**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**



**BÁO CÁO ĐỒ ÁN**

**MÔN CÁC MÔ HÌNH HỌC SÂU VÀ ỨNG DỤNG**

**Đề tài: All Attention You Need**

GVHD: TS. Nguyễn Vinh Tiệp

Nhóm sinh viên thực hiện:

1. Lê Thanh Dũng MSSV: 230101074

2. Nguyễn Lê Nam Anh MSSV: 230101070

3. Trần Quốc Huy MSSV: 230101048

4. Nguyễn Xuân Trường MSSV: 230104027

Tp. Hồ Chí Minh, 26/06/2024

# GIỚI THIỆU MÔ HÌNH TRANSFORMER

## Giới thiệu

Trong những năm gần đây, lĩnh vực trí tuệ nhân tạo (AI) và học máy (Machine Learning) đã chứng kiến sự phát triển vượt bậc với sự ra đời của nhiều mô hình học sâu (Deep Learning). Các mô hình như Recurrent Neural Networks (RNN) và Convolutional Neural Networks (CNN) đã từng thống trị trong các nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP) và xử lý ảnh. Tuy nhiên, sự xuất hiện của Transformer đã mang đến một cách tiếp cận mới, mang lại hiệu suất vượt trội trong nhiều bài toán phức tạp.

## Định nghĩa

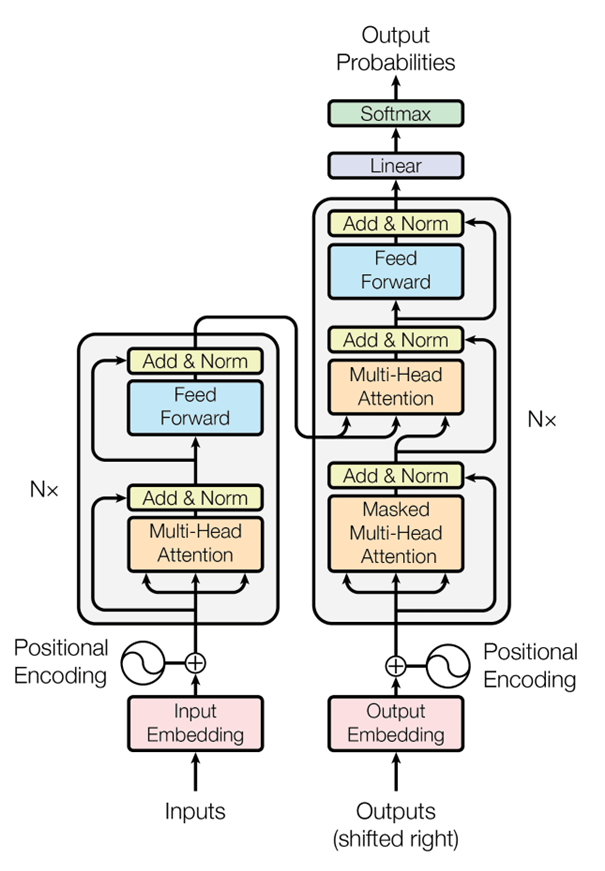
Transformer là một mô hình học sâu được thiết kế để phục vụ giải quyết nhiều bài toán trong xử lý ngôn ngữ và tiếng nói, ví dụ như bài toán dịch tự động, bài toán sinh ngôn ngữ, phân loại, nhận dạng thực thể, nhận dạng tiếng nói, chuyển văn bản thành tiếng nói. Tuy nhiên, khác với RNNs, Transformer không xử lý các phần tử trong một chuỗi một cách tuần tự. Nếu dữ liệu đầu vào là một câu ngôn ngữ tự nhiên, Transformer không cần phải xử lý phần đầu câu trước rồi mới tới phần cuối câu. Do tính năng này, Transformer có thể tận dụng khả năng tính toán song song của GPU và giảm thời gian xử lý đáng kể. Khác với các mô hình RNN và CNN, Transformer không dựa vào cấu trúc tuần tự để xử lý dữ liệu mà sử dụng cơ chế Attention để học và nắm bắt mối quan hệ giữa các từ trong câu.

Cơ chế Attention được dùng để giải quyết một số hạn chế về tính tuần tự cho phép các mô hình tập trung vào các phần liên quan của chuỗi đầu vào mà không phụ thuộc vào vị trí của chúng. Mặc dù thành công, các cơ chế này thường được kết hợp với cấu trúc hồi quy, điều này vẫn đặt ra các hạn chế về xử lý tuần tự.

# KIẾN TRÚC CỦA TRANSFORMER

Kiến trúc Transformer khác biệt bằng cách loại bỏ hoàn toàn hồi quy và tích chập. Nó sử dụng cơ chế tự chú ý để xử lý các phụ thuộc trong chuỗi, cho phép song song hóa hiệu quả hơn. Kiến trúc bao gồm hai phần chính: Encoder (bộ mã hóa) và Decoder (bộ giải mã), mỗi phần bao gồm nhiều lớp (layers).

Cả bộ Encoder và bộ Decoder đều được cấu tạo từ N lớp giống nhau (N=6 trong mô hình cơ sở). Mỗi lớp mã hóa có hai tầng phụ: một cơ chế multi-head self-attention và một position-wise fully connected feed-forward network. Các lớp Decoder bao gồm một tầng phụ bổ sung thực hiện multi-head attention qua đầu ra của bộ Decoder.



Hình 1. The Transformer- model architecture.

## Encoder

Encoder gồm nhiều lớp giống nhau, mỗi lớp bao gồm hai thành phần chính:

* **Multi-Head Self-Attention Mechanism:** Cho phép mô hình tập trung vào các phần khác nhau của câu đầu vào. Cơ chế này bao gồm nhiều đầu Attention (Multi-Head) để mô hình có thể chú ý đến nhiều phần khác nhau của dữ liệu.
* **Feed-Forward Neural Network:** Sau khi qua cơ chế Attention, dữ liệu được truyền qua một mạng nơ-ron truyền thẳng để thực hiện các phép biến đổi phi tuyến.

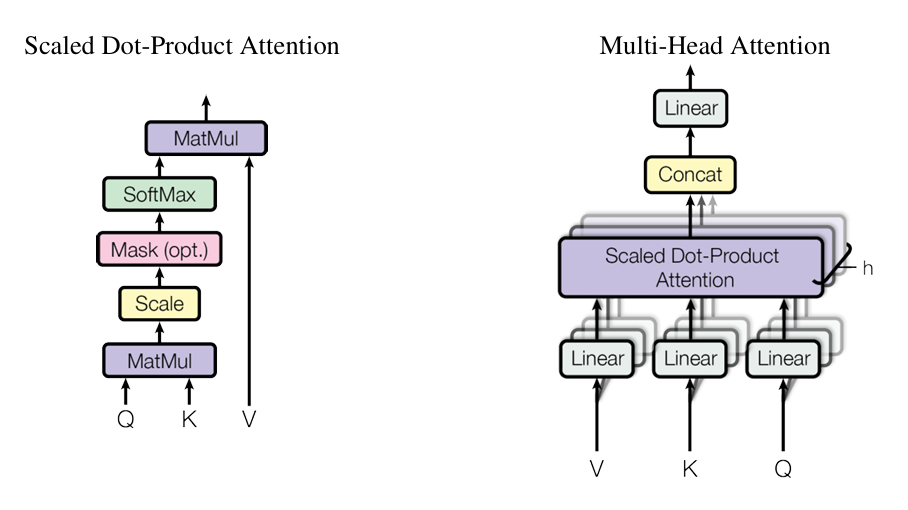
## Decoder

Decoder cũng gồm nhiều lớp giống nhau, với các thành phần tương tự như Encoder nhưng có thêm một lớp Attention:

* **Masked Multi-Head Self-Attention Mechanism:** Tương tự như trong Encoder, nhưng được điều chỉnh để đảm bảo mô hình không nhìn thấy các từ tương lai trong quá trình dự đoán.
* **Multi-Head Attention:** Tập trung vào các phần quan trọng của dữ liệu đầu vào từ Encoder.
* **Feed-Forward Neural Network:** Như trong Encoder, để biến đổi dữ liệu sau khi qua cơ chế Attention.
* **Masked Multi-Head Self-Attention Mechanism:** Tương tự như trong Encoder, nhưng được điều chỉnh để đảm bảo mô hình không nhìn thấy các từ tương lai trong quá trình dự đoán.
  1. **Cơ chế Attention**

Cơ chế Attention là điểm mấu chốt của Transformer, cho phép mô hình xử lý toàn bộ dữ liệu cùng một lúc thay vì tuần tự như trong RNN. Cơ chế này hoạt động như sau:

* **Scaled Dot-Product Attention:** Đầu tiên, tính toán sản phẩm dot giữa Query và Key, sau đó chia cho căn bậc hai của kích thước của Key (để tránh các giá trị quá lớn). Kết quả được đưa qua hàm Softmax để lấy trọng số Attention, sau đó nhân với Value để tạo ra đầu ra.
* **Multi-Head Attention:** Thay vì tính toán Attention một lần duy nhất, Multi-Head Attention thực hiện quá trình này nhiều lần song song với các trọng số khác nhau (head khác nhau), rồi kết hợp lại kết quả.

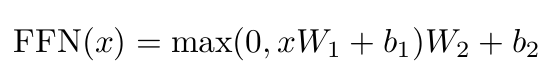


Hình 2. (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel

* 1. **Position-wise Feed-Forward Networks**

Mỗi lớp trong bộ mã hóa và bộ giải mã của Transformer chứa một mạng truyền thẳng được áp dụng cho từng vị trí trong chuỗi một cách độc lập và giống hệt nhau. Mạng này giúp mô hình học các biểu diễn phi tuyến tính và cải thiện khả năng học của mô hình. Mạng truyền thẳng bao gồm hai biến đổi tuyến tính với một kích hoạt ReLU ở giữa:

* **Biến đổi tuyến tính đầu tiên:** Chuyển đổi đầu vào thành không gian ẩn có chiều lớn hơn.
* **Kích hoạt ReLU:** Được áp dụng sau biến đổi tuyến tính đầu tiên.
* **Biến đổi tuyến tính thứ hai:** Chuyển đổi đầu ra từ không gian ẩn về kích thước ban đầu.

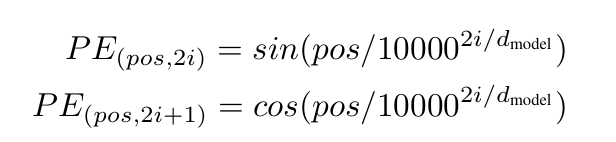


* 1. **Embeddings and Softmax**

Transformer sử dụng các embeddings học được để chuyển đổi các token đầu vào và đầu ra thành các vectơ số thực học được trong quá trình huấn luyện. Sử dụng một phép biến đổi tuyến tính học được và hàm softmax để chuyển đổi đầu ra của bộ giải mã thành các xác suất dự đoán cho token tiếp theo. Trọng số của các lớp embedding và phép biến đổi tuyến tính trước softmax được chia sẻ giữa các lớp embedding đầu vào và đầu ra. Điều này giúp cải thiện hiệu quả huấn luyện và giảm số lượng tham số cần học.

* 1. **Positional Encoding**

Do Transformer không sử dụng hồi quy hoặc tích chập, và nó cần có thông tin về vị trí các token trong chuỗi để mô hình có thể sử dụng thứ tự của các token. Mã hóa vị trí sử dụng các hàm sin và cos với các tần số khác nhau để mã hóa vị trí của các token. Và các mã hóa vị trí có cùng chiều với các embeddings, cho phép chúng được cộng vào embeddings đầu vào. Chúng giúp mô hình Transformer hiểu và xử lý được thứ tự và vị trí của các token trong chuỗi đầu vào, điều mà các mạng hồi quy và tích chập truyền thống thực hiện tự nhiên thông qua cấu trúc của chúng.



# SO SÁNH TRANSFORMER VỚI RNN VÀ CNN

* 1. **Recurrent Neural Networks (RNN)**
* **Ưu điểm:** RNN thích hợp cho các bài toán liên quan đến dữ liệu tuần tự như dịch máy, dự đoán chuỗi thời gian và nhận dạng giọng nói nhờ khả năng ghi nhớ các trạng thái trước đó.
* **Nhược điểm:** RNN gặp vấn đề với các chuỗi dữ liệu dài do gradient biến mất (vanishing gradient) và gradient bùng nổ (exploding gradient). Việc tính toán tuần tự cũng làm giảm tốc độ huấn luyện.
  1. **Convolutional Neural Networks (CNN)**
* **Ưu điểm:** CNN nổi bật trong xử lý dữ liệu hình ảnh nhờ khả năng phát hiện các đặc trưng cục bộ qua các lớp convolution. CNN cũng được sử dụng trong một số bài toán NLP để phát hiện các mẫu đặc trưng.
* **Nhược điểm:** CNN không phù hợp lắm cho việc xử lý các chuỗi dữ liệu dài hoặc có mối quan hệ phức tạp giữa các phần tử không liền kề.
  1. **Transformer**
* **Ưu điểm:** Transformer xử lý các chuỗi dữ liệu dài một cách hiệu quả nhờ cơ chế Attention. Không cần xử lý tuần tự, Transformer có thể song song hóa quá trình tính toán, giúp tăng tốc độ huấn luyện. Mô hình này cũng nắm bắt được mối quan hệ dài hạn trong dữ liệu một cách xuất sắc.
* **Nhược điểm:** Yêu cầu tài nguyên tính toán và bộ nhớ lớn do số lượng tham số và độ phức tạp của cơ chế Attention.

# KẾT LUẬN

Transformer đã đem đến một bước tiến lớn trong lĩnh vực học sâu và xử lý ngôn ngữ tự nhiên. So với các mô hình trước đó như RNN và CNN, Transformer không chỉ cải thiện hiệu suất mà còn mở ra nhiều cơ hội mới cho các ứng dụng trí tuệ nhân tạo. Tuy nhiên, việc áp dụng Transformer cũng đòi hỏi tài nguyên tính toán lớn và cần phải được tối ưu hóa để phù hợp với từng bài toán cụ thể.

Tương lai của nghiên cứu và ứng dụng Transformer rất hứa hẹn, với khả năng tiếp tục cải thiện và mở rộng trong nhiều lĩnh vực khác nhau.

# 

# TÀI LIỆU THAM KHẢO

1. <https://www.ibm.com/topics/transformer-model>
2. <https://baiblanc.github.io/2020/06/21/RNN-vs-CNN-vs-Transformer/>
3. Britz, D., Goldie, A., Luong, M.-T., & Le, Q. V. (2017). Massive exploration of neural machine translation architectures. arXiv preprint arXiv:1703.03906.
4. Lei Ba, J., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.
5. Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.