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

BÁO CÁO CUỐI KÌ MÔN HỌC SÂU

Tìm hiểu cơ chế Attention trong LLMs và ứng dụng vào bài toán OCR sử dụng CNN-Transformer

Người hướng dẫn: **PGS.TS. LÊ ANH CƯỜNG**
Người thực hiện: **HÀ TRỌNG NGUYỄN - 52200148**
ĐỖ THỊ KIỀU THANH - 52200144

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

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

BÁO CÁO CUỐI KÌ MÔN HỌC SÂU

Tìm hiểu cơ chế Attention trong LLMs và ứng dụng vào bài toán OCR sử dụng CNN-Transformer

Người hướng dẫn: **PGS.TS. LÊ ANH CƯỜNG**
Người thực hiện: **HÀ TRỌNG NGUYỄN - 52200148**
ĐỖ THỊ KIỀU THANH - 52200144

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025

LỜI CẢM ƠN

Chúng tôi xin gửi lời cảm ơn đến thầy Lê Anh Cường, giảng viên phụ trách giảng dạy chúng tôi môn Học sâu. Nhờ sự giúp đỡ tận tình của thầy, các thắc mắc liên quan đến bài học và dự án này của chúng tôi đã được sáng tỏ. Chúng tôi đã hiểu rõ hơn về kiến thức mình được học, từ đó cô đọng và hoàn chỉnh được báo cáo cũng như dự án này. Tuy nhiên vậy, trong quá trình viết báo cáo cũng không tránh được các sai sót không mong muốn, chúng tôi mong thầy có thể đưa ra nhận xét giúp chúng tôi hoàn thiện bài báo cáo này. Chúng tôi xin cảm ơn và chúc thầy cũng như thầy cô tổ bộ môn có nhiều sức khoẻ.

BÁO CÁO ĐƯỢC HOÀN THÀNH
TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Chúng tôi xin cam đoan đây là sản phẩm của riêng chúng tôi và được sự hướng dẫn của PGS.TS. Lê Anh Cường. Các nội dung nghiên cứu, kết quả trong báo cáo 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 báo cáo 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 báo cáo 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 tháng năm

Tác giả

(ký tên và ghi rõ họ tên)

Hà Trọng Nguyễn

Đỗ Thị Kiều Thanh

TÓM TẮT

Báo cáo này tập trung nghiên cứu hai vấn đề chính đó là hảo sát và phân tích các cơ chế attention tiên tiến trong Mô hình Ngôn ngữ Lớn (LLMs), và ứng dụng kiến trúc kết hợp giữa Mạng Nơ-ron Tích chập (CNN) và Transformer-Decoder vào bài toán Nhận dạng Ký tự Quang học (OCR) để trích xuất văn bản từ ảnh.

Chương 1: Đi sâu vào tìm hiểu bảy cơ chế attention quan trọng đang được sử dụng và phát triển trong các LLM hiện đại, bao gồm Self-Attention gốc, Multi-Query Attention (MQA), Grouped-Query Attention (GQA), FlashAttention (v1 và v2), Linear Attention, Sparse Attention và Rotary Positional Embedding (RoPE). Đối với mỗi cơ chế, báo cáo trình bày chi tiết về nguyên lý hoạt động, công thức toán học, độ phức tạp tính toán, hiệu quả sử dụng bộ nhớ, cũng như phân tích các ưu điểm và hạn chế. Mục tiêu là cung cấp cái nhìn tổng quan về sự phát triển của các kỹ thuật attention nhằm giải quyết những thách thức về hiệu suất và khả năng xử lý chuỗi dài của cơ chế Self-Attention truyền thống. Các cơ chế này được so sánh và phân tích để đưa ra hướng dẫn lựa chọn phù hợp cho các tình huống và yêu cầu mô hình khác nhau, đồng thời thảo luận về xu hướng phát triển trong tương lai.

Chương 2: Trình bày ứng dụng cụ thể của kiến trúc kết hợp CNN và Transformer-Decoder cho bài toán OCR. Báo cáo giải thích vai trò của CNN (ví dụ: ResNet) trong việc trích xuất các đặc trưng hình ảnh từ ảnh đầu vào và vai trò của Transformer-Decoder trong việc sinh chuỗi văn bản một cách tuần tự. Cơ chế chú ý chéo (cross-attention) đóng vai trò cầu nối, cho phép bộ giải mã tập trung vào các vùng đặc trưng ảnh phù hợp tại mỗi bước sinh ký tự, giúp mô hình xử lý hiệu quả văn bản trong các ảnh phức tạp. Các thành phần chính của Transformer-Decoder như Masked Multi-Head Self-Attention, Multi-Head Cross-Attention, Feed Forward Network và mã hóa vị trí (Positional Encoding) cũng được làm rõ trong ngữ cảnh OCR.

Chương 3: Đây là phần thực nghiệm để đánh giá hiệu quả của mô hình CNN-Transformer được xây dựng. Phần này mô tả tập dữ liệu được sử dụng và các phương

pháp đánh giá hiệu suất phổ biến trong OCR (như Accuracy, Precision, Recall, F1-score, CER, WER). Kết quả thực nghiệm (dự kiến) sẽ được trình bày để minh họa khả năng nhận dạng văn bản của mô hình trên tập dữ liệu kiểm tra, từ đó rút ra các kết luận về tính hiệu quả và những hạn chế của phương pháp tiếp cận này.

MỤC LỤC

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

DANH MỤC HÌNH

DANH MỤC BẢNG

CHƯƠNG 1 – CƠ CHẾ ATTENTION TRONG LLMS

1.1 Giới thiệu

Trong những năm gần đây, mô hình ngôn ngữ lớn (Large Language Models - LLMs) đã trở thành một trong những thành tựu nổi bật nhất trong lĩnh vực trí tuệ nhân tạo, mang lại những bước tiến đáng kể trong việc xử lý ngôn ngữ tự nhiên. Sự phát triển của các mô hình như GPT, LLaMA, PaLM và Mistral đã chứng minh khả năng hiểu và tạo ra văn bản gần như con người, mở ra nhiều ứng dụng mới trong dịch thuật, tóm tắt văn bản, trả lời câu hỏi, và thậm chí cả sáng tạo nội dung. Tuy nhiên, đằng sau những thành tựu ấn tượng này là một cơ chế quan trọng đã cách mạng hóa kiến trúc mạng nơ-ron: cơ chế attention.

Cơ chế attention, được giới thiệu lần đầu trong bài báo mang tính đột phá "Attention Is All You Need" của Vaswani và cộng sự vào năm 2017, đã trở thành nền tảng cho kiến trúc Transformer - kiến trúc cốt lõi của hầu hết các mô hình ngôn ngữ lớn hiện đại. Cơ chế này cho phép mô hình tập trung vào các phần khác nhau của dữ liệu đầu vào với các mức độ quan tâm khác nhau, giúp nắm bắt được các mối quan hệ phức tạp và phụ thuộc dài hạn trong dữ liệu. Đây là một bước tiến quan trọng so với các kiến trúc trước đó như RNN (Recurrent Neural Networks) và LSTM (Long Short-Term Memory), vốn gặp khó khăn trong việc xử lý các chuỗi dài do vấn đề gradient biến mất hoặc bùng nổ.

Tuy nhiên, cơ chế attention truyền thống có một hạn chế đáng kể: độ phức tạp tính toán và bộ nhớ tăng theo hàm bậc hai của độ dài chuỗi đầu vào ($O(n^2)$). Điều này tạo ra một rào cản lớn khi mở rộng quy mô mô hình để xử lý các chuỗi dài hơn hoặc tăng kích thước mô hình. Khi các mô hình ngôn ngữ lớn ngày càng phát triển về quy mô và khả năng, việc tối ưu hóa cơ chế attention trở nên cấp thiết hơn bao giờ hết.

Trong những năm gần đây, nhiều biến thể của cơ chế attention đã được đề xuất nhằm giải quyết những thách thức này. Các cơ chế như Multi-Query Attention (MQA), Grouped-Query Attention (GQA), FlashAttention, Linear Attention, Sparse Attention và Rotary Positional Embedding (RoPE) đã được phát triển với mục tiêu cải thiện hiệu quả

tính toán, giảm yêu cầu bộ nhớ, hoặc nâng cao khả năng mô hình hóa các mối quan hệ phụ thuộc trong dữ liệu. Mỗi cơ chế đều có những ưu điểm và hạn chế riêng, phù hợp với các tình huống và yêu cầu khác nhau.

Báo cáo này nhằm mục đích cung cấp một phân tích toàn diện về bảy cơ chế attention quan trọng trong các mô hình ngôn ngữ lớn hiện đại. Chúng tôi sẽ đi sâu vào nguyên lý hoạt động, công thức toán học, độ phức tạp tính toán, ưu điểm và hạn chế của từng cơ chế. Ngoài ra, chúng tôi cũng sẽ so sánh hiệu suất của các cơ chế này trong các tình huống khác nhau và thảo luận về ứng dụng của chúng trong các mô hình ngôn ngữ lớn hiện đại.

Việc hiểu rõ các cơ chế attention không chỉ quan trọng đối với các nhà nghiên cứu và phát triển trong lĩnh vực học máy, mà còn cung cấp những hiểu biết quý giá về cách thức hoạt động bên trong của các mô hình ngôn ngữ lớn - những công cụ đang ngày càng trở nên phổ biến và có ảnh hưởng trong cuộc sống hàng ngày của chúng ta. Thông qua báo cáo này, chúng tôi hy vọng sẽ cung cấp một nguồn tài liệu tham khảo hữu ích cho những ai quan tâm đến sự phát triển và tương lai của các mô hình ngôn ngữ lớn và trí tuệ nhân tạo nói chung.

1.2 Self-Attention: Nền tảng của các mô hình transformer

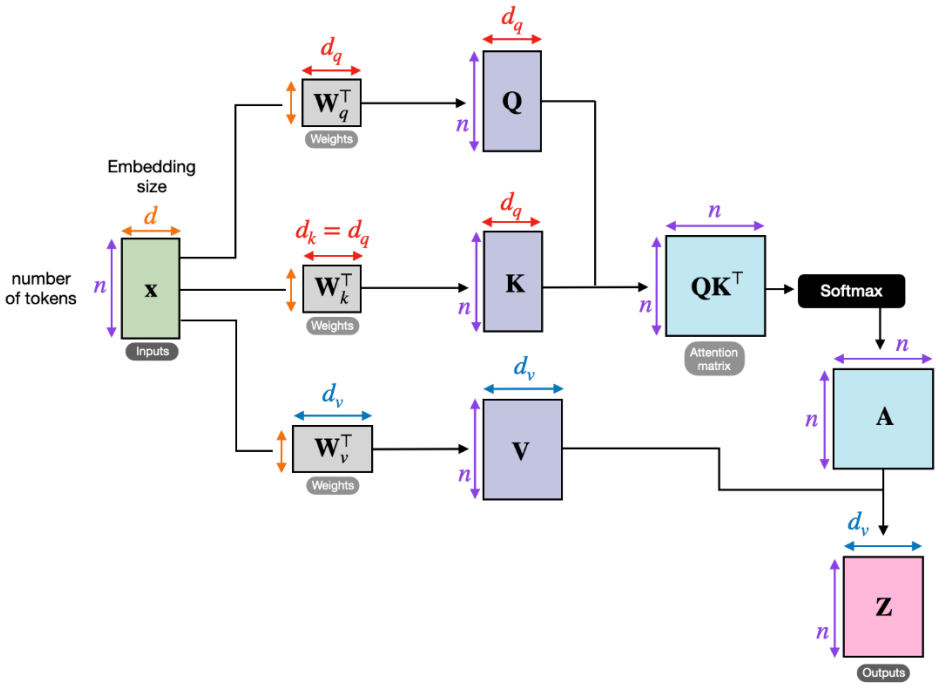
Self-Attention, hay còn được gọi là cơ chế tự chú ý, là thành phần cốt lõi và nền tảng của kiến trúc Transformer, được giới thiệu lần đầu trong bài báo “Attention Is All You Need” của Vaswani và cộng sự vào năm 2017. Cơ chế này đã tạo ra một bước ngoặt trong lĩnh vực xử lý ngôn ngữ tự nhiên, thay thế hoàn toàn các kiến trúc dựa trên mạng nơ-ron hồi quy (RNN) truyền thống và mở ra kỷ nguyên mới cho các mô hình ngôn ngữ lớn.

1.2.1 Nguyên lý hoạt động của self-attention

Ý tưởng cốt lõi của self-attention là cho phép mỗi phần tử trong một chuỗi đầu vào tương tác với tất cả các phần tử khác trong cùng chuỗi đó, từ đó học được mối quan hệ và sự phụ thuộc giữa chúng. Điều này khác biệt so với các kiến trúc RNN truyền thống,

nơi thông tin chỉ được truyền tuần tự từ phần tử này sang phần tử tiếp theo, gây ra vấn đề mất thông tin khi chuỗi quá dài.

Trong self-attention, mỗi token (từ hoặc phần tử) trong chuỗi đầu vào được biểu diễn bởi ba vector: Query (Q), Key (K) và Value (V). Các vector này được tạo ra bằng cách nhân vector nhúng của token với ba ma trận trọng số khác nhau (WQ, WK, WV). Quá trình này có thể được hình dung như sau: mỗi token đặt "câu hỏi" (query) cho tất cả các token khác, so sánh câu hỏi này với "chìa khóa" (key) của mỗi token, và dựa vào độ tương đồng để quyết định mức độ "chú ý" dành cho "giá trị" (value) của token đó.



Hình 1.1 Mô tả nguyên lý hoạt động của Self-Attention

1.2.2 Công thức toán học và biểu diễn ma trận

Về mặt toán học, self-attention được tính toán thông qua các bước sau:

1. Tạo ra các vector Query, Key và Value cho mỗi token:
   - $Q = X \times WQ$
   - $K = X \times WK$
   - $V = X \times WV$

Trong đó X là ma trận đầu vào (kích thước [batch_size, seq_length, d_model]), và WQ, WK, WV là các ma trận trọng số (kích thước [d_model, d_k] hoặc [d_model, d_v]).

2. Tính toán điểm số attention bằng cách nhân ma trận Q với K chuyển vị:

- $Scores = Q \times K\text{^}T$

3. Chia điểm số cho căn bậc hai của kích thước vector key để ổn định gradient:

- $Scores = \frac{Scores}{\sqrt{d_k}}$

4. Áp dụng hàm softmax để chuyển điểm số thành trọng số attention:

- $Attention\_weights = softmax(Scores)$

5. Nhân trọng số attention với ma trận Value để có được đầu ra cuối cùng:

- $Output = Attention\_weights \times V$

Công thức tổng quát của self-attention có thể được biểu diễn như sau:

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

Trong kiến trúc Transformer, self-attention thường được mở rộng thành multi-head attention, trong đó quá trình attention được thực hiện song song trên nhiều "đầu" (head) khác nhau, mỗi đầu học một loại mối quan hệ khác nhau giữa các token. Kết quả từ các đầu sau đó được nối lại và truyền qua một lớp tuyến tính.

1.2.3 Độ phức tạp tính toán

Một trong những hạn chế chính của self-attention là độ phức tạp tính toán và bộ nhớ. Với chuỗi đầu vào có độ dài n, ma trận điểm số attention (Scores) có kích thước [n × n], dẫn đến độ phức tạp tính toán và bộ nhớ là $O(n^2)$. Điều này tạo ra thách thức lớn khi xử lý các chuỗi dài, đặc biệt là trong các ứng dụng như xử lý văn bản dài, phân tích video, hoặc xử lý âm thanh.

Ví dụ, với chuỗi độ dài 1,000 token, ma trận attention sẽ có 1,000,000 phần tử. Khi tăng lên 10,000 token, con số này trở thành 100,000,000 phần tử, đòi hỏi bộ nhớ và sức mạnh tính toán đáng kể. Đây là lý do chính khiến các mô hình Transformer ban đầu như

BERT và GPT-2 bị giới hạn ở độ dài chuỗi tương đối ngắn (thường là 512 hoặc 1,024 token).

1.2.4 Ưu điểm và hạn chế

Self-attention có nhiều ưu điểm nổi bật:

1. **Khả năng mô hình hóa phụ thuộc dài hạn**: Self-attention cho phép mỗi token tương tác trực tiếp với tất cả các token khác, bất kể khoảng cách giữa chúng trong chuỗi, giúp nắm bắt được các mối quan hệ phụ thuộc dài hạn.
2. **Tính song song cao**: Không giống như RNN, self-attention có thể được tính toán song song cho tất cả các token trong chuỗi, giúp tăng tốc đáng kể quá trình huấn luyện.
3. **Khả năng giải thích**: Ma trận trọng số attention cung cấp thông tin về cách mô hình "chú ý" đến các phần khác nhau của đầu vào, giúp hiểu rõ hơn về cách mô hình đưa ra quyết định.
4. **Hiệu quả trong nhiều tác vụ**: Self-attention đã chứng minh hiệu quả trong nhiều tác vụ xử lý ngôn ngữ tự nhiên, thị giác máy tính, và thậm chí cả xử lý âm thanh.

Tuy nhiên, self-attention cũng có một số hạn chế đáng kể:

1. **Độ phức tạp bậc hai**: Như đã đề cập, độ phức tạp tính toán và bộ nhớ $O(n^2)$ là một rào cản lớn khi xử lý các chuỗi dài.
2. **Thiếu thông tin vị trí tự nhiên**: Self-attention không tự nhiên mã hóa thông tin về vị trí của các token trong chuỗi, đòi hỏi phải thêm vào các embedding vị trí.
3. **Chi phí tính toán cao**: Mặc dù có tính song song cao, nhưng tổng chi phí tính toán của self-attention vẫn rất lớn, đặc biệt là khi số lượng tham số tăng lên.

1.2.5 Ứng dụng trong các mô hình LLM hiện đại

Self-attention là nền tảng của hầu hết các mô hình ngôn ngữ lớn hiện đại, mặc dù nhiều mô hình đã áp dụng các biến thể hoặc tối ưu hóa để giải quyết các hạn chế của nó:

1. **BERT và các mô hình mã hóa**: Sử dụng self-attention hai chiều, cho phép mỗi token chú ý đến tất cả các token khác trong chuỗi, phù hợp cho các tác vụ như phân loại văn bản và trích xuất thông tin.
2. **GPT và các mô hình tạo sinh**: Sử dụng self-attention một chiều (masked self-attention), trong đó mỗi token chỉ có thể chú ý đến các token trước nó, phù hợp cho việc tạo sinh văn bản tự động.
3. **T5 và các mô hình encoder-decoder**: Kết hợp cả self-attention hai chiều trong encoder và self-attention một chiều trong decoder, phù hợp cho các tác vụ như dịch máy và tóm tắt văn bản.

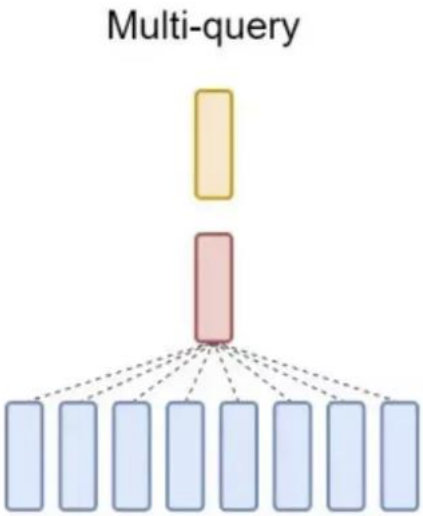
Mặc dù có những hạn chế, self-attention vẫn là một trong những đột phá quan trọng nhất trong lĩnh vực học máy và xử lý ngôn ngữ tự nhiên trong thập kỷ qua. Sự thành công của nó đã thúc đẩy làn sóng nghiên cứu về các biến thể và cải tiến, nhằm giải quyết các hạn chế và mở rộng khả năng của các mô hình Transformer. Trong các phần tiếp theo, chúng ta sẽ khám phá những biến thể này và cách chúng đã góp phần vào sự phát triển của các mô hình ngôn ngữ lớn hiện đại.

1.3 Multi-Query Attention (MQA)

Multi-Query Attention (MQA) là một biến thể quan trọng của cơ chế self-attention truyền thống, được giới thiệu bởi Shazeer và cộng sự trong bài báo “Fast Transformer Decoding: One Write-Head is All You Need” vào năm 2019. MQA được phát triển với mục tiêu chính là giảm yêu cầu bộ nhớ và tăng tốc độ suy luận (inference) trong các mô hình transformer, đặc biệt là trong giai đoạn giải mã tự hồi quy (autoregressive decoding).

1.3.1 Nguyên lý hoạt động và sự khác biệt so với self-attention truyền thống

Trong self-attention truyền thống với nhiều đầu (multi-head attention), mỗi đầu attention có ba tập ma trận trọng số riêng biệt để tạo ra các vector Query (Q), Key (K) và Value (V). Điều này có nghĩa là với H đầu attention, mô hình cần lưu trữ và tính toán 3H bộ ma trận trọng số và các vector tương ứng.



Hình 1.2 Nguyên lý hoạt động Multi-Query Attention (MQA)

Multi-Query Attention đưa ra một cách tiếp cận khác biệt: thay vì mỗi đầu attention có các ma trận Key và Value riêng biệt, MQA sử dụng chung một ma trận Key và một ma trận Value cho tất cả các đầu attention. Chỉ có ma trận Query là khác nhau giữa các đầu. Cụ thể:

1. Trong multi-head attention truyền thống:
   - Mỗi đầu i có: $Q_i = X \times W^{Q_i,K_i} = X \times W^{K_i,V_i} = X \times W^{V_i}$
   - Tổng cộng có 3H ma trận trọng số
2. Trong multi-query attention:
   - Mỗi đầu i có: $Q_i = X \times W^{Q_i}$
   - Nhưng tất cả các đầu đều dùng chung: K = X × W^K, V = X × W^V
   - Tổng cộng có H+2 ma trận trọng số

Sự khác biệt này có vẻ đơn giản nhưng mang lại tác động đáng kể đến hiệu suất và yêu cầu bộ nhớ của mô hình, đặc biệt trong quá trình suy luận.

1.3.2 Công thức toán học và biểu diễn ma trận

Về mặt toán học, Multi-Query Attention được tính toán như sau:

1. Tạo ra các vector Query cho mỗi đầu attention:

- $Q_i = X \times W^{Q_i}$ ($với\ i$ từ 1 đến H)

2. Tạo ra một vector Key và một vector Value dùng chung:
   - $K = X \times W^K$
   - $V = X \times W^V$
3. Tính toán điểm số attention cho mỗi đầu:
   - $Scores_i = \frac{(Q_i \times K^T)}{\sqrt{d_k}}$
4. Áp dụng hàm softmax để chuyển điểm số thành trọng số attention:
   - $Attention\_weights_i = softmax(Scores_i)$
5. Nhân trọng số attention với vector Value để có được đầu ra cho mỗi đầu:
   - $Output_i = Attention\_weights_i \times V$
6. Nối (concatenate) đầu ra từ tất cả các đầu và truyền qua một lớp tuyến tính:
   - $Output = Concat(Output_1, \ldots, Output_H) \times W^O$

   Công thức tổng quát của MQA có thể được biểu diễn như sau:

$$MQA(X) = Concat\left(\frac{softmax((Q_1 \times K^T)}{\sqrt{d_k}}\right) \times V, \ldots, \frac{softmax((Q_H \times K^T)}{\sqrt{d_k}}) V) \times W^O$$

1.3.3 Độ phức tạp tính toán và hiệu quả bộ nhớ

Lợi ích chính của Multi-Query Attention nằm ở hiệu quả bộ nhớ và tốc độ suy luận:

1. **Giảm yêu cầu bộ nhớ**: Trong quá trình suy luận tự hồi quy, các ma trận Key và Value từ các token đã được tạo ra cần được lưu trữ để tính toán attention cho token tiếp theo. Với multi-head attention truyền thống, cần lưu trữ H cặp ma trận Key và Value. Với MQA, chỉ cần lưu trữ một cặp duy nhất, giảm đáng kể yêu cầu bộ nhớ.
2. **Tăng tốc độ suy luận**: Việc giảm số lượng ma trận cần tính toán và lưu trữ dẫn đến tăng tốc độ suy luận, đặc biệt là trên các thiết bị có băng thông bộ nhớ hạn chế.

3. **Độ phức tạp tính toán**: Về mặt lý thuyết, độ phức tạp tính toán vẫn là $O(n^2)$ như self-attention truyền thống, nhưng hằng số nhân nhỏ hơn đáng kể do giảm số lượng phép nhân ma trận.

Cụ thể, với chuỗi độ dài n và H đầu attention:

- Multi-head attention truyền thống yêu cầu lưu trữ $2 \times H \times n \times d_k$ phần tử (cho các ma trận Key và Value)
- Multi-query attention chỉ yêu cầu lưu trữ $2 \times n \times d_k$ phần tử

Với các mô hình lớn có nhiều đầu attention (ví dụ: GPT-3 có 96 đầu attention), sự khác biệt này trở nên rất đáng kể.

1.3.4 Ưu điểm và hạn chế

Multi-Query Attention có nhiều ưu điểm nổi bật:

1. **Hiệu quả bộ nhớ cao**: Giảm đáng kể yêu cầu bộ nhớ trong quá trình suy luận, cho phép xử lý các chuỗi dài hơn hoặc batch lớn hơn với cùng một lượng bộ nhớ.
2. **Tốc độ suy luận nhanh hơn**: Giảm số lượng phép tính và truy cập bộ nhớ, dẫn đến tốc độ suy luận nhanh hơn, đặc biệt là trên các thiết bị có băng thông bộ nhớ hạn chế.
3. **Dễ triển khai**: Có thể dễ dàng áp dụng cho các kiến trúc transformer hiện có mà không cần thay đổi lớn về kiến trúc.
4. **Khả năng chuyển đổi từ mô hình đã huấn luyện**: Các mô hình đã được huấn luyện với multi-head attention có thể được chuyển đổi sang multi-query attention thông qua việc trung bình hóa các ma trận Key và Value.

Tuy nhiên, MQA cũng có một số hạn chế:

1. **Giảm nhẹ chất lượng**: Việc sử dụng chung ma trận Key và Value có thể dẫn đến giảm nhẹ chất lượng mô hình so với multi-head attention truyền thống, đặc biệt là trong các tác vụ phức tạp.

2. **Không hiệu quả trong quá trình huấn luyện**: Lợi ích chính của MQA nằm ở giai đoạn suy luận, trong khi trong quá trình huấn luyện, sự cải thiện về hiệu suất không đáng kể.
3. **Khả năng biểu diễn hạn chế hơn**: Việc sử dụng chung ma trận Key và Value có thể hạn chế khả năng của mô hình trong việc nắm bắt các mối quan hệ phức tạp trong dữ liệu.

1.3.5 Ứng dụng trong các mô hình như PaLM, StarCoder, Falcon

Multi-Query Attention đã được áp dụng thành công trong nhiều mô hình ngôn ngữ lớn hiện đại:

1. **PaLM (Pathways Language Model)**: Google đã sử dụng MQA trong mô hình PaLM để tăng tốc độ suy luận và giảm yêu cầu bộ nhớ, cho phép mô hình xử lý các chuỗi dài hơn với hiệu quả cao hơn.
2. **StarCoder**: Mô hình StarCoder, được phát triển để hiểu và tạo mã nguồn, sử dụng MQA để cải thiện hiệu suất khi làm việc với các tệp mã nguồn dài.
3. **Falcon**: Mô hình Falcon của Technology Innovation Institute (TII) đã áp dụng MQA như một phần của chiến lược tối ưu hóa, giúp mô hình đạt được hiệu suất cao hơn với cùng một lượng tài nguyên tính toán.
4. **MPT (MosaicML Pretrained Transformer)**: Mô hình MPT cũng sử dụng MQA để cải thiện hiệu quả bộ nhớ và tốc độ suy luận.

Các mô hình này đã chứng minh rằng MQA có thể được áp dụng hiệu quả trong các mô hình quy mô lớn mà không làm giảm đáng kể chất lượng. Trong một số trường hợp, việc giảm yêu cầu bộ nhớ cho phép tăng kích thước batch hoặc độ dài chuỗi, dẫn đến cải thiện hiệu suất tổng thể.

Multi-Query Attention là một ví dụ điển hình về cách các nhà nghiên cứu đang tìm cách cân bằng giữa hiệu suất mô hình và hiệu quả tính toán. Mặc dù đơn giản, nhưng nó đã trở thành một công cụ quan trọng trong bộ công cụ của các nhà phát triển mô hình

ngôn ngữ lớn, đặc biệt là khi triển khai các mô hình này trong môi trường có tài nguyên hạn chế hoặc khi cần xử lý các chuỗi đầu vào dài.

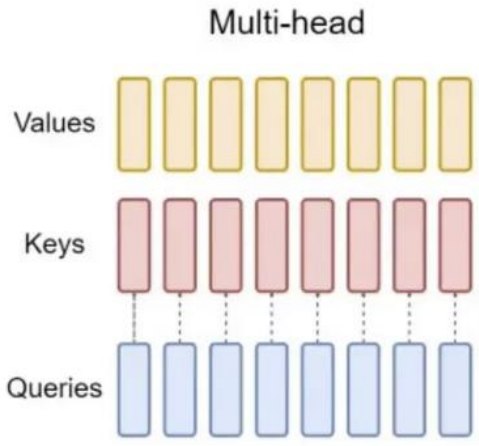
1.4 Grouped-Query Attention (GQA)

Grouped-Query Attention (GQA) là một cơ chế attention được phát triển như một sự cân bằng giữa Multi-Head Attention (MHA) truyền thống và Multi-Query Attention (MQA). Được giới thiệu bởi Ainslie và cộng sự trong bài báo "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints" vào năm 2023, GQA nhằm mục đích kết hợp ưu điểm của cả hai phương pháp: độ chính xác cao của MHA và hiệu quả bộ nhớ của MQA.

1.4.1 Nguyên lý hoạt động và mối quan hệ với MHA và MQA

Để hiểu rõ GQA, chúng ta cần nhìn lại cả MHA và MQA:

1. **Multi-Head Attention (MHA)**: Mỗi đầu attention có bộ ma trận trọng số riêng biệt cho Query (Q), Key (K) và Value (V). Với H đầu attention, có tổng cộng 3H ma trận trọng số.



Hình 1.3 Nguyên lý hoạt động của Multi-Head Attention (MHA)

2. **Multi-Query Attention (MQA)**: Mỗi đầu attention có ma trận trọng số Query (Q) riêng biệt, nhưng tất cả các đầu dùng chung một ma trận Key (K) và một ma trận Value (V). Với H đầu attention, có tổng cộng H+2 ma trận trọng số.

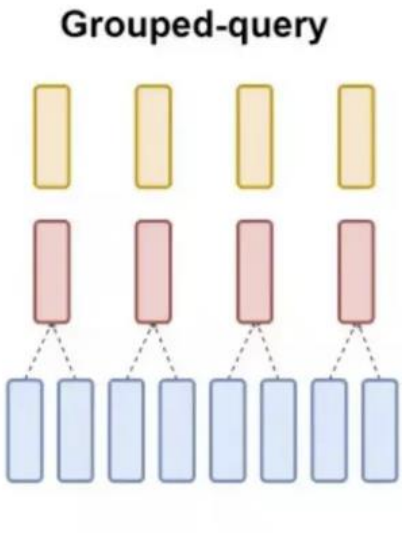
Grouped-Query Attention đưa ra một cách tiếp cận trung gian: các đầu attention được chia thành G nhóm (với G < H), và mỗi nhóm dùng chung một ma trận Key và một ma trận Value. Cụ thể:

- Mỗi đầu attention vẫn có ma trận Query (Q) riêng biệt
- Các đầu attention trong cùng một nhóm dùng chung ma trận Key (K) và Value (V)
- Với H đầu attention và G nhóm, có tổng cộng H+2G ma trận trọng số

Như vậy, GQA có thể được xem là một trường hợp tổng quát hóa của cả MHA và MQA:

- Khi G = H (số nhóm bằng số đầu), GQA trở thành MHA
- Khi G = 1 (tất cả các đầu trong một nhóm), GQA trở thành MQA

Điều này cho phép các nhà nghiên cứu và kỹ sư linh hoạt điều chỉnh sự cân bằng giữa hiệu suất mô hình và hiệu quả tính toán bằng cách chọn số lượng nhóm phù hợp.



Hình 1.4 Nguyên lý hoạt động của Grouped Query Attention (GQA)

1.4.2 Công thức toán học và biểu diễn ma trận

Về mặt toán học, Grouped-Query Attention được tính toán như sau:

1. Tạo ra các vector Query cho mỗi đầu attention:

- $Q_i = X \times W^{Q_i}$ ($với\ i$ từ 1 đến H)

2. Tạo ra các vector Key và Value cho mỗi nhóm:
   - $K_g = X \times W^{K_g}$ ($với\ g$ từ 1 đến G)
   - $V_g = X \times W^{V_g}$ ($với\ g$ từ 1 đến G)
3. Tính toán điểm số attention cho mỗi đầu, sử dụng Key của nhóm tương ứng:
   - $Scores_i = \frac{(Q_i \times K_g{}^T)}{\sqrt{d_k}}$ ($với\ g$ là $nhóm\ chứa$ đầu i)
4. Áp dụng hàm softmax để chuyển điểm số thành trọng số attention:
   - $Attention\_weights_i = softmax(Scores_i)$
5. Nhân trọng số attention với vector Value của nhóm tương ứng:
   - $Output_i = Attention\_weights_i \times V_g$ ($với\ g$ là $nhóm\ chứa$ đầu i)
6. Nối (concatenate) đầu ra từ tất cả các đầu và truyền qua một lớp tuyến tính:
   - $Output = Concat(Output_1, \ldots, Output_H) \times W^O$

Công thức tổng quát của GQA có thể được biểu diễn như sau:

$$GQA(X) = Concat\left(\frac{softmax\left(Q_1 \times K_{g1}{}^T\right)}{\left(\sqrt{d_k}\right)}\right) \times V_{g1}, \ldots,$$

$$\frac{softmax\left(Q_H \times K_{gH}{}^T\right)}{\left(\sqrt{d_k}\right)} \times V_{gH} \times W^O$$

Trong đó g1, ..., gH là các nhóm tương ứng với mỗi đầu attention.

1.4.3 Độ phức tạp tính toán và hiệu quả bộ nhớ

GQA cung cấp một sự cân bằng giữa hiệu suất mô hình và hiệu quả tính toán:

1. **Yêu cầu bộ nhớ**: Trong quá trình suy luận tự hồi quy, GQA yêu cầu lưu trữ $2 \times G \times n \times d_k$ phần tử (cho các ma trận Key và Value), so với $2 \times H \times n \times d_k$ của MHA và $2 \times n \times d_k$ của MQA. Với G < H, GQA tiết kiệm bộ nhớ đáng kể so với MHA.

2. **Băng thông bộ nhớ**: GQA giảm lượng dữ liệu cần truyền giữa bộ nhớ và đơn vị xử lý, đặc biệt quan trọng trên các thiết bị có băng thông bộ nhớ hạn chế như GPU.
3. **Độ phức tạp tính toán**: Về mặt lý thuyết, độ phức tạp tính toán vẫn là $O(n^2)$ như self-attention truyền thống, nhưng hằng số nhân nhỏ hơn do giảm số lượng phép nhân ma trận.

Ví dụ, với một mô hình có 32 đầu attention, nếu sử dụng 8 nhóm trong GQA, yêu cầu bộ nhớ sẽ giảm 4 lần so với MHA truyền thống, nhưng vẫn cao hơn 8 lần so với MQA.

1.4.4 Ưu điểm và hạn chế

GQA có nhiều ưu điểm nổi bật:

1. **Cân bằng giữa hiệu suất và hiệu quả**: GQA cung cấp một sự cân bằng linh hoạt giữa chất lượng mô hình (như MHA) và hiệu quả bộ nhớ/tính toán (như MQA).
2. **Khả năng mở rộng**: Với các mô hình lớn hơn, số lượng đầu attention thường tăng lên, nhưng số lượng nhóm có thể được giữ cố định hoặc tăng với tốc độ chậm hơn, giúp cải thiện hiệu quả mà không làm giảm đáng kể chất lượng.
3. **Dễ triển khai**: Tương tự như MQA, GQA có thể dễ dàng áp dụng cho các kiến trúc transformer hiện có.
4. **Khả năng chuyển đổi từ mô hình đã huấn luyện**: Các mô hình đã được huấn luyện với MHA có thể được chuyển đổi sang GQA thông qua việc trung bình hóa các ma trận Key và Value trong mỗi nhóm.

Tuy nhiên, GQA cũng có một số hạn chế:

1. **Phức tạp hơn MQA**: GQA phức tạp hơn MQA về mặt triển khai và có yêu cầu bộ nhớ cao hơn.
2. **Vẫn có sự đánh đổi**: Mặc dù GQA cung cấp sự cân bằng tốt hơn, nhưng vẫn có sự đánh đổi giữa chất lượng mô hình và hiệu quả tính toán.

3. **Cần lựa chọn số lượng nhóm phù hợp**: Hiệu suất của GQA phụ thuộc vào việc lựa chọn số lượng nhóm phù hợp, đòi hỏi thử nghiệm và điều chỉnh.

1.4.5 So sánh với self-attention gốc

So với self-attention truyền thống (MHA), GQA có những điểm khác biệt quan trọng:

1. **Hiệu quả bộ nhớ**: GQA yêu cầu ít bộ nhớ hơn đáng kể trong quá trình suy luận, đặc biệt là với các chuỗi dài.
2. **Tốc độ suy luận**: GQA thường có tốc độ suy luận nhanh hơn do giảm số lượng phép tính và truy cập bộ nhớ.
3. **Chất lượng mô hình**: GQA có thể duy trì chất lượng mô hình gần với MHA, đặc biệt khi số lượng nhóm được chọn phù hợp.
4. **Khả năng mở rộng**: GQA có khả năng mở rộng tốt hơn với các mô hình lớn và chuỗi dài.

1.4.6 Ứng dụng trong các mô hình LLM hiện đại như LLaMA-2, Mistral7B

GQA đã được áp dụng thành công trong nhiều mô hình ngôn ngữ lớn hiện đại:

1. **LLaMA-2**: Meta đã sử dụng GQA trong mô hình LLaMA-2 để cải thiện hiệu quả suy luận mà không làm giảm đáng kể chất lượng mô hình. Điều này đã giúp LLaMA-2 có thể xử lý các chuỗi dài hơn với hiệu quả cao hơn.
2. **Mistral7B**: Mô hình Mistral7B cũng áp dụng GQA như một phần của chiến lược tối ưu hóa, giúp mô hình đạt được sự cân bằng tốt giữa hiệu suất và hiệu quả.
3. **Claude**: Anthropic đã sử dụng GQA trong mô hình Claude để cải thiện hiệu quả bộ nhớ và tốc độ suy luận, đặc biệt là khi xử lý các chuỗi dài.
4. **Gemma**: Google đã áp dụng GQA trong mô hình Gemma để tối ưu hóa hiệu suất trên các thiết bị có tài nguyên hạn chế.

Các mô hình này đã chứng minh rằng GQA có thể được áp dụng hiệu quả trong các mô hình quy mô lớn, cung cấp một sự cân bằng tốt giữa chất lượng mô hình và hiệu quả tính toán. Đặc biệt, GQA đã trở thành một lựa chọn phổ biến cho các mô hình mã nguồn mở, nơi hiệu quả triển khai trên nhiều loại phần cứng khác nhau là một yếu tố quan trọng.

Grouped-Query Attention là một ví dụ điển hình về cách các nhà nghiên cứu đang tìm cách cải thiện hiệu quả của các mô hình ngôn ngữ lớn mà không làm giảm đáng kể chất lượng. Bằng cách cung cấp một phương pháp tổng quát hóa của cả MHA và MQA, GQA cho phép các nhà phát triển linh hoạt điều chỉnh sự cân bằng giữa hiệu suất và hiệu quả dựa trên yêu cầu cụ thể của ứng dụng và tài nguyên có sẵn.

1.5 FlashAttention và FlashAttention v2

FlashAttention là một cơ chế tối ưu hóa đột phá cho phép tính toán attention nhanh hơn và hiệu quả hơn về mặt bộ nhớ, được giới thiệu bởi Dao và cộng sự trong bài báo "FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness" vào năm 2022. Sau đó, nhóm nghiên cứu đã phát triển phiên bản cải tiến FlashAttention-2 vào năm 2023, với nhiều cải tiến về hiệu suất. Khác với các phương pháp khác như MQA, GQA, Linear Attention hay Sparse Attention - vốn thay đổi công thức attention hoặc cấu trúc mô hình - FlashAttention tập trung vào việc tối ưu hóa cách thức tính toán attention trên phần cứng GPU mà không thay đổi kết quả cuối cùng.

1.5.1 Nguyên lý hoạt động và cải tiến về IO-Awareness

Ý tưởng cốt lõi của FlashAttention dựa trên nhận thức về "IO-Awareness" (nhận thức về đầu vào/đầu ra), tức là hiểu và tối ưu hóa cách dữ liệu di chuyển giữa các cấp độ bộ nhớ khác nhau trong GPU. Trong kiến trúc GPU hiện đại, có ba cấp độ bộ nhớ chính:

1. **SRAM (Static Random-Access Memory)**: Bộ nhớ trên chip, có dung lượng nhỏ (khoảng 20MB trên GPU A100) nhưng băng thông cực cao (khoảng 19TB/s).
2. **HBM (High Bandwidth Memory)**: Bộ nhớ ngoài chip nhưng trên card, có dung lượng lớn hơn (40-80GB) nhưng băng thông thấp hơn (khoảng 1.5-2TB/s).
3. **DRAM (Dynamic Random-Access Memory)**: Bộ nhớ RAM của hệ thống, có dung lượng lớn nhưng băng thông thấp nhất (khoảng 12.8GB/s).

Trong cài đặt attention truyền thống, các ma trận Query, Key, Value và kết quả trung gian đều được lưu trữ trong HBM, dẫn đến nhiều lần đọc/ghi dữ liệu giữa HBM và

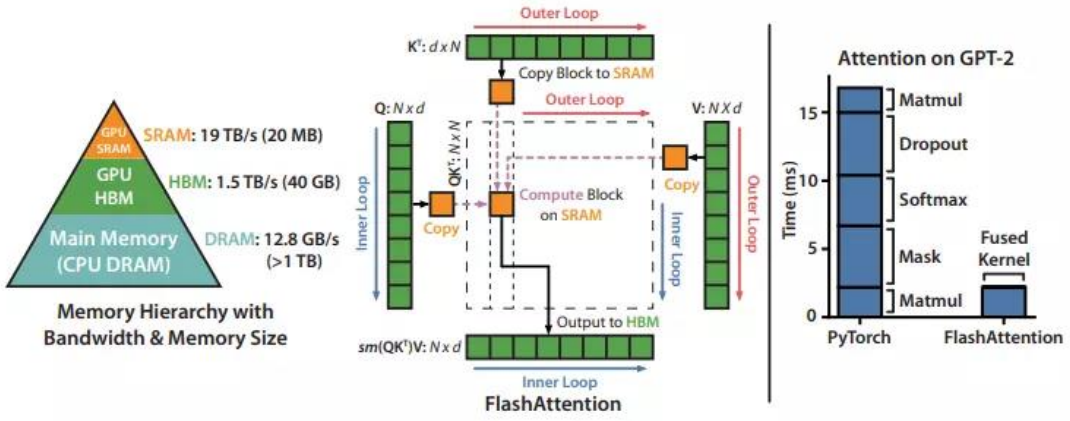
SRAM. Điều này tạo ra một "nút thắt cổ chai" về băng thông bộ nhớ, làm chậm quá trình tính toán đáng kể.

FlashAttention giải quyết vấn đề này bằng cách:

1. **Tiled Attention**: Chia nhỏ ma trận attention thành các "ô" (tiles) có kích thước vừa với SRAM.
2. **Kernel Fusion**: Kết hợp nhiều phép tính (nhân ma trận, softmax, v.v.) trong một kernel GPU duy nhất để giảm thiểu việc đọc/ghi dữ liệu từ HBM.
3. **Recomputation**: Tính toán lại một số kết quả trung gian thay vì lưu trữ chúng, đánh đổi tính toán để tiết kiệm bộ nhớ.

FlashAttention-2 cải tiến hơn nữa bằng cách:

1. **Cải thiện phân chia công việc**: Phân phối công việc tốt hơn giữa các thread block và warp trên GPU.
2. **Giảm số lượng phép tính không phải nhân ma trận**: Tối ưu hóa các phép tính phụ trợ.
3. **Song song hóa tính toán attention**: Cho phép tính toán song song ngay cả cho một đầu attention duy nhất.



Hình 1.5 Nguyên lý hoạt động của Flash Attetion

1.5.2 Công thức toán học và tối ưu hóa tính toán

Về mặt toán học, FlashAttention vẫn sử dụng công thức attention chuẩn:

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

Tuy nhiên, cách tính toán được tổ chức lại để tối ưu hóa hiệu suất. Cụ thể, FlashAttention chia ma trận Q, K, V thành các block nhỏ và tính toán attention theo từng block:

1. Chia Q thành các block Q_i có kích thước $B_r \times d$
2. Chia K và V thành các block K_j và V_j có kích thước $B_c \times d$
3. Tính toán attention cho từng cặp block (Q_i, K_j, V_j)

Đối với mỗi block, FlashAttention tính toán:

- $S_{ij} = Q_i \times \frac{K_j^T}{\sqrt{d_k}}$ (ma trận điểm số)
- $P_{ij} = softmax(S_{ij})$ (ma trận trọng số attention)
- $O_{ij} = P_{ij} \times V_j$ (đầu ra của block)

Để tính toán softmax chính xác trên toàn bộ chuỗi, FlashAttention sử dụng kỹ thuật "online softmax" với các biến tích lũy:

- m_i: giá trị lớn nhất của hàng i
- l_i: tổng của $exp(s_{ij} - m_i)$ cho hàng i
- o_i: đầu ra tích lũy cho hàng i

Các biến này được cập nhật khi xử lý từng block, cho phép tính toán softmax chính xác mà không cần lưu trữ toàn bộ ma trận điểm số.

FlashAttention-2 cải tiến hơn nữa bằng cách tối ưu hóa cách phân chia công việc giữa các thread block và warp, giảm thiểu giao tiếp qua shared memory và tăng cường song song hóa.

1.5.3 Độ phức tạp tính toán

Một trong những đóng góp quan trọng nhất của FlashAttention là giảm đáng kể yêu cầu bộ nhớ:

1. **Độ phức tạp bộ nhớ**: FlashAttention giảm độ phức tạp bộ nhớ từ $O(n^2)$ xuống $O(n)$, cho phép xử lý các chuỗi dài hơn nhiều với cùng một lượng bộ nhớ GPU.
2. **Độ phức tạp tính toán**: Về mặt lý thuyết, độ phức tạp tính toán vẫn là $O(n^2)$ như attention truyền thống, nhưng FlashAttention thực hiện các phép tính này hiệu quả hơn nhiều.

Cụ thể, với chuỗi độ dài n và kích thước mô hình d:

- Attention truyền thống: $O(n^2d)$ phép tính và $O(n^2)$ bộ nhớ
- FlashAttention: $O(n^2d)$ phép tính nhưng chỉ $O(nd)$ bộ nhớ

Trong thực tế, FlashAttention có thể nhanh hơn 2-4 lần so với các cài đặt attention tối ưu khác, và FlashAttention-2 nhanh hơn khoảng 2 lần so với FlashAttention ban đầu.

1.5.4 Ưu điểm và hạn chế

FlashAttention có nhiều ưu điểm nổi bật:

1. **Tính toán chính xác**: Không giống như nhiều phương pháp khác, FlashAttention tính toán attention chính xác, không có xấp xỉ, đảm bảo chất lượng mô hình không bị ảnh hưởng.
2. **Hiệu quả bộ nhớ cao**: Giảm đáng kể yêu cầu bộ nhớ, cho phép xử lý các chuỗi dài hơn hoặc tăng kích thước batch.
3. **Tốc độ nhanh hơn**: Tăng tốc đáng kể cả quá trình huấn luyện và suy luận.
4. **Dễ tích hợp**: Có thể dễ dàng tích hợp vào các kiến trúc transformer hiện có mà không cần thay đổi kiến trúc mô hình.
5. **Khả năng mở rộng tốt**: Hiệu suất cải thiện càng rõ rệt với các mô hình lớn và chuỗi dài.

Tuy nhiên, FlashAttention cũng có một số hạn chế:

1. **Phụ thuộc vào phần cứng**: Hiệu suất phụ thuộc vào kiến trúc GPU cụ thể và yêu cầu CUDA, hạn chế khả năng triển khai trên các nền tảng khác.
2. **Phức tạp trong triển khai**: Cài đặt FlashAttention đòi hỏi hiểu biết sâu về lập trình GPU và tối ưu hóa kernel.

3. **Vẫn có độ phức tạp tính toán $O(n^2)$**: Mặc dù nhanh hơn, nhưng vẫn có giới hạn khi xử lý các chuỗi cực dài (hàng trăm nghìn token).
4. **Không giải quyết vấn đề cơ bản của attention**: FlashAttention tối ưu hóa cách tính toán attention, nhưng không thay đổi bản chất của cơ chế attention.

1.5.5 Tác động đến hiệu suất huấn luyện và suy luận của mô hình

FlashAttention đã có tác động đáng kể đến cả quá trình huấn luyện và suy luận của các mô hình ngôn ngữ lớn:

1. **Huấn luyện nhanh hơn**: FlashAttention cho phép huấn luyện nhanh hơn 2-4 lần, giúp tiết kiệm thời gian và chi phí đáng kể. FlashAttention-2 còn nhanh hơn nữa, đạt tới 225 TFLOPs/s trên GPU A100 (72% hiệu suất lý thuyết tối đa).
2. **Xử lý chuỗi dài hơn**: Với yêu cầu bộ nhớ giảm từ $O(n^2)$ xuống $O(n)$, FlashAttention cho phép xử lý các chuỗi dài hơn nhiều. Điều này đặc biệt quan trọng cho các ứng dụng như xử lý tài liệu dài, mã nguồn, hoặc dữ liệu đa phương tiện.
3. **Mở rộng kích thước mô hình**: Hiệu quả bộ nhớ cao hơn cho phép huấn luyện các mô hình lớn hơn với cùng một lượng tài nguyên.
4. **Cải thiện chất lượng mô hình**: Khả năng xử lý chuỗi dài hơn và kích thước batch lớn hơn có thể dẫn đến cải thiện chất lượng mô hình.

FlashAttention đã được áp dụng rộng rãi trong nhiều mô hình ngôn ngữ lớn hiện đại, bao gồm:

1. **LLaMA và LLaMA-2**: Meta đã sử dụng FlashAttention để tăng tốc quá trình huấn luyện và cho phép xử lý chuỗi dài hơn.
2. **MPT**: MosaicML đã tích hợp FlashAttention vào mô hình MPT để cải thiện hiệu suất.
3. **Falcon**: Technology Innovation Institute đã sử dụng FlashAttention trong mô hình Falcon.

4. **RedPajama**: Dự án RedPajama đã áp dụng FlashAttention để tăng tốc quá trình huấn luyện.
5. **Mistral**: Mô hình Mistral cũng sử dụng FlashAttention để cải thiện hiệu suất.

Ngoài ra, FlashAttention đã được tích hợp vào nhiều thư viện phổ biến như PyTorch, JAX, và các framework huấn luyện mô hình ngôn ngữ lớn như Megatron-LM và DeepSpeed, giúp cộng đồng nghiên cứu và phát triển dễ dàng tiếp cận và sử dụng.

FlashAttention là một ví dụ điển hình về cách tối ưu hóa phần cứng có thể mang lại những cải tiến đáng kể cho hiệu suất mô hình mà không cần thay đổi kiến trúc mô hình. Bằng cách hiểu và tối ưu hóa cách dữ liệu di chuyển giữa các cấp độ bộ nhớ, FlashAttention đã mở ra khả năng xử lý các chuỗi dài hơn và huấn luyện các mô hình lớn hơn, góp phần vào sự phát triển nhanh chóng của các mô hình ngôn ngữ lớn trong những năm gần đây.

1.6 Linear Attention

Linear Attention là một biến thể quan trọng của cơ chế attention truyền thống, được thiết kế để giải quyết vấn đề độ phức tạp bậc hai ($O(n^2)$) của self-attention thông thường. Được giới thiệu trong bài báo "Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention" của Katharopoulos và cộng sự vào năm 2020, Linear Attention đã mở ra một hướng tiếp cận mới để xử lý các chuỗi cực dài với độ phức tạp tuyến tính ($O(n)$) về cả thời gian và bộ nhớ.

1.6.1 Nguyên lý hoạt động và phương pháp xấp xỉ hàm tương đồng

Ý tưởng cốt lõi của Linear Attention dựa trên việc biểu diễn lại công thức attention thông qua việc sử dụng các hàm ánh xạ đặc trưng (feature maps) và tận dụng tính chất kết hợp của phép nhân ma trận.

Trong self-attention truyền thống, công thức tính toán attention có dạng:

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

Trong đó, phép nhân QK^T tạo ra ma trận attention có kích thước n×n, dẫn đến độ phức tạp $O(n^2)$.

Linear Attention biểu diễn lại công thức này bằng cách sử dụng một hàm ánh xạ đặc trưng φ(x) thay cho hàm softmax:

$$Attention(Q, K, V) = \varphi(Q) \times \varphi(K)^T \times V$$

Bằng cách áp dụng tính chất kết hợp của phép nhân ma trận, chúng ta có thể tính toán theo thứ tự khác:

$$Attention(Q, K, V) = \varphi(Q) \times (\varphi(K)^T \times V)$$

Với cách tính này, thay vì tạo ra ma trận n×n, chúng ta chỉ cần tính $(\varphi(K)^T \times V)$ trước (có kích thước d×d), sau đó nhân với φ(Q). Điều này giảm độ phức tạp từ $O(n^2)$ xuống O(n).

Một trong những thách thức chính là tìm ra hàm ánh xạ φ(x) phù hợp để xấp xỉ hàm softmax. Các nhà nghiên cứu đã đề xuất nhiều lựa chọn khác nhau, trong đó phổ biến nhất là:

1. **ELU+1**: $\varphi(x) = ELU(x) + 1$, trong đó ELU (Exponential Linear Unit) là hàm kích hoạt phi tuyến.
2. **ReLU**: $\varphi(x) = max(0, x)$
3. **Hàm mũ**: $\varphi(x) = exp(x)$

Hàm ELU+1 được ưa chuộng vì nó có một số tính chất tương tự như hàm mũ (luôn dương, bảo toàn thứ tự tương đối) nhưng tính toán hiệu quả hơn.

1.6.2 Công thức toán học và biểu diễn ma trận

Về mặt toán học, Linear Attention được tính toán như sau:

1. Áp dụng hàm ánh xạ đặc trưng φ cho các ma trận Query và Key:

$$\varphi(Q) = \varphi(X \times W^Q)$$
$$\varphi(K) = \varphi(X \times W^K)$$

2. Tính toán ma trận trung gian KV:

$$KV = \varphi(K)^T \times V$$

3. Tính toán đầu ra cuối cùng:

$$Output = \varphi(Q) \times KV$$

4. Chuẩn hóa đầu ra (tùy chọn):

$$Z = 1 / (\varphi(Q) \times \varphi(K)^T \times 1)$$

$$Output\_normalized = Output \odot Z$$

Trong đó $\odot$ là phép nhân Hadamard (nhân từng phần tử).

Đối với attention tự hồi quy (autoregressive attention), Linear Attention có thể được cài đặt một cách hiệu quả bằng cách duy trì hai ma trận trạng thái S và Z:

- $S_t = S_{t-1} + \varphi(k_t) \times v_t^T$
- $Z_t = Z_{t-1} + \varphi(k_t)$

Và đầu ra tại mỗi vị trí t được tính như sau:

- $o_t = \varphi(q_t)^T \times S_t / (\varphi(q_t)^T \times Z_t)$

Cách tiếp cận này cho phép tính toán attention một cách lặp lại, tương tự như trong RNN, dẫn đến tên gọi "Transformers are RNNs" trong bài báo gốc.

1.6.3 Độ phức tạp tính toán

Một trong những đóng góp quan trọng nhất của Linear Attention là giảm đáng kể độ phức tạp tính toán và bộ nhớ:

1. **Độ phức tạp thời gian**: Linear Attention giảm độ phức tạp từ $O(n^2d)$ xuống $O(nd^2)$, trong đó n là độ dài chuỗi và d là kích thước mô hình. Khi $n >> d$ (thường gặp trong các chuỗi dài), đây là một cải thiện đáng kể.
2. **Độ phức tạp bộ nhớ**: Tương tự, độ phức tạp bộ nhớ giảm từ $O(n^2)$ xuống $O(n)$.

Điều này có ý nghĩa đặc biệt quan trọng khi xử lý các chuỗi cực dài. Ví dụ, với chuỗi độ dài 100,000 token, self-attention truyền thống sẽ yêu cầu lưu trữ ma trận attention có 10 tỷ phần tử, trong khi Linear Attention chỉ cần lưu trữ các ma trận có kích thước tuyến tính với độ dài chuỗi.

Trong thực tế, Linear Attention có thể nhanh hơn đến 4000 lần so với self-attention truyền thống khi xử lý các chuỗi cực dài trong quá trình suy luận tự hồi quy.

1.6.4 Ưu điểm và hạn chế

Linear Attention có nhiều ưu điểm nổi bật:

1. **Độ phức tạp tuyến tính**: Cho phép xử lý các chuỗi cực dài mà self-attention truyền thống không thể xử lý được.
2. **Hiệu quả bộ nhớ cao**: Giảm đáng kể yêu cầu bộ nhớ, đặc biệt quan trọng trên các thiết bị có tài nguyên hạn chế.
3. **Tốc độ suy luận nhanh**: Đặc biệt hiệu quả trong quá trình suy luận tự hồi quy, có thể nhanh hơn nhiều lần so với self-attention truyền thống.
4. **Khả năng mở rộng tốt**: Hiệu suất cải thiện càng rõ rệt với các chuỗi càng dài.
5. **Cài đặt lặp lại**: Có thể được cài đặt theo cách lặp lại, tương tự như RNN, giúp tiết kiệm bộ nhớ hơn nữa.

Tuy nhiên, Linear Attention cũng có một số hạn chế đáng kể:

1. **Giảm chất lượng mô hình**: Việc xấp xỉ hàm softmax bằng các hàm ánh xạ đặc trưng có thể dẫn đến giảm chất lượng mô hình trong một số tác vụ.
2. **Khó khăn trong huấn luyện**: Các mô hình sử dụng Linear Attention thường khó huấn luyện hơn và có thể đòi hỏi các kỹ thuật ổn định đặc biệt.
3. **Không phù hợp với mọi tác vụ**: Hiệu quả của Linear Attention có thể thay đổi đáng kể tùy thuộc vào tác vụ cụ thể.
4. **Hạn chế trong biểu diễn**: Một số mối quan hệ phức tạp có thể không được nắm bắt tốt như trong self-attention truyền thống.

1.6.5 So sánh hiệu suất với self-attention truyền thống

So sánh giữa Linear Attention và self-attention truyền thống cho thấy những điểm khác biệt quan trọng:

1. **Tốc độ và hiệu quả bộ nhớ**: Linear Attention vượt trội về tốc độ và hiệu quả bộ nhớ, đặc biệt với các chuỗi dài. Các thử nghiệm cho thấy tốc độ có thể nhanh hơn đến 4000 lần trong một số trường hợp.

2. **Chất lượng mô hình**: Self-attention truyền thống thường cho kết quả tốt hơn về chất lượng mô hình, đặc biệt trong các tác vụ đòi hỏi nắm bắt các mối quan hệ phức tạp.
3. **Khả năng mở rộng**: Linear Attention có khả năng mở rộng tốt hơn nhiều với độ dài chuỗi, trong khi self-attention truyền thống gặp giới hạn nghiêm trọng.
4. **Tính linh hoạt**: Self-attention truyền thống linh hoạt hơn và có thể áp dụng cho nhiều loại tác vụ khác nhau, trong khi Linear Attention có thể không hiệu quả cho một số tác vụ cụ thể.

Các thử nghiệm thực tế cho thấy Linear Attention đặc biệt hiệu quả trong các tác vụ như mô hình hóa ngôn ngữ tự hồi quy, tổng hợp âm thanh, và tạo sinh hình ảnh - những tác vụ đòi hỏi xử lý các chuỗi dài và có tính tuần tự cao.

Linear Attention đã được áp dụng trong nhiều mô hình và ứng dụng, bao gồm:

1. **Mô hình ngôn ngữ tự hồi quy**: Cho phép xử lý và tạo sinh các văn bản dài hơn nhiều so với các mô hình sử dụng self-attention truyền thống.
2. **Mô hình tạo sinh hình ảnh**: Được sử dụng trong các mô hình như DALL-E và Imagen để xử lý các chuỗi token hình ảnh dài.
3. **Xử lý âm thanh và video**: Cho phép xử lý các chuỗi dữ liệu đa phương tiện dài mà không gặp giới hạn về bộ nhớ.
4. **Mô hình hóa chuỗi thời gian**: Hiệu quả trong việc xử lý các chuỗi thời gian dài trong các ứng dụng như dự báo tài chính và phân tích dữ liệu cảm biến.

Linear Attention là một bước tiến quan trọng trong việc mở rộng khả năng của các mô hình transformer để xử lý các chuỗi cực dài. Mặc dù có những hạn chế về chất lượng mô hình, nhưng lợi ích về hiệu suất và khả năng mở rộng đã khiến nó trở thành một công cụ quan trọng trong bộ công cụ của các nhà nghiên cứu và phát triển mô hình ngôn ngữ lớn, đặc biệt là khi làm việc với các chuỗi dài hoặc trên các thiết bị có tài nguyên hạn chế.

1.7 Sparse Attention

Sparse Attention là một biến thể của cơ chế attention truyền thống, được thiết kế để giải quyết vấn đề độ phức tạp bậc hai ($O(n^2)$) của self-attention thông thường bằng cách chỉ tính toán attention cho một tập con các cặp token thay vì tất cả các cặp có thể. Được giới thiệu trong bài báo "Generating Long Sequences with Sparse Transformers" của Child và cộng sự tại OpenAI vào năm 2019, Sparse Attention đã mở ra khả năng xử lý các chuỗi cực dài với độ phức tạp thấp hơn đáng kể.

1.7.1 Nguyên lý hoạt động và các mẫu attention thưa

Ý tưởng cốt lõi của Sparse Attention dựa trên nhận xét rằng trong nhiều trường hợp, không phải tất cả các token đều cần chú ý đến tất cả các token khác. Thay vào đó, mỗi token chỉ cần chú ý đến một tập con các token quan trọng, chẳng hạn như các token lân cận (local attention) hoặc các token ở vị trí đặc biệt (global attention).

Sparse Transformer đề xuất hai mẫu attention thưa chính:

1. **Mẫu dải băng cục bộ (Local banded pattern)**: Mỗi token chỉ chú ý đến một cửa sổ cố định các token lân cận. Điều này dựa trên giả định rằng các token gần nhau thường có mối quan hệ mạnh mẽ hơn.
2. **Mẫu nhảy cách đều (Strided pattern)**: Mỗi token chú ý đến các token cách đều nhau trong chuỗi, cho phép nắm bắt các mối quan hệ dài hạn mà không cần tính toán attention cho tất cả các cặp.

Trong Sparse Transformer, các mẫu attention này được sắp xếp xen kẽ qua các lớp, với một lớp sử dụng mẫu dải băng cục bộ và lớp tiếp theo sử dụng mẫu nhảy cách đều. Điều này cho phép mô hình nắm bắt cả mối quan hệ cục bộ và mối quan hệ dài hạn mà không cần tính toán ma trận attention đầy đủ.

Ngoài ra, còn có nhiều biến thể khác của Sparse Attention đã được đề xuất:

1. **Block Sparse Attention**: Chia ma trận attention thành các khối và chỉ tính toán attention cho một tập con các khối.
2. **Longformer Attention**: Kết hợp attention cục bộ với một số token toàn cục (như token [CLS]) có thể chú ý đến tất cả các token khác.

3. **Big Bird**: Kết hợp ba loại attention: cục bộ, ngẫu nhiên và toàn cục, cho phép nắm bắt các mối quan hệ phức tạp với độ phức tạp tuyến tính.

1.7.2 Công thức toán học và biểu diễn ma trận

Về mặt toán học, Sparse Attention có thể được biểu diễn bằng cách sử dụng một ma trận mặt nạ thưa (sparse mask) M để xác định các cặp token nào được phép tương tác:

$$Attention(Q,K,V) = softmax(QK^T \odot \frac{M}{\sqrt{d_k}}) \times V$$

Trong đó $\odot$ là phép nhân Hadamard (nhân từng phần tử), và M là ma trận nhị phân với $M_{ij} = 1$ nếu token i được phép chú ý đến token j, và $M_{ij} = 0$ nếu không.

Đối với mẫu dải băng cục bộ với kích thước cửa sổ w, ma trận M có dạng: $M_{ij} = 1$ nếu $|i - j| \leq \frac{w}{2}$, và $M_{ij} = 0$ nếu ngược lại

Đối với mẫu nhảy cách đều với bước nhảy s, ma trận M có dạng: $M_{ij} = 1$ nếu (i - j) mod s = 0, và $M_{ij} = 0$ nếu ngược lại

Trong thực tế, việc cài đặt Sparse Attention thường không sử dụng ma trận mặt nạ rõ ràng, mà thay vào đó sử dụng các thuật toán hiệu quả để chỉ tính toán các phần tử attention cần thiết. Điều này có thể được thực hiện thông qua các phép toán ma trận thưa hoặc các kỹ thuật lập trình động.

1.7.3 Độ phức tạp tính toán

Một trong những đóng góp quan trọng nhất của Sparse Attention là giảm đáng kể độ phức tạp tính toán và bộ nhớ:

1. **Độ phức tạp thời gian**: Sparse Attention giảm độ phức tạp từ $O(n^2d)$ xuống $O(n\sqrt{n} \cdot d)$ hoặc thậm chí $O(n \cdot log(n) \cdot d)$ tùy thuộc vào mẫu attention cụ thể, trong đó n là độ dài chuỗi và d là kích thước mô hình.
2. **Độ phức tạp bộ nhớ**: Tương tự, độ phức tạp bộ nhớ giảm từ $O(n^2)$ xuống $O(n\sqrt{n})$ hoặc $O(n \cdot log(n))$.

Điều này có ý nghĩa đặc biệt quan trọng khi xử lý các chuỗi dài. Ví dụ, với chuỗi độ dài 10,000 token, self-attention truyền thống sẽ yêu cầu tính toán và lưu trữ 100 triệu phần tử, trong khi Sparse Attention với độ phức tạp O(n√n) chỉ cần khoảng 1 triệu phần tử.

Trong thực tế, Sparse Transformer đã được sử dụng để xử lý các chuỗi có độ dài lên đến 16,384 token, vượt xa giới hạn của các mô hình transformer truyền thống tại thời điểm đó.

1.7.4 Ưu điểm và hạn chế

Sparse Attention có nhiều ưu điểm nổi bật:

1. **Độ phức tạp thấp hơn**: Cho phép xử lý các chuỗi dài hơn nhiều so với self-attention truyền thống.
2. **Hiệu quả bộ nhớ cao**: Giảm đáng kể yêu cầu bộ nhớ, đặc biệt quan trọng khi xử lý các chuỗi dài.
3. **Khả năng mở rộng tốt**: Hiệu suất cải thiện càng rõ rệt với các chuỗi càng dài.
4. **Nắm bắt cả mối quan hệ cục bộ và dài hạn**: Thông qua việc kết hợp các mẫu attention khác nhau, Sparse Attention có thể nắm bắt cả mối quan hệ cục bộ và dài hạn.
5. **Linh hoạt trong thiết kế mẫu**: Có thể thiết kế các mẫu attention khác nhau phù hợp với đặc điểm của dữ liệu và tác vụ cụ thể.

Tuy nhiên, Sparse Attention cũng có một số hạn chế đáng kể:

1. **Có thể bỏ lỡ mối quan hệ quan trọng**: Việc chỉ tính toán attention cho một tập con các cặp token có thể dẫn đến việc bỏ lỡ một số mối quan hệ quan trọng.
2. **Khó khăn trong cài đặt hiệu quả**: Cài đặt Sparse Attention hiệu quả trên phần cứng hiện đại như GPU và TPU có thể phức tạp hơn so với self-attention truyền thống.
3. **Phụ thuộc vào mẫu attention**: Hiệu suất của Sparse Attention phụ thuộc nhiều vào việc lựa chọn mẫu attention phù hợp với dữ liệu và tác vụ.

4. **Khó khăn trong huấn luyện**: Các mô hình sử dụng Sparse Attention có thể đòi hỏi các kỹ thuật huấn luyện đặc biệt để đạt được hiệu suất tốt.

1.7.5 Ứng dụng trong mô hình hóa chuỗi dài

Sparse Attention đã được áp dụng thành công trong nhiều mô hình và ứng dụng, đặc biệt là những ứng dụng đòi hỏi xử lý các chuỗi dài:

1. **Mô hình hóa ngôn ngữ**: Sparse Transformer đã được sử dụng để huấn luyện các mô hình ngôn ngữ trên các chuỗi dài, đạt được kết quả tốt trên các bộ dữ liệu như Enwik8 và text8.
2. **Tạo sinh hình ảnh**: Sparse Attention đã được áp dụng trong các mô hình tạo sinh hình ảnh như ImageGPT và DALL-E, cho phép xử lý các chuỗi token hình ảnh dài.
3. **Xử lý âm thanh và video**: Sparse Attention đã được sử dụng để xử lý các chuỗi dữ liệu âm thanh và video dài, nơi các mối quan hệ cục bộ thường đóng vai trò quan trọng.
4. **Xử lý tài liệu dài**: Các biến thể của Sparse Attention như Longformer và Big Bird đã được sử dụng để xử lý các tài liệu dài trong các tác vụ như trả lời câu hỏi và tóm tắt văn bản.

Một số mô hình nổi bật sử dụng Sparse Attention hoặc các biến thể của nó bao gồm:

1. **Sparse Transformer (OpenAI)**: Mô hình gốc sử dụng Sparse Attention, đã đạt được kết quả tốt trong việc tạo sinh văn bản, hình ảnh và âm thanh.
2. **Longformer (Allen Institute for AI)**: Sử dụng sự kết hợp của attention cục bộ và toàn cục, đạt được kết quả tốt trong các tác vụ xử lý tài liệu dài.
3. **Big Bird (Google Research)**: Kết hợp attention cục bộ, ngẫu nhiên và toàn cục, đã được chứng minh là có khả năng biểu diễn tương đương với attention đầy đủ trong nhiều trường hợp.
4. **Reformer (Google Research)**: Sử dụng kỹ thuật hashing nhạy cảm với vị trí (LSH) để xác định các cặp token tương tự, giảm độ phức tạp xuống $O(n \cdot \log(n))$.

Sparse Attention là một bước tiến quan trọng trong việc mở rộng khả năng của các mô hình transformer để xử lý các chuỗi dài. Mặc dù có những hạn chế nhất định, nhưng lợi ích về hiệu suất và khả năng mở rộng đã khiến nó trở thành một công cụ quan trọng trong bộ công cụ của các nhà nghiên cứu và phát triển mô hình ngôn ngữ lớn, đặc biệt là khi làm việc với các chuỗi dài hoặc dữ liệu có cấu trúc đặc biệt.

1.8 Rotary Positional Embedding (RoPE)

Rotary Positional Embedding (RoPE) là một phương pháp mã hóa thông tin vị trí trong các mô hình transformer, được giới thiệu bởi Jianlin Su và cộng sự trong bài báo "RoFormer: Enhanced Transformer with Rotary Position Embedding" vào năm 2021. Khác với các phương pháp mã hóa vị trí truyền thống, RoPE kết hợp thông tin vị trí trực tiếp vào quá trình tính toán attention thông qua phép xoay trong không gian phức, mang lại nhiều ưu điểm đáng kể về hiệu suất và khả năng tổng quát hóa.

1.8.1 Nguyên lý hoạt động và cách mã hóa thông tin vị trí

Ý tưởng cốt lõi của RoPE dựa trên việc biểu diễn thông tin vị trí thông qua phép xoay trong không gian phức. Thay vì cộng thêm các vector vị trí vào các embedding token như trong positional encoding truyền thống, RoPE áp dụng một phép xoay phụ thuộc vào vị trí cho các vector query và key trước khi tính toán attention.

Cụ thể, RoPE biểu diễn mỗi cặp chiều liền kề (2i, 2i+1) trong vector embedding như một số phức, và áp dụng phép xoay với góc phụ thuộc vào vị trí của token và tần số của chiều embedding. Điều này có thể được hình dung như việc xoay các vector trong không gian 2D với góc xoay tăng dần theo vị trí.

Phép xoay này có một tính chất quan trọng: nó bảo toàn tích vô hướng tương đối. Điều này có nghĩa là tích vô hướng giữa hai vector sau khi xoay chỉ phụ thuộc vào khoảng cách tương đối giữa chúng, không phụ thuộc vào vị trí tuyệt đối. Tính chất này cho phép RoPE kết hợp cả thông tin vị trí tuyệt đối và tương đối trong cùng một cơ chế.

Một cách trực quan, RoPE hoạt động như sau:

1. Mỗi token ở vị trí m được biểu diễn bởi một vector embedding x_m

2. RoPE áp dụng một phép xoay R_m phụ thuộc vào vị trí m cho vector này
3. Khi tính toán attention giữa các token ở vị trí m và n, tích vô hướng giữa các vector đã xoay sẽ phụ thuộc vào khoảng cách (m-n)

1.8.2 Công thức toán học và biểu diễn ma trận

Về mặt toán học, RoPE được định nghĩa như sau:

Cho một vector embedding $x \in \mathbb{R}^d$, chúng ta xem mỗi cặp chiều liền kề (x_{2i}, x_{2i+1}) như một số phức $x_{2i} + ix_{2i+1}$. Phép xoay RoPE cho token ở vị trí m được định nghĩa bởi:

$$R_{\theta(x,m)} = [R_\theta^{0(x_0,x_1,m)}, R_\theta^{1(x_2,x_3,m)}, \dots, R_\theta^{\left(\frac{d}{2}-1\right)(x_{d-2},x_{d-1},m)}]$$

Trong đó R_θ^i là phép xoay 2D được định nghĩa bởi:

$$R_\theta^{i(x_{2i},x_{2i+1},m)} = [x_{2i}cos(\theta_i \cdot m) - x_{2i+1}sin(\theta_i \cdot m),$$
$$x_{2i}sin(\theta_i \cdot m) + x_{2i+1}cos(\theta_i \cdot m)]$$

Với $\theta_i = 10000^{-\frac{2i}{d}}$ là tần số cho chiều thứ i.

Trong ngữ cảnh của self-attention, RoPE được áp dụng cho cả query (q) và key (k):

$$q'_m = R_\theta(q_m, m) k'_n = R_\theta(k_n, n)$$

Khi tính toán tích vô hướng giữa q'_m và k'_n, chúng ta có:

$$\langle q'_m, k'_n \rangle = \langle R_\theta(q_m, m), R_\theta(k_n, n) \rangle = f(q_m, k_n, m - n)$$

Điều này cho thấy tích vô hướng sau khi áp dụng RoPE chỉ phụ thuộc vào khoảng cách tương đối (m-n), không phụ thuộc vào vị trí tuyệt đối.

Trong thực tế, RoPE thường được cài đặt bằng cách sử dụng các phép toán ma trận hiệu quả. Một cách tiếp cận phổ biến là sử dụng ma trận xoay phức:

$$e^{i\theta_j \cdot m} = cos(\theta_j \cdot m) \; + \; i \cdot sin(\theta_j \cdot m)$$

Và áp dụng phép nhân phức cho các cặp chiều liền kề trong vector embedding.

1.8.3 Ưu điểm và hạn chế

RoPE có nhiều ưu điểm nổi bật:

1. **Kết hợp cả thông tin vị trí tuyệt đối và tương đối**: RoPE mã hóa cả vị trí tuyệt đối của các token (thông qua phép xoay) và vị trí tương đối giữa chúng (thông qua tính chất bảo toàn tích vô hướng tương đối).
2. **Khả năng ngoại suy tốt**: RoPE cho phép mô hình ngoại suy tốt hơn đến các độ dài chuỗi lớn hơn so với các phương pháp mã hóa vị trí khác, đặc biệt khi được kết hợp với các kỹ thuật như interpolation hoặc dynamic NTK scaling.
3. **Tương thích với các cơ chế attention hiệu quả**: RoPE có thể được kết hợp dễ dàng với các cơ chế attention hiệu quả như Linear Attention, không đòi hỏi ma trận attention đầy đủ.
4. **Không làm tăng độ phức tạp mô hình**: RoPE không thêm tham số mới vào mô hình, chỉ thay đổi cách tính toán attention.
5. **Hiệu quả tính toán**: RoPE có thể được cài đặt hiệu quả bằng cách sử dụng các phép toán ma trận tối ưu.

Tuy nhiên, RoPE cũng có một số hạn chế:

1. **Phức tạp trong cài đặt**: Cài đặt RoPE có thể phức tạp hơn so với các phương pháp mã hóa vị trí đơn giản như sinusoidal positional encoding.
2. **Giới hạn trong ngoại suy**: Mặc dù RoPE có khả năng ngoại suy tốt hơn so với các phương pháp khác, nhưng vẫn có giới hạn khi ngoại suy đến các độ dài chuỗi lớn hơn nhiều so với dữ liệu huấn luyện.
3. **Yêu cầu điều chỉnh tần số**: Hiệu suất của RoPE phụ thuộc vào việc lựa chọn tần số θ_i phù hợp, đòi hỏi điều chỉnh cẩn thận.
4. **Khó khăn trong phân tích lý thuyết**: Tính chất toán học phức tạp của RoPE có thể gây khó khăn trong việc phân tích lý thuyết và hiểu rõ hành vi của nó.

1.8.4 So sánh với các phương pháp mã hóa vị trí khác

So với các phương pháp mã hóa vị trí khác, RoPE có những điểm khác biệt quan trọng:

1. **Absolute Positional Encoding (APE)**: APE (như trong Transformer gốc) cộng thêm các vector vị trí cố định vào các embedding token. So với APE, RoPE có khả năng ngoại suy tốt hơn và kết hợp cả thông tin vị trí tương đối.
2. **Relative Positional Encoding (RPE)**: RPE (như trong T5) thêm các bias phụ thuộc vào khoảng cách tương đối vào ma trận attention. So với RPE, RoPE không yêu cầu ma trận attention đầy đủ và có thể được kết hợp với các cơ chế attention hiệu quả.
3. **ALiBi (Attention with Linear Biases)**: ALiBi thêm các bias tuyến tính vào ma trận attention dựa trên khoảng cách tương đối. So với ALiBi, RoPE có khả năng biểu diễn các mối quan hệ phức tạp hơn và tương thích tốt hơn với các cơ chế attention hiệu quả.
4. **Learned Positional Encoding**: Các phương pháp này học các embedding vị trí từ dữ liệu. So với chúng, RoPE không yêu cầu tham số bổ sung và có khả năng ngoại suy tốt hơn.

Các thử nghiệm thực tế cho thấy RoPE thường vượt trội hơn các phương pháp khác về hiệu suất, đặc biệt là trong các tác vụ đòi hỏi hiểu các mối quan hệ phụ thuộc dài hạn và khả năng ngoại suy đến các độ dài chuỗi lớn hơn.

1.8.5 Ứng dụng trong các mô hình LLM hiện đại

RoPE đã được áp dụng rộng rãi trong nhiều mô hình ngôn ngữ lớn hiện đại:

1. **LLaMA và LLaMA-2**: Meta đã sử dụng RoPE trong các mô hình LLaMA và LLaMA-2, góp phần vào hiệu suất ấn tượng của chúng, đặc biệt là khả năng xử lý các chuỗi dài.
2. **Mistral**: Mô hình Mistral sử dụng RoPE kết hợp với kỹ thuật Sliding Window Attention để xử lý hiệu quả các chuỗi dài.
3. **Falcon**: Technology Innovation Institute đã áp dụng RoPE trong mô hình Falcon, kết hợp với Multi-Query Attention để cải thiện hiệu suất.

4. **MPT**: MosaicML đã sử dụng RoPE trong mô hình MPT, góp phần vào khả năng xử lý chuỗi dài của nó.
5. **Gemma**: Google đã áp dụng RoPE trong mô hình Gemma, kết hợp với các kỹ thuật khác để cải thiện hiệu suất.
6. **Claude**: Anthropic đã sử dụng RoPE trong mô hình Claude, góp phần vào khả năng hiểu và tạo ra văn bản dài của nó.

Ngoài ra, nhiều biến thể và cải tiến của RoPE đã được đề xuất và áp dụng:

1. **xPos**: Mở rộng RoPE với các tần số phức tạp hơn để cải thiện khả năng ngoại suy.
2. **YaRN (Yet another RoPE with Normalization)**: Kết hợp RoPE với kỹ thuật chuẩn hóa để cải thiện khả năng ngoại suy.
3. **Dynamic NTK scaling**: Điều chỉnh tần số của RoPE dựa trên lý thuyết Neural Tangent Kernel để cải thiện khả năng ngoại suy.
4. **Position Interpolation**: Nội suy các vị trí trong RoPE để mở rộng cửa sổ ngữ cảnh mà không cần huấn luyện lại.

Các kỹ thuật này đã cho phép các mô hình được huấn luyện với cửa sổ ngữ cảnh giới hạn (ví dụ: 2048 hoặc 4096 token) mở rộng đến các cửa sổ lớn hơn nhiều (32K, 100K hoặc thậm chí 1M token) mà không cần huấn luyện lại hoàn toàn.

RoPE đã trở thành một trong những phương pháp mã hóa vị trí phổ biến nhất trong các mô hình ngôn ngữ lớn hiện đại, nhờ vào khả năng kết hợp hiệu quả thông tin vị trí tuyệt đối và tương đối, cũng như khả năng ngoại suy tốt đến các độ dài chuỗi lớn hơn. Sự phát triển và cải tiến liên tục của RoPE đã góp phần quan trọng vào việc mở rộng khả năng của các mô hình ngôn ngữ lớn để xử lý các chuỗi ngày càng dài, mở ra nhiều ứng dụng mới trong xử lý ngôn ngữ tự nhiên.

1.9 So sánh và phân tích các cơ chế attention

Sau khi đã nghiên cứu chi tiết về bảy cơ chế attention khác nhau, phần này sẽ cung cấp một phân tích so sánh toàn diện giữa chúng, tập trung vào độ phức tạp tính toán, hiệu

quả bộ nhớ, ưu nhược điểm trong các tình huống khác nhau, và hướng dẫn lựa chọn cơ chế attention phù hợp cho các ứng dụng cụ thể.

Cơ chế Attention	**Độ phức tạp thời gian**	**Độ phức tạp bộ nhớ**	**Chất lượng mô hình**
Self-Attention (MHA)	$O(n^2 d)$	$O(n^2)$	Cao (chuẩn)
Multi-Query Attention (MQA)	$O(n^2 d)$	$O(nd)$	Trung bình-cao
Grouped-Query Attention (GQA)	$O(n^2 d)$	$O(nGd)$ với $G < H$	Cao
FlashAttention	$O(n^2 d)$	O(nd)	Cao (giống MHA)
Linear Attention	$O(nd^2)$	O(nd)	Trung bình
Sparse Attention	$O(n\sqrt{n} \cdot d)$	$O(n\sqrt{n})$	Trung bình-cao
Rotary Positional Embedding (RoPE)	Không thay đổi	Không thay đổi	Cao

Bảng 1.1 Bảng so sánh độ phức tạp tính toán

Trong bảng trên:

- n là độ dài chuỗi
- d là kích thước mô hình
- H là số đầu attention
- G là số nhóm trong GQA

Lưu ý rằng RoPE không thay đổi độ phức tạp tính toán hoặc bộ nhớ của cơ chế attention cơ bản, mà chỉ thay đổi cách mã hóa thông tin vị trí.

Cơ chế Attention	Yêu cầu bộ nhớ trong suy luận	Yêu cầu bộ nhớ trong huấn luyện	Khả năng xử lý chuỗi dài
Self-Attention (MHA)	$2 \times H \times n \times d_k$	$O(n^2)$	Hạn chế (thường < 4K tokens)
Multi-Query Attention (MQA)	$2 \times n \times d_k$	$O(nd)$	Tốt (có thể xử lý 8-16K tokens)
Grouped-Query Attention (GQA)	$2 \times G \times n \times d_k$	$O(nGd)$	Tốt (có thể xử lý 8-16K tokens)
FlashAttention	$O(nd)$	$O(nd)$	Rất tốt (có thể xử lý 16-32K tokens)
Linear Attention	$O(nd)$	$O(nd)$	Xuất sắc (có thể xử lý > 100K tokens)
Sparse Attention	$O(n\sqrt{n})$	$O(n\sqrt{n})$	Rất tốt (có thể xử lý 16-64K tokens)
Rotary Positional Embedding (RoPE)	Không thay đổi	Không thay đổi	Cải thiện khả năng ngoại suy

Bảng 1.2 Bảng so sánh hiệu quả bộ nhớ

Trong bảng trên:

- H là số đầu attention
- G là số nhóm trong GQA
- d_k là kích thước của vector key/query

1.9.1 Phân tích ưu nhược điểm trong các tình huống khác nhau

❖ **Tình huống 1: Xử lý văn bản ngắn với tài nguyên dồi dào**

- **Lựa chọn tốt nhất**: Self-Attention (MHA) hoặc FlashAttention
- **Lý do**: Với văn bản ngắn và tài nguyên dồi dào, độ phức tạp O(n²) không phải vấn đề lớn, và MHA cung cấp chất lượng mô hình tốt nhất. FlashAttention cung cấp cùng chất lượng nhưng nhanh hơn.

❖ **Tình huống 2: Xử lý văn bản dài với tài nguyên hạn chế**

- **Lựa chọn tốt nhất**: Linear Attention hoặc Sparse Attention
- **Lý do**: Với văn bản dài và tài nguyên hạn chế, độ phức tạp tuyến tính hoặc gần tuyến tính là cần thiết. Linear Attention và Sparse Attention cung cấp sự cân bằng tốt giữa hiệu quả và chất lượng.

❖ **Tình huống 3: Suy luận trên thiết bị di động hoặc edge**

- **Lựa chọn tốt nhất**: Multi-Query Attention (MQA) hoặc Grouped-Query Attention (GQA)
- **Lý do**: Trên thiết bị có tài nguyên hạn chế, MQA và GQA cung cấp sự cân bằng tốt giữa hiệu quả bộ nhớ và chất lượng mô hình, đặc biệt là trong quá trình suy luận.

❖ **Tình huống 4: Huấn luyện mô hình lớn**

- **Lựa chọn tốt nhất**: FlashAttention kết hợp với RoPE
- **Lý do**: FlashAttention cung cấp tốc độ huấn luyện nhanh hơn đáng kể mà không làm giảm chất lượng mô hình, trong khi RoPE cải thiện khả năng nắm bắt mối quan hệ phụ thuộc vị trí.

❖ **Tình huống 5: Xử lý chuỗi cực dài (>100K tokens)**

- **Lựa chọn tốt nhất**: Linear Attention kết hợp với RoPE
- **Lý do**: Độ phức tạp tuyến tính của Linear Attention là cần thiết cho các chuỗi cực dài, trong khi RoPE cải thiện khả năng ngoại suy đến các độ dài chuỗi lớn hơn.

1.9.2 Hướng dẫn lựa chọn cơ chế attention phù hợp

Khi lựa chọn cơ chế attention phù hợp cho một ứng dụng cụ thể, cần xem xét các yếu tố sau:

1. **Độ dài chuỗi đầu vào**:
   - Chuỗi ngắn (<2K tokens): Self-Attention (MHA) hoặc FlashAttention

- Chuỗi trung bình (2K-16K tokens): GQA, FlashAttention, hoặc Sparse Attention
- Chuỗi dài (>16K tokens): Linear Attention hoặc Sparse Attention

2. **Tài nguyên tính toán có sẵn**:
   - Tài nguyên dồi dào (GPU/TPU hiệu năng cao): FlashAttention hoặc MHA
   - Tài nguyên trung bình: GQA hoặc Sparse Attention
   - Tài nguyên hạn chế: MQA hoặc Linear Attention
3. **Yêu cầu về chất lượng mô hình**:
   - Chất lượng cao nhất: MHA hoặc FlashAttention
   - Cân bằng chất lượng và hiệu quả: GQA hoặc Sparse Attention
   - Ưu tiên hiệu quả: MQA hoặc Linear Attention
4. **Giai đoạn sử dụng**:
   - Huấn luyện: FlashAttention hoặc MHA
   - Suy luận: MQA, GQA, hoặc Linear Attention
5. **Khả năng ngoại suy**:
   - Cần ngoại suy đến độ dài lớn hơn: RoPE kết hợp với kỹ thuật như dynamic NTK scaling
6. **Loại tác vụ**:
   - Tạo sinh văn bản: MHA, GQA, hoặc FlashAttention
   - Xử lý tài liệu dài: Sparse Attention hoặc Linear Attention
   - Tạo sinh hình ảnh/âm thanh: Sparse Attention hoặc Linear Attention

Ngoài ra, cần lưu ý rằng nhiều mô hình hiện đại kết hợp nhiều cơ chế attention khác nhau để tận dụng ưu điểm của từng loại. Ví dụ:

- LLaMA-2 kết hợp GQA với RoPE
- Mistral kết hợp Sliding Window Attention với RoPE
- Falcon kết hợp MQA với RoPE

Việc lựa chọn cơ chế attention phù hợp là một phần quan trọng trong thiết kế mô hình transformer, và có thể có tác động đáng kể đến hiệu suất, hiệu quả và khả năng mở rộng của mô hình.

1.10 Kết luận và hướng phát triển tương lai

Trong báo cáo này, chúng tôi đã tiến hành phân tích toàn diện về bảy cơ chế attention quan trọng trong các mô hình ngôn ngữ lớn hiện đại như Self-Attention, Multi-Query Attention (MQA), Grouped-Query Attention (GQA), FlashAttention/FlashAttention v2, Linear Attention, Sparse Attention và Rotary Positional Embedding (RoPE). Mỗi cơ chế đều có những ưu điểm và hạn chế riêng, phù hợp với các tình huống và yêu cầu khác nhau.

1.10.1 Tổng kết các cơ chế attention đã nghiên cứu

Self-Attention là nền tảng của kiến trúc Transformer, cho phép mỗi token tương tác với tất cả các token khác trong chuỗi. Mặc dù cung cấp chất lượng mô hình cao, nhưng nó có độ phức tạp tính toán và bộ nhớ $O(n^2)$, gây khó khăn khi xử lý các chuỗi dài.

Multi-Query Attention (MQA) giảm yêu cầu bộ nhớ bằng cách sử dụng chung một ma trận Key và một ma trận Value cho tất cả các đầu attention, chỉ có ma trận Query là khác nhau giữa các đầu. Điều này giúp tăng tốc độ suy luận và giảm yêu cầu bộ nhớ, mặc dù có thể làm giảm nhẹ chất lượng mô hình.

Grouped-Query Attention (GQA) cung cấp một sự cân bằng giữa MHA và MQA bằng cách chia các đầu attention thành các nhóm, mỗi nhóm dùng chung một ma trận Key và một ma trận Value. GQA cho phép điều chỉnh linh hoạt sự cân bằng giữa hiệu suất và hiệu quả.

FlashAttention/FlashAttention v2 tối ưu hóa cách tính toán attention trên phần cứng GPU thông qua IO-Awareness, giảm độ phức tạp bộ nhớ từ $O(n^2)$ xuống $O(n)$ mà không làm thay đổi kết quả cuối cùng. FlashAttention tăng tốc đáng kể cả quá trình huấn luyện và suy luận.

Linear Attention giảm độ phức tạp tính toán và bộ nhớ từ $O(n^2)$ xuống $O(n)$ bằng cách sử dụng các hàm ánh xạ đặc trưng thay cho hàm softmax. Điều này cho phép xử lý các chuỗi cực dài, mặc dù có thể làm giảm chất lượng mô hình.

Sparse Attention chỉ tính toán attention cho một tập con các cặp token thay vì tất cả các cặp có thể, giảm độ phức tạp xuống $O(n\sqrt{n})$ hoặc thậm chí $O(n \cdot \log(n))$. Sparse Attention cho phép nắm bắt cả mối quan hệ cục bộ và dài hạn thông qua việc kết hợp các mẫu attention khác nhau.

Rotary Positional Embedding (RoPE) mã hóa thông tin vị trí thông qua phép xoay trong không gian phức, kết hợp cả thông tin vị trí tuyệt đối và tương đối. RoPE có khả năng ngoại suy tốt đến các độ dài chuỗi lớn hơn và tương thích với các cơ chế attention hiệu quả.

1.10.2 Xu hướng phát triển mới trong tối ưu hóa attention

Lĩnh vực tối ưu hóa attention đang phát triển nhanh chóng, với nhiều xu hướng mới nổi:

1. **Kết hợp nhiều cơ chế attention**: Các mô hình hiện đại thường kết hợp nhiều cơ chế attention khác nhau để tận dụng ưu điểm của từng loại. Ví dụ, kết hợp GQA với RoPE, hoặc Sparse Attention với FlashAttention.
2. **Attention với độ phức tạp tuyến tính**: Ngoài Linear Attention, nhiều phương pháp khác đang được phát triển để đạt được độ phức tạp tuyến tính, như Performer, Linformer, và Nyströmformer.
3. **Attention dựa trên state space models**: Các mô hình như Mamba và S4 đang kết hợp ý tưởng từ state space models với attention để đạt được độ phức tạp tuyến tính và khả năng mô hình hóa chuỗi dài.
4. **Attention với kiến trúc hỗn hợp**: Các mô hình như Mixture-of-Experts (MoE) kết hợp attention với kiến trúc hỗn hợp để cải thiện hiệu suất và hiệu quả.

5. **Attention với tối ưu hóa phần cứng**: Các cơ chế attention được thiết kế đặc biệt cho các kiến trúc phần cứng cụ thể, như GPU, TPU, hoặc thậm chí là các chip AI chuyên dụng.
6. **Attention với khả năng ngoại suy tốt hơn**: Các cải tiến của RoPE như xPos, YaRN, và dynamic NTK scaling đang được phát triển để cải thiện khả năng ngoại suy đến các độ dài chuỗi lớn hơn.
7. **Attention với cửa sổ trượt và bộ nhớ cache**: Các kỹ thuật như Sliding Window Attention và KV Cache đang được tối ưu hóa để cải thiện hiệu suất suy luận.

1.10.3 Các thách thức còn tồn tại và hướng nghiên cứu tiềm năng

Mặc dù đã có nhiều tiến bộ, vẫn còn nhiều thách thức trong lĩnh vực tối ưu hóa attention:

1. **Cân bằng giữa hiệu quả và chất lượng**: Hầu hết các phương pháp tối ưu hóa attention đều có sự đánh đổi giữa hiệu quả tính toán và chất lượng mô hình. Tìm ra phương pháp có thể cải thiện hiệu quả mà không làm giảm chất lượng vẫn là một thách thức lớn.
2. **Xử lý chuỗi cực dài**: Mặc dù đã có nhiều cải tiến, việc xử lý các chuỗi cực dài (hàng triệu token) vẫn là một thách thức lớn, đặc biệt là khi cần nắm bắt các mối quan hệ phụ thuộc dài hạn.
3. **Tối ưu hóa cho các thiết bị có tài nguyên hạn chế**: Triển khai các mô hình transformer hiệu quả trên các thiết bị di động hoặc edge vẫn là một thách thức lớn.
4. **Khả năng ngoại suy**: Cải thiện khả năng ngoại suy của các mô hình đến các độ dài chuỗi lớn hơn nhiều so với dữ liệu huấn luyện vẫn là một hướng nghiên cứu quan trọng.
5. **Hiểu biết lý thuyết**: Phát triển hiểu biết lý thuyết sâu sắc hơn về các cơ chế attention và tác động của chúng đến hiệu suất mô hình.

Các hướng nghiên cứu tiềm năng bao gồm:

1. **Attention với cấu trúc phân cấp**: Phát triển các cơ chế attention có thể nắm bắt cấu trúc phân cấp trong dữ liệu, từ cấp độ token đến cấp độ đoạn và tài liệu.
2. **Attention đa phương thức**: Tối ưu hóa attention cho các mô hình đa phương thức, nơi cần xử lý và tích hợp thông tin từ nhiều loại dữ liệu khác nhau (văn bản, hình ảnh, âm thanh, v.v.).
3. **Attention với khả năng thích ứng**: Phát triển các cơ chế attention có thể thích ứng động với đặc điểm của dữ liệu đầu vào, tự động điều chỉnh mẫu attention hoặc độ phức tạp tính toán.
4. **Attention với bộ nhớ ngoài**: Kết hợp attention với các cơ chế bộ nhớ ngoài để xử lý hiệu quả các chuỗi cực dài hoặc thông tin từ nhiều nguồn khác nhau.
5. **Attention với khả năng giải thích**: Phát triển các cơ chế attention có khả năng giải thích cao hơn, giúp hiểu rõ hơn về cách mô hình đưa ra quyết định.

Tóm lại, lĩnh vực tối ưu hóa attention đang phát triển nhanh chóng và có nhiều hướng nghiên cứu đầy hứa hẹn. Việc tiếp tục cải tiến các cơ chế attention sẽ đóng vai trò quan trọng trong việc mở rộng khả năng của các mô hình ngôn ngữ lớn, cho phép chúng xử lý các chuỗi dài hơn, hiệu quả hơn, và với chất lượng cao hơn.

CHƯƠNG 2 – ỨNG DỤNG MÔ HÌNH CNN VÀ TRANSFORMER – DECODER TRONG NHẬN DẠNG VĂN BẢN ẢNH

2.1 Giới thiệu OCR

OCR (Nhận dạng Ký tự Quang học) là công nghệ giúp chuyển đổi văn bản trong hình ảnh (bao gồm văn bản viết tay, đánh máy hoặc in) thành dữ liệu văn bản có thể chỉnh sửa và tìm kiếm được. Công nghệ này sử dụng các thuật toán phức tạp để phân tích hình ảnh, nhận dạng các ký tự và chuyển chúng thành dữ liệu số. Chính vì vậy, OCR ngày càng trở nên quan trọng trong kỷ nguyên số, với nhiều ứng dụng thực tế như số hóa tài liệu lưu trữ, tự động hóa nhập liệu hóa đơn và biểu mẫu, nhận dạng biển số xe, hỗ trợ người khiếm thị, và trích xuất thông tin từ hình ảnh trên mạng xã hội hoặc trong các môi trường thực tế.

Quá trình OCR thường bao gồm các bước thu nhận hình ảnh, bằng cách sử dụng máy quét hoặc thiết bị chụp ảnh để thu nhận hình ảnh chứa văn bản. Sau khi thu thập, ảnh sẽ được tiền xử lý để loại bỏ nhiễu, điều chỉnh độ sáng, độ tương phản và căn chỉnh văn bản cho phù hợp. Khi đã có dữ liệu đầy đủ, OCR sẽ áp dụng các thuật toán nhận dạng mẫu hoặc trích xuất đặc trưng để nhận diện các ký tự trong ảnh. Cuối cùng, các kỹ thuật như kiểm tra chính tả và ngữ cảnh được sử dụng để cải thiện độ chính xác của văn bản đã nhận dạng.

Có một số loại OCR phổ biến như Nhận dạng Ký tự Thông minh (ICR), giúp nhận dạng chữ viết tay, cải thiện khả năng nhận diện văn bản viết tay. Bên cạnh đó, Nhận dạng Từ Thông minh giúp xử lý toàn bộ từ thay vì từng ký tự riêng lẻ, mang lại hiệu quả nhanh chóng và chính xác hơn trong việc nhận diện văn bản. Ngoài ra, Nhận dạng Dấu Quang học (OMR) giúp nhận diện các dấu hiệu, chẳng hạn như các ô trên phiếu khảo sát hoặc biển số xe, hỗ trợ tự động hóa quá trình thu thập dữ liệu.

Tuy nhiên, việc xây dựng hệ thống OCR hiệu quả, đặc biệt đối với văn bản xuất hiện trong các ảnh tự nhiên (scene text) như văn bản trên biển báo, thẻ, hóa đơn, biển hiệu..., gặp phải nhiều thách thức đáng kể. Văn bản trong ảnh tự nhiên thường bị ảnh hưởng bởi các yếu tố như phối cảnh biến dạng, ánh sáng không đồng đều, phông chữ đa dạng, độ phân giải thấp, nền lộn xộn, và hướng văn bản không chuẩn. Các phương pháp OCR truyền thống thường gặp khó khăn trong việc xử lý những yếu tố này.

Để giải quyết các thách thức này, các kiến trúc học sâu hiện đại đã được phát triển và cho thấy những thành tựu ấn tượng. Một trong những phương pháp mạnh mẽ và phổ biến hiện nay là sự kết hợp giữa Mạng Nơ-ron Tích chập (CNN) và Transformer Decoder. Trong đó, CNN đóng vai trò trích xuất các đặc trưng mạnh mẽ từ ảnh đầu vào, còn Transformer Decoder với cơ chế attention tinh vi giúp chuyển các chuỗi đặc trưng thành chuỗi văn bản tương ứng. Cơ chế attention giúp mô hình tập trung vào các phần quan trọng của ảnh khi sinh ra văn bản, cải thiện độ chính xác của việc trích xuất văn bản từ ảnh.

2.2 Nguyên lý kiến trúc CNN – Transformer cho OCR

2.2.1 Vai trò của mạng nơ-ron tích chập (CNN) trong OCR

Mạng nơ-ron Tích chập (Convolutional Neural Network – CNN) đóng vai trò nền tảng và không thể thiếu trong các hệ thống nhận dạng ký tự quang học (OCR) hiện nay, đặc biệt là khi xử lý các văn bản trong ảnh tự nhiên (scene text). Nhiệm vụ chính của CNN trong kiến trúc CNN-Transformer chính là trích xuất các đặc trưng hình ảnh dạng lưới (grid of features) hoặc một vector đặc trưng tổng thể mạnh mẽ và có ý nghĩa từ ảnh đầu vào. Những đặc trưng này sẽ được cung cấp cho thành phần Transformer Decoder để giải mã thành các chuỗi ký tự tương ứng.

2.2.1.1 Nguyên lý cơ bản

CNN là một lớp các mạng nơ-ron sâu (deep neural networks) được thiết kế đặc biệt để xử lý dữ liệu có cấu trúc dạng lưới, chẳng hạn như hình ảnh. Điểm đặc biệt của CNN nằm ở việc sử dụng các lớp tích chập (convolutional layers) để tự động và thích ứng học

các hệ thống phân cấp đặc trưng từ dữ liệu đầu vào. Các thành phần chính của một CNN điển hình bao gồm:

- Lớp Tích chập (Convolutional Layer) là lớp quan trọng trong CNN. Lớp này áp dụng một tập hợp các bộ lọc (filters hoặc kernels) có thể học được lên ảnh đầu vào. Mỗi bộ lọc trượt (convolve) trên toàn bộ chiều rộng và chiều cao của ảnh, tính toán tích chập giữa các trọng số của bộ lọc và vùng ảnh cục bộ mà nó đang bao phủ. Quá trình này tạo ra một bản đồ đặc trưng (feature map) cho mỗi bộ lọc, làm nổi bật các mẫu cụ thể (như cạnh, góc, kết cấu) tại các vị trí khác nhau trong ảnh. Một ưu điểm quan trọng của lớp tích chập là khả năng chia sẻ tham số (parameter sharing), nghĩa là cùng một bộ lọc được sử dụng trên toàn bộ ảnh, giúp giảm đáng kể số lượng tham số cần học và làm cho mạng có khả năng bất biến với sự dịch chuyển (translation invariant).
- Hàm Kích hoạt (Activation Function), Sau mỗi lớp tích chập, một hàm kích hoạt phi tuyến thường được áp dụng, phổ biến nhất là ReLU (Rectified Linear Unit), với công thức f(x) = max (0, x). Hàm này giúp mạng học được các mối quan hệ phức tạp và phi tuyến trong dữ liệu. ReLU đơn giản chỉ trả về giá trị đầu vào nếu nó dương và trả về 0 nếu nó âm.
- Lớp Gộp (Pooling Layer) thường được chèn vào giữa các lớp tích chập liên tiếp. Mục đích chính của nó là giảm dần kích thước không gian (chiều rộng và chiều cao) của các bản đồ đặc trưng, từ đó giảm số lượng tham số và tính toán trong mạng, đồng thời giúp kiểm soát overfitting. Các phương pháp gộp phổ biến là Max Pooling (lấy giá trị lớn nhất trong một vùng cục bộ) và Average Pooling (lấy giá trị trung bình). Lớp gộp cũng góp phần tạo ra sự bất biến với các biến dạng nhỏ hoặc dịch chuyển cục bộ trong ảnh.
- Lớp Kết nối đầy đủ (Fully Connected Layer) thường nằm ở cuối mạng CNN (trong các tác vụ phân loại cổ điển), lớp này kết nối mọi nơ-ron từ lớp trước với mọi nơ-ron trong lớp hiện tại. Tuy nhiên, trong các kiến trúc OCR hiện đại

như CNN-Transformer, các lớp kết nối đầy đủ truyền thống ít được sử dụng trực tiếp sau phần trích xuất đặc trưng chính. Thay vào đó, các đặc trưng không gian từ CNN được xử lý tiếp bởi các cơ chế khác (như Attention).

2.2.1.2 Cách CNN trích xuất đặc trưng hình ảnh cho OCR

Các kiến trúc CNN như ResNet (Residual Network), VGG, cùng với các phiên bản nhẹ hơn, là những lựa chọn phổ biến khi sử dụng làm backbone trong các mô hình OCR (Nhận dạng ký tự quang học). Việc lựa chọn kiến trúc CNN phụ thuộc vào yêu cầu về độ chính xác và hiệu suất tính toán của ứng dụng cụ thể.

Chẳng hạn, ResNet sử dụng kết nối tắt (shortcut connections), giúp giải quyết vấn đề suy giảm độ dốc (vanishing gradient) khi huấn luyện các mạng nơ-ron rất sâu. Điều này cho phép mô hình có thể học và trích xuất được các đặc trưng ngữ nghĩa phong phú từ ảnh mà không gặp phải sự mất mát thông tin qua các lớp sâu.

Sau khi ảnh đầu vào được đưa qua các lớp CNN, nó sẽ được chuyển đổi thành một bản đồ đặc trưng đa chiều, ví dụ với kích thước là Height x Width x Channels. Bản đồ này chứa đựng các đặc trưng không gian và ngữ nghĩa, như vị trí và hình dạng của các ký tự, cũng như các mối quan hệ giữa các ký tự và các phần khác trong ảnh. Những đặc trưng này sẽ được cung cấp cho bộ giải mã Transformer (Transformer Decoder), nơi chúng sẽ được xử lý để sinh ra chuỗi văn bản từ ảnh.

Mô hình CNN này đóng vai trò quan trọng trong việc trích xuất các đặc trưng đặc biệt từ ảnh, giúp nhận diện và hiểu văn bản hiệu quả, từ đó phục vụ cho các ứng dụng OCR như nhận dạng chữ viết tay, biển hiệu, hóa đơn...

2.2.1.3 Ưu và nhược điểm

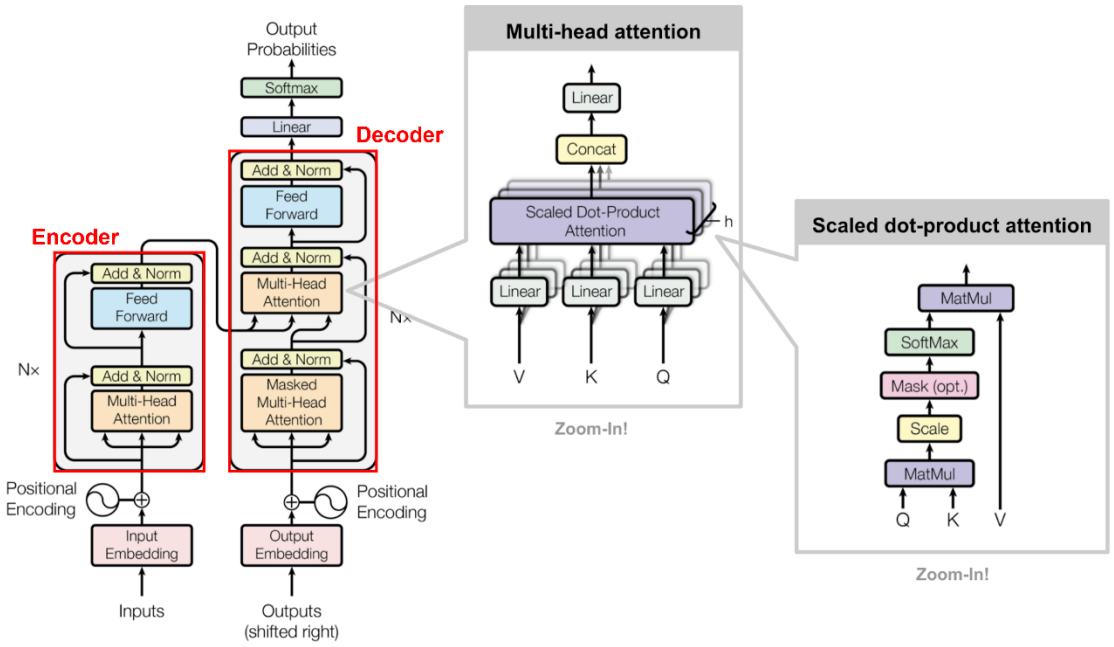
CNN có khả năng học đặc trưng tự động và phân cấp, giúp mô hình nhận diện các đặc trưng quan trọng trong ảnh mà không cần sự can thiệp thủ công. Nó cũng có tính bất biến với dịch chuyển cục bộ, tức là có thể nhận diện các ký tự trong ảnh dù chúng có thay đổi vị trí nhỏ. Thêm vào đó, việc chia sẻ tham số trong CNN giúp giảm số lượng tham số, làm giảm độ phức tạp của mô hình và giúp tiết kiệm tài nguyên tính toán.

Một trong những hạn chế lớn của CNN là mô hình này chủ yếu tập trung vào thông tin cục bộ thông qua các cửa sổ tích chập (convolutional windows), điều này khiến CNN gặp khó khăn trong việc nắm bắt các phụ thuộc ngữ cảnh xa hoặc các mối quan hệ phức tạp trên toàn bộ ảnh. Hơn nữa, kích thước đầu ra của CNN phụ thuộc vào kích thước đầu vào, điều này đòi hỏi các kỹ thuật bổ sung như pooling thích ứng để xử lý ảnh có kích thước thay đổi hoặc để tạo ra chuỗi đặc trưng có độ dài cố định hoặc linh hoạt cho các mô hình tuần tự tiếp theo.

Chính những hạn chế của CNN trong việc mô hình hóa các mối quan hệ tuần tự và phụ thuộc xa đã thúc đẩy việc kết hợp CNN với kiến trúc Transformer, đặc biệt là Transformer Decoder. Cơ chế attention mạnh mẽ của Transformer có khả năng xử lý các phụ thuộc dài hạn và học mối quan hệ phức tạp giữa các phần tử trong chuỗi, điều mà CNN khó có thể làm được chỉ với các tầng tích chập cục bộ. Nhờ vậy, kết hợp CNN và Transformer Decoder giúp tạo ra các mô hình OCR hiệu quả hơn, có khả năng nhận diện và sinh văn bản chính xác từ ảnh, bất kể độ phức tạp hay biến dạng của văn bản trong ảnh.

2.2.2 Vai trò của Transformer Decoder trong OCR

Bộ giải mã Transformer Decoder đóng vai trò quan trọng trong việc chuyển đổi các đặc trưng hình ảnh từ CNN thành chuỗi văn bản có ý nghĩa, thực hiện dự đoán ký tự hoặc từ tuần tự dựa trên các đặc trưng hình ảnh và văn bản đã được sinh ra trước đó. Kiến trúc Transformer Decoder hoạt động dựa trên cơ chế self-attention và cross-attention, giúp mô hình học và nhận diện các mối quan hệ giữa các ký tự và đặc trưng hình ảnh, từ đó sinh ra văn bản chính xác. Một Transformer Decoder bao gồm một stack gồm các lớp decoder (thường là N lớp) với các thành phần nhất định.



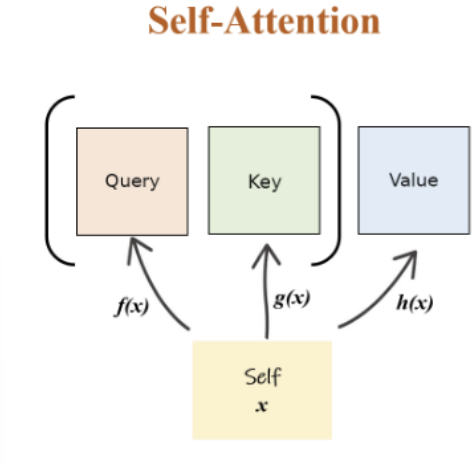
Hình 2.1 Tổng quan kiến trúc mô hình Transformer

2.2.2.1 Masked Multi – Head Self – Attention

Cơ chế hoạt động: Trong quá trình sinh văn bản, cơ chế Masked Multi-Head Self-Attention trong bộ giải mã Transformer đảm bảo rằng mỗi vị trí trong chuỗi đầu ra chỉ có thể chú ý đến các vị trí đã được sinh ra trước đó, không thể "nhìn thấy" các vị trí tương lai. Điều này được thực hiện thông qua việc áp dụng một ma trận mặt nạ (mask) lên ma trận tương tác giữa các vị trí trong chuỗi, ngăn cản việc truyền thông tin từ các vị trí chưa được xử lý.

Ứng dụng: Việc áp dụng mặt nạ trong cơ chế self-attention giúp duy trì tính tự hồi quy (auto - regressive) của mô hình, nghĩa là mỗi ký tự được sinh ra dựa trên các ký tự đã được sinh ra trước đó, từ trái qua phải. Điều này đặc biệt quan trọng trong các bài toán như nhận dạng văn bản từ ảnh (OCR), nơi thứ tự của các ký tự trong chuỗi đầu ra cần

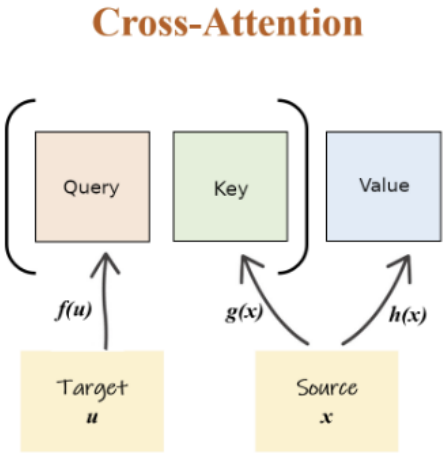
được duy trì để đảm bảo tính chính xác và ý nghĩa của văn bản được sinh ra.



Hình 2.2 Cách tính toán Q, K và V trong Self - Attention

2.2.2.2 Multi – Head Cross – Attention

Cơ chế hoạt động: Trong kiến trúc Transformer, cơ chế Cross-Attention cho phép bộ giải mã (decoder) tập trung vào các phần khác nhau của chuỗi đặc trưng hình ảnh do CNN tạo ra. Cụ thể, tại mỗi bước giải mã, trạng thái hiện tại của bộ giải mã (được sinh ra từ các bước trước đó) sẽ được sử dụng làm Query (Q), trong khi chuỗi đặc trưng hình ảnh từ CNN đóng vai trò là Key (K) và Value (V). Cơ chế này giúp mô hình xác định phần nào trong ảnh là quan trọng nhất để sinh ra ký tự tiếp theo trong chuỗi văn bản.



Hình 2.3 Cách tính toán Q, K và V trong Cross - Attention

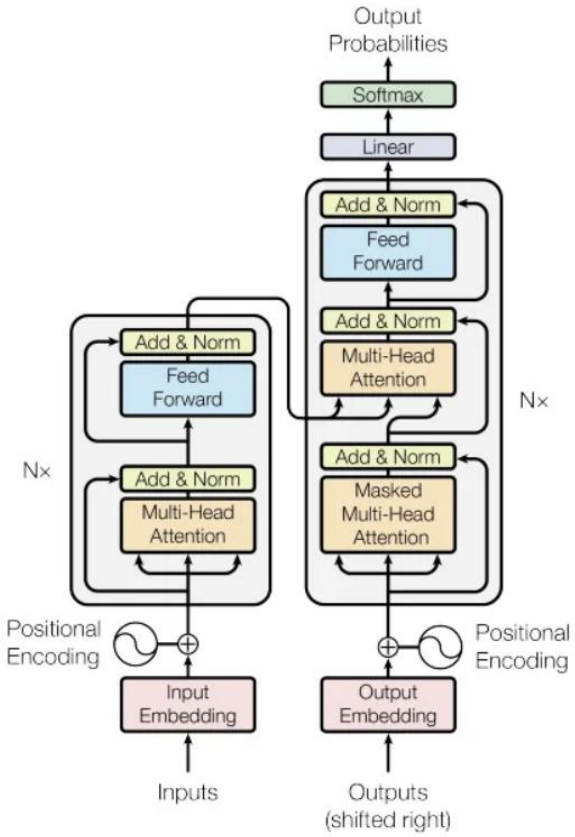
Ứng dụng: Trong bài toán OCR, việc nhận dạng văn bản từ hình ảnh không chỉ đơn giản là nhận diện các ký tự riêng lẻ mà còn phải hiểu được mối quan hệ giữa các ký tự và cấu trúc ngữ nghĩa của văn bản. Cơ chế Multi-Head Cross-Attention giúp mô hình tập trung vào các phần quan trọng của ảnh tương ứng với các ký tự hoặc từ trong văn bản, từ đó cải thiện độ chính xác của việc nhận dạng. Ngoài ra, việc sử dụng nhiều đầu attention (multi-head) giúp mô hình học được nhiều khía cạnh khác nhau của mối quan hệ giữa văn bản và hình ảnh, từ đó tăng cường khả năng tổng quát và độ chính xác của mô hình.

2.2.2.3 Feed Forward Network

Trong kiến trúc Transformer, Mạng Nơ-ron Truyền thẳng (Feed-Forward Network - FFN) đóng vai trò quan trọng trong việc xử lý và tinh chỉnh các đặc trưng sau khi chúng đã được xử lý qua cơ chế attention. FFN giúp mô hình học được các biểu diễn phi tuyến tính, tăng cường khả năng phân loại và nhận diện trong các tác vụ như nhận dạng văn bản từ ảnh (OCR).

FFN trong Transformer bao gồm hai lớp tuyến tính (fully connected) với một hàm kích hoạt phi tuyến ở giữa, thường là ReLU. Cụ thể như sau:

- Lớp đầu tiên dùng để chuyển đổi đầu vào có kích thước d_{model} thành một không gian ẩn có kích thước d_{ff} ($d_{ff} > d_{model}$).
- Hàm kích hoạt thường sẽ áp dụng hàm ReLU để thêm tính phi tuyến tính cho mô hình.
- Lớp thứ hai sẽ chuyển đổi đầu ra từ lớp đầu tiên trở lại không gian kích thước d_{model}. Công thức toán học cơ bản cho Feed Forward Network $FFN(x) = \max(0, xW_1 + b_1)\, W_2 + b_2$. Trong đó W_1, W_2 là các ma trận trọng số và b_1, b_2 là các vector bias.



Hình 2.4 Minh hoạ FFN trong một khối Transformer

Trong bài toán nhận dạng văn bản từ ảnh (OCR), Mạng Nơ-ron Truyền thẳng (FFN) đóng vai trò quan trọng trong việc cải thiện hiệu suất của mô hình. Đầu tiên, FFN giúp mô hình xử lý các mối quan hệ phi tuyến giữa các đặc trưng, điều này rất quan trọng trong việc nhận dạng văn bản có ngữ nghĩa phức tạp. Sau khi các đặc trưng hình ảnh đã được trích xuất và xử lý qua cơ chế attention, FFN giúp mô hình phân loại chính xác các ký tự hoặc từ trong văn bản. Bên cạnh đó, FFN còn giúp tinh chỉnh các đặc trưng đã học, làm cho chúng phù hợp hơn với tác vụ nhận dạng văn bản. Theo các nghiên cứu, FFN trong kiến trúc Transformer giúp mô hình học được các biểu diễn phức tạp, từ đó cải thiện hiệu suất trong các tác vụ như OCR.

2.2.2.4 Lớp tuyến tính và Softmax

Trong mô hình Transformer áp dụng cho bài toán Nhận dạng Ký tự Quang học (OCR), sau khi các đặc trưng hình ảnh được trích xuất từ mạng nơ-ron tích chập (CNN), chúng cần được chuyển đổi thành chuỗi văn bản có ý nghĩa. Quá trình này được thực hiện bởi bộ giải mã Transformer (Transformer Decoder), và một phần quan trọng trong quá trình này là lớp tuyến tính (linear layer) kết hợp với hàm softmax.

Lớp tuyến tính trong Transformer Decoder có nhiệm vụ ánh xạ đầu ra từ các lớp attention (có kích thước đặc trưng là d_{model}) thành không gian kích thước của bộ từ vựng (với kích thước là vocab_size). Điều này cho phép mô hình dự đoán xác suất cho từng ký tự trong bộ từ vựng tại mỗi bước giải mã. Cụ thể, lớp tuyến tính thực hiện phép toán $logits = Linear(h_t)$. Trong đó, h_t là đầu ra của bộ giải mã tại bước thời gian t.

Sau khi có logits từ lớp tuyến tính, hàm softmax được áp dụng để chuyển đổi các giá trị này thành phân phối xác suất. Hàm softmax tính toán xác suất cho mỗi ký tự trong bộ từ vựng bằng cách chuẩn hóa các logits, đảm bảo tổng xác suất bằng 1. Công thức của hàm softmax là:

$$\mathrm{P(y = j \mid x)} = \frac{e^{z_j}}{\sum_{k=1}^{V} e^{z_k}} .$$

Trong đó z_j là giá trị của logits tương ứng với ký tự thứ j, và V là kích thước của bộ từ vựng. Việc sử dụng hàm softmax giúp mô hình có thể chọn ra ký tự có xác suất cao nhất từ đó sunh ra văn bản một cách tuần tự.

Hàm softmax giúp chuyển đổi các giá trị logits thành xác suất, từ đó cho phép mô hình đưa ra dự đoán có ý nghĩa, xác định ký tự nào sẽ xuất hiện tiếp theo trong chuỗi văn bản. Việc sử dụng lớp tuyến tính kết hợp với softmax không chỉ giúp mô hình chuyển đổi các đặc trưng từ ảnh thành văn bản, mà còn đảm bảo rằng quá trình sinh văn bản diễn ra tuần tự, từ trái sang phải, giống như cách con người đọc và viết. Đồng thời, sự kết hợp

này cũng tối ưu hóa quá trình học của mô hình, giúp nó hiểu được các đặc trưng ngữ nghĩa từ ảnh và sinh ra văn bản chính xác hơn.

2.2.2.4 Positional encoding

Trong các mô hình Transformer, mã hóa vị trí đóng vai trò quan trọng trong việc cung cấp thông tin về thứ tự của các phần tử trong chuỗi đầu vào. Vì Transformer không có cơ chế tuần tự như RNN hay LSTM, việc sử dụng mã hóa vị trí là cần thiết để mô hình có thể nhận thức được thứ tự của các phần tử trong chuỗi.

Mã hóa vị trí được thêm vào các vector nhúng (embedding vectors) của các phần tử trong chuỗi đầu vào để cung cấp thông tin về vị trí của chúng. Trong mô hình Transformer ban đầu, mã hóa vị trí được tính toán bằng các hàm sin và cos với các tần số khác nhau, cho phép mô hình nhận biết được vị trí tương đối của các phần tử trong chuỗi. Công thức tính mã hóa vị trí cho một phần tử tại vị trí pospospos trong chuỗi với chiều dài d_{model} là:

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

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

Trong đó, pos là vị trí phần tử trong chuỗi, i là chỉ số của chiều trong vector mã hoá vị trí, d_{model} là kích thước của vector nhúng. Mã hoá ở các vị trí này sẽ được thêm vào các vector nhúng của các phần tử trong chuỗi đầu vào giúp mô hình nhận biết được thứ tự của chúng.

Trong bài toán nhận dạng văn bản từ ảnh (OCR), thứ tự của các ký tự trong chuỗi văn bản là rất quan trọng. Việc sử dụng mã hóa vị trí giúp mô hình Transformer nhận biết được thứ tự của các ký tự trong chuỗi đầu ra, từ đó cải thiện khả năng sinh ra văn bản chính xác. Ví dụ, trong một chuỗi văn bản như “Hà Nội”, việc nhận biết được thứ tự của các ký tự là rất quan trọng để mô hình có thể sinh ra chuỗi văn bản đúng. Nếu không

có thông tin về vị trí, mô hình có thể hiểu nhầm và sinh ra chuỗi văn bản sai như "Nội Hà".

Ngoài phương pháp mã hóa vị trí sử dụng hàm sin và cos, còn có một số phương pháp khác được đề xuất để cải thiện hiệu quả của mã hóa vị trí trong mô hình Transformer. Một trong những phương pháp đó là Mã hóa vị trí học được (Learnable Positional Encoding), trong đó các vector mã hóa vị trí được học trong quá trình huấn luyện, giúp mô hình tự điều chỉnh và tối ưu mã hóa vị trí sao cho phù hợp với dữ liệu. Thêm vào đó, Mã hóa vị trí xoay (Rotary Positional Encoding - RoPE) là một phương pháp sử dụng các phép toán hình học để mã hóa vị trí, giúp mô hình xử lý hiệu quả hơn các mối quan hệ vị trí trong không gian đa chiều. Cuối cùng, Mã hóa vị trí tương đối (Relative Positional Encoding) không chỉ mã hóa vị trí tuyệt đối mà còn mã hóa khoảng cách giữa các phần tử trong chuỗi, giúp mô hình hiểu được mối quan hệ giữa các phần tử dựa trên khoảng cách giữa chúng.

2.2.2.5 Quá trình sinh văn bản trong Transformer decoder

Bước 1: Khởi tạo chuỗi đầu vào, quá trình sinh văn bản bắt đầu với một token đặc biệt, thường là <SOS> (Start of sequence), nhằm đánh dấu điểm bắt đầu của chuỗi đầu ra.

Bước 2: Nhúng token và kết hợp với đặc trưng hình ảnh chính là các token được chuyển thành vector nhúng (embedding) và được kết hợp với các đặc trưng hình ảnh đã được trích xuất từ CNN. Các đặc trưng này cung cấp thông tin ngữ nghĩa từ hình ảnh nhằm hỗ trợ quá trình sinh văn bản.

Bước 3: Áp dụng Masked Multi-Head Self-Attention, các cơ chế self-attention cho phép bộ giải mã tập trung vào các phần khác nhau trong chuỗi đầu ra đã được sinh ra một phần. Việc sử dụng mặt nạ (masking) đảm bảo rằng khi dự đoán ký tự tại một ví trí t, mô hình chỉ có thể chú ý đến các ký tự từ vị trí 1 đến t - 1, ngăn chặn việc "nhìn" vào tương lai.

Bước 4: Áp dụng Multi-Head Cross-Attention, Cơ chế cross-attention cho phép bộ giải mã tập trung vào các phần khác nhau của bản đồ đặc trưng hình ảnh (đầu ra từ CNN). Tại mỗi bước giải mã, bộ giải mã sử dụng trạng thái hiện tại của nó (từ self-attention) làm truy vấn (query) để "hỏi" bản đồ đặc trưng ảnh (đóng vai trò là khóa - key và giá trị - value), xác định xem phần nào của ảnh là quan trọng nhất để dự đoán ký tự tiếp theo.

Bước 5: Xử lý qua FFN, sau các lớp attention, đầu ra được đưa qua một mạng nơ-ron truyền thẳng (feed-forward network) đơn giản để xử lý thêm.

Bước 6: Lớp tuyến tính kết hợp Softmax, đầu ra từ khối giải mã cuối cùng được đưa qua một lớp tuyến tính (linear layer) và hàm softmax để tạo ra phân phối xác suất trên tập các ký tự có thể có. Ký tự có xác suất cao nhất thường được chọn làm ký tự dự đoán tiếp theo.

Cuối cùng ký tự được sinh ra sẽ được thêm vào chuỗi và quá trình sẽ tiếp tục cho đến khi mô hình sinh ra token kết thúc (<EOS> - End of Sequence) hay đạt đến một độ dài tối đa cho phép.

2.3 Cơ chế Attention giữa đặc trưng ảnh và trình giải mã văn bản

Trong bài toán Nhận dạng Ký tự Quang học (OCR), cơ chế chú ý, đặc biệt là chú ý chéo (cross-attention), giữa đặc trưng hình ảnh và bộ giải mã văn bản đóng vai trò quan trọng trong việc kết nối thông tin không gian từ ảnh với chuỗi văn bản đầu ra. Cơ chế này cho phép mô hình học cách liên kết động giữa các vùng trong ảnh và các ký tự trong chuỗi văn bản, từ đó cải thiện độ chính xác của việc nhận dạng văn bản.

Có thể nói cơ chế cross-attention hoạt động bằng cách tính toán một tập hợp các trọng số chú ý (attention weights). Các trọng số này cho biết mức độ quan trọng của từng phần (hoặc pixel, hoặc vùng đặc trưng) trong bản đồ đặc trưng CNN đối với việc dự đoán ký tự hiện tại trong bộ giải mã. Cụ thể, tại mỗi bước giải mã, trạng thái ẩn của bộ giải mã (query) sẽ được so sánh với tất cả các vector đặc trưng trong bản đồ đặc trưng CNN (keys). Mức độ tương đồng (thường được tính bằng tích vô hướng - dot product) sẽ xác định trọng số chú ý. Các vector đặc trưng CNN (values) sau đó được tổng hợp lại dựa

trên các trọng số này để tạo ra một vector ngữ cảnh (context vector), tập trung vào các thông tin hình ảnh liên quan nhất.

Còn đối với đặc trưng hình ảnh có cấu trúc 2D, cơ chế attention có thể được thiết kế để hoạt động trực tiếp trên lưới đặc trưng 2D này (2D Attention). Điều này cho phép mô hình tập trung vào các vùng không gian cụ thể trên ảnh khi sinh ra từng ký tự, thay vì chỉ làm việc với một vector đặc trưng phẳng hóa. Việc áp dụng attention 2D giúp mô hình nhận diện tốt hơn các văn bản có bố cục phức tạp, chẳng hạn như văn bản cong, nghiêng hoặc có khoảng cách giữa các ký tự lớn.

Cơ chế Attention trong OCR cho phép bộ giải mã tập trung có chọn lọc vào các phần liên quan của ảnh tại mỗi bước sinh ký tự, giúp mô hình xử lý tốt hơn các vấn đề như văn bản bị nghiêng, cong, hoặc có bố cục phức tạp. Có thể giảm đi sự phụ thuộc vào việc căn chỉnh cứng nhắc giữa ảnh và văn bản, làm cho mô hình linh hoạt và chính xác hơn, đặc biệt đối với văn bản trong cảnh tự nhiên vốn rất đa dạng và không đều.

2.4 Ưu và nhược điểm mô hình CNN-Transformer cho OCR

Ưu điểm: Đầu tiên, mô hình này có hiệu suất cao khi xử lý văn bản phức tạp, đặc biệt là với văn bản bị biến dạng, cong vẹo hoặc có bố cục không đều. Cơ chế attention của Transformer giúp mô hình tập trung vào các vùng đặc trưng quan trọng mà không bị ràng buộc bởi vị trí của chúng trong chuỗi đặc trưng, từ đó cải thiện độ chính xác. Thứ hai, Transformer vượt trội trong việc mô hình hóa phụ thuộc xa, cho phép nắm bắt các mối quan hệ ngữ cảnh giữa các ký tự hoặc từ ở khoảng cách xa, điều mà các mô hình RNN hoặc LSTM khó thực hiện. Bên cạnh đó, khả năng song song hóa của Transformer giúp giảm đáng kể thời gian huấn luyện, đặc biệt khi sử dụng phần cứng GPU, giúp tăng hiệu suất và tiết kiệm tài nguyên tính toán. Kiến trúc này cũng khá linh hoạt, cho phép tối ưu hóa cấu hình của các thành phần như CNN backbone hoặc Transformer Decoder để phù hợp với các yêu cầu cụ thể về độ chính xác và hiệu suất.

Nhược điểm: Cả CNN và Transformer đều yêu cầu tài nguyên tính toán lớn, đặc biệt là bộ nhớ GPU, và cơ chế self-attention trong Transformer có độ phức tạp bậc hai theo

độ dài chuỗi, điều này có thể gây khó khăn khi xử lý ảnh lớn hoặc chuỗi đặc trưng dài. Hơn nữa, mô hình học sâu này đòi hỏi một lượng lớn dữ liệu huấn luyện đã được gán nhãn để đạt hiệu suất tối ưu, điều này có thể tạo ra khó khăn trong việc thu thập và xử lý dữ liệu OCR, đặc biệt là cho văn bản trong ảnh tự nhiên. Việc diễn giải mô hình, đặc biệt là cơ chế attention, vẫn là một thách thức, mặc dù các kỹ thuật trực quan hóa có thể giúp giải thích một phần quá trình quyết định của mô hình. Cuối cùng, khi xử lý văn bản cực kỳ dài, mặc dù Transformer có thể xử lý phụ thuộc xa tốt hơn RNN, nhưng độ phức tạp tính toán và yêu cầu bộ nhớ vẫn là một vấn đề cần giải quyết.

CHƯƠNG 3 – THỰC NGHIỆM

3.1 Tập dữ liệu và phương pháp đánh giá

VinText là một tập dữ liệu nhân tạo được xây dựng nhằm phục vụ hai tác vụ quan trọng trong thị giác máy tính: phát hiện văn bản (text detection) và nhận dạng văn bản (text recognition) trong môi trường tự nhiên. Đây là tập dữ liệu đầu tiên được thiết kế chuyên biệt cho tiếng Việt trong lĩnh vực nhận diện văn bản tự động, giúp giải quyết một khoảng trống lớn trong nghiên cứu OCR đa ngôn ngữ, đặc biệt đối với các ngôn ngữ có dấu phức tạp như tiếng Việt.

Một trong những thách thức lớn đối với OCR tiếng Việt là sự đa dạng về hình thức của chữ viết, bao gồm chữ in hoa, in thường, chữ hoa đầu câu, chữ có dấu, và chữ ghép. Hơn nữa, các bối cảnh thực tế như biển hiệu, poster, bìa sách, đường phố tại Việt Nam thường có nền phức tạp, nhiều nhiễu ảnh, và góc nhìn khó. Để giải quyết những vấn đề này, VinText được xây dựng dựa trên phương pháp tổng hợp hình ảnh văn bản nhân tạo (synthetic scene text), giúp tạo ra một tập dữ liệu lớn, đa dạng và có chất lượng gán nhãn hoàn hảo mà chi phí thu thập thấp hơn rất nhiều so với phương pháp gán nhãn thủ công.

Một điểm nổi bật của VinText là nó không chỉ cung cấp gán nhãn ở mức hộp chứa (bounding box) cho từng từ trong ảnh mà còn đi kèm chuỗi ký tự chính xác của từ đó. Điều này cho phép VinText được sử dụng đồng thời cho hai tác vụ quan trọng: phát hiện văn bản (localization) và nhận dạng văn bản (recognition), giúp phát triển các hệ thống end-to-end mà không cần nhiều nguồn dữ liệu khác nhau.

Về mặt quy mô và đa dạng, VinText vượt trội so với các tập dữ liệu tiếng Việt trước đây vốn nhỏ lẻ và khó tiếp cận. Tập dữ liệu này bao phủ hơn 2000 phông chữ, nhiều kiểu phối cảnh, và cả những trường hợp khó như văn bản cong, bị che khuất hoặc mờ. Ngoài ra, việc sử dụng tiếng Việt — một ngôn ngữ giàu dấu câu và dấu thanh — mang đến thách thức thú vị mà các tập dữ liệu tiếng Anh hoặc Latin không có, đồng thời giúp đánh giá và thúc đẩy sự phát triển của các thuật toán OCR đa ngôn ngữ.

Dữ liệu thô bao gồm các ảnh và nhãn tương ứng với từng ảnh, trong đó nhãn của ảnh là tập hợp các dòng, mỗi dòng được viết theo nguyên tắc <x1, y1, x2, y2, x3, y3, x4, y4 SCRIPT>. Trong đó, x1, y1, x2, y2, x3, y3, x4, y4 đại diện cho tọa độ của 4 điểm của vùng bao quanh chữ, còn SCRIPT là văn bản mà vùng ảnh đó thể hiện. Tập dữ liệu hướng đến tác vụ xử lý nhận dạng ký tự quang học cấp kí tự, giúp nhận dạng và trả về kết quả đúng trên từng ký tự.

Tuy nhiên, nếu truyền trực tiếp các kí tự này vào mô hình để làm đầu vào, mô hình sẽ khó để hiểu được do các mô hình chỉ hoạt động hiệu quả trên dữ liệu dạng số. Vì thế tokenizer hóa là bước chuẩn bị cần thiết để giúp dữ liệu dễ hiểu hơn dưới góc nhìn của mô hình. Để tokenizer hóa, bước đầu tiên cần thực hiện là xây dựng tập từ vựng (vocab) để mô hình biết sẽ có những kí tự nào xuất hiện trong tập dữ liệu. Do tập dữ liệu đã phân sẵn thành các từ nên chỉ cần thêm một bước là tạo token (thường là dạng số) tương ứng với từng từ phân biệt. Tiếp đến, nhóm tiến hành tokenizer hóa dữ liệu bằng cách tạo một vec tơ nhúng là một tập hợp các số tương ứng với từng kí tự trong nhãn, các kí tự lạ không xuất hiện trong tập từ vựng sẽ được thay thế bằng <UNK> và kết thúc vec tơ là <EOS>. Ngoài ra, để đồng bộ độ dài của các từ, những từ không đủ độ dài kí tự so với độ dài tối đa đã quy định sẽ được thêm các kí tự <PAD> vào để lắp đầy những chỗ còn thiếu.

Mô hình sẽ được đánh giá dựa trên ba thang đo là CER (Character Error Rate), tỉ lệ dự đoán đúng trên ký tự và tỉ lệ dự đoán đúng trên từ. Cụ thể hơn:

CER là chỉ số đo tỷ lệ lỗi giữa ký tự của chuỗi dự đoán so với chuỗi chuẩn, thể hiện số thao tác cần thiết để chỉnh sửa (thêm, xóa, sửa ký tự) chuỗi dự đoán thành chuỗi chuẩn.

$$CER = \frac{S + D + I}{N}$$

Trong đó:

- S: số ký tự sai
- D: số ký tự bị xóa

- I: số ký tự bị chèn thêm
- N: tổng số ký tự trong ground truth

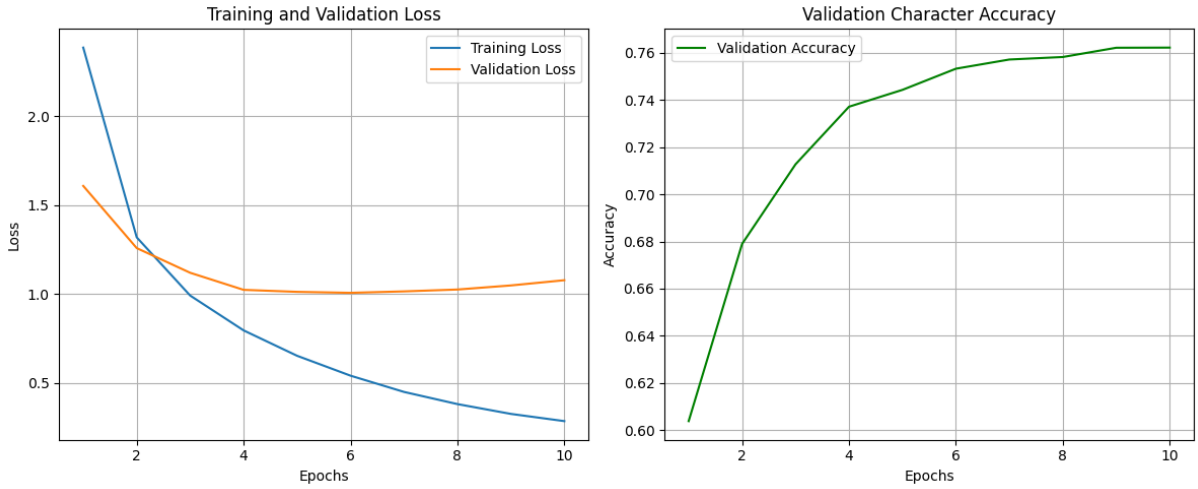
Áp dụng vào mô hình, nhóm sử dụng khoảng cách chỉnh sửa Leveneshtein để mô phỏng tương tự công thức trên, với ý nghĩa càng gần 0 thì mô hình càng tốt.

Đối với tỉ lệ dự đoán đúng trên từ và tỉ lệ dự đoán đúng trên ký tự, đây đơn thuần là độ dự đoán chính xác của mô hình dựa vào tỉ lệ số dự đoán đúng hoàn toàn trên trong tổng số dự đoán, ứng với việc tỉ lệ dự đoán đúng trên từ là chỉ số thể hiện tỉ lệ từ dự đoán đúng trong một vùng ảnh, còn tỉ lệ dự đoán đúng trên từ là tỉ lệ ký tự dự đoán đúng trong một chuỗi ký tự trong một vùng ảnh nhất định. Tỉ lệ dự đoán đúng trên từ có nhược điểm là khá nhạy cảm với lỗi nhỏ, khi mà chỉ cần một ký tự trong chuỗi dự đoán sai so với chuỗi chuẩn thì sẽ bị coi là sai, ảnh hưởng lớn đến độ chính xác dự đoán. Vì thế nhóm đã đánh giá mô hình thêm thông qua chỉ số tỉ lệ dự đoán trên ký tự để thể hiện rõ hơn mô hình có thực sự tệ khi tỉ lệ dự đoán chính xác trên từ không cao không. Các chỉ số này đánh giá mô hình trong khoảng từ [0;1] và càng gần 1 là càng tốt.

3.2 Kết quả thực nghiệm

Để đánh giá hiệu quả của mô hình CNN-Transformer trên tập dữ liệu VinText, nhóm đã tiến hành huấn luyện mô hình trên GPU P100 16GB với các thông số: batch size là 32, learning rate là 1×10^{-4}, vocab size là 231, và số vòng huấn luyện là 10 epochs. Kết quả huấn luyện và kiểm thử được ghi nhận thông qua các biểu đồ đánh giá quá trình huấn luyện và số liệu hiệu suất trên hai tập dữ liệu: tập Test và tập Unseen Test.

Quá trình huấn luyện



Hình 3.1 Biểu đồ quá trình huấn luyện

Biểu đồ bên trái cho thấy hàm mất mát huấn luyện (Training Loss) và kiểm định (Validation Loss) đều giảm đều đặn qua các epoch. Cụ thể, Training Loss giảm từ khoảng 2.2 xuống dưới 0.5, trong khi Validation Loss giảm từ khoảng 1.5 xuống dưới 1.0. Điều này cho thấy mô hình đã học tốt trên tập huấn luyện và có khả năng tổng quát hóa trên tập kiểm định, với khoảng cách giữa hai đường cong không quá lớn, báo hiệu không có dấu hiệu overfitting nghiêm trọng. Tuy nhiên, Validation Loss có xu hướng giảm chậm hơn sau epoch thứ 8, cho thấy mô hình có thể đã tiến gần đến điểm hội tụ.

Biểu đồ bên phải thể hiện độ chính xác ký tự trên tập kiểm định (Validation Character Accuracy) tăng từ khoảng 0.60 lên khoảng 0.76 sau 10 epochs. Sự tăng trưởng này khá ổn định, đặc biệt trong 6 epoch đầu tiên, nhưng bắt đầu chậm lại từ epoch 8. Điều này cho thấy mô hình đã đạt được hiệu suất khá tốt, nhưng có thể cần thêm các kỹ thuật tối ưu hóa hoặc tăng số epoch để cải thiện thêm độ chính xác.

Hiệu suất trên tập Test và Unseen Test

Chỉ số	Tập Test	Tập Unseen Test
Character Error Rate (CER)	35.94%	30.84%
Word Accuracy	49.46%	53.46%
Character Accuracy	65.59%	70.03%
Average Edit Distance	1.38 ký tự/từ	1.22 ký tự/từ
Sample Count	7,220	10,086

Bảng 3. 1 Tổng hợp hiệu suất của mô hình trên tập Test và tập Unseen Test.

Kết quả cho thấy mô hình đạt hiệu suất tốt hơn trên tập Unseen Test so với tập Test, điều này khá bất ngờ vì tập Unseen Test thường chứa các mẫu dữ liệu chưa từng thấy, có thể đa dạng và phức tạp hơn. Cụ thể, CER giảm từ 35.94% trên tập Test xuống 30.84% trên tập Unseen Test, trong khi Character Accuracy tăng từ 65.59% lên 70.03%. Tương tự, Word Accuracy cũng tăng từ 49.46% lên 53.46%, và khoảng cách chỉnh sửa trung bình (Average Edit Distance) giảm từ 1.38 xuống 1.22 ký tự/từ. Sự cải thiện này có thể được giải thích bởi sự đa dạng của tập Unseen Test, với số lượng mẫu lớn hơn (10,086 so với 7,220), giúp mô hình thể hiện tốt hơn khả năng tổng quát hóa trên các trường hợp mới.

Phân tích lỗi

Các lỗi chủ yếu tập trung vào việc nhận dạng dấu thanh và dấu phụ tiếng Việt, đặc biệt là sự nhầm lẫn giữa các ký tự tương tự như “u” và “ư”, hoặc “o” và “ơ”. Ngoài ra, mô hình gặp khó khăn khi xử lý văn bản dài và ảnh có chất lượng kém (nhiễu, mờ, hoặc ánh sáng không đồng đều). Điều này phù hợp với đặc điểm của tập dữ liệu VinText, vốn bao gồm các trường hợp phức tạp như văn bản cong, bị che khuất, hoặc có nền lộn xộn.

Hình 3.2 bên dưới minh họa các kết quả đầu ra trên các mẫu dữ liệu là biển hiệu màu xanh với chữ trắng. Mặc dù không có phân tích chi tiết về các lỗi trong báo cáo gốc, nhưng dựa trên các hình ảnh này, có thể suy ra rằng mô hình đã nhận dạng được văn bản

trên các biển hiệu, nhưng có thể gặp lỗi ở các ký tự có dấu hoặc trong các trường hợp văn bản nghiêng, cong, hoặc bị ảnh hưởng bởi ánh sáng.

Hình 3.2 Kết quả của đầu ra của mô hình khi thử với một mẫu dữ liệu

3.3 Kết luận

Nguyên lý hoạt động của kiến trúc kết hợp mạng nơ-ron Tích chập (CNN) và Transformer Decoder trong lĩnh vực nhận dạng ký tự quang học (OCR), đây là cấu trúc kết hợp khả năng mạnh mẽ của CNN trong việc trích xuất đặc trưng hình ảnh và khả năng giải mã chuỗi văn bản của Transformer Decoder thông qua cơ chế attention tinh vi. Mô hình này không chỉ giải quyết tốt các bài toán văn bản phức tạp trong ảnh tự nhiên mà còn đảm bảo khả năng xử lý phụ thuộc tuần tự và ngữ cảnh một cách hiệu quả.

CNN đóng vai trò là bộ trích xuất đặc trưng mạnh mẽ, nắm bắt các thông tin không gian trong ảnh đầu vào. Sau khi các đặc trưng hình ảnh được trích xuất, Transformer Decoder tiếp nhận chúng, sử dụng cơ chế cross-attention để tạo ra văn bản từ chuỗi đặc trưng hình ảnh. Sự kết hợp này giúp mô hình nhận thức được các mối quan hệ phức tạp giữa các ký tự và từ trong chuỗi, giúp cải thiện hiệu suất nhận dạng, đặc biệt trong môi trường ảnh phức tạp.

Kiến trúc CNN-Transformer đã chứng minh hiệu quả trong việc nhận dạng văn bản tiếng Việt từ ảnh tự nhiên, đặc biệt trên tập dữ liệu VinText với các đặc trưng phức tạp như dấu thanh, văn bản cong, và nền lộn xộn. Kết quả thực nghiệm cho thấy mô hình đạt độ chính xác ký tự khá tốt (65.59% trên tập Test và 70.03% trên tập Unseen Test), nhưng độ chính xác từ chỉ đạt khoảng 49-53%. Khoảng cách giữa độ chính xác ký tự và từ cho thấy mô hình thường mắc lỗi ở một vài ký tự trong từ, dẫn đến việc từ đó bị đánh giá là

sai hoàn toàn. Điều này đặc biệt rõ ràng với các từ tiếng Việt, nơi một lỗi nhỏ ở dấu thanh (ví dụ: “hà” thành “hà”) có thể làm thay đổi ý nghĩa từ.

Hơn nữa, sự cải thiện hiệu suất trên tập Unseen Test so với tập Test là một điểm sáng, cho thấy mô hình có khả năng tổng quát hóa tốt trên các mẫu dữ liệu mới. Điều này có thể được giải thích bởi sự đa dạng của tập Unseen Test, cũng như việc mô hình đã được huấn luyện để xử lý các trường hợp phức tạp trong tập VinText. Tuy nhiên, CER vẫn ở mức khá cao (30.84% trên tập Unseen Test), cho thấy vẫn còn nhiều lỗi cần khắc phục, đặc biệt là với các ký tự có dấu và văn bản dài.

Dựa trên kết quả và phân tích lỗi, có thể xem xét một số hướng cải thiện như sau:

1. **Tăng cường xử lý dấu thanh và ký tự tương tự**:
   - **Data augmentation**: Tăng số lượng mẫu dữ liệu chứa các ký tự dễ nhầm lẫn (u/ư, o/ơ) bằng cách sử dụng các kỹ thuật biến đổi ảnh như thay đổi ánh sáng, thêm nhiễu, hoặc xoay văn bản.
   - **Cơ chế attention đặc biệt cho dấu**: Thêm một lớp attention phụ để tập trung vào việc nhận dạng dấu thanh, có thể kết hợp với các kỹ thuật như dictionary-guided correction để sửa lỗi sau khi sinh văn bản.
2. **Tối ưu hóa kiến trúc mô hình**:
   - **Thử nghiệm backbone CNN mạnh hơn**: Thay vì ResNet18 có thể sử dụng ResNet50 hoặc EfficientNet để trích xuất đặc trưng hình ảnh tốt hơn, đặc biệt với các ảnh có chất lượng kém.
   - **Điều chỉnh Transformer Decoder**: Tăng số lớp hoặc số đầu attention trong Transformer Decoder để cải thiện khả năng xử lý văn bản dài và các phụ thuộc phức tạp. Ngoài ra, có thể thử nghiệm các biến thể như RoPE (Rotary Positional Embedding) để mã hóa vị trí tốt hơn, như đã đề cập trong Chương 1.

3. **Cải thiện quá trình huấn luyện**:
   - **Tăng số epoch**: Với biểu đồ huấn luyện cho thấy Validation Loss và Character Accuracy chưa hoàn toàn hội tụ sau 10 epochs, nhóm nên thử tăng số vòng huấn luyện (ví dụ: lên 20 epochs) để cải thiện hiệu suất.
   - **Fine-tune hyperparameters**: Điều chỉnh learning rate (ví dụ: thử các giá trị nhỏ hơn như 5×10^{-5}) hoặc sử dụng scheduler để giảm learning rate theo thời gian, giúp mô hình hội tụ tốt hơn.
   - **Thêm regularization**: Áp dụng các kỹ thuật như dropout hoặc weight decay để giảm nguy cơ overfitting, đặc biệt khi Validation Loss có dấu hiệu chững lại.
4. **Xử lý văn bản dài và ảnh chất lượng thấp**:
   - **Sliding Window Attention**: Áp dụng kỹ thuật Sliding Window Attention (như đã đề cập trong Chương 1) để giảm độ phức tạp tính toán khi xử lý chuỗi đặc trưng dài từ ảnh lớn.
   - **Tiền xử lý ảnh**: Tăng cường các bước tiền xử lý ảnh (điều chỉnh độ sáng, giảm nhiễu, căn chỉnh văn bản) để cải thiện chất lượng đầu vào cho mô hình.

Ngoài các cải thiện trên, một số hướng nghiên cứu tiềm năng có thể được xem xét để nâng cao hiệu quả của mô hình CNN-Transformer trong bài toán OCR tiếng Việt:

- **Học tự giám sát hoặc bán giám sát**: Sử dụng các kỹ thuật như contrastive learning hoặc masked language modeling để tận dụng dữ liệu không gán nhãn, giảm phụ thuộc vào dữ liệu gán nhãn thủ công vốn tốn kém với tiếng Việt.
- **Tích hợp thông tin đa phương thức**: Kết hợp thông tin ngữ cảnh từ từ điển tiếng Việt hoặc các nguồn dữ liệu khác (ví dụ: âm thanh, ngữ nghĩa) để cải thiện độ chính xác, đặc biệt trong việc sửa lỗi dấu thanh.
- **Triển khai trên thiết bị di động**: Tối ưu hóa mô hình để triển khai trên các thiết bị có tài nguyên hạn chế, chẳng hạn bằng cách sử dụng các kỹ thuật như

pruning, quantization, hoặc chuyển sang các backbone nhẹ hơn như MobileNet.

Kiến trúc CNN-Transformer đã thể hiện tiềm năng lớn trong việc giải quyết bài toán OCR tiếng Việt trên tập dữ liệu VinText, với khả năng xử lý văn bản phức tạp trong các điều kiện tự nhiên. Tuy nhiên, các thách thức liên quan đến nhận dạng dấu thanh, văn bản dài, và ảnh chất lượng thấp vẫn cần được giải quyết để nâng cao hiệu suất. Những cải tiến đề xuất, cùng với các hướng nghiên cứu tương lai, hứa hẹn sẽ giúp mô hình đạt được độ chính xác cao hơn và mở rộng ứng dụng trong các kịch bản thực tế, từ số hóa tài liệu đến nhận dạng văn bản trên biển hiệu và hóa đơn.

TÀI LIỆU THAM KHẢO

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

2. Shazeer, N. (2019). Fast Transformer Decoding: One Write-Head is All You Need. *arXiv preprint arXiv:1911.02150*.

3. Ainslie, J., Engers, F., Ontanon, S., Pueyo, P., Teodorescu, A., & Wang, T. (2023). GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. *arXiv preprint arXiv:2305.13245*.

4. Dao, T., Fu, D. Y., Ermon, S., Rudra, A., & Ré, C. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems*, 35.

5. Dao, T., Fu, D. Y., Saab, K. K., Thomas, A., Rudra, A., & Ré, C. (2023). FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *arXiv preprint arXiv:2307.08691*.

6. Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. *International Conference on Machine Learning*.

7. Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating Long Sequences with Sparse Transformers. *arXiv preprint arXiv:1904.10509*.

8. Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., & Liu, Y. (2021). RoFormer: Enhanced Transformer with Rotary Position Embedding. *arXiv preprint arXiv:2104.09864*.

9. Biderman, S., Black, S., Foster, C., Gao, L., Hallahan, E., He, H., Wang, B., & Wang, P. (2021). Rotary Embeddings: A Relative Revolution. *EleutherAI Blog*.

10. Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The Long-Document Transformer. *arXiv preprint arXiv:2004.05150*.

11. Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L., & Ahmed, A. (2020). Big Bird: Transformers for Longer Sequences. *Advances in Neural Information Processing Systems*, 33.

12. Kitaev, N., Kaiser, Ł., & Levskaya, A. (2020). Reformer: The Efficient Transformer. *International Conference on Learning Representations*.

13. Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., Belanger, D., Colwell, L., & Weller, A. (2021). Rethinking Attention with Performers. *International Conference on Learning Representations*.

14. Press, O., Smith, N. A., & Lewis, M. (2021). Train Short, Test Long: Attention with Linear Biases Enables Input Length Extrapolation. *arXiv preprint arXiv:2108.12409*.

15. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and Efficient Foundation Language Models. *arXiv preprint arXiv:2302.13971*.

16. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., & others. (2023). LLaMA 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.

17. Jiang, D., Mosallanezhad, A., Cao, Y., Bisk, Y., Xu, H., Riedl, M., & Zhu, Y. (2023). YaRN: Efficient Context Window Extension of Large Language Models. *arXiv preprint arXiv:2309.00071*.

18. Peng, B., Quach, V., Tay, Y., Fu, X., Dehghani, M., Kalyan, A., Chung, H. W., Tran, T., Baevski, A., Auli, M., & others. (2023). RWKV: Reinventing RNNs for the Transformer Era. *arXiv preprint arXiv:2305.13048*.

19. Gu, A., & Dao, T. (2022). FlashAttention: Fast Attention with Low Memory Footprint. *Stanford University Blog*.

20. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., & others. (2022). PaLM: Scaling Language Modeling with Pathways. *arXiv preprint arXiv:2204.02311*.

21. Jiang, Y., Sablayrolles, A., Mensch, A., Bamford, C., Singh, D. J., Bressand, Y., Passos, A., Jegou, H., Saulnier, N., Sharma, P. & others. (2023). Mistral 7B. *arXiv preprint arXiv:2310.06825*.

22. Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Debbah, M., Goffinet, E., Heslow, D., Launay, J., Malartic, Q., & others. (2023). Falcon: Scaling Language Models to 180 Billion Parameters. *arXiv preprint arXiv:2311.16867*.

23. Raschka, S. (2023). Self-Attention from Scratch. *Sebastian Raschka's Blog*.

24. Phuong, M., & Hutter, M. (2022). Formal Algorithms for Transformers. *arXiv preprint arXiv:2207.09238*.

25. Elhage, N., Nanda, N., Olsson, C., Henighan, T., Joseph, N., Mann, B., Askell, A., Bai, Y., Chen, A., Conerly, T., & others. (2021). A Mathematical Framework for Transformer Circuits. *Anthropic Research*.

26. Paleti, N. C. (2024, November 21). Positional Encoding Explained: A Deep Dive into Transformer PE. *Medium*.

27. Herath, S. (2024, November 28). The feedforward Network (FFN) in the transformer model. *Medium*.

28. *Cơ chế Attention và mô hình Transformer*. (2023, July 17).