Report paper: Attention Is All You Need

Giáp Ngọc Hiệu

**Tóm tắt nội dung**

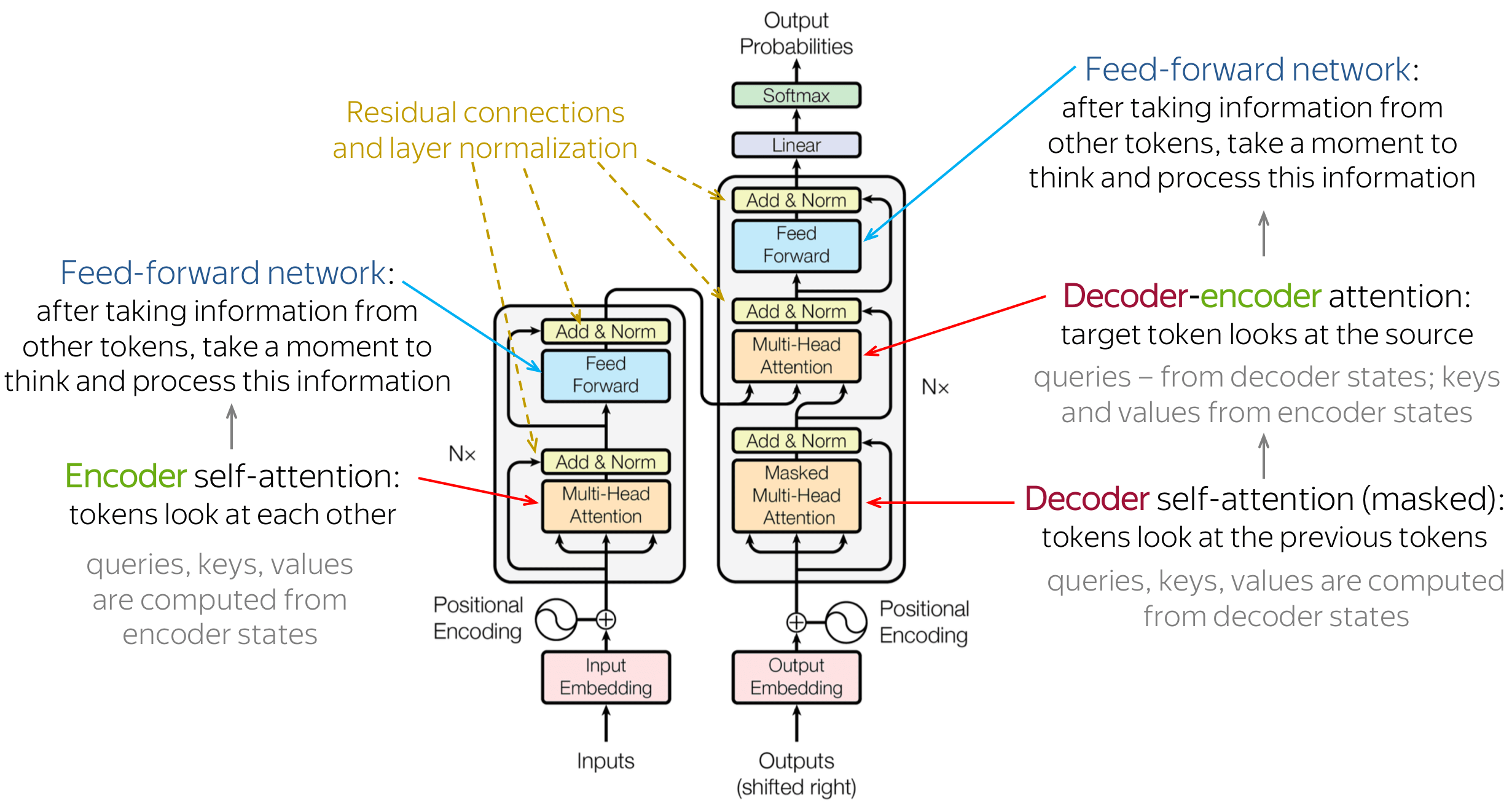
Bài report này là phần tóm tắt lại ý tưởng, nội dung chính, những điều làm được của paper

Attention Is All You Need

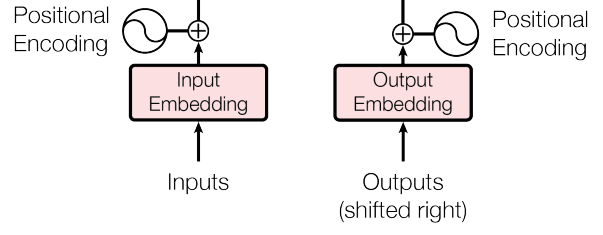
1. **Ý tưởng**

Các sequence transduction models tốt nhất hiện tại dựa trên các mạng recurrent hoặc convolutional phức tạp bao gồm encoder và decoder. Các mô hình hoạt động tốt nhất cũng kết nối encoder và decoder thông qua cơ chế attention. Tác giả đề xuất một kiến ​​trúc mạng mới, gọi là Transformer, chỉ dựa trên các cơ chế self- attention, loại bỏ hoàn toàn recurrent hoặc convolutional. Đồng thời có khả năng song song hóa cao hơn và cần ít thời gian đào tạo hơn đáng kể so với các kiến trúc hiện tại. Hiệu suất cho thấy kiến trúc này vượt trội hơn so với các kiến trúc hiện tại.

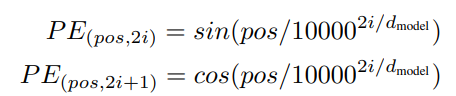
1. **Transformer: Model Architecture**



* 1. **Positional Encoding**

****

* Vì Transformer không chứa recurrence và convolution nên cần phải đưa vào một số thông tin về vị trí tương đối hoặc tuyệt đối của các tokens trong chuỗi.
* “Positional encodings” được thêm vào phần input embedding ở phía dưới cùng của encoder và decoder. Các positional encodings có cùng chiều dmodel với các phần embedding, do đó cả hai có thể được tính tổng.
* Các hàm sin và cosin có tần số khác nhau được sử dụng:

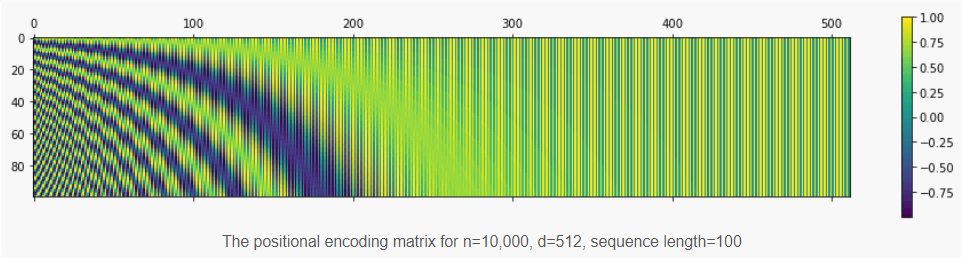


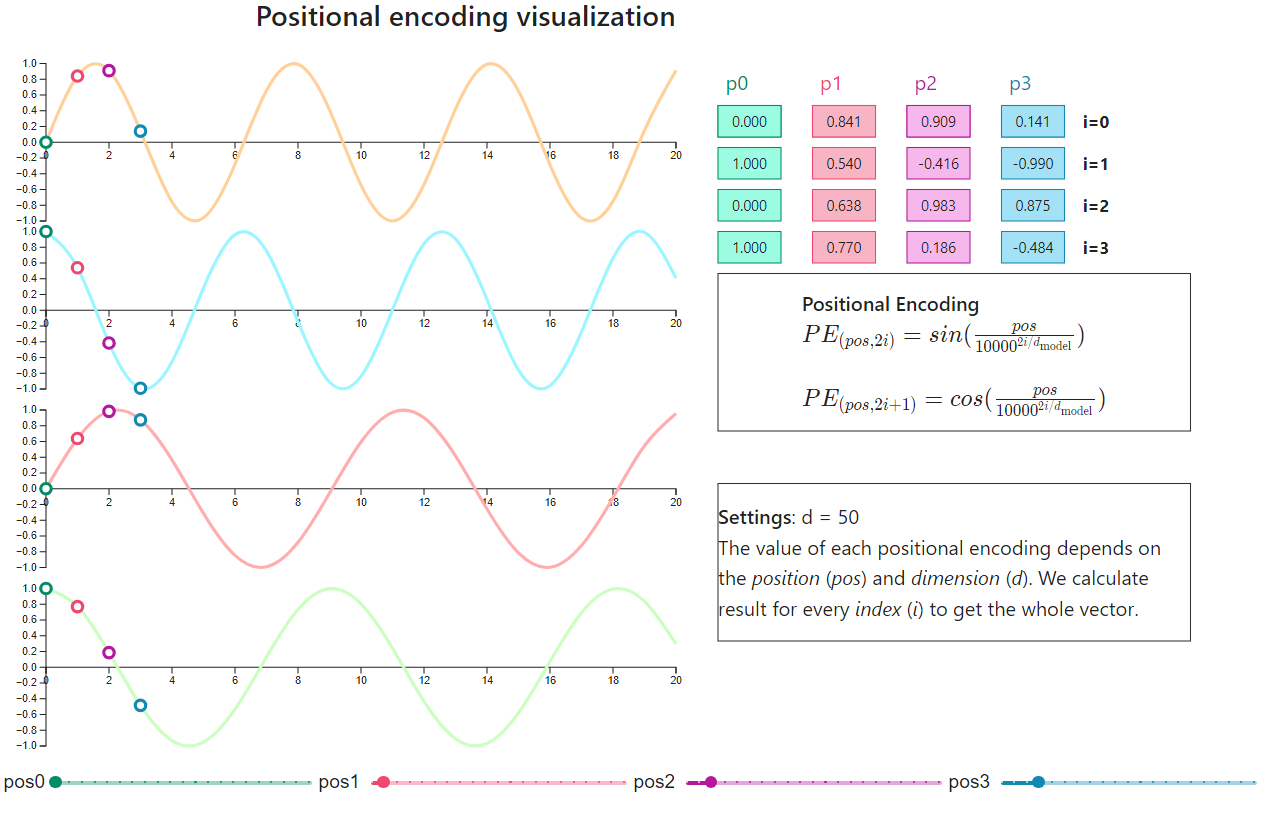
trong đó pos là vị trí và i là kích thước. Mỗi chiều của positional encoding tương ứng với một sinusoid, tác giả đưa ra giả thuyết cho rằng nó sẽ cho phép mô hình dễ dàng học cách tham dự theo các vị trí tương đối.

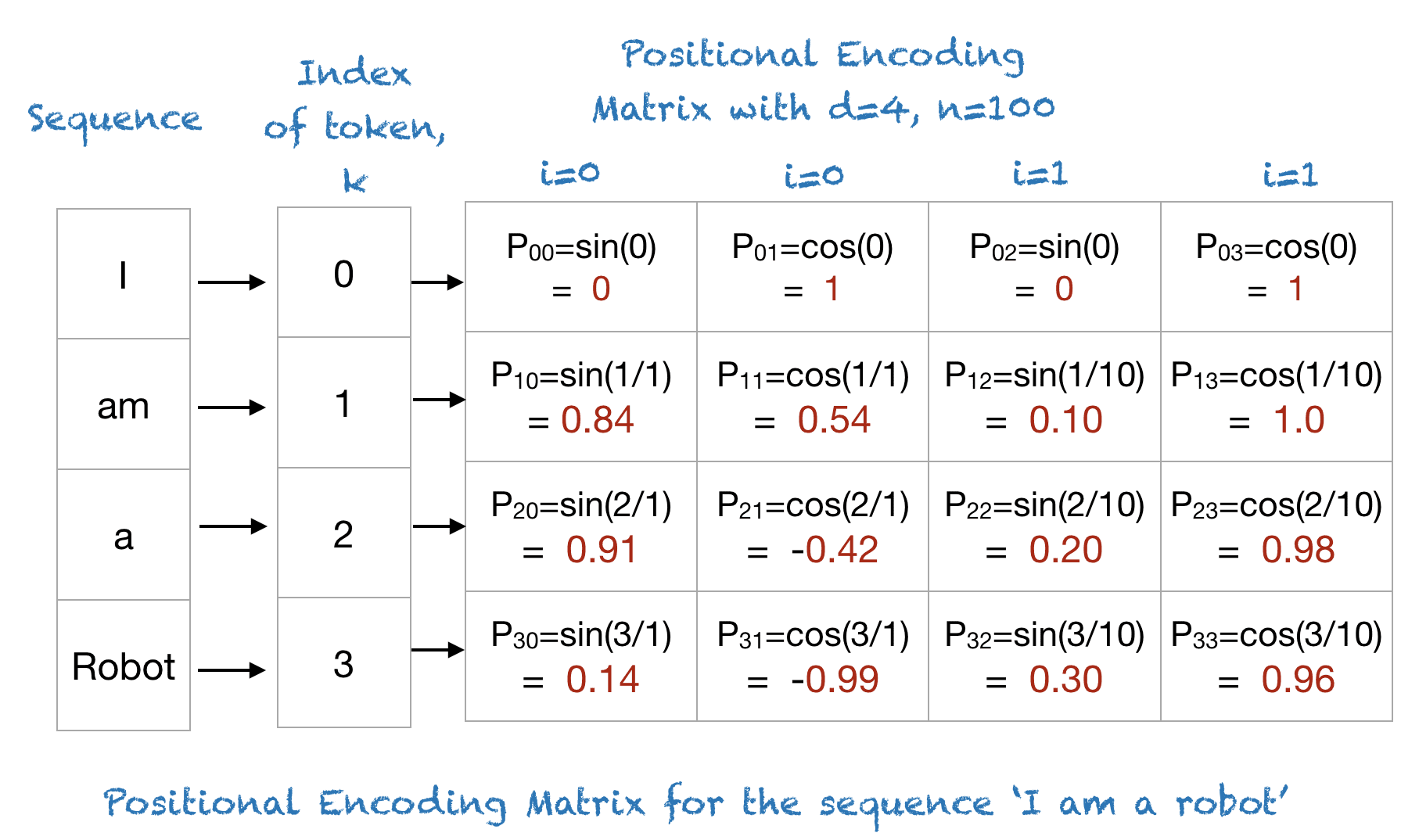
* Nếu nhìn kỹ vào chức năng mã hóa vị trí, bạn có thể thấy rằng bước sóng của một tín hiệu cố định i được cho bởi:

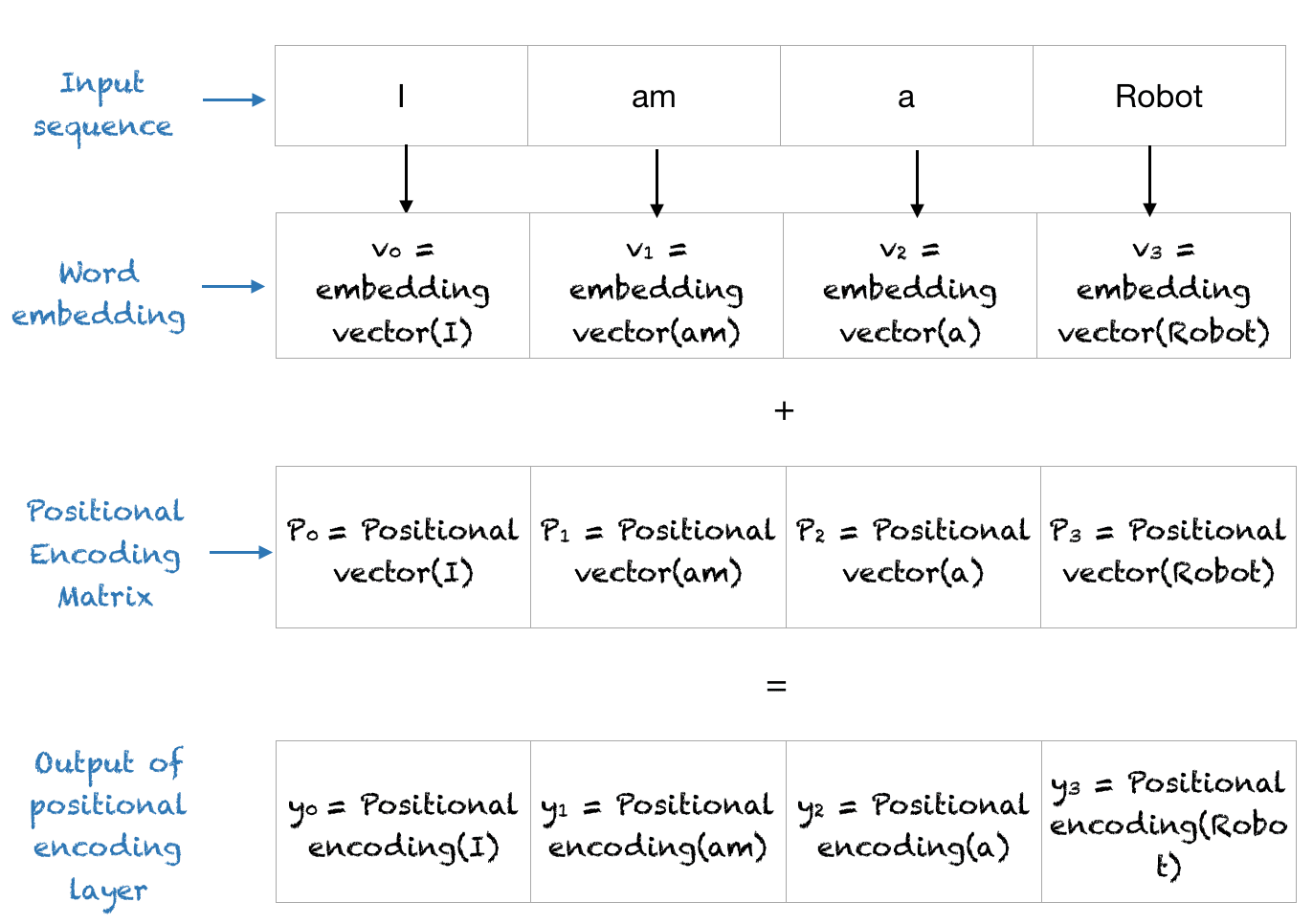


* Do đó, bước sóng của các hình sin tạo thành một cấp số nhân và thay đổi từ 2π to 10000 · 2π

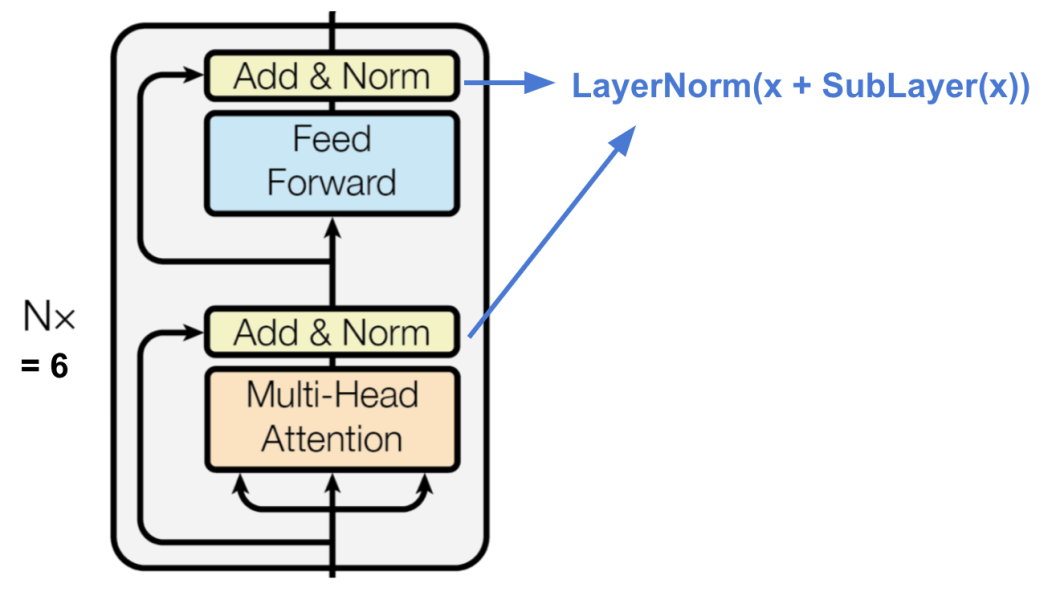




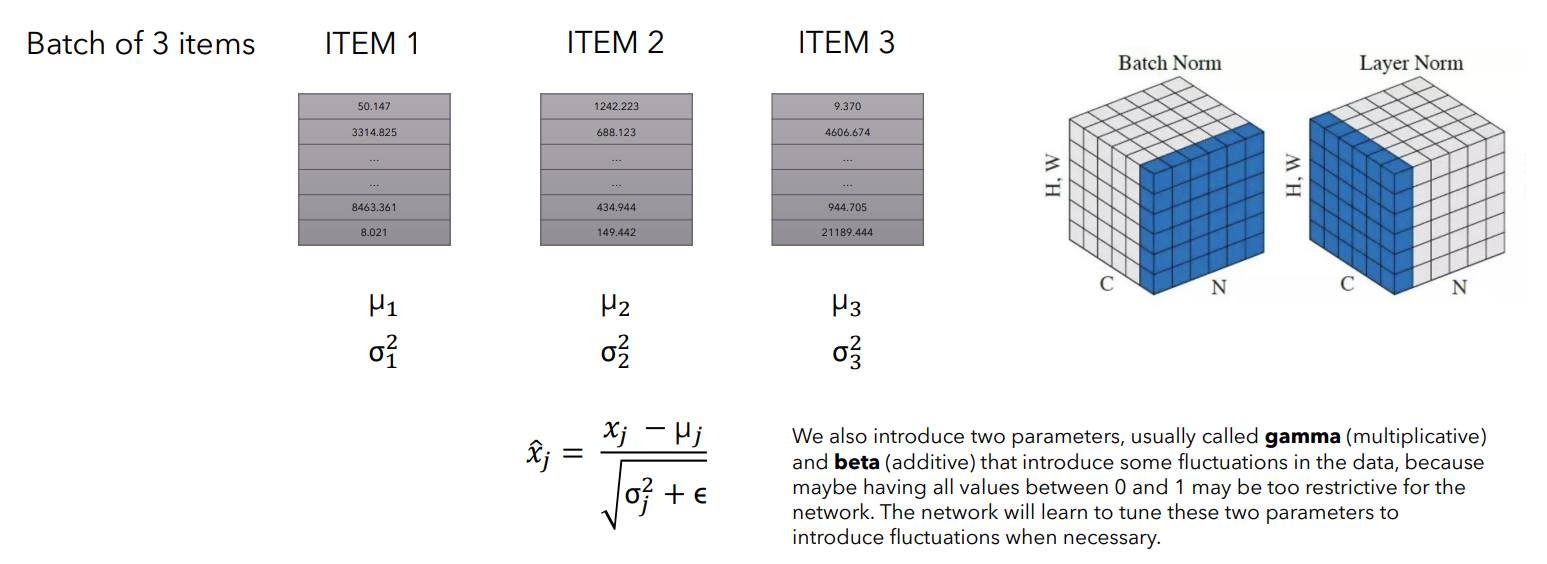


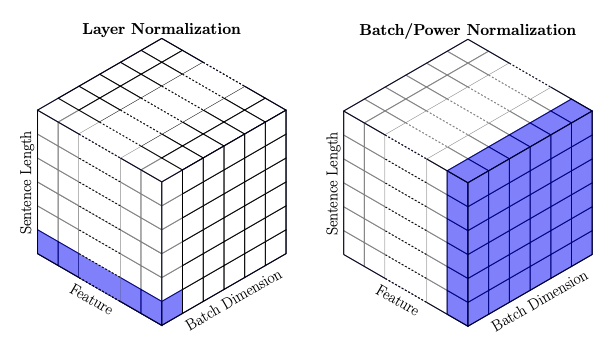


* Cách encode này có một số ưu điểm:
  + Các hàm sin và cosin có các giá trị trong [-1, 1], giữ các giá trị của ma trận mã hóa vị trí trong phạm vi chuẩn hóa.
  + Vì hình sin cho mỗi vị trí là khác nhau nên ta có một cách mã hóa riêng cho từng vị trí.
  + Ta có cách đo lường hoặc định lượng sự giống nhau giữa các vị trí khác nhau, từ đó cho phép ta mã hóa vị trí tương đối của các từ.
  1. **Encoder**

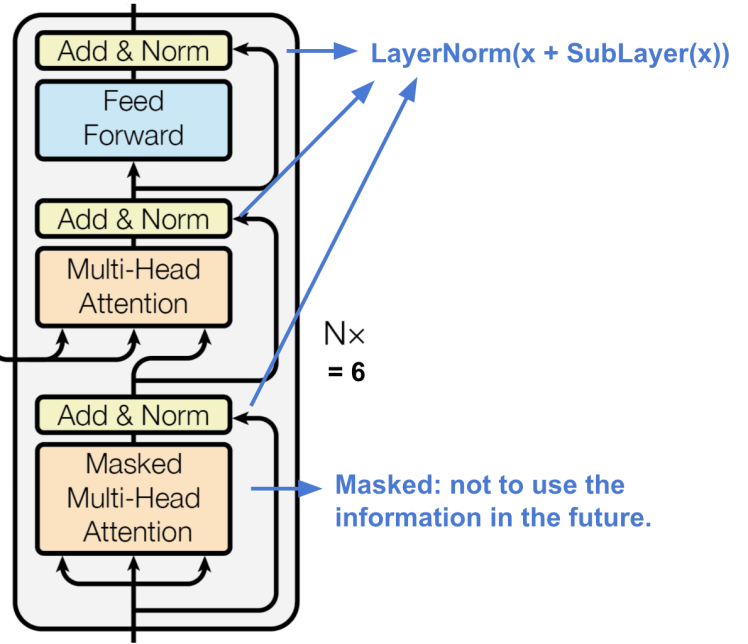


* Encoder bao gồm một stack N = 6 layer tương tự nhau.
* Mỗi layer bao gồm multi-head self-attention và point wise feed forward network.
* Residual connection và layer normalization được sử dụng với mỗi sub-layer.
* Nghĩa là, đầu ra của mỗi sub-layer là LayerNorm(x + Sublayer(x)).
* Tất cả các sub-layer trong mô hình cũng như các embedding layers đều cho đầu ra có kích thước dmodel=512.
* Residual connection được sử dụng ở đây ngoài giảm gradient vanishing còn giúp bảo toàn thông tin về vị trí (position encoding) xuyên suốt quá trình biến đổi.
* LayerNorm:
  + Trong các tác vụ NLP, thường có sự biến động lớn về độ dài câu, tần suất từ và phân bố. Làm cho việc áp dụng BatchNorm kém hiệu quả, vì các thống kê được sử dụng để chuẩn hóa có thể thay đổi mạnh từ batch này sang batch khác và phụ thuộc vào batch size.
  + LayerNorm thực hiện chuẩn hóa theo chiều đặc trưng (feature dimension) nên không phụ thuộc vào kích thước của batch hay độ dài câu.
  + LayerNorm đảm bảo rằng các giá trị đặc trưng trong cùng một layer có cùng phân phối chuẩn và không bị ảnh hưởng bởi biến động trong độ dài câu hay kích thước của batch.

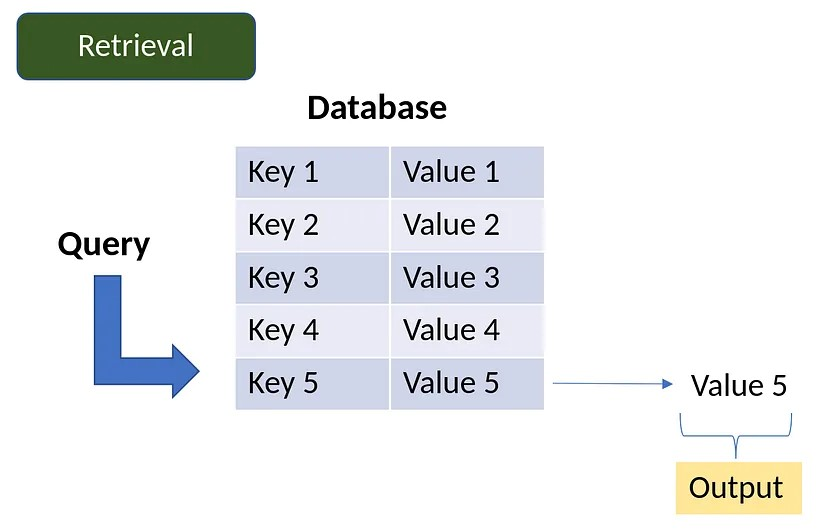




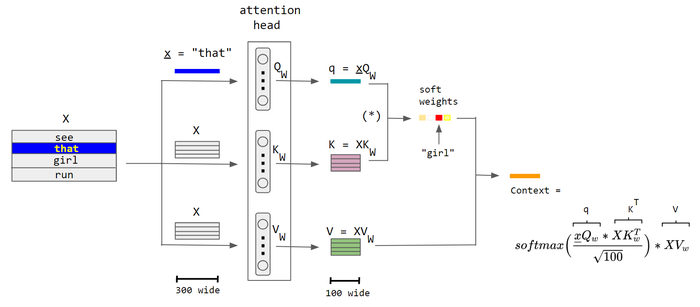
* 1. **Decoder**

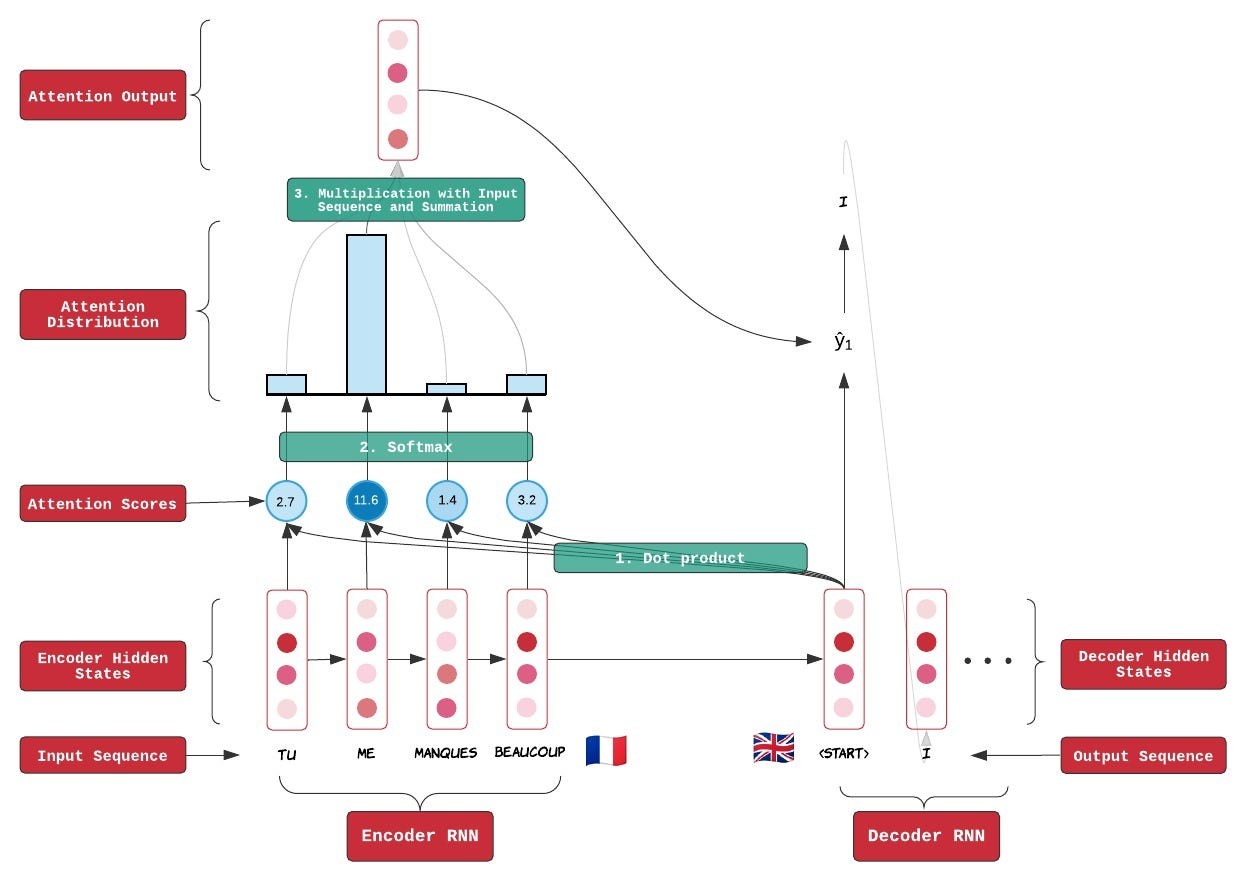


* Decoder cũng bao gồm một stack gồm N = 6 layer tương tự nhau.
* Ngoài hai sub-layer trong mỗi lớp encoder, decoder còn chèn một sub-layer thứ ba, lớp này thực hiện multi-head attention đối với đầu ra của encoder stack.
* Tương tự như encoder, các residual connection và layer normalization cũng được sử dụng.
* Self-attention sub-layer trong decoder stack được sửa đổi để đảm bảo rằng các dự đoán cho vị trí i chỉ có thể phụ thuộc vào các đầu ra đã biết ở các vị trí nhỏ hơn i.
  1. **Attention**
* Khái niệm về key/value/query tương tự như các hệ thống truy xuất.
* Ví dụ: khi tìm kiếm video trên Youtube, công cụ tìm kiếm sẽ ánh xạ query (văn bản trong thanh tìm kiếm) theo một bộ key (tiêu đề video, mô tả, v.v.) được liên kết với các video trong cơ sở dữ liệu, sau đó hiển thị video phù hợp nhất (value).

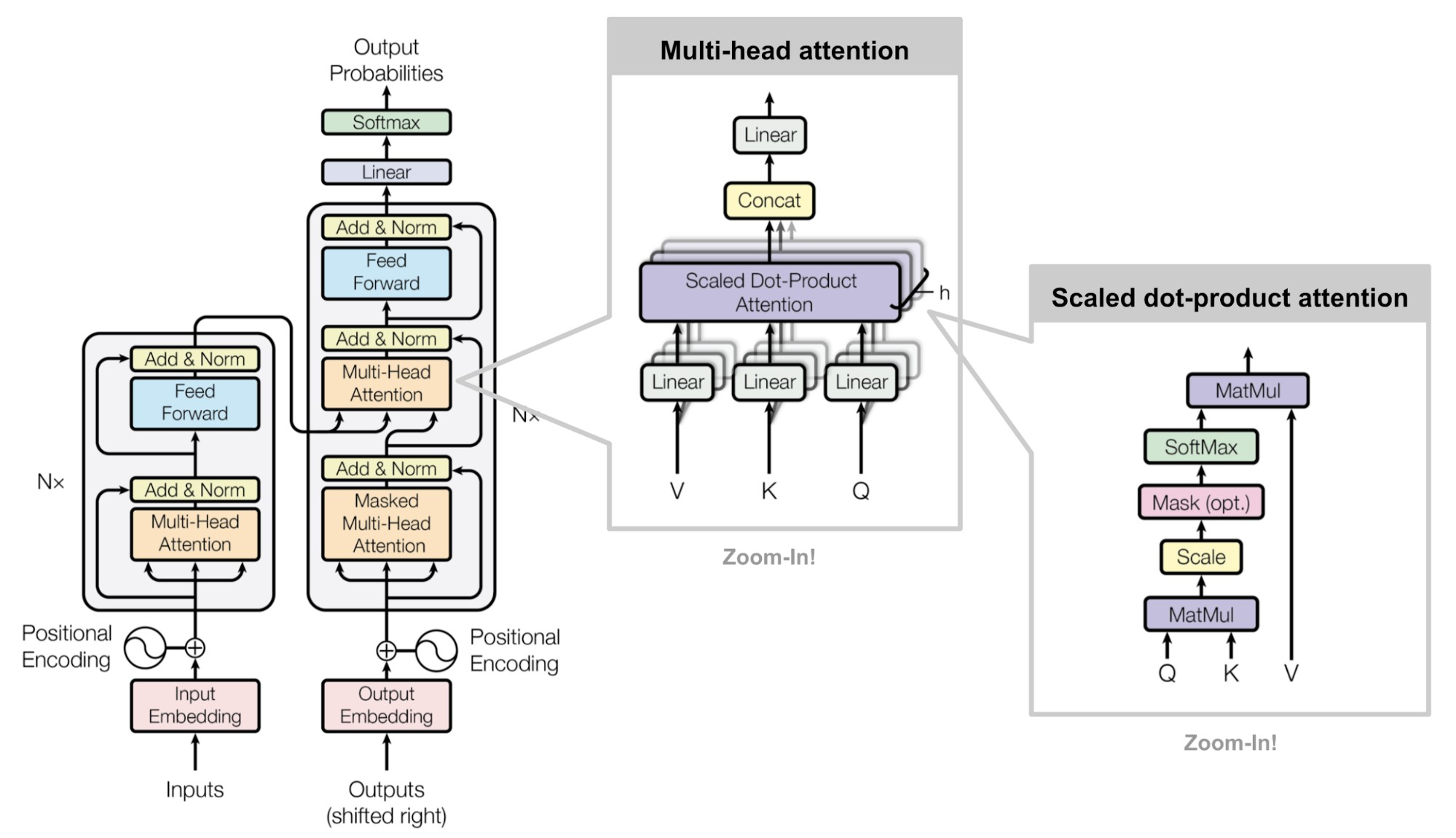


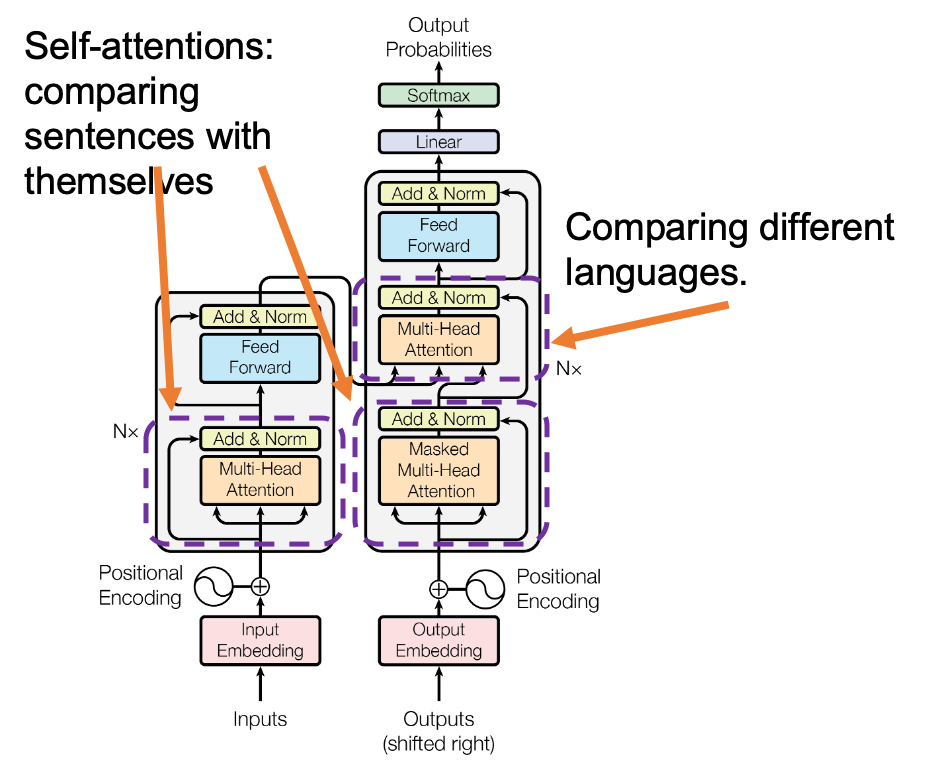
**Attention mechanism in sequence models (như RNN, LSTM, GRU, …)**



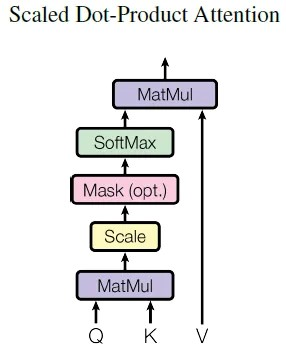


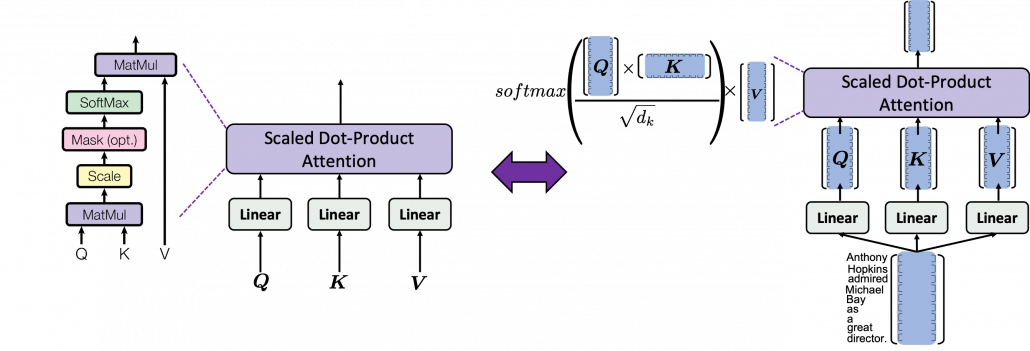
**Attention mechanism in Transformer**





* + 1. **Scaled Dot-Product Attention**

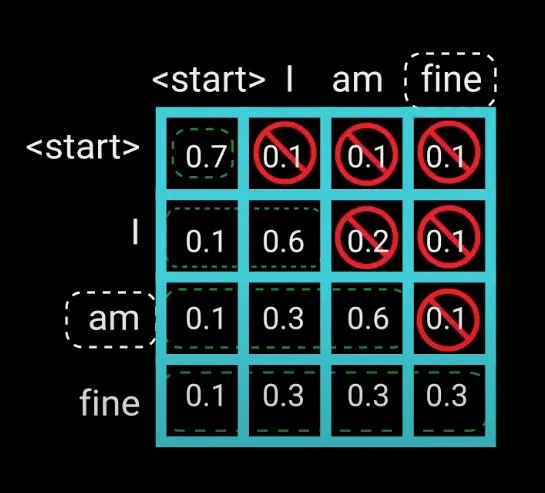


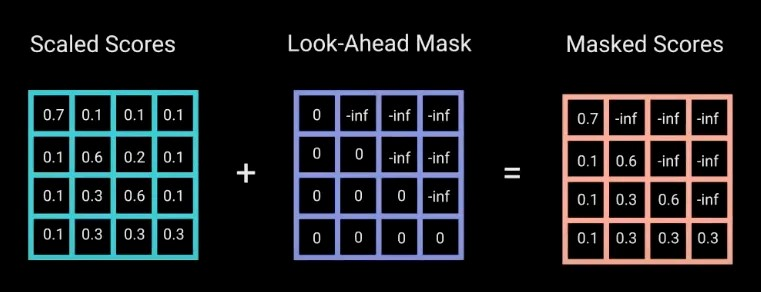


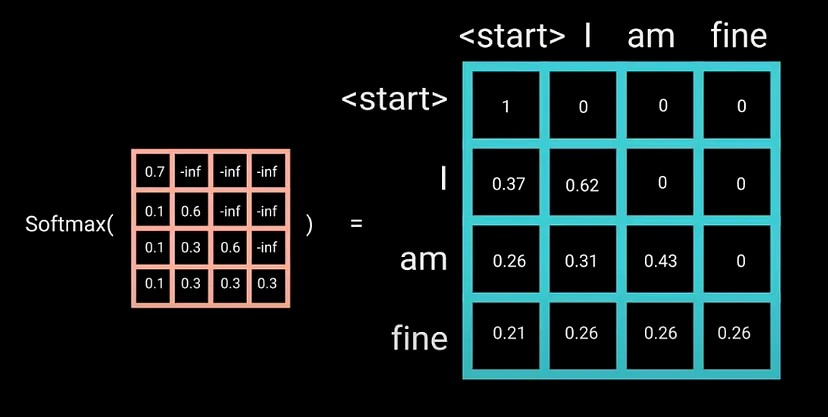
**Hoạt động**

* Đầu vào bao gồm các queries và keys có kích thước dk và các values có kích thước dv.
* Dot products của query với tất cả keys được tính toán, mỗi kết quả chia cho √(dk) và hàm softmax được áp dụng để thu được trọng số trên các giá trị. Trong thực tế, attention function có thể được tính toán đồng thời trên một tập hợp query, được đóng gói thành ma trận Q. Các key và value cũng được đóng gói thành ma trận K và V. Ma trận đầu ra là:



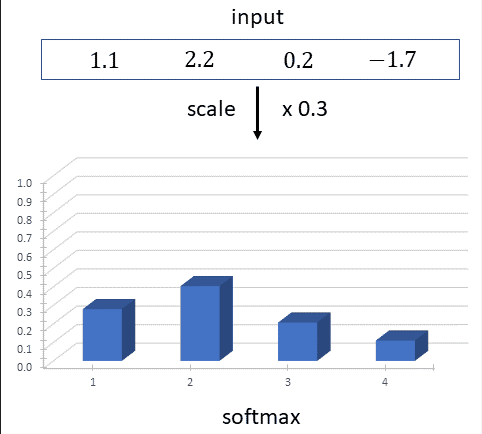
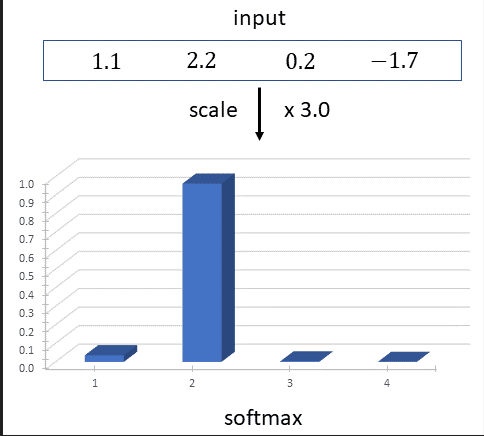
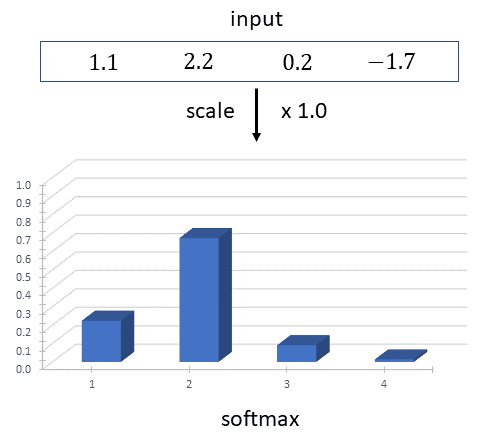




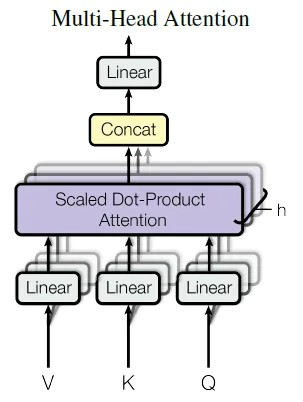


**Lí do sử dụng Dot Product Attention thay vì Additive Attention**

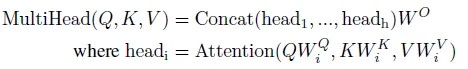
* Hai attention functions được sử dụng phổ biến nhất là additive attention và dot-product (multiplicative) attention.
* Additive attention tính toán độ tương đồng bằng cách sử dụng feed-forward network với một lớp ẩn duy nhất.
* Mặc dù cả hai đều giống nhau về độ phức tạp về mặt lý thuyết, nhưng
* dot-product attention nhanh hơn và tiết kiệm không gian hơn trong thực tế vì nó có thể được thực hiện bằng cách sử dụng highly optimized matrix multiplication code.
* Trong khi đối với các giá trị nhỏ của dk, hai cơ chế hoạt động tương tự nhau. Đối với các giá trị lớn của dk, dot-product tăng theo độ lớn dk, đẩy hàm softmax vào các vùng có gradient cực nhỏ.
* Để khắc phục điều này, tác giả chia tỷ lệ dot-product với 1/√(dk).



* + 1. **Multi-Head Attention**

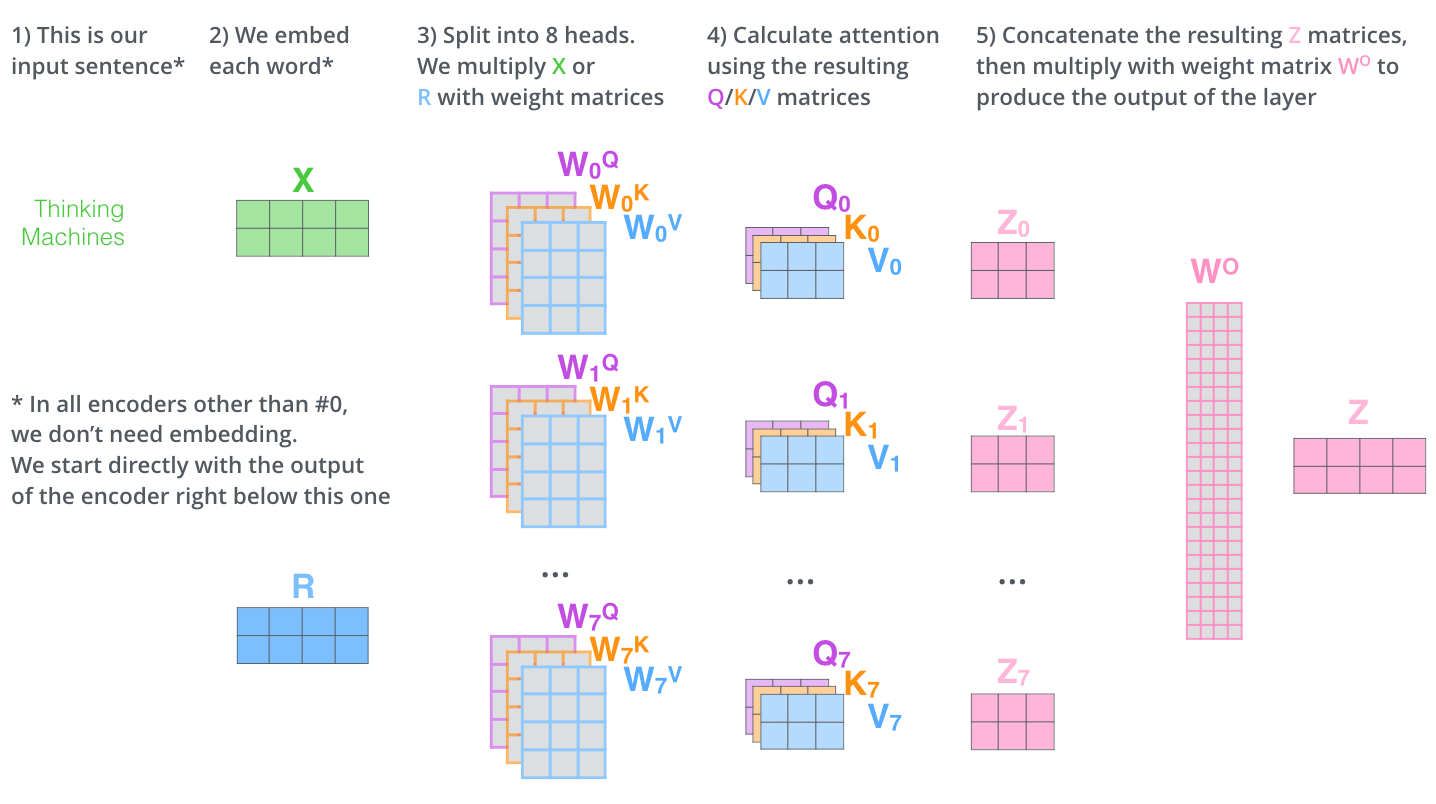


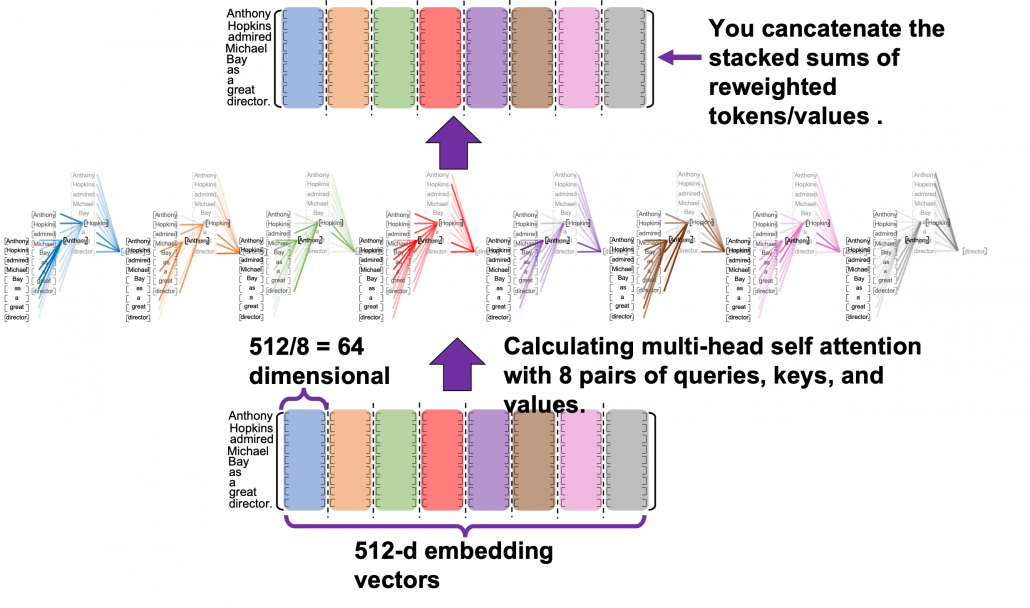
* Thay vì thực hiện một attention function duy nhất với các keys, values và queries theo chiều dmodel, tác giả thấy rằng sẽ có lợi khi linearly project các queries, keys và values h lần với các linearly project đã học khác nhau cho các kích thước dk, dk và dv tương ứng.
* Trên mỗi projected versions ​​của queries, keys và values, attention function được thực hiện song song, mang lại giá trị đầu ra theo chiều dv. Chúng được concat với nhau và một lần nữa được project:

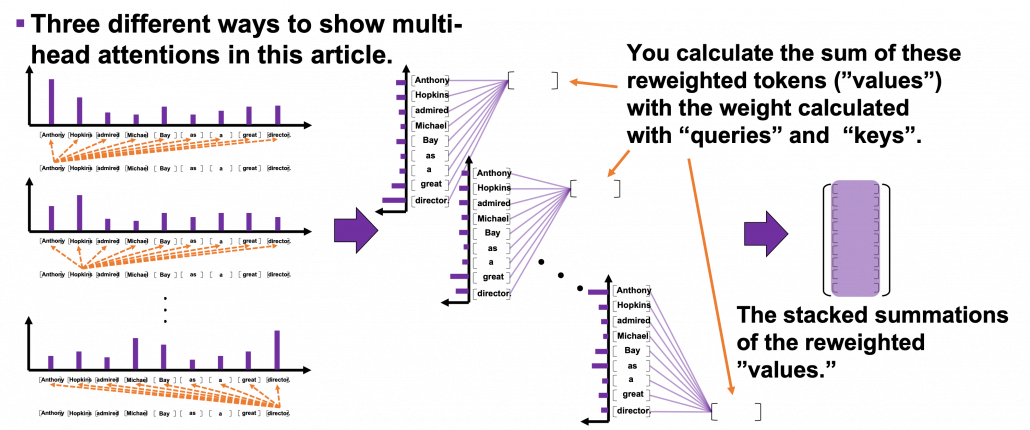


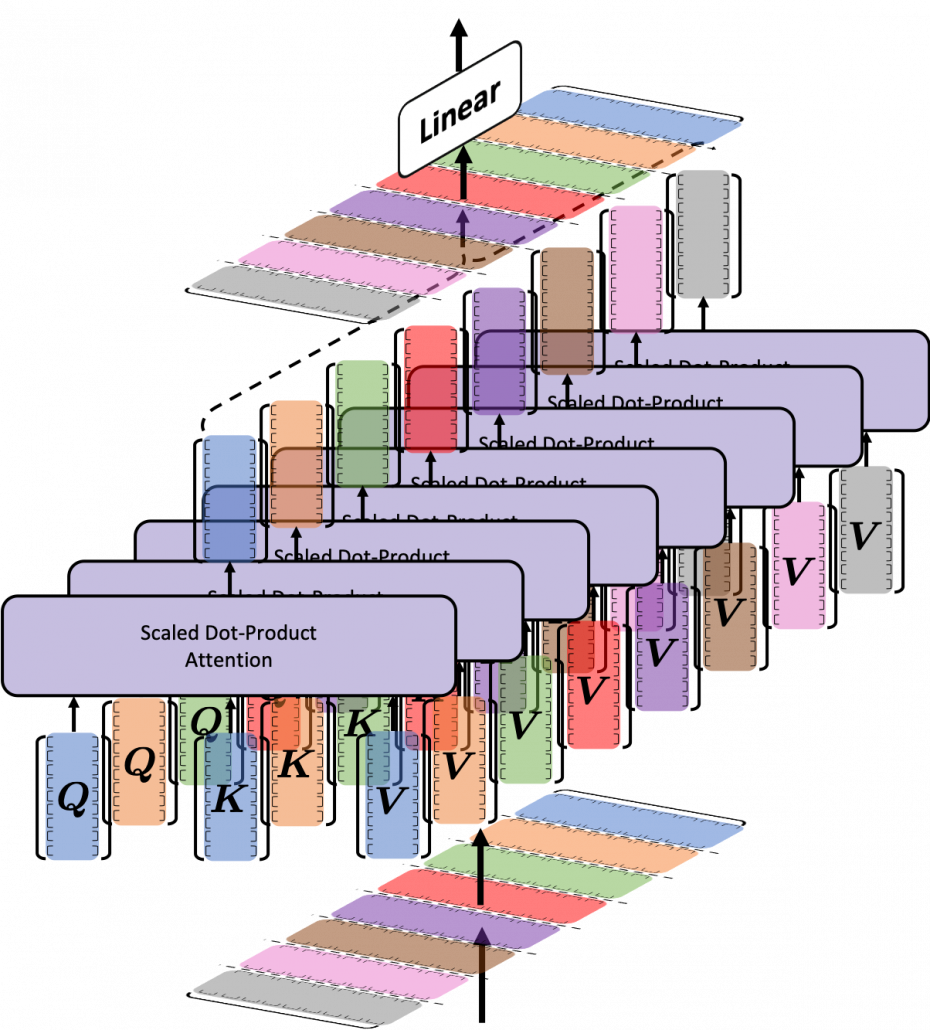
trong đó các phép chiếu là các ma trận tham số WQi, WKi, WVi.

* Trong mô hình này, h=8 lớp attention song song, hoặc các heads.
* Đối với mỗi trong số này, dk=dv=dmodel/h=64. Do kích thước của mỗi đầu bị giảm nên tổng chi phí tính toán tương tự như chi phí tính toán của một đầu với đầy đủ chiều.
* Multi-head attention có một số lợi thế so với one head attention:
  + Nó có thể nắm bắt các loại mối quan hệ khác nhau giữa token input và output, chẳng hạn như cú pháp, ngữ nghĩa hoặc vị trí. Ví dụ, một head có thể tập trung vào sự hòa hợp giữa chủ ngữ và động từ, trong khi một head khác có thể chú ý đến trật tự từ.
  + Nó có thể tăng công suất và tính biểu cảm của mô hình, vì mỗi đầu có thể học bởi các thông số và attention function của riêng mình.
  + Nó có thể giảm chi phí tính toán và mức sử dụng bộ nhớ vì chiều của mỗi vectơ con nhỏ hơn vectơ ban đầu.

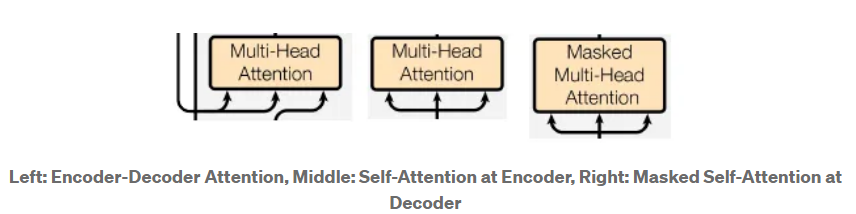








* + 1. **Applications of Attention in Transformer**



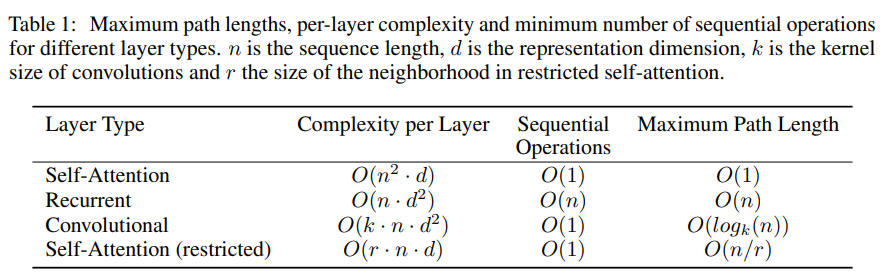
* + Transformer sử dụng multi-head attention theo ba cách khác nhau (3 khối màu cam):
    - Trong lớp “encoder-decoder attention”, các queries đến từ lớp decoder trước đó, đồng thời các keys và values bộ nhớ đến từ đầu ra của encoder. Điều này cho phép mọi vị trí trong decoder tham gia trên tất cả các vị trí trong chuỗi đầu vào.
    - Encoder chứa các lớp self-attention. Trong lớp self-attention, tất cả keys, values và queries đều đến từ cùng một nơi, trong trường hợp này là đầu ra của lớp trước đó trong encoder. Mỗi vị trí trong encoder có thể tham gia tất cả các vị trí ở lớp trước của encoder.
    - Tương tự, các lớp self-attention trong decoder cho phép mỗi vị trí trong decoder tham gia tất cả các vị trí trong decoder và bao gồm cả vị trí đó. Để ngăn luồng thông tin sang trái trong decoder nhằm bảo toàn thuộc tính tự động hồi quy, scaled dot-product attention được chia tỷ lệ được sửa đổi bằng cách che đi (đặt thành -∞) tất cả các giá trị trong đầu vào của softmax tương ứng với các illegal connections.
    1. **Position-wise Feed-Forward Networks**



* Ngoài attention sub-layers, mỗi lớp trong encoder và decoder chứa một fully connected feed-forward network, bao gồm 2 linear transformations với ReLU activation:

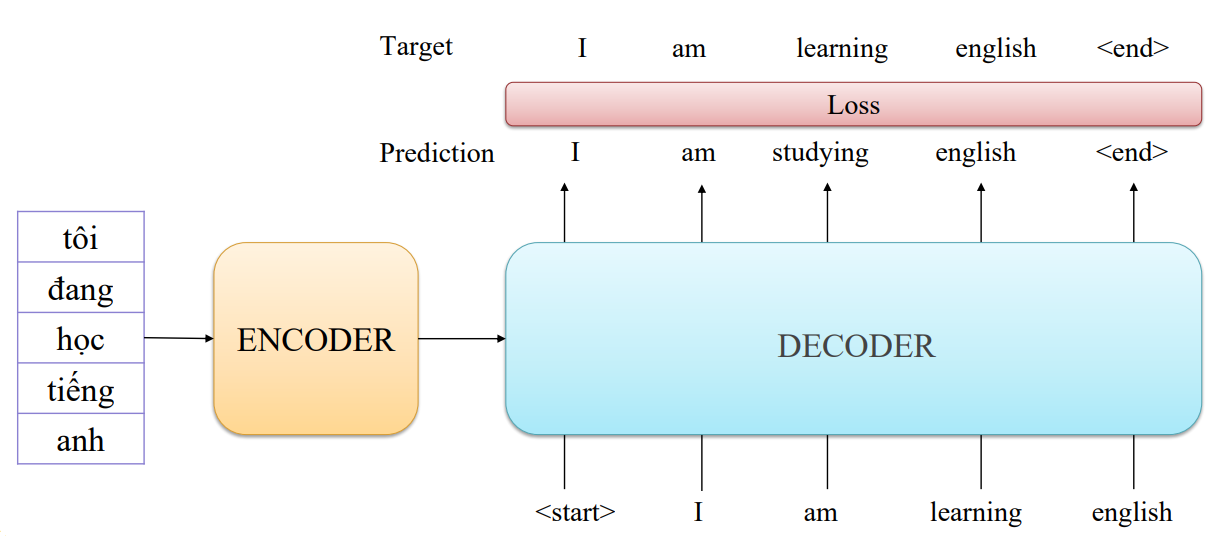


* Một cách khác để mô tả điều này là hai tích chập có kích thước kernel là 1. Chiều của đầu vào và đầu ra là dmodel = 512 và lớp bên trong có chiều dff = 2048.
  1. **Embeddings and Softmax**
* Learned embeddings được sử dụng để chuyển đổi input tokens và output tokens thành vectơ có chiều dmodel.
* Phép biến đổi tuyến tính đã học thông thường và hàm softmax được sử dụng để chuyển đổi đầu ra decoder thành xác suất next-token được dự đoán.
* Trong Transformer, weight matrix giống nhau được chia sẻ giữa embedding layers và pre-softmax linear transformation.
* Trong embedding layers, các weight đó được nhân với √(dmodel).
  1. **Why Self-Attention**

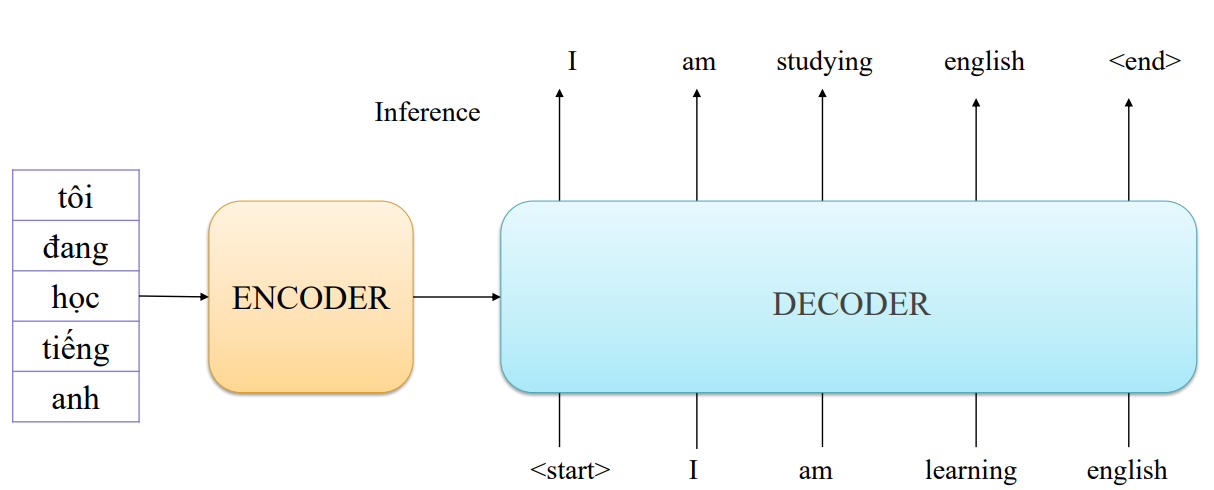
****

* Có 3 yếu tố để sử dụng self-attention:
* Tổng độ phức tạp tính toán trên mỗi layer.
* Số lượng tính toán có thể được song song.
* Độ dài đường dẫn giữa các long-range dependencies trong mạng.
* Self-attention layer kết nối tất cả các vị trí với số lượng thao tác được thực hiện tuần tự không đổi, trong khi recurrent layer yêu cầu các hoạt động tuần tự O(n).
* Về độ phức tạp tính toán, self-attention layer nhanh hơn các recurrent layer khi độ dài chuỗi n nhỏ hơn chiều biểu diễn d.
* Một single convolutional layer có độ rộng kernel k<n không kết nối tất cả các cặp vị trí input và output. Làm như vậy đòi hỏi một chồng các convolutional layers O(n=k) trong trường hợp các kernels liền kề, hoặc O(logk(n)) trong trường hợp dilated convolutions.
  1. **Training and Inference**

**Training**

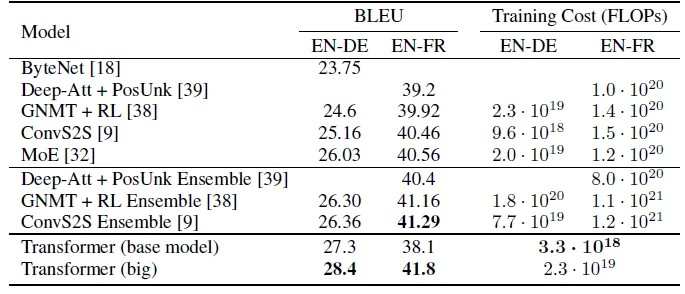


**Inference**

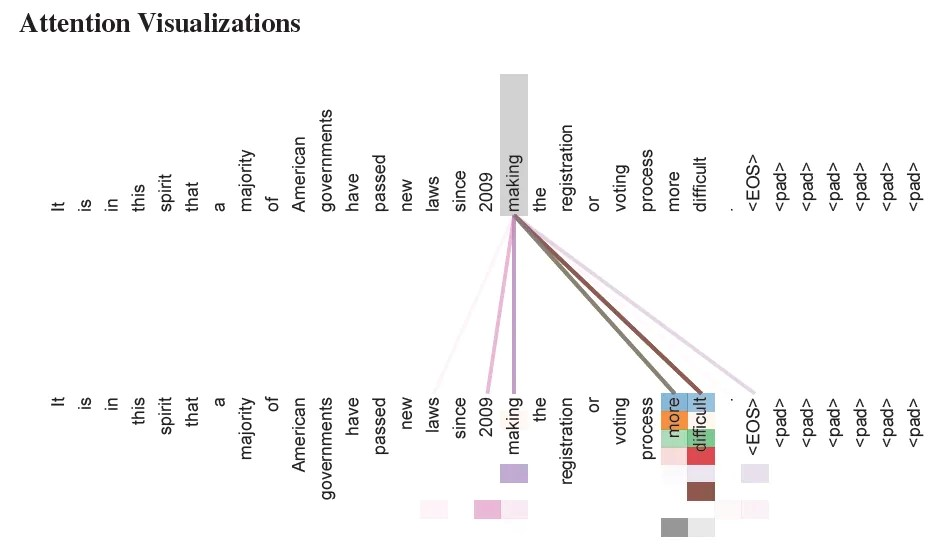
****

1. **Experimental Results**
   1. **Dataset**

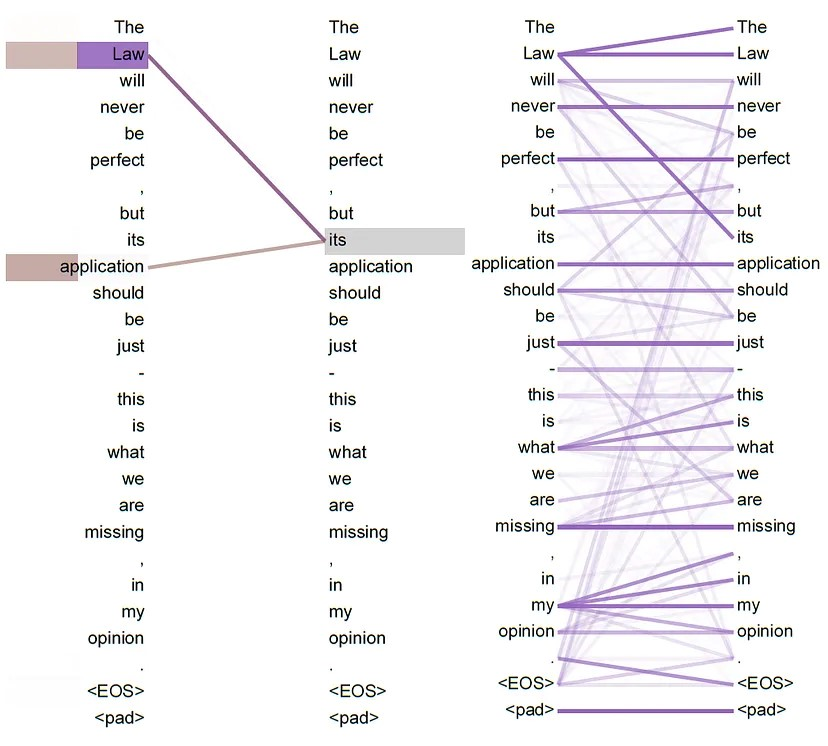
* WMT 2014 English-German dataset bao gồm khoảng 4,5 triệu cặp câu. Các câu được mã hóa bằng cách sử dụng byte-pair encoding, có source-target vocabulary khoảng 37000 tokens.
* WMT 2014 English-French dataset bao gồm 36 triệu câu và chia token thành vocabulary gồm 32000 từ. Mỗi đợt huấn luyện chứa một tập hợp các cặp câu chứa khoảng 25000 source tokens và 25000 target tokens.



* 1. **Attention Visualization**



Nhiều attention heads chú ý đến sự phụ thuộc xa vời của động từ 'making', hoàn thành cụm từ 'making...more difficult'. Sự chú ý ở đây chỉ được thể hiện đối với từ 'making'. Màu sắc khác nhau đại diện cho những heads khác nhau.



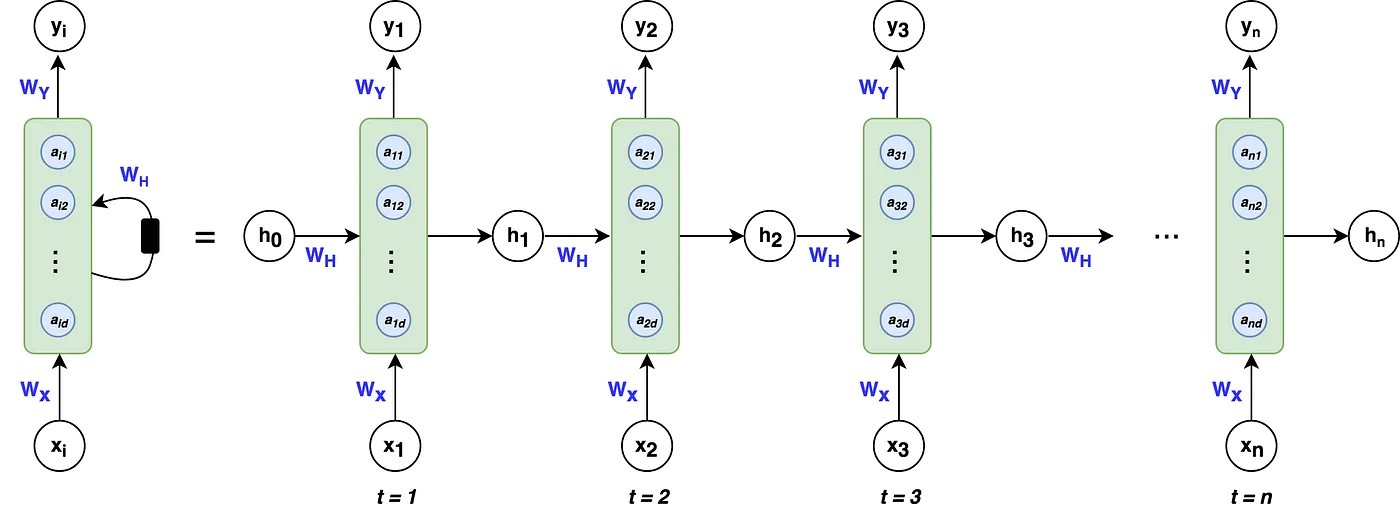
Phải: full attentions head 5.

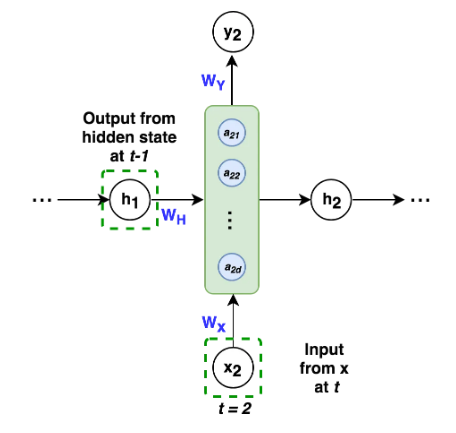
Trái: attention cho từ ‘its’ với head 5 và 6.

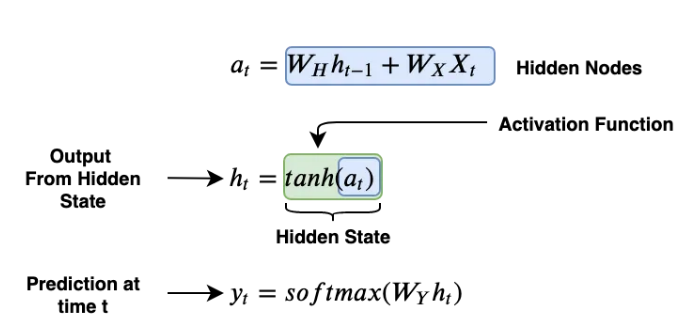
1. **Note**

* **Ưu điểm của Transformer**:
  + Có khả năng song song cao, có nghĩa là chúng có thể xử lý nhiều phần của một chuỗi cùng một lúc, giúp tăng tốc đáng kể việc đào tạo và suy luận.
  + Có thể nắm bắt được sự phụ thuộc dài hạn trong văn bản, cho phép model hiểu rõ hơn về bối cảnh chung và tạo ra văn bản mạch lạc hơn.
  + Linh hoạt và có thể mở rộng hơn, giúp dễ dàng điều chỉnh model theo các nhiệm vụ và lĩnh vực khác nhau.
* **Nhược điểm**:
  + Nhu cầu tính toán cao. Do quy mô và độ phức tạp của chúng, các mô hình dựa trên Transformers đòi hỏi một lượng lớn tài nguyên tính toán và thời gian đào tạo.
  + Ngoài ra, Transformers rất nhạy cảm với chất lượng và số lượng dữ liệu đào tạo. Nếu dữ liệu đào tạo bị giới hạn hoặc sai lệch, hiệu suất mô hình có thể bị ảnh hưởng xấu. Đây có thể là một thách thức trong các tình huống mà dữ liệu khan hiếm hoặc khó có được.

**RNN**

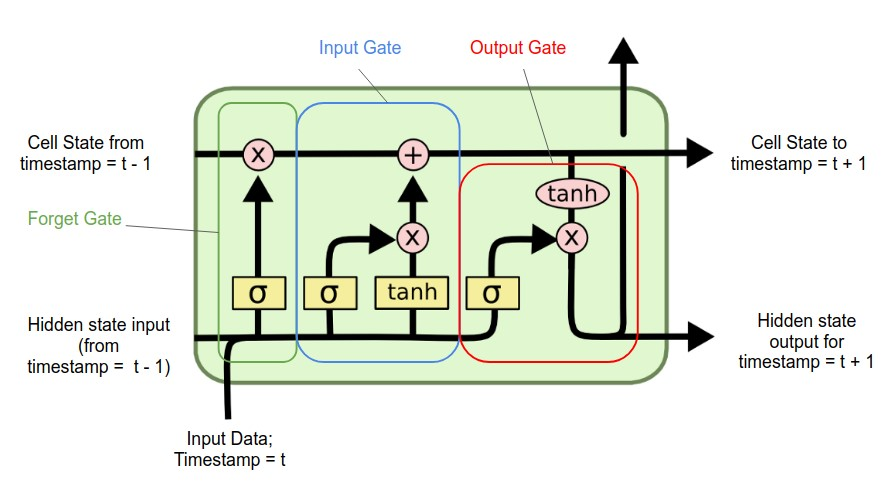






* Ưu điểm của RNN:
  + Xử lý chuỗi đầu vào có độ dài bất kỳ
  + Kích thước mô hình không tăng đối với độ dài chuỗi đầu vào dài hơn
  + Mỗi bước t sử dụng thông tin từ nhiều bước quay lại
  + Các trọng số giống nhau được áp dụng cho mọi dấu thời gian của đầu vào
* Nhược điểm
  + RNN là tuần tự, không thể song song
    - Tính toán chậm
  + Khó tiếp cận thông tin từ nhiều bước lùi
    - Biến mất và bùng nổ độ dốc

**LSTM**



* Ưu điểm LSTM:
  + LSTM có khả năng xử lý dữ liệu chuỗi phức tạp và dài hạn tốt hơn so với RNN
* Nhược điểm:
  + LSTM mất nhiều thời gian hơn để đào tạo
  + LSTM yêu cầu nhiều bộ nhớ hơn để đào tạo
  + LSTM rất dễ bị overfit

-------------------Ending Mệt vãi ---------------------