy trong quá trình thực hiện.

Convolution hay tích chập là nhân từng phần tử trong ma trận 3. Sliding Window hay còn gọi là kernel, filter hoặc feature detect là một ma trận có kích thước nhỏ như trong ví dụ trên là 3×3.

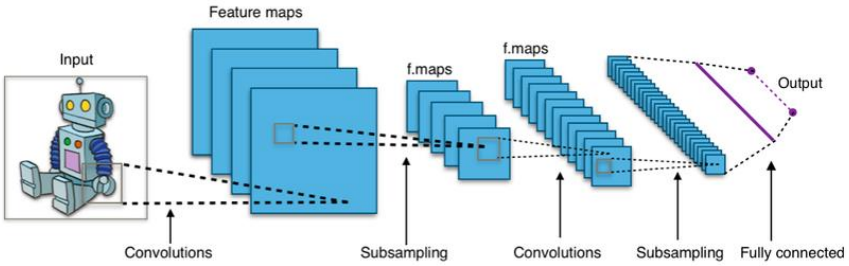
Convolution hay tích chập là nhân từng phần tử bên trong ma trận 3×3 với ma trận bên trái. Kết quả được một ma trận gọi là Convoled feature được sinh ra từ việc nhận ma trận Filter với ma trận ảnh 5×5 bên trái.

1.1.1.2 Pooling Layer

Trong mô hình CNN có 2 khía cạnh cần quan tâm là **tính bất biến** (Location Invariance) và **tính kết hợp** (Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các gốc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể.

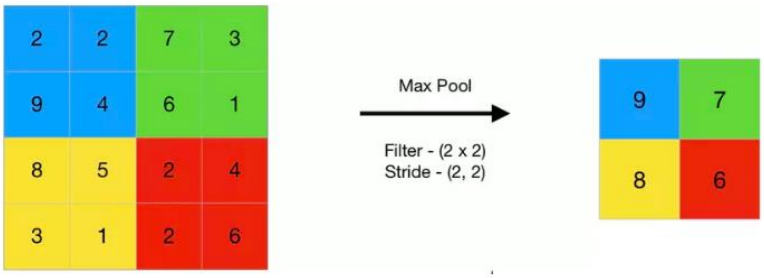
Lớp Pooling chịu trách nhiệm giảm kích thước không gian của Convolved Feature. Điều này nhằm giảm sức mạnh tính toán cần thiết để xử lý dữ liệu thông qua việc giảm chiều.

Hơn nữa, nó hữu ích để trích xuất các tính năng chính.



Hình 1. 2 Quá trình xử lý trong mạng nơ-ron tích chập

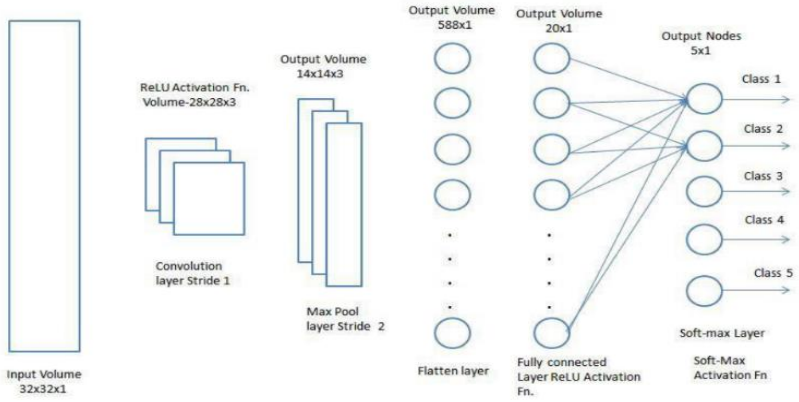
Lớp pooling(Maxpool) thường được sử dụng ngay sau lớp convulational để đơn giản hóa thông tin đầu ra để giảm bớt số lượng neuron.



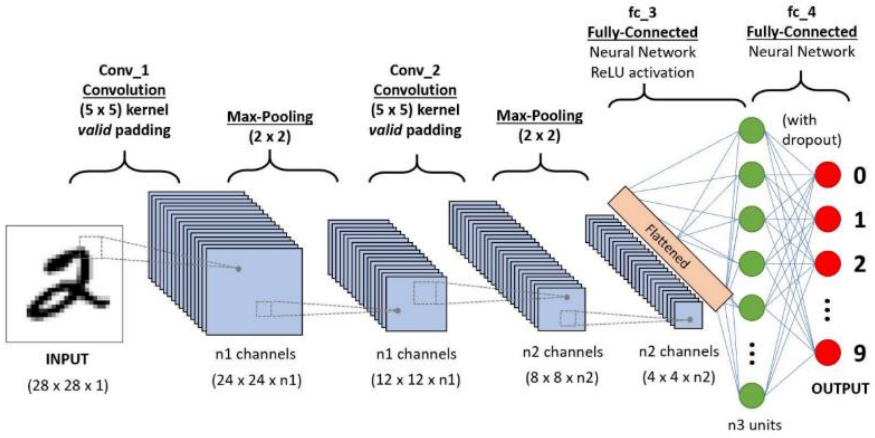
Hình 1. 3 Quá trình xử lí của hàm Maxpool

1.1.1.3 Fully Connected Layer

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh thì tensor của output của layer cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng nơ-ron. Với FC layer được kết hợp với các tính năng lại với nhau để tạo ra một mô hình. Cuối cùng sử dụng softmax hoặc sigmoid để phân loại đầu ra.



Hình 1. 4 Quá trinh CNN trích xuất đặc trưng



Hình 1. 5 Ví dụ quá trình trích xuất đặc trưng con số

1.1.1.4 Convolutional Neural Networks

CNN là gì? Hay còn gọi là mạng nơ-ron tích chập (CNN) chỉ đơn giản là mạng nơ-ron sử dụng phép toán tích chập, một loại phép toán tuyến tính chuyên biệt.

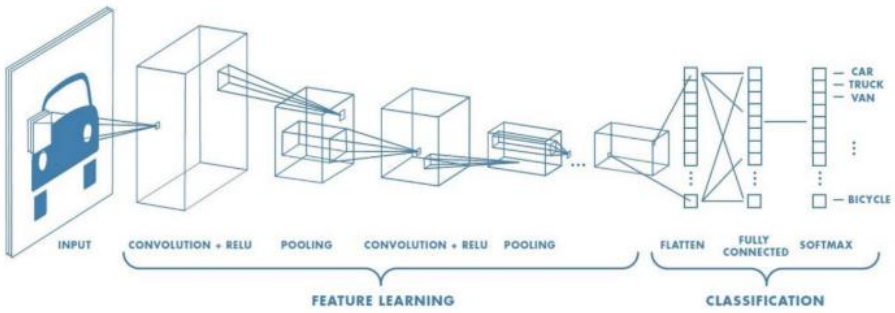
CNN là phiên bản được chuẩn hóa của perceptron nhiều lớp.

CNN tận dụng mô hình phân cấp trong dữ liệu và lắp ráp các mô hình phức tạp hơn bằng cách sử dụng các mô hình nhỏ hơn và đơn giản hơn.

CNN sử dụng các bộ lọc tích chập (convolution) để tự động trích xuất đặc trưng từ dữ liệu đầu vào.

Mạng CNN là một tập hợp các lớp Convolution chồng lên nhau và sử dụng các hàm nonlinear activation như ReLU và tanh để kích hoạt các trọng số trong các node. Mỗi một lớp sau khi thông qua các hàm kích hoạt sẽ tạo ra các thông tin trừu tượng hơn cho các lớp tiếp theo.

Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Như vậy mỗi neuron ở lớp kế tiếp sinh ra từ kết quả của filter áp đặt lên một vùng ảnh cục bộ của neuron trước đó.

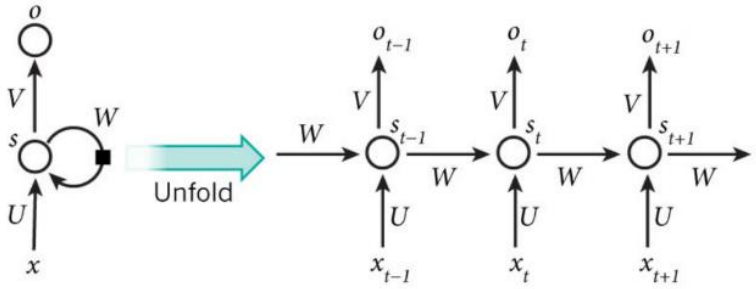


Hình 1. 6 Cấu trúc mô hình Convolutional Neural Networks(CNN)

1.1.2 Long Short-Term Memory(LSTM)

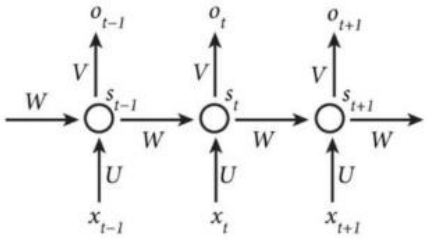
Mạng trí nhớ ngắn hạn định hướng dài hạn còn được viết tắt là LSTM làm một kiến trúc đặc biệt của RNN có khả năng học được sự phục thuộc trong dài hạn (long-term dependencies). Kiến trúc này đã được phổ biến và sử dụng rộng rãi cho tới ngày nay. LSTM khắc phục được rất nhiều những hạn chế của RNN về triệt tiêu đạo hàm. Tuy nhiên cấu trúc có phần phức tạp hơn nhưngvẫn dữ được tư tưởng chính của RNN là sự sao chép các kiến trúc theo dạng chuỗi.

Mô hình RNN có dạng:



Hình 1. 7 Mô hình mạng nơ-ron hồi quy(RNN)

1.1.2.1 Sơ lược về RNN(Recurrent Neural Network):



$$s_t = f(Ux_t + Ws_{t-1}).$$

$$o_t = \text{softmax}(Vs_t).$$

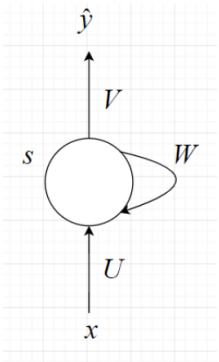
Hình 1. 8 Quá trình cập nhật trạng thái

- x_t là đầu vào tại thời điểm bước t. Ví dụ, x_1 có thể là một vectơ one-hot tương ứng với từ thứ hai của một câu.
- s_t là trạng thái ẩn tại thời điểm bước t. Đó là "bộ nhớ" của mạng. s_t được tính toán dựa trên trạng thái ẩn trước đó và đầu vào tại bước hiện tại: $s_t = f(U_x) + Ws_{t-1}$. Hàm f thường là một hàm phi tuyến tính(hàm kích hoạt) như tanh hoặc ReLU. s -1, cần thiết để tính trạng thái ẩn đầu tiên, thường được khởi tạo thành tất cả các số không.

- o_t là đầu ra tại bước t. Ví dụ, nếu chúng ta muốn dự đoán từ tiếp theo trong một câu, thì đó sẽ là một vectơ xác suất trên toàn bộ vốn từ vựng của chúng ta.

$$o_t = soft\max(Vs_t)$$

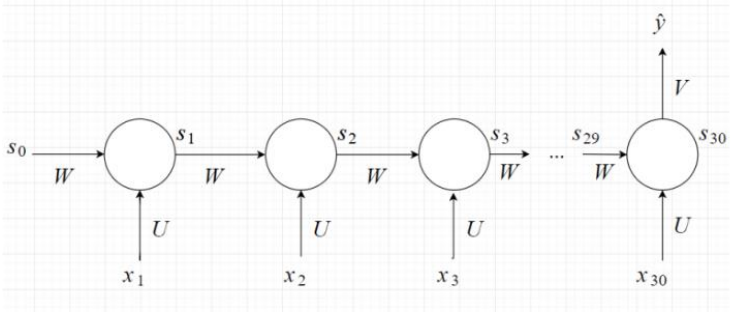
Ta thấy ở mỗi state các hệ số W, U là giống nhau nên model có thể được viết lại thành:



Hình 1. 9 Mô hình RNN rút gọn

Backpropagation Through Time (BPTT): Lan truyền ngược theo thời gian.

Ví dụ: Ta có mô hình có 30 input và 1 output, các input được cho vào model đúng với thứ tự ảnh trong video $x_1, x_2, ...x_{30}$.



Hình 1. 10 Ví dụ RNN thực thi 30 bước

Để thực hiện gradient descent, ta cần tính: $\frac{\partial L}{\partial U}, \frac{\partial L}{\partial V}, \frac{\partial L}{\partial W}$

L: Loss Function

Tính đạo đạo hàm với V: $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial \gamma} \cdot \frac{\partial \gamma}{\partial V}$

Tính đào hàm với U, W: $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \gamma} \cdot \frac{\partial \gamma}{\partial s_{30}} \cdot \frac{\partial s_{30}}{\partial W}$

Đạo hàm của L với W ở state thứ i : $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial \gamma} \cdot \frac{\partial \gamma}{\partial s_{30}} \cdot \frac{\partial s_{30}}{\partial s_i} \cdot \frac{\partial s_i'}{\partial W}$

Trong đó : $\frac{\partial s_{30}}{\partial s_i} = \prod_{j=i}^{29} \frac{\partial s_{j+1}}{\partial s_j}$

Giả sử hàm kích hoạt là hàm tanh: $s_t = \tanh(U \cdot x_t + W \cdot s_{t-1})$

$$\frac{\partial s_t}{\partial s_{t-1}} = (1 - s_t^2) \cdot W \Rightarrow \frac{\partial s_{30}}{\partial s_i} = W^{30-i} \cdot \prod_{j=i}^{29} (1 - s_j^2)$$

Ta có: $s_j < 1, W < 1$, với những state ở xa thì $\frac{\partial s_{30}}{\partial s_i} \approx 0 \; hay \; \frac{\partial L}{\partial W} \approx 0$

=> Vanishing Problem, ta có thể thấy là các state càng xa ở trước đó thì càng bị vanishing gradient và các hệ số không được update với các frame ở xa. Hay nói cách khác là RNN không học được từ các thông tin ở trước đó xa do vanishing gradient.

Kết luận:

Trong quá trình lan truyền ngược, mạng nơ-ron hồi quy gặp phải vấn đề về vanishing gradient. Gradient là các giá trị được sử dụng để cập nhật trọng số của mạng nơ-ron. Vấn đề về vanishing gradien là khi gradient co lại khi lan truyền ngược theo thời gian. Nếu giá trị gradient trở nên cực kỳ nhỏ, nó sẽ không đóng góp quá nhiều vào việc học.

Như vậy về lý thuyết là RNN có thể mang thông tin từ các layer trước đến các layer sau, nhưng thực tế là thông tin chỉ mang được qua một số lượng state nhất định, sau đó thì sẽ bị vanishing gradient, hay nói cách khác là model chỉ học được từ các state gần nó => short term memory. Cần một mô hình mới để giải quyết vấn đề này => Long short - term memory (LSTM) ra đời.

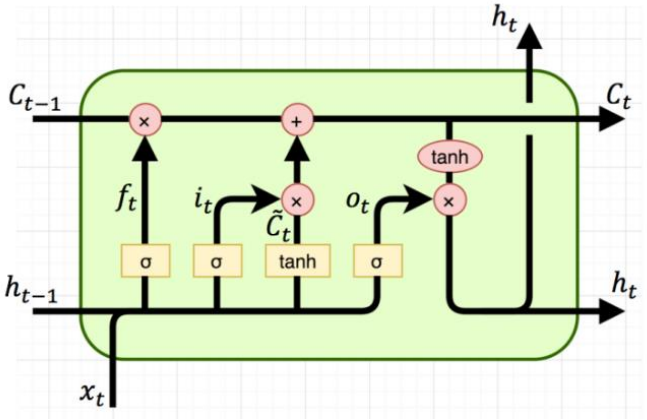
1.1.2.2 Long Short-Term Memory

So với RNN thì Long Short-Term Memory thì:

- Thêm cơ chế quên.

- Thêm cơ chế lưu.
- Khi mô hình nhìn thấy một hình ảnh mới, nó cần tìm hiểu xem có thông tin nào về hình ảnh đó đáng để sử dụng và lưu hay không.

LSTM cũng có một chuỗi tương tự như RNN nhưng phần kiến trúc lặp lại có cấu trúc khác biệt hơn. Thay vì chỉ có một tầng đơn, chúng có tới 4 tầng ẩn (3 sigmoid và 1 tanh) tương tác với nhau theo một cấu trúc đặc biệt.



Hình 1. 11 Mô hình Long short - term memory (LSTM)

Trong đó:

- Output: C_t, h_t, ta gọi c là cell state, h là hidden state.
- Input: C_{t-1}, h_{t-1}, x_t, input là x_t ở state thứ t. C_{t-1}, $h_t - 1$ là ouput của layer trước. h đóng vai trò khá giống như s ở RNN, trong khi c là điểm mới của LSTM.
- f_t, i_t, o_t : tương ứng với forget gate, input gate và output gate.
- $\overline{C_t} = \tanh(U_c \cdot x_t + W_c \cdot h_{t-1} + b_c)$ l hàm kích hoạt tương tự s_t ở RNN.

Forget gate: $f_t = \sigma(U_f \cdot x_t + W_f \cdot h_{t-1} + b_f)$

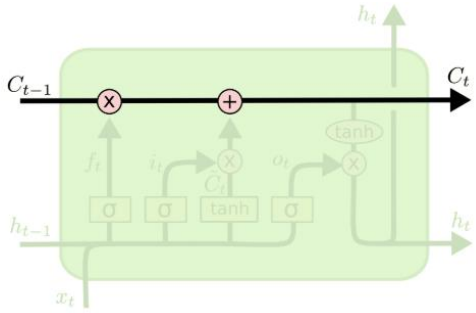
Input gate: $i_t = \sigma(U_i \cdot x_t + W_i \cdot h_{t-1} + b_i)$

Output gate: $o_t = \sigma(U_o \cdot x_t + W_o \cdot h_{t-1} + b_o)$

Giải thích:

- $0 < f_t, i_t, o_t < 1$.
- b_f, b_i, b_o là các hệ số bias.
- U, W là hệ số tương tự như ở mô hình RNN.

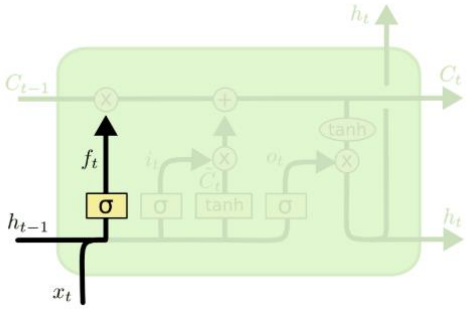
Trạng thái cell state: ý tưởng chính của LSTM là thành phần ô trạng thái (cell state) được thể hiện qua đường chạy ngang qua đỉnh đồ thị như hình vẽ bên dưới:



Hình 1. 12 Trạng thái cell state

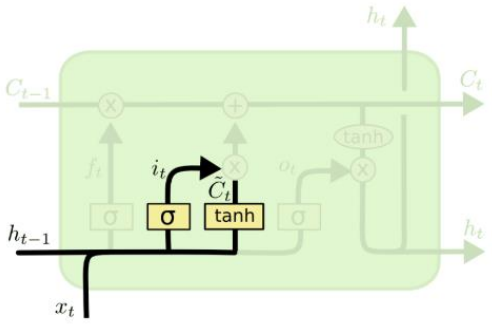
Thứ tự các bước của LSTM:

- **Bước 1:** Đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state). Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer). Nhận vào 2 giá trị là h_{t-1} và x_t và trả về một giá trị nằm trong khoảng [0,1] cho mỗi giá trị của ô trạng thái. Nếu giá trị bằng 1 thể hiện 'giữ toàn bộ thông tin' và bằng 0 thể hiện 'bỏ qua toàn bộ chúng'.



Hình 1. 13 Tầng cổng quên(Forget gate layer)

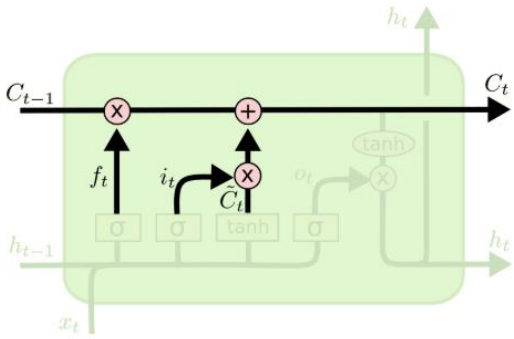
- **Bước 2:** Ở bước này sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Một tầng ẩn của hàm sigmoid được gọi là tầng cổng vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Tầng ẩn hàm tanh sẽ tạo ra một véc tơ của một giá trị trạng thái mới $\overline{C_t}$. Tiếp theo kết hợp kết quả của 2 tầng này để tạo thành một cập nhật cho trạng thái.



Hình 1. 14 Cập nhật giá trị cho ô trạng thái

- **Bước 3:** Đây là thời điểm để cập nhật một ô trạng thái cũ, C_{t-1} sang một trạng thái mới C_t.

  Ta có: $C_t = f_t \cdot C_{t-1} + i_t \cdot \overline{C_t}$



Hình 1. 15 Ô trạng thái mới

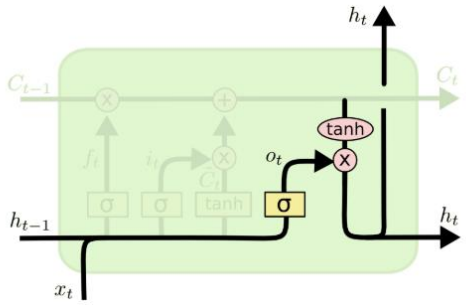
- **Bước 4:** Kết quả ở đầu ra sẽ dựa trên ô trạng thái. Đầu tiên, chúng ta chạy qua một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về khoảng -1 và 1) và nhân

nó với đầu ra của một cổng sigmoid, do đó chỉ trả ra phần mà chúng ta quyết định.

Ta có:

$$o_t = \sigma(U_o \cdot x_t + W_o \cdot h_{t-1} + b_o)$$

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



Hình 1. 16 Điều chỉnh thông tin ở đầu ra thông qua hàm tanh

LSTM chống vanishing gradient:

Ta cũng áp dụng thuật toán back propagation through time cho LSTM tương tự như RNN.

Thành phần chính gây là vanishing gradient trong RNN là: $\frac{\partial s_{t+1}}{\partial s_t} = (1 - s_t^2) \cdot W$, trong đó st, W < 1

Tương tự trong LSTM ta quan tâm đến $\frac{\partial c_t}{\partial c_{t-1}} = f_t$. Do $0 < f_t < 1$ nên về cơ bản thì LSTM vẫn bị vanishing gradient nhưng bị ít hơn so với RNN. Hơn thế nữa, khi mang thông tin trên cell state thì ít khi cần phải quên giá trị cell cũ, nên $f_t \approx 1$ => Tránh được vanishing gradient.

1.2 Tổng quan về cơ chế Attention

1.2.1 Encoder và decoder

Trong các bài toán xử lý ngôn ngữ tự nhiên (NLP) và thị giác máy tính (CV), mô hình thường không thể làm việc trực tiếp với dữ liệu thô như văn bản, hình ảnh hay âm thanh. Thay vào đó, chúng cần một cơ chế để chuyển đổi dữ liệu đầu vào thành dạng biểu diễn số hóa có thể học được. Đó chính là vai trò của **Encoder** và **Decoder**.

- **Encoder:** là quá trình chuyển đổi dữ liệu đầu vào thành một dạng biểu diễn có ý nghĩa trong không gian đặc trưng, giúp mô hình có thể hiểu và học được.
    - Trong **mạng nơ-ron truyền thống (Neural Network)**, Encoder thường là các hidden layers, nơi dữ liệu được ánh xạ sang không gian đặc trưng có ý nghĩa hơn.
    - Trong **mạng CNN (Convolutional Neural Networks)**, Encoder là chuỗi các lớp **Convolutional + MaxPooling**, giúp trích xuất đặc trưng từ hình ảnh.
    - Trong **mạng RNN (Recurrent Neural Networks)**, Encoder bao gồm các lớp **Embedding + RNN (LSTM, GRU, Transformer)**, giúp mô hình học được thông tin tuần tự từ văn bản hoặc chuỗi thời gian.
- **Decoder:** nhận đầu ra từ Encoder và thực hiện quá trình chuyển đổi ngược lại để tạo ra kết quả mong muốn.
    - Trong bài toán **dịch máy**, đầu ra của Encoder là vector ngữ nghĩa, và Decoder sẽ giải mã thành chuỗi văn bản đích.
    - Trong **hệ thống hỏi đáp hình ảnh (VQA)**, Encoder trích xuất đặc trưng từ hình ảnh, và Decoder sử dụng thông tin đó để sinh ra câu trả lời phù hợp.
    - Trong **mô hình tạo văn bản**, Decoder có thể dự đoán từng từ trong câu theo thứ tự thời gian.

1.2.2 Seq2Seq model

Seq2Seq (Sequence-to-Sequence) là một mô hình mạng nơ-ron nhân tạo được sử dụng phổ biến trong các bài toán xử lý chuỗi, đặc biệt là dịch máy, tạo phụ đề tự động và tóm tắt văn bản. Mô hình này có thể biến đổi một chuỗi đầu vào có độ dài bất kỳ thành một chuỗi đầu ra có độ dài khác bằng cách sử dụng hai thành phần chính: **Encoder và Decoder**.

- **Kiến trúc của Seq2Seq:** Mô hình Seq2Seq bao gồm hai phần chính:

- **Encoder**: Tiếp nhận một chuỗi đầu vào và mã hóa nó thành một biểu diễn cố định gọi là **context vector** (vector ngữ cảnh).
- **Decoder**: Nhận context vector từ Encoder và tạo ra từng phần tử của chuỗi đầu ra theo từng bước thời gian.

Ban đầu, Seq2Seq được xây dựng dựa trên **Recurrent Neural Networks (RNNs)** và các biến thể mạnh hơn như **Long Short-Term Memory (LSTM)** hoặc **Gated Recurrent Units (GRU)**.

- **Cách hoạt động của Seq2Seq:** Quá trình hoạt động của Seq2Seq có thể được chia thành hai bước chính:
- **Bước 1: Encoding**
  - Chuỗi đầu vào được đưa vào mạng RNN, LSTM hoặc GRU.
  - Tại mỗi bước thời gian, một **hidden state** được cập nhật dựa trên hidden state trước đó và đầu vào hiện tại.
  - Sau khi toàn bộ chuỗi đầu vào được xử lý, hidden state cuối cùng của Encoder sẽ được sử dụng làm context vector.
- **Bước 2: Decoding**
  - Context vector được dùng làm hidden state ban đầu cho Decoder.
  - Ở mỗi bước thời gian, Decoder dự đoán phần tử tiếp theo trong chuỗi đầu ra dựa trên hidden state hiện tại và phần tử đầu ra trước đó.
  - Quá trình tiếp tục cho đến khi đạt được token kết thúc (ví dụ: <EOS> trong dịch máy).

1.2.3 Cơ chế Attention

1.2.3.1 Scale dot product attention

Scale Dot Product Attention là một cơ chế cốt lõi trong Attention, được sử dụng rộng rãi trong các mô hình học sâu, đặc biệt trong các bài toán xử lý ngôn ngữ tự nhiên (NLP) và thị giác máy tính (CV). Cơ chế này giúp mô hình tập trung vào những phần quan trọng của dữ liệu đầu vào bằng cách tính toán mức độ liên quan giữa các phần tử.

Cơ chế **Scale Dot Product Attention** hoạt động dựa trên ba ma trận đầu vào:

- **Query (Q)**: Đại diện cho vector truy vấn.
- **Key (K)**: Đại diện cho vector khóa.
- **Value (V)**: Đại diện cho vector giá trị.

Quá trình tính toán Attention diễn ra qua các bước sau:

- **Bước 1: Tính điểm tương quan (Score Calculation)**: Điểm attention giữa từng cặp từ trong câu được tính bằng cách thực hiện phép nhân dot product giữa Query và Key:

  $$score = Q.K^T$$

  Phép nhân này giúp xác định mức độ liên quan giữa mỗi từ trong câu với các từ khác.
- **Bước 2: Chuẩn hóa điểm số bằng Softmax**: Để đưa các điểm số về dạng phân phối xác suất, ta áp dụng hàm Softmax:

  $$attention\_weights = softmax(\frac{Q.K^T}{\sqrt{d_k}})$$

  Trong đó d_k, là số chiều của vector Key, giúp tránh trường hợp giá trị attention quá lớn khi số chiều tăng cao.
- **Bước 3: Tính toán Attention Output**: Ma trận attention weights được nhân với ma trận Value để tạo ra đầu ra cuối cùng:

  $$output = attention\_weights * V$$

  Vector đầu ra này sẽ được sử dụng trong quá trình tiếp theo của mô hình.

1.2.3.2 Multi-head Attention

Multi-head Attention mở rộng cơ chế **Scale Dot Product Attention** bằng cách sử dụng nhiều đầu attention song song để học các mối quan hệ khác nhau trong dữ liệu. Quy trình Multi-head Attention:

1. **Tạo nhiều ma trận Query, Key và Value:** Đầu vào được chiếu thành nhiều nhóm trọng số khác nhau, mỗi nhóm tạo ra một cặp *Q, K, V*.

2. **Tính Attention trên từng đầu (head):** Mỗi cặp *Q, K, V* được đưa vào cơ chế Scale Dot Product Attention để tạo ra ma trận attention tương ứng.
3. **Ghép nối (concatenate) các đầu Attention:** Các ma trận attention từ các đầu được nối lại theo chiều kênh (channel) để tạo thành một ma trận tổng hợp.
4. **Dự đoán đầu ra:** Ma trận tổng hợp được nhân với ma trận trọng số W^O để đưa về kích thước mong muốn.

Công thức toán học của Multi-head Attention:

$$MultiHead(Q, K, V) = Concat(head_1, head_2, \dots head_n)W^O$$

Với mỗi $head_i$,:

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

Trong đó:

- QW_i^Q, KW_i^K, VW_i^V là ma trận trọng số cho từng đầu attention.
- W^O là ma trận trọng số cuối cùng dùng để chiếu ma trận attention tổng hợp về kích thước ban đầu.

Nhờ Multi-head Attention, mô hình có thể học được nhiều ngữ cảnh khác nhau trong cùng một thời điểm, giúp cải thiện hiệu quả của attention so với khi chỉ sử dụng một đầu duy nhất.

1.2.3.3 Quá trình encoder và decoder trong Attention

Quá trình Encoder và Decoder trong Attention đóng vai trò quan trọng trong việc biến đổi đầu vào thành đầu ra trong các mô hình xử lý ngôn ngữ tự nhiên. Dưới đây là cách hoạt động của hai quá trình này trong hệ thống Seq2Seq có áp dụng Attention:

Quá trình Encoder: Encoder được thiết kế để tiếp nhận chuỗi đầu vào và biến đổi nó thành một tập hợp các vector biểu diễn có ý nghĩa. Trong mô hình Attention, quá trình Encoder gồm nhiều lớp xếp chồng lên nhau, mỗi lớp bao gồm hai thành phần chính:

- **Multi-head Attention Layer**: Thành phần này giúp mô hình học được mối quan hệ giữa các từ trong câu. Thông qua Scale Dot Product Attention, mỗi

từ trong câu có thể điều chỉnh trọng số chú ý của nó đến các từ khác, giúp hệ thống nắm bắt ngữ cảnh hiệu quả hơn.

- **Feed-forward Layer**: Sau khi trải qua quá trình Attention, dữ liệu đi qua một mạng nơ-ron truyền thẳng (fully connected feed-forward layer) để tăng cường khả năng biểu diễn. Để ổn định quá trình học, mỗi lớp trong Encoder sử dụng **cơ chế Residual Connection** kết hợp với **Layer Normalization** để duy trì thông tin và tránh mất mát tín hiệu.
- Quá trình trên được lặp lại **N lần** (thường là 6 lần trong mô hình Transformer), tạo thành một khối Encoder hoàn chỉnh.

Quá trình Decoder: Nhận đầu ra của Encoder và từng bước tạo ra chuỗi đầu ra mong muốn. Decoder cũng gồm nhiều lớp tương tự như Encoder, nhưng có một số khác biệt quan trọng:

- **Masked Multi-head Attention Layer**: Lớp Attention này hoạt động tương tự như Multi-head Attention nhưng có thêm cơ chế Masking, giúp đảm bảo rằng tại mỗi bước thời gian, mô hình chỉ có thể nhìn thấy những từ đã được tạo ra trước đó mà không bị ảnh hưởng bởi các từ trong tương lai.
- **Multi-head Attention với đầu ra từ Encoder**: Ở bước tiếp theo, đầu ra từ Encoder được đưa vào một lớp Attention thứ hai, giúp mô hình tập trung vào những phần quan trọng của câu đầu vào để đưa ra kết quả chính xác hơn.
- **Feed-forward Layer**: Sau các bước Attention, dữ liệu tiếp tục đi qua một mạng nơ-ron truyền thẳng để tinh chỉnh đầu ra.
- **Residual Connection và Layer Normalization**: Tương tự như trong Encoder, mỗi lớp trong Decoder cũng sử dụng cơ chế Residual Connection kết hợp với Layer Normalization để đảm bảo tính ổn định trong quá trình huấn luyện.

CHƯƠNG 2 – BÀI TOÁN VISUAL QUESTION ANSWERING (VQA)

2.1 Giới thiệu bài toán VQA

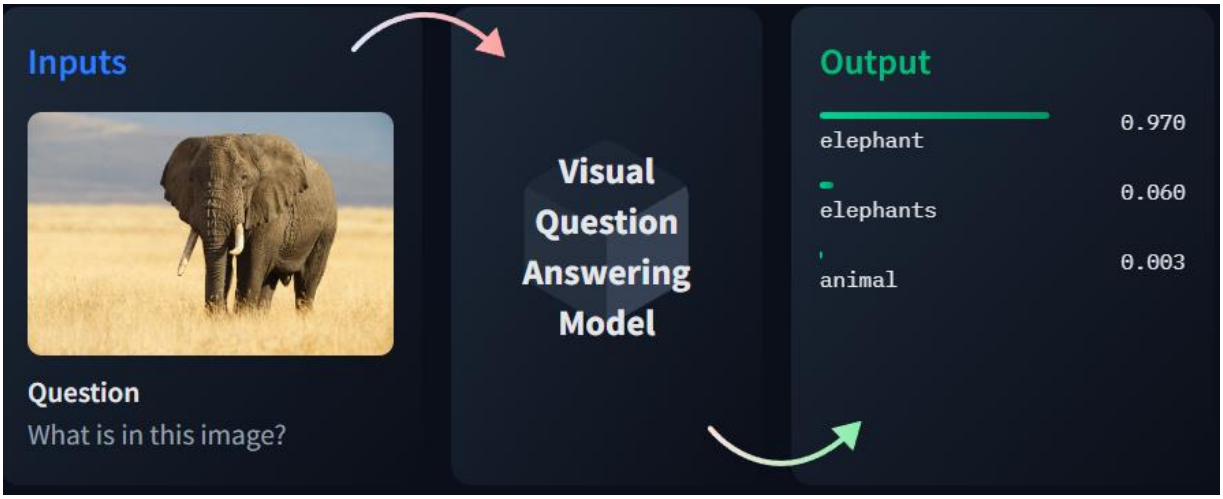
2.1.1 Định nghĩa bài toán

Visual Question Answering (VQA) là một bài toán trong lĩnh vực **Trí tuệ nhân tạo (AI)**, kết hợp giữa **Thị giác máy tính (Computer Vision)** và **Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP)**.

Cụ thể, bài toán VQA yêu cầu một mô hình AI có khả năng **trả lời câu hỏi về nội dung của một hình ảnh đầu vào**. Mô hình phải **hiểu cả hình ảnh và câu hỏi** để đưa ra câu trả lời chính xác.

Đầu vào & Đầu ra của bài toán VQA

- **Đầu vào**:
  - **Hình ảnh** (Image)
  - **Câu hỏi tự nhiên** liên quan đến hình ảnh (Question)
- **Đầu ra**:
  - **Câu trả lời chính xác** (Answer)



Hình 2. 1 Đầu vào và đầu ra của bài toán VQA

2.1.2 Tầm quan trọng của VQA

Bài toán **Visual Question Answering (VQA)** có ý nghĩa quan trọng không chỉ trong nghiên cứu trí tuệ nhân tạo mà còn trong nhiều ứng dụng thực tế. Dưới đây là một số lý do vì sao VQA đóng vai trò quan trọng:

- **Kết hợp giữa Thị giác máy tính và Xử lý ngôn ngữ tự nhiên:** VQA là một trong những bài toán hiếm hoi yêu cầu mô hình AI phải **hiểu đồng thời cả hình ảnh và ngôn ngữ**. Điều này giúp thúc đẩy nghiên cứu trong cả hai lĩnh vực **Computer Vision (CV)** và **Natural Language Processing (NLP)**, đồng thời phát triển các phương pháp **học đa phương thức (multi-modal learning)**.
- **Ứng dụng trong đời sống thực**
    - **Hỗ trợ người khiếm thị**: Các hệ thống VQA có thể giúp người khiếm thị hiểu nội dung hình ảnh bằng cách trả lời câu hỏi về hình ảnh đó. Ví dụ, một người dùng có thể hỏi: *"Trong ảnh có bao nhiêu người?"* và hệ thống sẽ trả lời dựa trên phân tích hình ảnh.
    - **Trợ lý ảo thông minh**: Các trợ lý ảo như Siri, Google Assistant có thể tích hợp VQA để giúp người dùng hiểu nội dung hình ảnh.
    - **Tìm kiếm bằng hình ảnh**: Thay vì nhập từ khóa, người dùng có thể đặt câu hỏi về một bức ảnh và nhận được câu trả lời chính xác.
- **Ứng dụng trong y tế**: VQA có thể được sử dụng trong **chẩn đoán hình ảnh y tế**, giúp bác sĩ phân tích hình ảnh X-quang, MRI, CT scan và cung cấp câu trả lời dựa trên dữ liệu y khoa. Ví dụ:
    - Có dấu hiệu tổn thương nào trong ảnh X-quang không?
    - Có một khối u nhỏ trong phổi.

- **Ứng dụng trong giám sát và an ninh:** Hệ thống VQA có thể hỗ trợ trong giám sát giao thông, an ninh công cộng bằng cách tự động phân tích hình ảnh từ camera và trả lời các câu hỏi như:
    - Có bao nhiêu xe đang vượt đèn đỏ?
    - Hai xe màu đen và một xe màu trắng.
- **Nâng cao khả năng tương tác giữa con người và máy tính:** VQA giúp máy móc giao tiếp với con người một cách **tự nhiên hơn**, thay vì chỉ nhận dạng hình ảnh đơn thuần, máy có thể **trả lời câu hỏi**, giúp người dùng có trải nghiệm trực quan và tiện lợi hơn.

2.1.3 Các thách thức chính

Mặc dù Visual Question Answering (VQA) là một lĩnh vực tiềm năng với nhiều ứng dụng quan trọng, nhưng nó cũng đặt ra nhiều **thách thức lớn**. Dưới đây là một số vấn đề quan trọng mà các nhà nghiên cứu cần giải quyết khi phát triển mô hình VQA:

- **Hiểu đồng thời cả hình ảnh và ngôn ngữ**
    - VQA không chỉ yêu cầu **xử lý hình ảnh** như các bài toán thị giác máy tính thông thường (nhận diện vật thể, phân loại ảnh) mà còn phải hiểu **ngôn ngữ tự nhiên**.
    - Mô hình cần **liên kết thông tin giữa hai miền dữ liệu khác nhau (hình ảnh và văn bản)**, điều này rất phức tạp vì cách biểu diễn thông tin của hai miền này là hoàn toàn khác nhau.
    - ***Ví dụ:*** **Câu hỏi:** "Có bao nhiêu con chó trong ảnh?"
      → Mô hình cần nhận diện chính xác đối tượng "chó" trong ảnh và đếm số lượng của chúng.
- **Sự đa dạng và mơ hồ của câu hỏi**
    - Một hình ảnh có thể có **nhiều loại câu hỏi khác nhau**, từ câu hỏi về đối tượng (*"Có bao nhiêu con mèo?"*), về màu sắc (*"Chiếc xe màu gì?"*),

đến câu hỏi mang tính suy luận (*"Người đàn ông trong ảnh đang làm gì?"*).

- Một số câu hỏi có thể **không rõ ràng hoặc có nhiều cách diễn giải khác nhau**, gây khó khăn cho mô hình trong việc đưa ra câu trả lời chính xác.
- ***Ví dụ:***
  - *"Người trong ảnh có hạnh phúc không?"* → Cần suy luận dựa trên cảm xúc khuôn mặt.
  - *"Có gì trên bàn?"* → Nếu có quá nhiều vật thể, mô hình có thể khó xác định thông tin nào là quan trọng.

- **Thiếu dữ liệu cân bằng và hiện tượng "học vẹt"**
  - Các mô hình VQA dễ **học theo xu hướng thống kê** thay vì thực sự hiểu câu hỏi.
  - Ví dụ, nếu trong tập dữ liệu có quá nhiều câu hỏi dạng *"Bầu trời màu gì?"* và câu trả lời phổ biến là *"Xanh"*, mô hình có thể trả lời *"Xanh"* ngay cả khi bầu trời trong ảnh có màu **cam lúc hoàng hôn**.
  - Các bộ dữ liệu như VQA v2 đã cố gắng giải quyết vấn đề này bằng cách tạo **các câu hỏi đối nghịch (counterexamples)** để buộc mô hình phải nhìn vào hình ảnh thay vì chỉ dựa vào thống kê.
  - ***Ví dụ:***
    - **Câu hỏi 1**: *"Có bao nhiêu quả táo trên bàn?"* → 3
    - **Câu hỏi 2 (counterexample)**: *"Có bao nhiêu quả chuối trên bàn?"* → 1
      → Nếu mô hình chỉ dựa vào thống kê, nó có thể đoán sai số lượng quả chuối.

- **Xử lý câu trả lời có nhiều cách diễn đạt khác nhau**

- Câu trả lời trong VQA có thể có nhiều cách diễn đạt khác nhau nhưng vẫn đúng. Điều này gây khó khăn cho việc **đánh giá độ chính xác** của mô hình.
- Các thước đo truyền thống như **Accuracy, BLEU, ROUGE** có thể không phản ánh đúng chất lượng mô hình trong các trường hợp này.
- ***Ví dụ:***
  - **Câu hỏi**: *"What is the man doing?"*
  - **Các câu trả lời hợp lệ**: *"He is playing soccer", "Playing football", "Kicking a ball".*
  - Mô hình cần hiểu rằng những câu trả lời này có nghĩa tương đương dù khác nhau về mặt ngôn ngữ.

2.2 Cấu trúc tổng quan của một hệ thống VQA

2.2.1 Đầu vào và đầu ra của hệ thống

Một hệ thống **Visual Question Answering (VQA)** hoạt động dựa trên việc **nhận diện thông tin từ hình ảnh và câu hỏi**, sau đó **tạo ra câu trả lời phù hợp**.

Đầu vào của hệ thống VQA

Hệ thống VQA nhận hai loại đầu vào chính:

1. **Hình ảnh (Image Input)**
   - Là một bức ảnh chứa thông tin mà câu hỏi đề cập đến.
   - Hình ảnh có thể thuộc nhiều lĩnh vực khác nhau, chẳng hạn như:
     - **Ảnh tự nhiên** (COCO, ImageNet)
     - **Ảnh y khoa** (CT scan, MRI)
     - **Ảnh bản đồ, vệ tinh**
   - Hệ thống cần **trích xuất đặc trưng từ ảnh** để hiểu nội dung trong ảnh.
2. **Câu hỏi (Question Input)**
   - Là một câu hỏi bằng **ngôn ngữ tự nhiên** liên quan đến hình ảnh.
   - Câu hỏi có thể thuộc nhiều loại:

        - **Câu hỏi về đối tượng** → *"Có bao nhiêu con mèo?"*
        - **Câu hỏi về thuộc tính** → *"Chiếc xe màu gì?"*
        - **Câu hỏi mang tính suy luận** → *"Người đàn ông đang làm gì?"*
    - Hệ thống cần **mã hóa câu hỏi thành biểu diễn số** để xử lý.

Ví dụ về đầu vào:

- **Hình ảnh**: Ảnh chụp một người đang cưỡi ngựa.

Hình 2. 2 Ví dụ ảnh chụp một người đang cưỡi ngựa.

- **Câu hỏi**: *"What is the color of the horse?"*

Đầu ra của hệ thống là một câu trả lời được mô hình dự đoán dựa trên **sự kết hợp giữa thông tin từ ảnh và câu hỏi**.

1. **Dạng đầu ra phổ biến:**
    - **Câu trả lời ngắn (Single-word answer)** → *"A chair"*
    - **Câu trả lời dài hơn (Phrase/Sentence answer)** → *"The dog is sitting on a chair."*
    - **Câu trả lời dạng số (Numerical answer)** → *"3"* (số lượng vật thể)

- **Câu trả lời Yes/No (Binary answer)** → *"Yes", "No"*

2. **Cách đánh giá đầu ra:**
   - **So khớp chính xác (Exact Match Accuracy)** → So sánh câu trả lời mô hình với câu trả lời thực.
   - **BLEU, ROUGE, METEOR** → Đánh giá dựa trên độ tương đồng giữa câu trả lời mô hình và câu trả lời thực tế.
   - **Human Evaluation** → Con người đánh giá tính hợp lý của câu trả lời.

Ví dụ về đầu ra của ví dụ trên:

- **Câu trả lời dự đoán**: *"Black and white."*

2.2.2 Các bước xử lý đầu vào trong hệ thống VQA

2.2.2.1 Xử lý hình ảnh

Hình ảnh đầu vào cần được chuyển thành dạng có thể sử dụng cho mô hình học sâu. Quá trình này bao gồm:

- **Tiền xử lý hình ảnh**: Điều chỉnh kích thước, chuẩn hóa pixel, và chuyển đổi định dạng.
- **Trích xuất đặc trưng**: Sử dụng **mạng CNN (ResNet, EfficientNet, etc.)** hoặc **transformer-based models (ViT, Swin Transformer, etc.)** để tạo vector biểu diễn hình ảnh.
- **Chú ý vùng ảnh quan trọng**: Một số mô hình VQA sử dụng **Attention Mechanism** để tập trung vào các vùng ảnh liên quan đến câu hỏi.

2.2.2.2 Xử lý câu hỏi

Câu hỏi đầu vào là văn bản, cần được chuyển thành dạng số để mô hình có thể hiểu và xử lý:

- **Tiền xử lý văn bản**: Loại bỏ ký tự đặc biệt, chuyển về chữ thường, tokenization.
- **Biểu diễn câu hỏi**: Mã hóa câu hỏi bằng **Word Embeddings (GloVe, Word2Vec)** hoặc sử dụng **Transformer-based models (BERT, T5, etc.)** để trích xuất đặc trưng ngữ nghĩa.
- **Mô hình hóa chuỗi (Sequence Modeling)**: Dùng **LSTM, GRU** hoặc Transformer để xử lý câu hỏi và tạo vector biểu diễn câu hỏi.

2.2.3 Xử lý đầu ra trong hệ thống VQA

2.2.3.1 Kết hợp thông tin hình ảnh và văn bản

Hình ảnh và câu hỏi được biểu diễn dưới dạng vector đặc trưng riêng biệt. Để mô hình hiểu và suy luận, hai nguồn thông tin này cần được kết hợp hiệu quả. Một số phương pháp phổ biến:

- **Concatenation (Ghép nối vector)**: Kết hợp đơn giản bằng cách ghép hai vector đặc trưng lại với nhau.
- **Attention Mechanism (Cơ chế chú ý)**: Giúp mô hình tập trung vào các vùng quan trọng trong ảnh tương ứng với câu hỏi.
- **Multimodal Fusion (Tổng hợp đa phương thức)**: Sử dụng các kỹ thuật như **MLB (Multimodal Low-rank Bilinear pooling)**, **MCB (Multimodal Compact Bilinear pooling)** hoặc **Transformer-based fusion** để kết hợp thông tin mạnh mẽ hơn.

2.2.3.2 Dự đoán câu trả lời

Sau khi kết hợp thông tin, hệ thống sử dụng một mô hình để sinh hoặc lựa chọn câu trả lời:

- **Mô hình phân loại (Classification-based VQA)**: Mô hình dự đoán câu trả lời từ một tập hợp giới hạn các câu trả lời khả thi.
- **Mô hình sinh văn bản (Generative-based VQA)**: Sử dụng kiến trúc **Seq2Seq, Transformer** để tạo ra câu trả lời mới, đặc biệt quan trọng trong các bài toán yêu cầu câu trả lời chi tiết.
- **Mô hình kết hợp (Hybrid Approaches)**: Kết hợp cả hai phương pháp trên để tận dụng điểm mạnh của từng phương pháp.

2.3 Phương pháp tiếp cận bài toán VQA

Một trong những phương pháp phổ biến để giải quyết bài toán **Visual Question Answering (VQA)** là sử dụng sự kết hợp giữa **mạng nơ-ron tích chập (CNN)** để trích xuất đặc trưng hình ảnh và **mạng LSTM** để xử lý câu hỏi. Kết hợp với **Attention**

Mechanism, mô hình có thể tập trung vào các vùng quan trọng của hình ảnh tương ứng với nội dung của câu hỏi.

Kiến trúc tổng quan:

- **Trích xuất đặc trưng hình ảnh (CNN):**
  - Sử dụng một mô hình CNN đã được huấn luyện trước như **ResNet** để trích xuất đặc trưng hình ảnh.
  - Kết quả đầu ra là một **vector đặc trưng hình ảnh** hoặc một **bản đồ đặc trưng (feature map)** chứa thông tin không gian.
- **Mã hóa câu hỏi (LSTM):**
  - Một mạng **LSTM** tiếp nhận các vector từ câu hỏi để tạo ra một **vector đặc trưng đại diện cho toàn bộ câu hỏi**.
- **Cơ chế Attention:**
  - Attention được sử dụng để tính trọng số giữa các vùng quan trọng của ảnh dựa trên thông tin từ câu hỏi.
  - Các trọng số này giúp mô hình tập trung vào các vùng có liên quan trong ảnh, thay vì xử lý toàn bộ hình ảnh một cách đồng đều.
- **Ghép nối đặc trưng và dự đoán:**
  - Đặc trưng hình ảnh có trọng số từ Attention được kết hợp với đặc trưng câu hỏi từ LSTM.
  - Một **fully connected layer (FC layer)** hoặc **MLP** sẽ dự đoán câu trả lời dựa trên đặc trưng đã kết hợp.

CHƯƠNG 3 – TẬP DỮ LIỆU

3.1 Tổng quan về COCO và VQA

3.1.1 Tổng quan về COCO

COCO (Common Objects in Context) là một tập dữ liệu lớn được sử dụng rộng rãi trong các bài toán thị giác máy tính, đặc biệt là nhận diện và phân đoạn ảnh. Tập dữ liệu này được phát triển bởi Microsoft và lần đầu tiên được công bố vào năm 2014 nhằm cung cấp một nguồn dữ liệu phong phú cho các nghiên cứu về nhận diện đối tượng, phân đoạn ảnh và chú thích hình ảnh.

COCO chứa một lượng lớn hình ảnh thực tế với nhiều vật thể xuất hiện trong bối cảnh tự nhiên, giúp mô hình học sâu có khả năng tổng quát hóa tốt hơn khi áp dụng vào các tình huống thực tế. Các nhãn trong COCO không chỉ bao gồm thông tin về đối tượng trong ảnh mà còn có chú thích dạng văn bản, tọa độ bounding box, và phân vùng segmentation của các vật thể.

- **Số lượng và đặc điểm dữ liệu:** COCO bao gồm nhiều tập con khác nhau phục vụ cho từng mục đích nghiên cứu, trong đó tập **COCO train 2014** được sử dụng trong bài toán VQA của đồ án này. Dưới đây là một số thống kê về tập dữ liệu COCO train 2014:
    - Số lượng ảnh: hơn **82.000** ảnh
    - Số lượng đối tượng được gán nhãn: hơn **500.000**
    - Số lượng danh mục đối tượng: **80 lớp khác nhau**
- Tuy nhiên, để tránh hiện tượng dữ liệu thưa trong bài toán VQA, tập dữ liệu được lựa chọn chỉ bao gồm các ảnh chứa **động vật** từ COCO train 2014. Điều này giúp đảm bảo rằng dữ liệu đầu vào có sự liên kết chặt chẽ với các câu hỏi trong VQA dataset, đặc biệt là các câu hỏi dạng **“what animal is”** và **“how many”**.

Hình 2. 3 Ảnh mẫu với id 312.jpg trong dataset của COCO

- **Mục đích sử dụng trong bài toán VQA:** Trong bài toán VQA (Visual Question Answering), tập COCO được sử dụng làm nguồn cung cấp ảnh đầu vào. Các câu hỏi trong VQA dataset sẽ được liên kết với các ảnh trong COCO, cho phép mô hình học cách suy luận dựa trên cả hình ảnh và câu hỏi dạng văn bản.

3.1.2 Tổng quan về VQA

VQA (Visual Question Answering) là một tập dữ liệu lớn dành cho bài toán hỏi-đáp dựa trên hình ảnh, trong đó mô hình cần tạo ra câu trả lời dựa trên cả hình ảnh và câu hỏi đi kèm. Tập dữ liệu này lần đầu tiên được công bố vào năm 2015 và đã trải qua nhiều phiên bản nâng cấp để cải thiện chất lượng dữ liệu.

Trong đồ án này, tập dữ liệu VQA v2.0 (2017) được sử dụng, bao gồm Training annotations 2017 v2.0 và Training questions 2017 v2.0. Phiên bản này khắc phục vấn đề thiên lệch trong dữ liệu bằng cách cung cấp các câu hỏi tương tự cho nhiều hình ảnh khác nhau, đảm bảo rằng mô hình phải dựa vào hình ảnh để đưa ra câu trả lời chính xác thay vì chỉ dựa vào tần suất xuất hiện của câu trả lời trong tập huấn luyện.

- **Đặc điểm của VQA v2.0 (2017):** Tập **Training annotations 2017 v2.0** và **Training questions 2017 v2.0** chứa dữ liệu từ tập ảnh **COCO train 2014**. Dữ liệu được tổ chức dưới dạng JSON với các thông tin chính:
    - **Training questions 2017 v2.0:** Chứa danh sách các câu hỏi về hình ảnh, với thông tin:
        - ***image_id*:** ID của ảnh tương ứng.
        - ***question_id*:** ID của câu hỏi.
        - ***question*:** Nội dung câu hỏi.

```
{
  "image_id": 312,
  "question": "How many baby elephants are there?",
  "question_id": 312001
},
```

    - **Training annotations 2017 v2.0:** Cung cấp các câu trả lời cho từng câu hỏi, với thông tin:
        - ***question_type* :** Loại câu của câu hỏi tương ứng.
        - ***multiple_choice_answer*:** Câu trả lời phổ biến nhất.
        - ***answers*:** Danh sách các câu trả lời từ nhiều người gán nhãn (thường là 10 câu trả lời/câu hỏi). Mỗi câu trả lời trong danh sách này sẽ có độ tin cậy là biến ***answer_confidence***
        - ***image_id*:** ID của ảnh tương ứng.
        - ***answer_type***: Kiểu câu trả lời (ví dụ: "yes/no", "number", "other").
        - ***question_id*:** ID của câu hỏi tương ứng.

```
{
  "question_type": "how many",
  "multiple_choice_answer": "1",
  "answers": [
      {
          "answer": "1",
          "answer_confidence": "yes",
          "answer_id": 1
```

```
        },
        {
            "answer": "1",
            "answer_confidence": "yes",
            "answer_id": 2
        },
        .......
        {
            "answer": "1",
            "answer_confidence": "yes",
            "answer_id": 10
        }
    ],
    "image_id": 312,
    "answer_type": "number",
    "question_id": 312001
},
```


- **Mục tiêu sử dụng VQA dataset trong bài toán:** Trong bài toán VQA của đồ án này, tập VQA v2.0 được sử dụng để cung cấp câu hỏi và câu trả lời cho các ảnh trong COCO train 2014. Để tối ưu hóa dữ liệu và tránh hiện tượng dữ liệu thưa, đồ án này tập trung vào hai nhóm câu hỏi chính:
    - **Câu hỏi về nhận dạng loài động vật:** Các câu hỏi dạng *"What animal is this?"*, giúp mô hình nhận diện và suy luận về loài động vật trong ảnh.
    - **Câu hỏi đếm số lượng:** Các câu hỏi dạng *"How many animals are there?"*, giúp mô hình học cách đếm số lượng đối tượng.

3.2 Cấu trúc của tập dữ liệu

3.2.1 Lựa chọn dữ liệu từ COCO và VQA

Tập dữ liệu trong đồ án được xây dựng từ hai nguồn chính:

- **COCO train 2014:** Cung cấp ảnh.
- **VQA v2.0 (2017):** Cung cấp câu hỏi và câu trả lời tương ứng.

Tuy nhiên, để tránh hiện tượng dữ liệu thưa và đảm bảo tính hiệu quả của mô hình, chỉ những ảnh chứa động vật trong **COCO train 2014** được sử dụng. Các ảnh này được

xác định dựa trên annotation trong tệp **instances_train2014.json**, bằng cách lọc theo ***category_id*** thuộc nhóm "animal" trong COCO.

Ngoài ra, chỉ các câu hỏi liên quan đến động vật từ **Training questions 2017 v2.0** được giữ lại. Cụ thể, hai dạng câu hỏi chính được chọn là:

- **Dạng nhận diện đối tượng**: "What animal is this?", giúp mô hình xác định tên động vật trong ảnh.
- **Dạng đếm số lượng**: "How many animals are there?", giúp mô hình học cách đếm số lượng động vật.

3.2.2 Tổ chức thư mục và dữ liệu

Để quản lý tập dữ liệu một cách có hệ thống và thuận tiện cho quá trình tiền xử lý, tập dữ liệu được tổ chức theo cấu trúc thư mục như sau:

```
midterm_deep_learning/
├── annotations/    # Chứa annotation của COCO & VQA
│   ├── instances_train2014.json # Annotation COCO train 2014 (lọc động vật)
│   ├── v2_mscoco_train2014_annotations.json  # Annotation VQA cho ảnh COCO train 2014
├── images/      # Chứa hình ảnh từ COCO train 2014
│   ├── 100014.jpg
│   ├── 100037.jpg
│   ├── 10005.jpg
│   ├── 100064.jpg
│   ├── ...  (chỉ bao gồm ảnh về động vật)
├── questions/    # Chứa câu hỏi từ VQA
│   ├── v2_OpenEnded_mscoco_train2014_questions.json  # Chỉ chứa câu hỏi về động vật
├── dataset_vqa.csv        # dataframe lưu thông tin dataset
```

- **Thư mục *annotations:*** Chứa các annotation từ VQA và COCO.
    - **instances_train2014.json:** Annotation từ COCO, giúp phân loại các đối tượng có trong dataset.

- **v2_mscoco_train2014_annotations.json:** Chứa thông tin câu trả lời từ VQA v2.0.

- **Thư mục *images:*** Chứa các ảnh từ tập COCO train 2014.
  - Chỉ giữ lại các ảnh có chứa động vật, được xác định dựa trên các instances là động vật từ COCO.
- **Thư mục *questions:*** Chứa danh sách câu hỏi từ tập VQA v2.0.
  - **v2_OpenEnded_mscoco_train2014_questions.json:** Chứa câu hỏi dạng mở liên quan đến động vật.
- **File qa_pairs_dataset.json:** Đây là dataset đã qua tiền xử lý, chứa các cặp câu hỏi - câu trả lời liên quan đến động vật.
  - Dữ liệu trong file được trích xuất và tổng hợp từ ***questions/*** và ***annotations/,*** chỉ giữ lại những câu hỏi thuộchai dạng chính:
    - "What animal is this?"
    - "How many animals are there?"
  - Cấu trúc của file **qa_pairs_dataset.json**:

```
{
  "image_id": 312,
  "question": "How many baby elephants are there?",
  "answer": "1"
},
{
  "image_id": 4587,
  "question": "What animal is in the picture?",
  "answer": "Zebra"
}
```

3.3 Tiền xử lý dữ liệu

3.3.1 Chọn lọc dữ liệu ảnh

Trong bài toán VQA, dữ liệu ảnh đầu vào đóng vai trò quan trọng trong việc đảm bảo mô hình có thể học được các đặc trưng phù hợp với câu hỏi. Tuy nhiên, tập dữ liệu COCO train 2014 chứa hàng trăm nghìn hình ảnh với nhiều đối tượng khác nhau, bao

gồm cả con người, vật dụng và cảnh quan. Do đó, để tránh hiện tượng dữ liệu thưa và tập trung vào bài toán nhận diện động vật, ta tiến hành lọc ra các ảnh chỉ chứa động vật.

- **Bước 1: Tải và trích xuất annotation của COCO:** Tập dữ liệu COCO cung cấp thông tin chi tiết về các đối tượng có trong từng ảnh thông qua file annotation ***instances_train2014.json***. File này chứa danh sách **categories** (nhóm đối tượng) và **annotations** (chú thích của từng ảnh).
    - Trước tiên, ta tải và giải nén file annotation, trong đây sẽ chứa file ***instances_train2014.json*** đây chính là file chứa thông tin các instances:
- **Bước 2: Xác định danh sách category của động vật**: Ta đọc danh sách **categories** để xác định ID của các nhóm động vật:
    - Từ danh sách categories trên, ta chọn các ID tương ứng với động vật là **[16, 17, 18, 19, 20, 21, 22, 23, 24, 25].** Đây là các ảnh của các loài động vật như Chó, Sói, Mèo, Ngựa, Hươu cao cổ, …
- **Bước 3: Lọc danh sách ảnh chứa động vật và lưu vào một tập hợp (set):**
    - Duyệt qua danh sách annotation, ta lấy danh sách ID của những ảnh chứa ít nhất một trong các category động vật trên.
    - Kết quả thu được ID của **16,740 ảnh** là ảnh có ít nhất một động vật trong ảnh.

3.3.2 Xây dựng dataframe cho tập dữ liệu

Bước 1: Đọc dữ liệu từ các file JSON

- Load file annotations (*v2_mscoco_train2014_annotations.json*) chứa câu trả lời.
- Load file questions (*v2_OpenEnded_mscoco_train2014_questions.json*) chứa câu hỏi.
- Tạo dict để tra cứu nhanh câu hỏi theo *question_id*.
- Tạo dict để tra cứu nhanh loại câu hỏi (*question_type*).

Bước 2: Lọc dữ liệu theo điều kiện

- Chỉ lấy dữ liệu có *image_id* nằm trong danh sách ảnh động vật (*animal_image_ids*).
- Chỉ lấy các câu hỏi thuộc nhóm *"what animal is", "how many", "what number is", "what is this", "what are"*.
- Tạo danh sách qa_pairs chứa các thông tin cần thiết (*image_id, question_id, question, question_type, multiple_choice_answer, answer list, answer_type*).

Bước 3: Tạo DataFrame từ danh sách qa_pairs

- Chuyển danh sách qa_pairs thành DataFrame (*df_vqa*).

Bước 4: Tiền xử lý dữ liệu

- Chuyển đổi chữ viết tắt thành dạng đầy đủ (*decontractions*).
- Chuẩn hóa văn bản: chữ thường, loại bỏ dấu câu không cần thiết, giữ lại chữ cái, số, dấu . hợp lệ.
- Áp dụng tiền xử lý cho cả cột "*question*" và danh sách "*answer*".

Bước 5: Trích xuất danh sách nhãn (labels)

- Lấy tất cả giá trị duy nhất từ cột *multiple_choice_answer* sau đó sử dụng label encoder để mã hóa nhãn.
- Giới hạn số lượng nhãn đầu ra (nếu vượt quá 1000).

Bước 6: Lưu DataFrame ra file CSV:

Các mẫu đầu tiên của dataframe:

image_id	question	multiple_choice_answer	answer	label
137045	what is this	bear	bear, bear, bear, bear, bear, bear, bear, bear, bear, bear	39
131093	how many sheeps are this	3	3, 3, 3, 3, 3, 0, 3, 3, 3, 3	297
25	how many giraffes are there	2	2, 2, 2, 1, 2, 1, 2, 2, 2, 2	89
25	how many animals are in this photo	2	2, 2, i can see total of 2 animals, 2, giraffe, 2, 2, 2, 2, 2	89
370986	how many elephants are in the water	2	2, 2, 1, 2, 2, 2, 3, 2, 2, 2	89
370986	how many tails can you see in this picture	3	3, 3, 3, 3, 3, 3, 3, 3, 3, 3	297
262204	how many tusks are visible	2	2, 2, 2, 2, 2, 2, 2, 2, 2, 2	89
262204	what is this animal	elephant	elephant, elephant, elephant, elephant, elephant, elephant, elephant, elephant, elephant, elephant	316
72	how many giraffes are in this image	2	2, 2, 2, 2, 2, 2, 2, 3, 2, 2	89
393292	how many birds are in the sink	2	1, 1, 2, 1, 1, 2, 2, 2, 2, 2	89

Hình 2. 4 Các mẫu đầu tiên của Datafame

3.3.3 Tiền xử lý ảnh đầu vào

Bước 1: Lấy danh sách ID ảnh cần xử lý

- Trích xuất danh sách image_id duy nhất từ DataFrame df_vqa.

Bước 2: Định nghĩa các phép biến đổi ảnh (transform)

- Resize ảnh về kích thước (224, 224).
- Chuyển ảnh thành tensor (ToTensor).
- Chuẩn hóa ảnh theo thống kê của tập ImageNet (Normalize).

Bước 3: Duyệt qua từng ảnh và tiền xử lý

- Mở ảnh từ đường dẫn và chuyển về định dạng RGB.
- Áp dụng các phép biến đổi (transform).
- Lưu tensor của ảnh vào dictionary image_tensors.
- Lưu danh sách ID ảnh đã xử lý vào image_id_tensors.

Bước 4: Chuyển danh sách ID ảnh thành tensor PyTorch

- image_id_tensors = torch.tensor(image_id_tensors).

Bước 5: Kiểm tra kết quả

- In số lượng ảnh đã được xử lý thành công.
- Cảnh báo nếu có ảnh bị thiếu.

3.3.4 Tiền xử lý câu hỏi đầu vào

Bước 1: Xây dựng từ điển (build_vocab)

1. **Đếm số lần xuất hiện của từ** trong câu hỏi và câu trả lời.
2. **Lọc những từ có tần suất >= min_freq** để đưa vào từ điển.
3. **Thêm các token đặc biệt**: "<PAD>", "<UNK>", "<START>", "<END>".
4. **Tạo ánh xạ** giữa từ và ID:
    - word2idx: ánh xạ từ → ID.
    - idx2word: ánh xạ ID → từ.

Bước 2: Ánh xạ câu hỏi thành vector (vectorize_question)

1. **Tách từ (word_tokenize)** trong câu hỏi.
2. **Chuyển từ thành ID** bằng word2idx, dùng "<UNK>" nếu từ không có trong từ điển.

3. **Căn chỉnh độ dài (max_length)**:
    - Nếu câu ngắn hơn max_length, thêm "<PAD>" vào cuối.
    - Nếu dài hơn max_length, cắt bớt.

Bước 3: Chuyển đổi toàn bộ câu hỏi thành vector

- Áp dụng hàm vectorize_question() cho toàn bộ cột "question" trong df_vqa.
- Lưu kết quả vào cột "question_vector".

CHƯƠNG 4 – XÂY DỰNG CÁC MÔ HÌNH

4.1 Tổng quan về Train from Scratch và Pre-Model

Trong Deep Learning, việc huấn luyện mô hình có thể được thực hiện theo hai hướng chính: huấn luyện từ đầu (Train From Scratch) hoặc sử dụng mô hình đã được huấn luyện trước (Pre-trained Model). Mỗi phương pháp có ưu và nhược điểm riêng, phù hợp với từng bài toán cụ thể. Báo cáo này sẽ phân tích chi tiết về Pre-trained model và Train from Scratch.

4.1.1 Pre-trained model:

Mô hình Pre-trained (huấn luyện sơ bộ) là những mô hình đã được huấn luyện trước trên một khối lượng dữ liệu lớn. Những mô hình này giúp tối ưu quá trình huấn luyện và cải thiện hiệu suất cho các nhiệm vụ mới.

Ở báo cáo này, nhóm tôi sử dụng Pre-trained model **ResNet (Residual Networks)** nhằm giải quyết bài toán này.

ResNet (Residual Networks): ResNet là một mạng nơ-ron sâu được phát triển để giải quyết vấn đề suy giảm gradient trong quá trình huấn luyện các mô hình có nhiều tầng. Mô hình này sử dụng kỹ thuật residual learning, trong đó các khối residual giúp duy trì và truyền tải thông tin hiệu quả bằng cách sử dụng các kết nối tắt (skip connections). Những kết nối này giúp giảm thiểu sự mất mát thông tin khi mạng ngày càng sâu, đồng thời tăng tốc độ hội tụ và cải thiện hiệu suất trên các bài toán thị giác máy tính như phân loại ảnh và nhận dạng đối tượng.

Ưu điểm:

- **Tiết kiệm tài nguyên:** Giảm đáng kể thời gian và chi phí tính toán so với việc huấn luyện từ đầu.
- **Yêu cầu ít dữ liệu hơn:** Mô hình đã học được nhiều đặc trưng quan trọng từ tập dữ liệu lớn, giúp nó có thể hoạt động tốt ngay cả khi dữ liệu huấn luyện hạn chế.

- **Tăng độ chính xác:** Các mô hình pre-trained đã được tối ưu hóa trên các bộ dữ liệu lớn, giúp chúng có độ chính xác cao ngay cả khi áp dụng vào các bài toán mới.

Nhược điểm:

- **Hạn chế tùy chỉnh:** Khó thay đổi kiến trúc mô hình theo ý muốn vì đã có các trọng số cố định từ trước.
- **Lệ thuộc vào dữ liệu gốc:** Nếu dữ liệu huấn luyện của mô hình pre-trained khác biệt quá lớn so với bài toán thực tế, mô hình có thể hoạt động kém hiệu quả.
- **Hiệu suất không tối ưu trong một số bài toán đặc thù:** Nếu bài toán yêu cầu một kiến trúc đặc biệt, việc sử dụng pre-trained model có thể không đạt hiệu suất tốt nhất.

4.1.2 Train from Scratch

Huấn luyện từ đầu (Train from Scratch) có nghĩa là xây dựng và huấn luyện một mô hình từ đầu, tức là bắt đầu với các trọng số ngẫu nhiên và cập nhật chúng trong quá trình huấn luyện.

Ưu điểm:

- **Tùy chỉnh cao:** Có thể thiết kế mô hình tối ưu hóa cho từng bài toán cụ thể.
- **Không bị lệ thuộc vào dữ liệu gốc của mô hình pre-trained:** Đôi khi, dữ liệu gốc của mô hình pre-trained không phù hợp với bài toán mới.
- **Tiềm năng đạt độ chính xác cao:** Nếu có đủ dữ liệu và tài nguyên, mô hình huấn luyện từ đầu có thể đạt độ chính xác cao hơn so với mô hình pre-trained.

Nhược điểm:

- **Cần lượng dữ liệu lớn:** Việc huấn luyện từ đầu yêu cầu một lượng dữ liệu lớn để mô hình có thể học được các đặc trưng hữu ích.
- **Tốn thời gian và tài nguyên:** Quá trình huấn luyện thường mất rất nhiều thời gian và yêu cầu phần cứng mạnh như GPU hoặc TPU.

- **Khả năng hội tụ kém:** Nếu không thiết lập siêu tham số (hyperparameters) hợp lý, mô hình có thể không hội tụ hoặc hội tụ rất chậm.

So sánh

Tiêu chí	Train From Scratch	Pre-trained Model
Dữ liệu yêu cầu	Lớn	Ít
Thời gian huấn luyện	Dài	Ngắn
Tài nguyên tính toán	Cao	Thấp
Độ chính xác ban đầu	Thấp	Cao
Khả năng tùy chỉnh	Cao	Thấp
Khả năng tổng quát hóa	Phụ thuộc vào dữ liệu	Tốt do đã học từ tập dữ liệu lớn

Bảng 4. 1 Bảng so sánh giữa Train from scratch và PreTrain model

Kết luận đưa ra:

Chọn Train From Scratch khi:

- Có một lượng dữ liệu đủ lớn và đa dạng.
- Bài toán yêu cầu một mô hình đặc thù, không có pre-trained model phù hợp.
- Muốn kiểm soát hoàn toàn quá trình huấn luyện và kiến trúc mô hình.

Chọn Pre-trained Model khi:

- Dữ liệu huấn luyện bị hạn chế.
- Muốn triển khai mô hình nhanh chóng.
- Bài toán có thể được giải quyết hiệu quả bằng cách fine-tune từ một mô hình có sẵn.

4.2 Pre-trained model và Train from scratch model

4.2.1 Cấu trúc Pre-trained model

Trích xuất đặc trưng hình ảnh bằng CNN (class ImageFeatureExtractor**)**

- Sử dụng ResNet-50 đã được huấn luyện trước trên tập dữ liệu ImageNet để trích xuất đặc trưng hình ảnh.
- Đóng băng trọng số của ResNet-50 để tránh làm thay đổi các tham số đã học trước đó, giúp mô hình ổn định hơn.
- Lấy đầu ra từ tầng cuối cùng của ResNet-50 trước lớp fully connected và chuyển đổi thành vector đặc trưng có kích thước 2048.
- Ánh xạ vector đặc trưng về không gian ẩn kích thước 512 thông qua một lớp fully connected nhằm giảm chiều dữ liệu và giữ lại thông tin quan trọng.

4.2.2 Cấu trúc Train from scratch model

Trích xuất đặc trưng hình ảnh bằng cách xây dựng các lớp tích chập CNN (class CNNTraining). Trong mô hình CNNTraining là một kiến trúc để trích xuất đặc trưng từ ảnh. Nó gồm 3 lớp Conv2D, batch normalization, pooling(Adaptive Pooling), và một fully connected layer để đưa về vector đặc trưng có kích thước mong muốn.

Ý nghĩa:

• 3 lớp Convolution (conv1, conv2, conv3) để học đặc trưng ảnh.

• BatchNorm (bn1, bn2, bn3) giúp chuẩn hóa dữ liệu để giảm overfitting.

• Số channels:

conv1: (3 kênh RGB) → 64 kênh

conv2: 64 kênh → 128 kênh

conv3: 128 kênh → 256 kênh

• Kernel size = 3x3 (là chuẩn của CNN để giữ thông tin không gian tốt).

• Stride = 1 & Padding = 1 để giữ kích thước ảnh không đổi.

Quá trình xử lí dữ liệu:

- Conv1 + BatchNorm + ReLU → MaxPool2D(2x2) → kích thước giảm một nửa.
- Conv2 + BatchNorm + ReLU → MaxPool2D(2x2) → tiếp tục giảm kích thước.
- Conv3 + BatchNorm + ReLU → Adaptive Pooling (1x1) → đặc trưng chính.
- Flatten và đưa vào Fully Connected Layer (fc1).
- Dropout để tránh overfitting.

- Thêm chiều thứ 2 bằng (unsqueeze) để phù hợp với mô hình sử dụng.

4.2.3 Các phương thức xử lí và huấn luyện mô hình

a. Các phương thức xử lí

Xử lý câu hỏi bằng LSTM (class QuestionLSTM**)**

- Sử dụng LSTM để xử lý chuỗi câu hỏi bằng cách chuyển đổi chúng thành dạng nhúng (embedding), giúp biểu diễn các từ trong không gian liên tục.
- Trước tiên, mỗi từ trong câu hỏi được ánh xạ sang một vector nhúng có kích thước cố định.
- Sau đó, kết hợp vector đặc trưng hình ảnh với các từ trong câu hỏi để tạo ra biểu diễn giàu ngữ cảnh hơn.
- Khi sử dụng Attention, mô hình sẽ xác định những phần quan trọng nhất trong câu hỏi để tập trung vào, giúp cải thiện chất lượng dự đoán.

Mô hình tổng thể (VQAModel) (class VQAModel**)**

- Kết hợp bộ trích xuất đặc trưng ảnh và LSTM để xử lý đồng thời hình ảnh và câu hỏi.
- Sử dụng một lớp fully connected để dự đoán câu trả lời dựa trên thông tin đã xử lý.
- Khi sử dụng Attention, mô hình có thể xác định các phần quan trọng nhất trong câu hỏi và hình ảnh, giúp tối ưu hóa quá trình suy luận và nâng cao độ chính xác.

b. Huấn luyện mô hình

Khởi tạo mô hình

Mô hình được khởi tạo với thông số `vocab_size` (kích thước từ vựng) và tùy chọn có sử dụng Attention hay không. Nếu sử dụng GPU, mô hình sẽ được đưa lên thiết bị để tăng tốc xử lý.

Hàm Mất Mát và Tối Ưu Hóa

Hàm mất mát **Cross Entropy Loss** được sử dụng để đo độ chính xác của mô hình, bỏ qua các token không quan trọng như `<PAD>`.

Bộ tối ưu hóa **Adam** giúp mô hình học nhanh hơn và ổn định hơn.

Huấn Luyện Mô Hình

Trong mỗi epoch, dữ liệu được chia thành từng batch nhỏ.

- Mô hình nhận vào hình ảnh và câu hỏi để dự đoán câu trả lời.
- Sai số (loss) được tính toán bằng Cross Entropy Loss.
- Gradient được tính toán và cập nhật để cải thiện mô hình.

Mỗi khi hoàn thành một batch, nếu batch đó trùng với mốc kiểm tra (`log_interval`), chương trình sẽ hiển thị thông tin về tiến trình huấn luyện.

Lưu Mô Hình Tốt Nhất

Mỗi epoch sẽ được kiểm tra xem loss có giảm hay không. Nếu loss tốt hơn so với các epoch trước, mô hình sẽ được lưu lại.

CHƯƠNG 5 – SO SÁNH HIỆU SUẤT, ĐÁNH GIÁ MÔ HÌNH

5.1 Pre-trained model và Train from scratch

5.1.1 Pre-trained model

a. Pre-trained model không sử dụng attention

Hình 5. 1 Biểu đồ Train Loss và Validation Loss PreTrain không Attention

Dưới đây là xu hướng của mô hình qua các epoch 15, 25, 50:

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
15	1.9730	40.55%	2.4830	36.27%
25	1.6052	48.39%	2.5442	36.46%
50	1.0824	64.23%	2.6553	37.51%

Bảng 5. 1 Xu hướng của mô hình PreTrain không attention

Hiệu suất trên tập huấn luyện

- Train loss giảm dần qua các epoch (1.9730 → 1.6052 → 1.0824), cho thấy mô hình đang học tốt hơn trên tập huấn luyện.

- Train accuracy tăng liên tục từ 40.55% → 48.39% → 64.23%, thể hiện khả năng mô hình ngày càng cao trong việc phân loại dữ liệu huấn luyện.
- Xu hướng hội tụ tốt, không có dấu hiệu của underfitting.

Hiệu suất trên tập kiểm tra (Validation set)

- Validation loss có xu hướng tăng nhẹ (2.4830 → 2.5442 → 2.6553), cho thấy mô hình có dấu hiệu overfitting sau Epoch 25.
- Validation accuracy chỉ tăng nhẹ (36.27% → 36.46% → 37.51%), chứng tỏ mô hình không cải thiện đáng kể khả năng tổng quát hóa lên dữ liệu mới.

Xu hướng chung

- Từ Epoch 15 đến Epoch 25, mô hình cải thiện đồng thời trên cả tập huấn luyện và kiểm tra.
- Từ Epoch 25 đến Epoch 50, mô hình tiếp tục cải thiện trên tập huấn luyện, nhưng trên tập kiểm tra, hiệu suất không tăng đáng kể, thậm chí validation loss còn có xu hướng tăng.
- Dấu hiệu overfitting xuất hiện rõ ràng từ Epoch 25 trở đi.

Kết luận:

- Mô hình có hiệu suất tốt trên tập huấn luyện nhưng chưa tối ưu trên tập kiểm tra.
- Overfitting xảy ra từ Epoch 25 trở đi, khi validation loss tăng dần và accuracy không cải thiện đáng kể.
- Cần có các biện pháp để tăng khả năng tổng quát hóa của mô hình, thay vì tiếp tục huấn luyện đến Epoch 50.

b. Pre-trained model có sử dụng attention

Hình 5. 2 Biểu đồ Train Loss và Validation Loss PreTrain có Attention

Dưới đây là xu hướng của mô hình sau khi áp dụng attention qua các epoch 15, 25, 50:

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
15	1.7585	43.18%	2.3002	37.12%
25	1.3561	54.22%	2.2636	39.43%
50	0.8049	73.08%	2.4043	40.91%

Bảng 5. 2 Xu hướng của mô hình PreTrain không attention

Hiệu suất trên tập huấn luyện

- Train Loss giảm đáng kể từ 1.7585 (Epoch 15) xuống 1.3561 (Epoch 25) và tiếp tục giảm xuống 0.8049 (Epoch 50). Điều này cho thấy mô hình đang học tốt hơn theo thời gian.
- Train Accuracy tăng mạnh từ 43.18% (Epoch 15) lên 54.22% (Epoch 25) và đạt 73.08% (Epoch 50), cho thấy mô hình ngày càng phân loại chính xác hơn trên tập huấn luyện.

Hiệu suất trên tập kiểm tra (Validation set)

- Validation Loss có xu hướng giảm nhẹ từ Epoch 15 (2.3002) đến Epoch 25 (2.2636), nhưng lại tăng lên 2.4043 tại Epoch 50.
- Validation Accuracy tăng từ 37.12% (Epoch 15) lên 39.43% (Epoch 25) và đạt 40.91% (Epoch 50).
- Dù Validation Accuracy có cải thiện, nhưng mức tăng không nhiều so với sự cải thiện mạnh của Train Accuracy, điều này có thể là dấu hiệu của overfitting.

So sánh xu hướng chung

- Mô hình học rất tốt trên tập huấn luyện (Train Loss giảm mạnh và Train Accuracy tăng nhanh).
- Tuy nhiên, Validation Loss không giảm đáng kể và thậm chí tăng lên ở Epoch 50, có thể là dấu hiệu của overfitting.
- Validation Accuracy có tăng nhưng không đáng kể, cho thấy mô hình có thể đang ghi nhớ dữ liệu huấn luyện thay vì học được các đặc trưng tổng quát hơn.

c. So sánh giữa mô hình có attention và không attention

So sánh hiệu suất huấn luyện

Train Loss giảm mạnh hơn khi áp dụng Attention

- Tại Epoch 15, Train Loss giảm từ 1.9730 xuống 1.7585 (~10.88% giảm).
- Tại Epoch 25, Train Loss giảm từ 1.6052 xuống 1.3561 (~15.53% giảm).
- Tại Epoch 50, Train Loss giảm từ 1.0824 xuống 0.8049 (~25.61% giảm).

⇨ Điều này cho thấy mô hình có Attention giúp hội tụ nhanh hơn và học tốt hơn trên tập huấn luyện.

Train Accuracy tăng đáng kể

- Ở Epoch 15, tăng từ 40.55% lên 43.18%.
- Ở Epoch 25, tăng từ 48.39% lên 54.22%.
- Ở Epoch 50, tăng từ 64.23% lên 73.08%.

⇨ Attention giúp mô hình học các đặc trưng quan trọng hơn, cải thiện đáng kể độ chính xác trên tập huấn luyện.

So sánh hiệu suất trên tập kiểm tra (Validation Set)

Validation Loss giảm sau khi áp dụng Attention ở Epoch 15 và 25

- Epoch 15: Từ 2.4830 xuống 2.3002 (~7.36% giảm).
- Epoch 25: Từ 2.5442 xuống 2.2636 (~11.03% giảm).
- Epoch 50: Validation Loss giảm so với Epoch 25 nhưng lại tăng nhẹ từ 2.6553 xuống 2.4043, có thể do mô hình bắt đầu overfitting.

Validation Accuracy tăng sau khi áp dụng Attention

- Ở Epoch 15, tăng từ 36.27% lên 37.12%.
- Ở Epoch 25, tăng từ 36.46% lên 39.43%.
- Ở Epoch 50, tăng từ 37.51% lên 40.91%.

⇨ Dù mức tăng không lớn như trên tập huấn luyện, nhưng cho thấy mô hình có Attention giúp cải thiện khả năng tổng quát hóa.

Kết luận

- Attention giúp mô hình học nhanh hơn và tốt hơn trên tập huấn luyện, thể hiện qua Train Loss giảm nhanh và Train Accuracy tăng mạnh.
- Trên tập kiểm tra, Attention giúp giảm Validation Loss và tăng Validation Accuracy, đặc biệt rõ ràng ở Epoch 15 và 25.
- Tuy nhiên, ở Epoch 50, Validation Loss lại tăng nhẹ, có thể do overfitting.

5.1.2 Train from scratch

Chúng tôi trình bày quá trình huấn luyện và so sánh hiệu suất giữa hai biến thể mô hình Trả Lời Câu Hỏi Hình Ảnh (VQA) - một mô hình sử dụng Attention và một không sử dụng Attention. Các mô hình này được huấn luyện trên tập dữ liệu VQA với kiến trúc CNN được xây dựng.

Phương pháp huấn luyện

- Mô hình được huấn luyện trên tập dữ liệu VQA sử dụng PyTorch.
- Sử dụng Cross-Entropy Loss và Adam Optimizer với learning rate 0.001.
- Số epoch huấn luyện: 5.
- So sánh hai biến thể mô hình: có Attention và không có Attention.

a. Train from scratch không sử dụng attention

Hình 5. 3 Biểu đồ Train Loss và Validation Loss Train from scratch không attention

Dưới đây là xu hướng của mô hình không áp dụng attention qua các epoch 15, 25, 50:

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
15	1.8854	42.38%	2.5279	29.87%
25	1.3344	57.25%	2.6277	31.96%
50	0.6770	77.86%	2.9773	33.01%

Bảng 5. 3 Xu hướng của mô hình Train from scratch không attention

Nhận xét huấn luyện trên tập huấn luyện:

- Train Loss giảm dần từ 1.8854 → 1.3344 → 0.6770, cho thấy mô hình đang học tốt hơn theo thời gian.
- Train Accuracy tăng mạnh từ 42.38% → 57.25% → 77.86%, chứng tỏ mô hình ngày càng phân loại chính xác hơn trên tập huấn luyện.
- Xu hướng hội tụ tốt, không có dấu hiệu underfitting.

Nhận xét hiệu suất trên tập kiểm tra (Validation set)

- Validation Loss tăng dần từ 2.5279 → 2.6277 → 2.9773, điều này cho thấy mô hình có dấu hiệu overfitting sau Epoch 25.
- Validation Accuracy chỉ tăng nhẹ từ 29.87% → 31.96% → 33.01%, điều này cho thấy mô hình không cải thiện đáng kể khả năng tổng quát hóa lên dữ liệu mới.

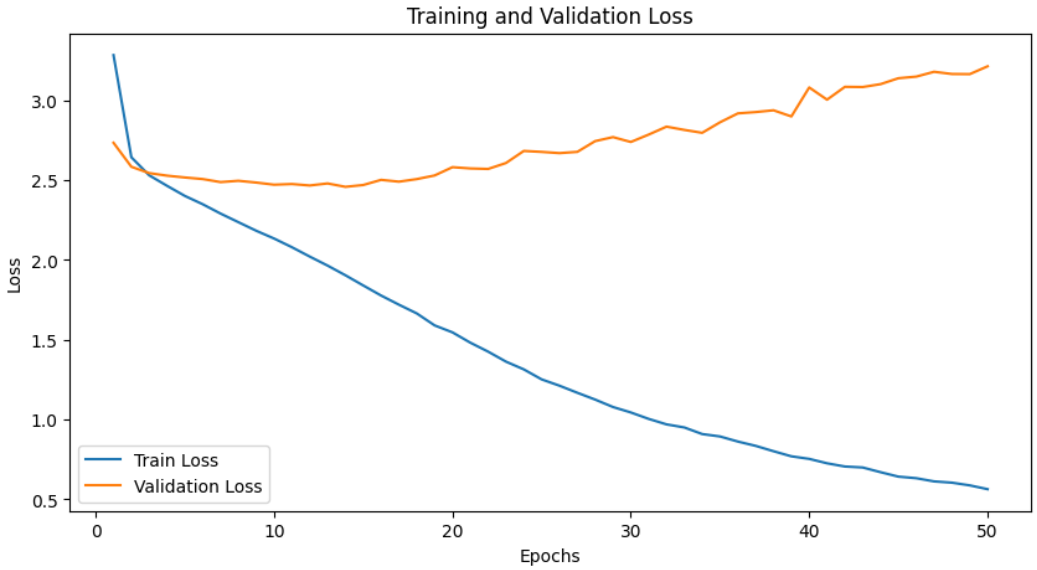
Xu hướng

- Từ Epoch 15 đến Epoch 25, mô hình cải thiện tốt trên cả tập huấn luyện và kiểm tra.
- Từ Epoch 25 đến Epoch 50, Train Accuracy tiếp tục tăng mạnh, nhưng Validation Accuracy chỉ tăng nhẹ, trong khi Validation Loss lại tăng. Điều này là dấu hiệu rõ ràng của overfitting.

Kết luận

- Mô hình có hiệu suất tốt trên tập huấn luyện nhưng chưa tối ưu trên tập kiểm tra.
- Overfitting xảy ra từ Epoch 25 trở đi, khi Validation Loss tăng dần nhưng Validation Accuracy không cải thiện đáng kể.
- Cần có biện pháp để giảm overfitting

b. Train from scratch có sử dụng Attention



Hình 5. 4 Biểu đồ Train Loss và Validation Loss Train from scratch có attention

Dưới đây là xu hướng của mô hình áp dụng attention qua các epoch 15, 25, 50:

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
15	1.8405	43.59%	2.4703	31.31%
25	1.2519	59.06%	2.6778	32.01%
50	0.5637	80.83%	3.2143	32.47%

Bảng 5. 4 Xu hướng của mô hình Train from scratch có attention

Hiệu suất trên tập huấn luyện

- Train Loss giảm mạnh từ 1.8405 → 1.2519 → 0.5637, chứng tỏ mô hình đang học tốt trên tập huấn luyện.
- Train Accuracy tăng đáng kể từ 43.59% → 59.06% → 80.83%, cho thấy mô hình ngày càng phân loại chính xác hơn trên tập huấn luyện.
- Xu hướng hội tụ tốt, không có dấu hiệu underfitting.

Hiệu suất trên tập kiểm tra (Validation Set)

- Validation Loss có xu hướng tăng từ 2.4703 → 2.6778 → 3.2143, cho thấy mô hình có dấu hiệu overfitting từ Epoch 25 trở đi.
- Validation Accuracy chỉ tăng rất ít từ 31.31% → 32.01% → 32.47%, chứng tỏ mô hình không cải thiện đáng kể khả năng tổng quát hóa lên dữ liệu mới.

Xu hướng chung

- Từ Epoch 15 đến Epoch 25, mô hình cải thiện tốt trên cả tập huấn luyện và kiểm tra.
- Từ Epoch 25 đến Epoch 50, Train Accuracy tiếp tục tăng mạnh, nhưng Validation Accuracy gần như không tăng, trong khi Validation Loss lại tăng. Điều này là dấu hiệu rõ ràng của overfitting.

Kết luận & Giải pháp cải thiện

- Mô hình có hiệu suất tốt trên tập huấn luyện nhưng chưa tối ưu trên tập kiểm tra.
- Overfitting xảy ra sau Epoch 25, khi Validation Loss tăng nhưng Validation Accuracy không cải thiện đáng kể.
- Cần áp dụng các kỹ thuật giảm overfitting

c. So sánh giữa mô hình có attention và không attention

So sánh hiệu suất huấn luyện

Train Loss giảm mạnh hơn khi áp dụng Attention

- Tại Epoch 15, Train Loss giảm từ 1.8854 xuống 1.8405 (~2.39% giảm).
- Tại Epoch 25, Train Loss giảm từ 1.3344 xuống 1.2519 (~6.19% giảm).
- Tại Epoch 50, Train Loss giảm từ 0.6770 xuống 0.5637 (~16.75% giảm).

⇨ Điều này cho thấy mô hình có Attention giúp hội tụ nhanh hơn và học tốt hơn trên tập huấn luyện.

Train Accuracy tăng đáng kể

- Ở Epoch 15, tăng từ 42.38% lên 43.59%.
- Ở Epoch 25, tăng từ 57.25% lên 59.06%.
- Ở Epoch 50, tăng từ 77.86% lên 80.83%.

⇨ Attention giúp mô hình học các đặc trưng quan trọng hơn, cải thiện đáng kể độ chính xác trên tập huấn luyện.

So sánh hiệu suất trên tập kiểm tra (Validation Set)

Validation Loss giảm sau khi áp dụng Attention ở Epoch 15

- Epoch 15: Từ 2.5279 xuống 2.4703 (~2.28% giảm).
- Epoch 25: Từ 2.6277 tăng nhẹ lên 2.6778 (~1.91% tăng).
- Epoch 50: Validation Loss tăng từ 2.9773 lên 3.2143, có thể do mô hình bắt đầu overfitting.

Validation Accuracy tăng sau khi áp dụng Attention

- Ở Epoch 15, tăng từ 29.87% lên 31.31%.
- Ở Epoch 25, tăng từ 31.96% lên 32.01%.
- Ở Epoch 50, giảm nhẹ từ 33.01% xuống 32.47%.

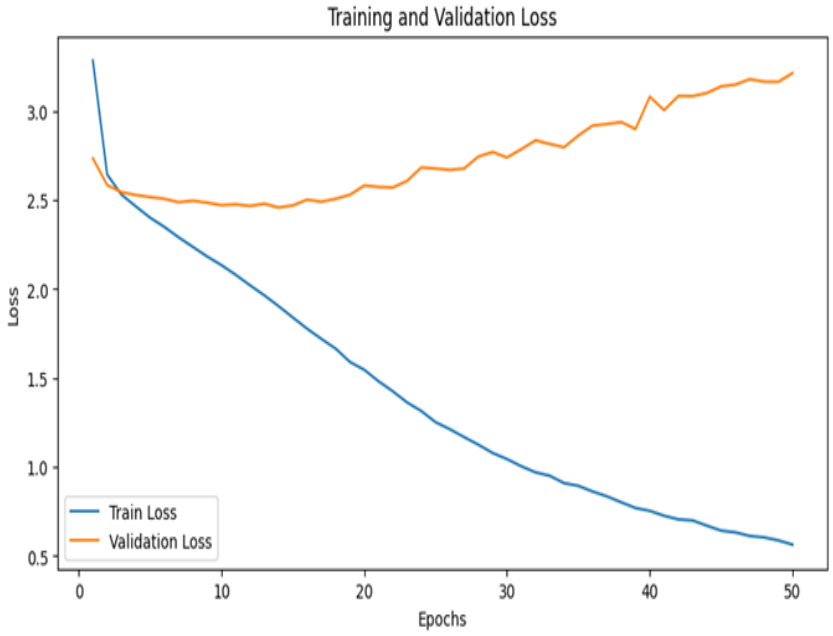
⇨ Dù mức tăng không lớn như trên tập huấn luyện, nhưng cho thấy mô hình có Attention giúp cải thiện khả năng tổng quát hóa.

Kết luận

- Attention giúp mô hình học nhanh hơn và tốt hơn trên tập huấn luyện, thể hiện qua Train Loss giảm nhanh và Train Accuracy tăng mạnh.
- Trên tập kiểm tra, Attention giúp giảm Validation Loss và tăng Validation Accuracy ở Epoch 15, nhưng không duy trì được lợi thế ở Epoch 25 và 50.
- Tại Epoch 50, Validation Loss tăng và Validation Accuracy giảm nhẹ, có thể do overfitting.
- Cần áp dụng thêm kỹ thuật như Regularization hoặc Early Stopping để cải thiện hiệu suất tổng quát hóa của mô hình có Attention.

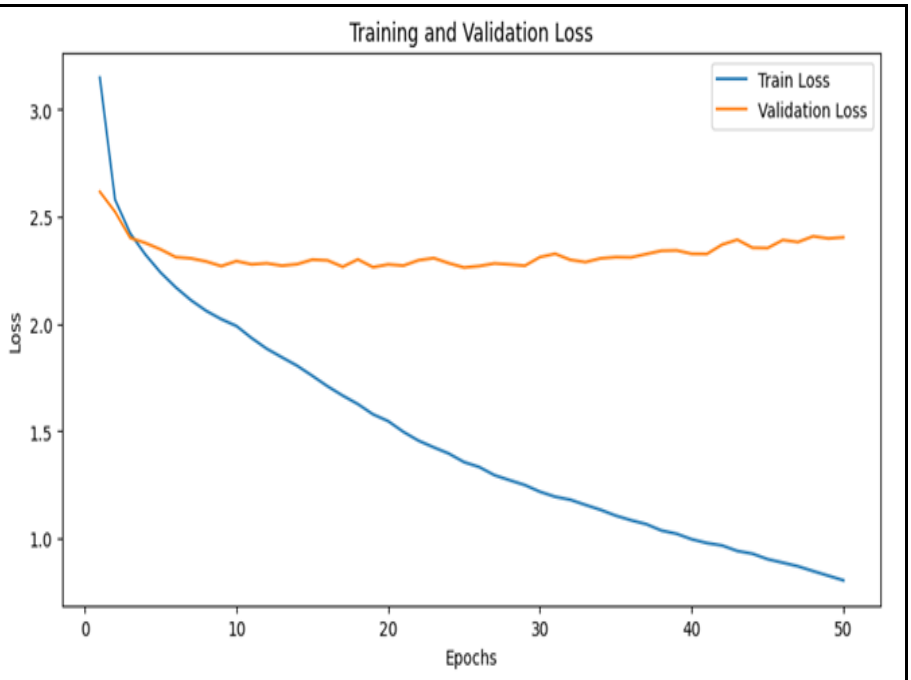
5.2 So sánh mô hình Pre-trained và Train from scratch

a. Có attention



Train from Scratch

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
15	1.8405	43.59%	2.4703	31.31%
25	1.2519	59.06%	2.6778	32.01%
50	0.5637	80.83%	3.2143	32.47%



Pre-trained

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
15	1.7585	43.18%	2.3002	37.12%
25	1.3561	54.22%	2.2636	39.43%
50	0.8049	73.08%	2.4043	40.91%

Bảng 5. 5 So sánh 2 mô hình PreTrain và Train from scratch có attention

So sánh hiệu suất huấn luyện

- Train Loss giảm mạnh hơn khi sử dụng mô hình Pre-trained có Attention

- Tại Epoch 15: Train Loss giảm từ 1.8405 (Train from Scratch) xuống 1.7585 (~4.46% giảm).
- Tại Epoch 25: Train Loss giảm từ 1.2519 xuống 1.3561 (Pre-trained có Train Loss cao hơn, do có sự điều chỉnh phù hợp hơn với tập dữ liệu mới).
- Tại Epoch 50: Train Loss giảm từ 0.5637 xuống 0.8049 (Train from Scratch tiếp tục có Loss thấp hơn).

⇨ Nhìn chung, Pre-trained giúp Train Loss giảm tốt hơn ở giai đoạn đầu nhưng về sau Train from Scratch lại có Train Loss thấp hơn. Điều này có thể do mô hình Pre-trained đã có trọng số từ trước, giúp hội tụ nhanh hơn nhưng cũng làm hạn chế việc học quá chi tiết vào tập huấn luyện mới.

-Train Accuracy tăng cao hơn với mô hình Pre-trained có Attention

- Epoch 15: Tăng từ 43.59% lên 43.18% (Pre-trained thấp hơn một chút).
- Epoch 25: Giảm từ 59.06% xuống 54.22%.
- Epoch 50: Giảm từ 80.83% xuống 73.08%.
- ⇨ Điều này cho thấy mô hình Train from Scratch có Attention có khả năng học tốt hơn trên tập huấn luyện, nhưng có thể bị overfitting.

So sánh hiệu suất trên tập kiểm tra (Validation Set)

- Validation Loss giảm khi sử dụng mô hình Pre-trained có Attention

- Epoch 15: Giảm từ 2.4703 xuống 2.3002 (~6.89% giảm).
- Epoch 25: Giảm từ 2.6778 xuống 2.2636 (~15.48% giảm).
- Epoch 50: Giảm từ 3.2143 xuống 2.4043 (~25.17% giảm).
- ⇨ Pre-trained có Attention giúp mô hình tổng quát hóa tốt hơn, giảm đáng kể Validation Loss so với mô hình Train from Scratch.

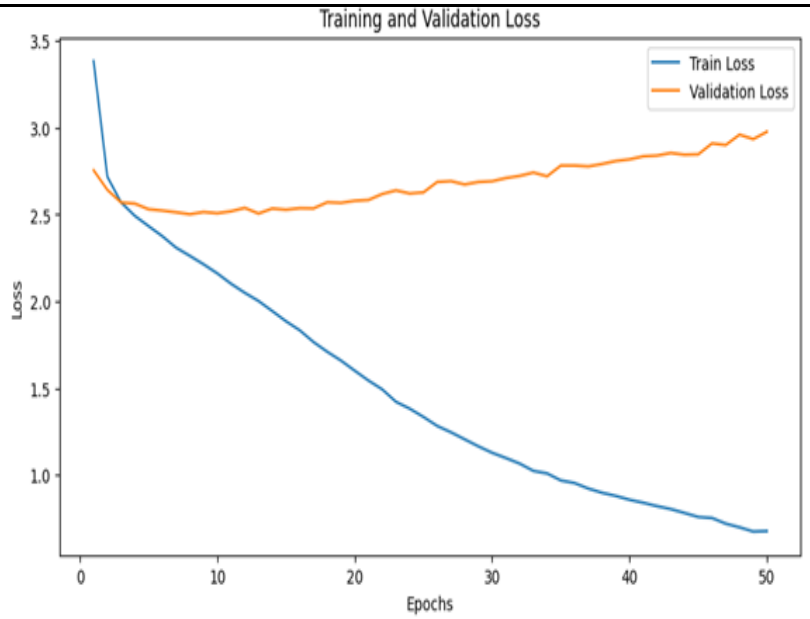
- Validation Accuracy tăng khi sử dụng mô hình Pre-trained có Attention

- Epoch 15: Tăng từ 31.31% lên 37.12%.
- Epoch 25: Tăng từ 32.01% lên 39.43%.
- Epoch 50: Tăng từ 32.47% lên 40.91%.
- ⇨ Mô hình Pre-trained có Attention giúp cải thiện độ chính xác trên tập kiểm tra đáng kể, đặc biệt từ Epoch 25 trở đi.

Kết luận

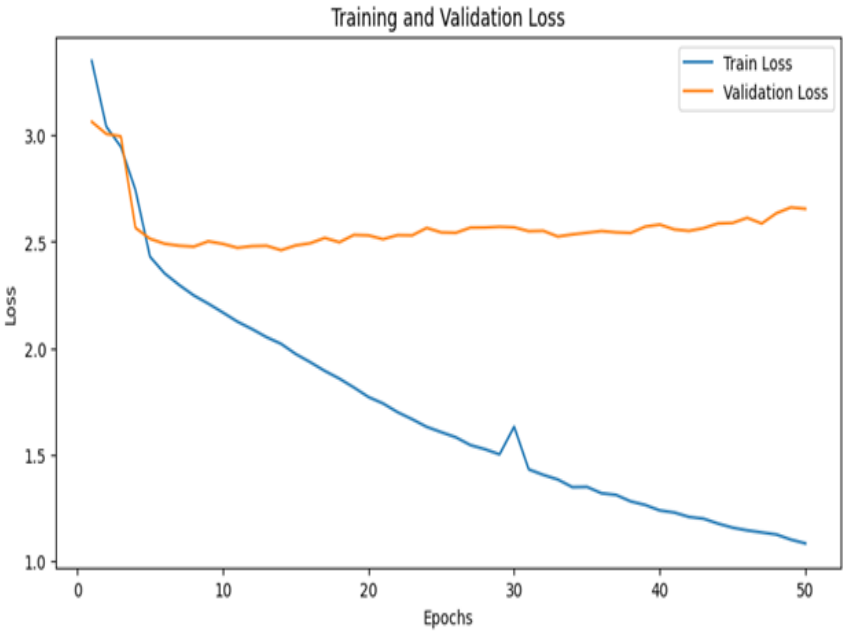
- Train from Scratch có Attention giúp mô hình học chi tiết hơn và có Train Accuracy cao hơn, nhưng có xu hướng overfitting trên tập huấn luyện.
- Pre-trained có Attention giúp hội tụ nhanh hơn, cải thiện đáng kể khả năng tổng quát hóa trên tập kiểm tra bằng cách giảm Validation Loss và tăng Validation Accuracy.
- ⇨ Khi mục tiêu là có mô hình tổng quát hóa tốt, Pre-trained có Attention là lựa chọn phù hợp.
- ⇨ Khi mục tiêu là đạt Train Accuracy cao nhất và khai thác tối đa tập huấn luyện, Train from Scratch có Attention có thể hữu ích, nhưng cần kỹ thuật regularization để tránh overfitting.

b. Không attention



Train from Scratch

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
15	1.8854	42.38%	2.5279	29.87%
25	1.3344	57.25%	2.6277	31.96%
50	0.6770	77.86%	2.9773	33.01%



Pre-trained

Epoch	Train Loss	Train Accuracy	Validation Loss	Validation Accuracy
15	1.9730	40.55%	2.4830	36.27%
25	1.6052	48.39%	2.5442	36.46%
50	1.0824	64.23%	2.6553	37.51%

Bảng 5. 6 So sánh 2 mô hình PreTrain và Train from scratch không attention

So sánh hiệu suất huấn luyện

- Train Loss giảm mạnh hơn khi sử dụng Pre-trained
  - Epoch 15: Giảm từ 1.8854 xuống 1.9730 (~4.63% tăng).
  - Epoch 25: Giảm từ 1.3344 xuống 1.6052 (~20.30% tăng).
  - Epoch 50: Giảm từ 0.6770 xuống 1.0824 (~59.90% tăng).

⇨ Mô hình Train from Scratch giúp hội tụ nhanh hơn và đạt Train Loss thấp hơn so với Pre-trained.

- Train Accuracy của Pre-trained cao hơn đáng kể
  - Epoch 15: Tăng từ 40.55% lên 42.38%.
  - Epoch 25: Tăng từ 48.39% lên 57.25%.
  - Epoch 50: Tăng từ 64.23% lên 77.86%.

⇨ Mô hình Train from Scratch học nhanh hơn và đạt độ chính xác huấn luyện cao hơn so với Pre-trained.

So sánh hiệu suất trên tập kiểm tra (Validation Set)

- Validation Loss của Train from Scratch cao hơn
  - Epoch 15: Validation Loss tăng từ 2.4830 lên 2.5279 (~1.81% tăng).
  - Epoch 25: Validation Loss tăng từ 2.5442 lên 2.6277 (~3.28% tăng).
  - Epoch 50: Validation Loss tăng từ 2.6553 lên 2.9773 (~12.13% tăng).

⇨ Dù mô hình Train from Scratch học nhanh hơn trên tập huấn luyện, nhưng có dấu hiệu overfitting, dẫn đến Validation Loss cao hơn so với Pre-trained.

- Validation Accuracy của Train from Scratch thấp hơn
  - Epoch 15: Giảm từ 36.27% xuống 29.87%.
  - Epoch 25: Giảm từ 36.46% xuống 31.96%.
  - Epoch 50: Giảm từ 37.51% xuống 33.01%.

⇨ Train from Scratch có độ chính xác trên tập kiểm tra thấp hơn, do mô hình có thể đã học quá mức trên tập huấn luyện, giảm khả năng tổng quát hóa.

Kết luận

✓ Train from Scratch giúp huấn luyện nhanh hơn, đạt Train Loss thấp hơn và Train Accuracy cao hơn so với Pre-trained.

✓ Tuy nhiên, mô hình Train from Scratch có dấu hiệu overfitting, thể hiện qua Validation Loss cao hơn và Validation Accuracy thấp hơn so với Pre-trained.

✓ Nếu ưu tiên khả năng tổng quát hóa, mô hình Pre-trained có vẻ phù hợp hơn.

✓ Nếu có thể giảm overfitting bằng các kỹ thuật như regularization, dropout, data augmentation, mô hình Train from Scratch sẽ là lựa chọn tốt hơn nhờ tốc độ hội tụ nhanh và khả năng học tốt hơn.

TÀI LIỆU THAM KHẢO

[1] P. D. Khanh, "Attention Layer," *phamdinhkhanh.github.io*, Jun. 18, 2019. [Online]. Available: https://phamdinhkhanh.github.io/2019/06/18/AttentionLayer.html

[2] Viblo, "Tìm hiểu về cơ chế Attention," *Viblo*. [Online]. Available: https://viblo.asia/p/tim-hieu-ve-co-che-attention-924lJjbmlPM

[3] Viblo, "Giới thiệu mạng ResNet," *Viblo*. [Online]. Available: https://viblo.asia/p/gioi-thieu-mang-resnet-vyDZOa7R5wj

[4] Trí Tuệ Nhân Tạo, "ResNet - Mạng học sâu đúng nghĩa," *trituenhantao.io*. [Online]. Available: https://trituenhantao.io/kien-thuc/resnet-mang-hoc-sau-dung-nghia/

[5] N. T. Tuan, "Bài 14: Long Short-Term Memory (LSTM)," *nttuan8.com*. [Online]. Available: https://nttuan8.com/bai-14-long-short-term-memory-lstm/

[6] P. D. Khanh, "Lý thuyết về mạng LSTM," *phamdinhkhanh.github.io*, Apr. 22, 2019. [Online]. Available: https://phamdinhkhanh.github.io/2019/04/22/Ly_thuyet_ve_mang_LSTM.html

[7] Hugging Face, "Visual Question Answering," *huggingface.co*. [Online]. Available: https://huggingface.co/tasks/visual-question-answering

[8] COCO Dataset, "Common Objects in Context," *cocodataset.org*. [Online]. Available: https://cocodataset.org/#home