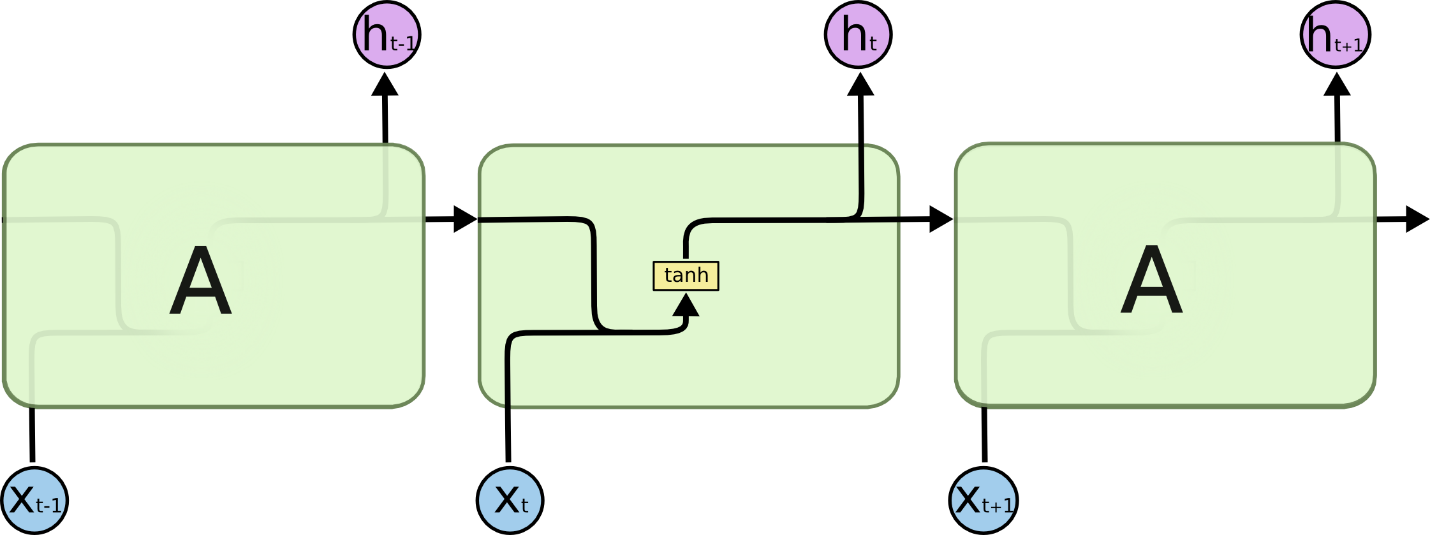
**BÁO CÁO TIẾN ĐỘ ĐỒ ÁN TỐT NGHIỆP**

**ĐỀ TÀI**: XÂY DỰNG MÔ HÌNH DỰ BÁO MỰC NƯỚC SỬ DỤNG MÔ HÌNH HỌC MÁY VỚI CƠ CHẾ ATTENTION

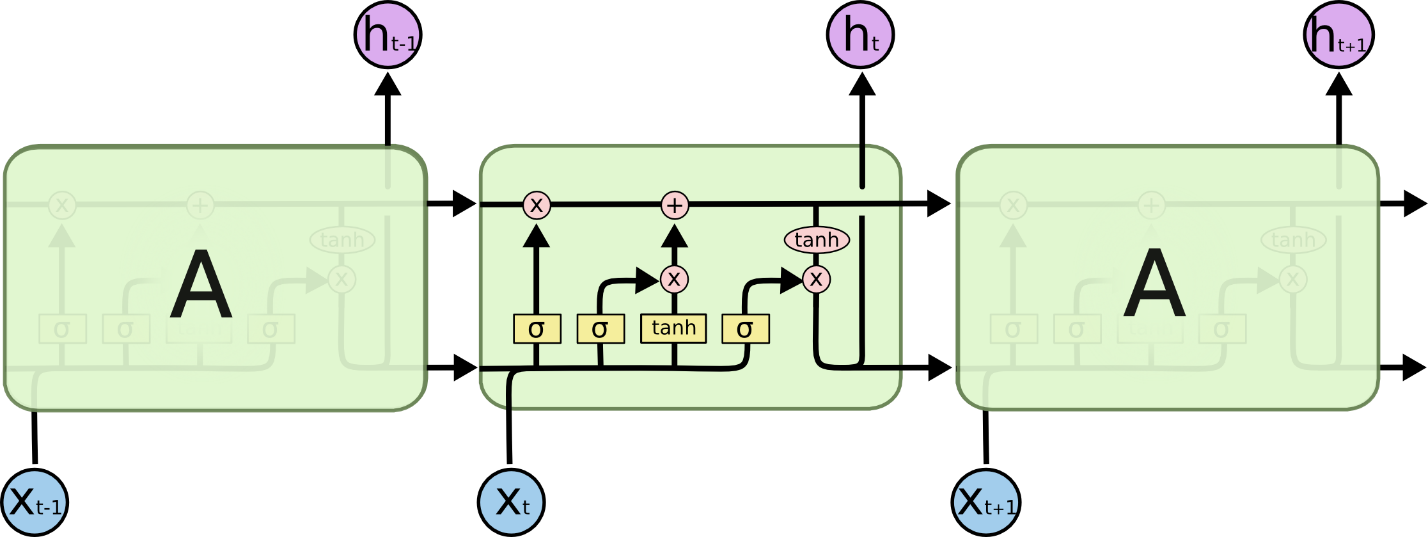
1.Thuật tuán Long Sort-Term Memory(LSTM)

- Mạng trí nhớ ngắn hạn định hướng dài hạn còn được viết tắt là LSTM làm một kiến trúc đặc biệt của RNN có khả năng học được sự phục thuộc trong dài hạn (long-term dependencies) được giới thiệu bởi [Hochreiter & Schmidhuber (1997)](http://www.bioinf.jku.at/publications/older/2604.pdf). Kiến trúc này đã được phổ biến và sử dụng rộng rãi cho tới ngày nay. LSTM đã tỏ ra khắc phục được rất nhiều những hạn chế của RNN trước đây về triệt tiêu đạo hàm. Tuy nhiên cấu trúc của chúng có phần phức tạp hơn mặc dù vẫn dữ được tư tưởng chính của RNN là sự sao chép các kiến trúc theo dạng chuỗi.

Một mạng RNN tiêu chuẩn sẽ có kiến trúc rất đơn giản chẳng hạn như đối với kiến trúc gồm một tầng ẩn là hàm tanh như bên dưới.



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



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



Ô trạng thái là một dạng băng chuyền chạy thẳng xuyên suốt toàn bộ chuỗi với chỉ một vài tương tác tuyến tính nhỏ giúp cho thông tin có thể truyền dọc theo đồ thị mạng nơ ron ổn định.

LSTM có khả năng xóa và thêm thông tin vào ô trạng thái và điều chỉnh các luồng thông tin này thông qua các cấu trúc gọi là cổng.

Cổng là cơ chế đặc biệt để điều chỉnh luồng thông tin đi qua. Chúng được tổng hợp bởi một tầng ẩn của hàm activation sigmoid và với một toán tử nhân như đồ thị.



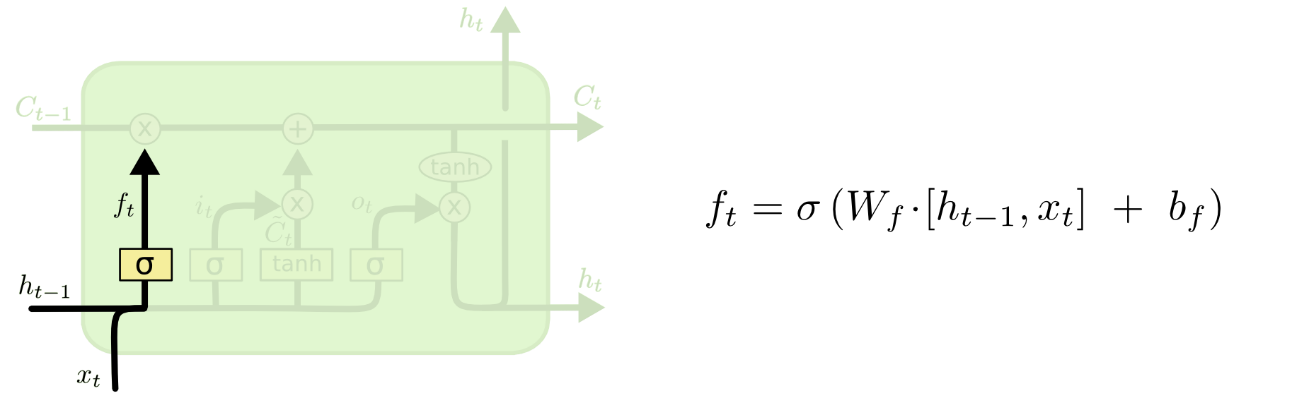
Hàm sigmoid sẽ cho đầu ra là một giá trị xác xuất nằm trong khoảng từ 0 đến 1, thể hiện rằng có bao nhiêu phần thông tin sẽ đi qua cổng. Giá trị bằng 0 ngụ ý rằng không cho phép thông tin nào đi qua, giá trị bằng 1 sẽ cho toàn bộ thông tin đi qua.

Một mạng LSTM sẽ có 3 cổng có kiến trúc dạng này để bảo vệ và kiểm soát các ô trạng thái.

***Thứ tự các bước của LSTM:***

Bước đầu tiên trong LSTM sẽ quyết định xem thông tin nào chúng ta sẽ cho phép đi qua ô trạng thái (cell state). Nó được kiểm soát bởi hàm sigmoid trong một tầng gọi là tầng quên (forget gate layer). Đầu tiên nó nhận đầu vào là 2 giá trị ht−1 và xt và trả về một giá trị nằm trong khoảng 0 và 1 cho mỗi giá trị của ô trạng thái Ct−1. Nếu giá trị bằng 1 thể hiện ‘giữ toàn bộ thông tin’ và bằng 0 thể hiện ‘bỏ qua toàn bộ chúng’.

Trở lại ví dụ về ngôn ngữ, chúng ta đang cố gắng dự báo từ tiếp theo dựa trên toàn bộ những từ trước đó. Trong những bài toán như vậy, ô trạng thái có thể bao gồm loại của chủ ngữ hiện tại, để cho đại từ ở câu tiếp theo được sử dụng chính xác. Chẳng hạn như chúng ta đang mô tả về một người bạn là con trai thì các đại từ nhân xưng ở tiếp theo phải là anh, thằng, hắn thay vì cô, con ấy (xin lỗi vì lấy ví dụ hơi thô). Tuy nhiên chủ ngữ không phải khi nào cũng cố định. Khi chúng ta nhìn thấy một chủ ngữ mới, chúng ta muốn quên đi loại của một chủ ngữ cũ. Do đó tầng quên cho phép cập nhật thông tin mới và lưu giữ giá trị của nó khi có thay đổi theo thời gian.



Bước tiếp theo chúng ta sẽ quyết định loại thông tin nào sẽ được lưu trữ trong ô trạng thái. Bước này bao gồm 2 phần. Phần đầu tiên là một tầng ẩn của hàm sigmoid được gọi là tầng cổng vào (input gate layer) quyết định giá trị bao nhiêu sẽ được cập nhật. Tiếp theo, tầng ẩn hàm tanh sẽ tạo ra một véc tơ của một giá trị trạng thái mới C~t mà có thể được thêm vào trạng thái. Tiếp theo kết hợp kết quả của 2 tầng này để tạo thành một cập nhật cho trạng thái.

Trong ví dụ của mô hình ngôn ngữ, chúng ta muốn thêm loại của một chủ ngữ mới vào ô trạng thái để thay thế phần trạng thái cũ muốn quên đi.



Đây là thời điểm để cập nhật một ô trạng thái cũ, Ct−1 sang một trạng thái mới Ct. Những bước trước đó đã quyết định làm cái gì, và tại bước này chỉ cần thực hiện nó.

Chúng ta nhân trạng thái cũ với ft tương ứng với việc quên những thứ quyết định được phép quên sớm. Phần tử đề cử it∗C~t là một giá trị mới được tính toán tương ứng với bao nhiêu được cập nhật vào mỗi giá trị trạng thái.



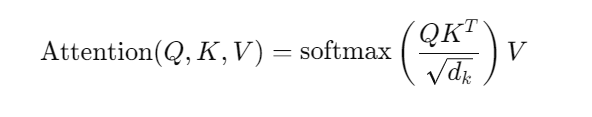
Cuối cùng cần quyết định xem đầu ra sẽ trả về bao nhiêu. Kết quả ở đầu ra sẽ dựa trên ô trạng thái, nhưng sẽ là một phiên bản được lọc. Đầu tiên, chúng ta chạy qua một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về khoảng -1 và 1) và nhân nó với đầu ra của một cổng sigmoid, do đó chỉ trả ra phần mà chúng ta quyết định.

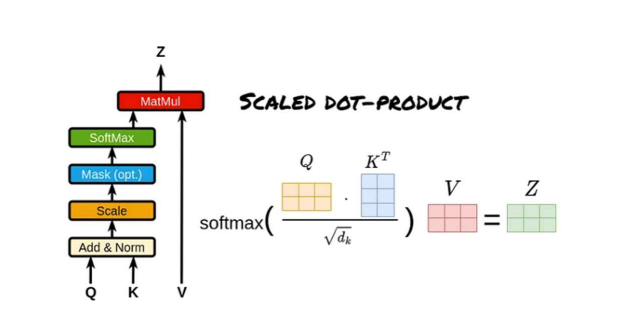
2.Cơ chế Attention

- Cơ chế chú ý có thể được mô tả như là ánh xạ một truy vấn và một tập hợp các cặp Key-Value đến một đầu ra, trong đó Query, Key, Value và đầu ra đều là các vector [13]. Đầu ra được tính toán dưới dạng tổng có trọng số của Value, trong đó trọng số được gán cho mỗi giá trị được tính toán bằng một hàm tương thích của truy vấn với khóa tương ứng.

2.1 Scaled Dot-Product Attention

Trong mô hình **Transformer** đã loại bỏ đi hoàn toàn khái niệm các vector hidden, memori và thay thế chúng bằng ba vector **query, keys, values**. Kết quả đầu ra bây giờ được tính bằng tổng các giá trị **values** đã được đánh trọng số. Trọng số này chính là hàm softmax tính dựa trên **query** và **key** tương ứng. Công thức tính attention weight có tên là **Scaled Dot-Product Attention**.

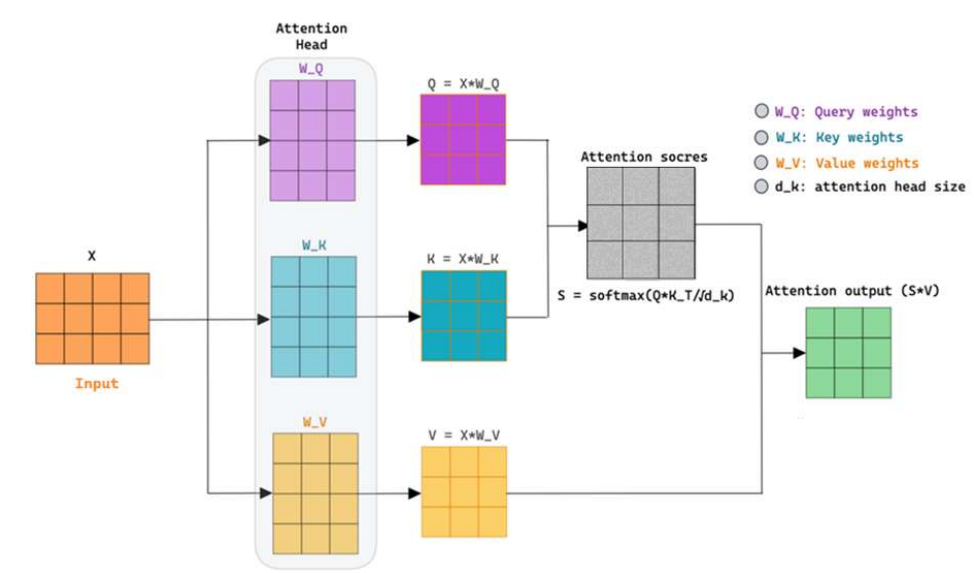




2.2 Seft-Attention

**Self-Attention** là một cơ chế trong đó mỗi phần tử trong chuỗi đầu vào "chú ý" đến tất cả các phần tử khác trong chuỗi, bao gồm chính nó. Nó tính toán trọng số attention cho từng phần tử đầu vào dựa trên tất cả các phần tử khác trong chuỗi, giúp nắm bắt mối quan hệ dài hạn trong chuỗi dữ liệu.

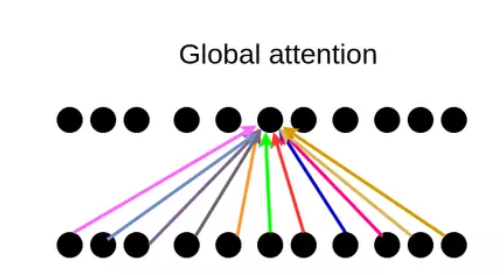
* **Ứng dụng chính**: Trong mô hình Transformer, Self-Attention là cốt lõi giúp nắm bắt mối quan hệ giữa các từ trong câu ở cả ngữ cảnh gần và xa.
* **Ưu điểm**: Tăng khả năng mô hình hiểu được các phụ thuộc dài hạn, đặc biệt hiệu quả trong các chuỗi dữ liệu như văn bản



2.3 Global Self-Attention

**Global Self-Attention** là một biến thể của Self-Attention nhưng mở rộng phạm vi "chú ý" ra toàn bộ chuỗi đầu vào. Tức là, nó cho phép mỗi phần tử trong chuỗi chú ý không chỉ các phần tử kế cận mà cả các phần tử ở bất kỳ khoảng cách nào. Nó đặc biệt hữu ích trong các trường hợp mà các mối quan hệ dài hạn trong chuỗi có ảnh hưởng mạnh đến kết quả.

* **Đặc điểm**: Không bị giới hạn trong một phạm vi cục bộ, do đó có thể nắm bắt mối quan hệ toàn cục.
* **Ứng dụng**: Rất hữu ích cho các mô hình dự đoán với dữ liệu có quan hệ dài hạn như thời gian, chuỗi sự kiện.



2.4 DropAttention

**DropAttention** là một biến thể của cơ chế attention, giúp giảm bớt các thành phần chú ý không cần thiết để tránh quá tải tính toán. Cơ chế này chỉ giữ lại một phần của các trọng số attention và bỏ qua các trọng số có mức độ chú ý thấp hơn, tương tự cách Dropout hoạt động.

* **Mục tiêu**: Giảm chi phí tính toán và tránh overfitting bằng cách loại bỏ các giá trị chú ý có trọng số thấp.
* **Cách hoạt động**: Thông thường, DropAttention sẽ đặt các trọng số chú ý có giá trị thấp về 0, chỉ giữ lại các trọng số có giá trị lớn hơn ngưỡng nhất định.

2.5 Multi-Head Attention

**Multi-Head Attention** là một kỹ thuật mở rộng của Self-Attention. Thay vì tính toán attention một lần duy nhất, Multi-Head Attention thực hiện nhiều phép tính attention song song trên các "đầu attention" (heads) khác nhau, sau đó kết hợp chúng lại để tăng cường khả năng nắm bắt các đặc trưng phức tạp của dữ liệu.

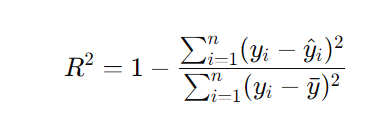
* **Ưu điểm**: Giúp mô hình có thể "nhìn" vào dữ liệu từ nhiều góc độ khác nhau cùng lúc, nắm bắt được các đặc trưng ở các mức độ khác nhau.
* **Ứng dụng**: Là thành phần chủ chốt trong mô hình Transformer, cho phép mô hình xử lý các quan hệ phức tạp giữa các phần tử trong chuỗi.

3. Độ đo đánh giá chất lượng mô hình

3.1 Độ đo R2(Hệ số xác định)

R² cho biết tỷ lệ phương sai của biến đầu ra (thực tế) được giải thích bởi biến đầu vào trong mô hình.

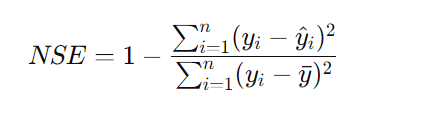
Giá trị R² nằm trong khoảng từ 0 đến 1 (đôi khi có thể nhỏ hơn 0 nếu mô hình kém). Giá trị càng gần 1 thì mô hình càng tốt vì nó giải thích được nhiều phương sai của dữ liệu thực tế.



3.2 Độ đo NSE

NSE đánh giá mức độ chính xác của dự đoán so với giá trị trung bình của dữ liệu thực tế. Chỉ số này thường được sử dụng trong các mô hình dự báo chuỗi thời gian.

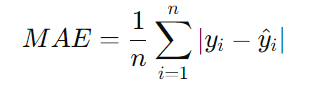
NSE dao động từ -∞ đến 1. Giá trị NSE càng gần 1 thì mô hình càng tốt. NSE < 0 nghĩa là mô hình dự đoán kém hơn dự đoán dựa trên trung bình của dữ liệu thực.



3.3 Độ đo MAE(Sai số tuyệt đối trung bình)

MAE đo lường sai số trung bình của các dự đoán so với giá trị thực tế mà không tính đến chiều của sai số (sai số âm hoặc dương đều tính như nhau).

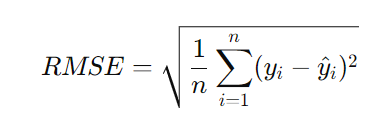
MAE là một số không âm, giá trị càng nhỏ thì mô hình càng chính xác.



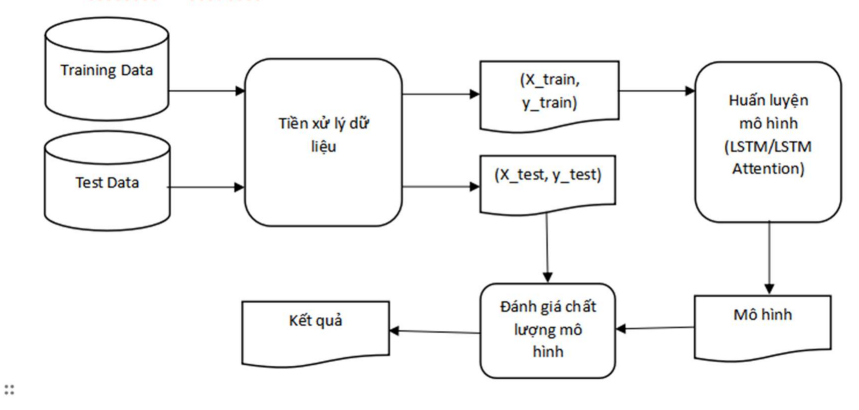
3.4 Độ đo RMSE(Căn bậc hai của sai số bình phương trung bình)

RMSE là độ đo lỗi phổ biến, nhấn mạnh vào các sai số lớn vì nó bình phương các sai số trước khi tính trung bình. RMSE lớn hơn khi có những sai số lớn, cho thấy mức độ nhạy cảm với các điểm dữ liệu bất thường.

RMSE là một số không âm, giá trị càng nhỏ thì mô hình càng chính xác.



Cách xây dựng mô hình dự báo mực nước:

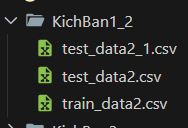


4.Mô hình dự báo mực nước

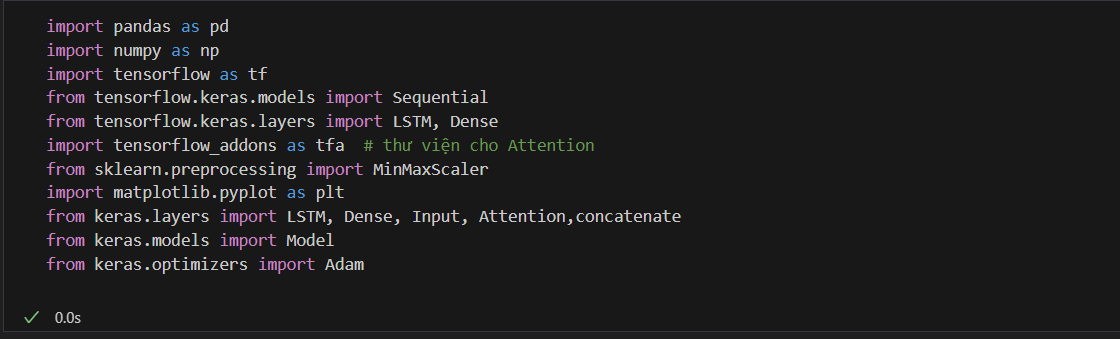
Sau quá trình nghiên cứu ta có thể xây dựng mô hình theo nhiều cách khác nhau :

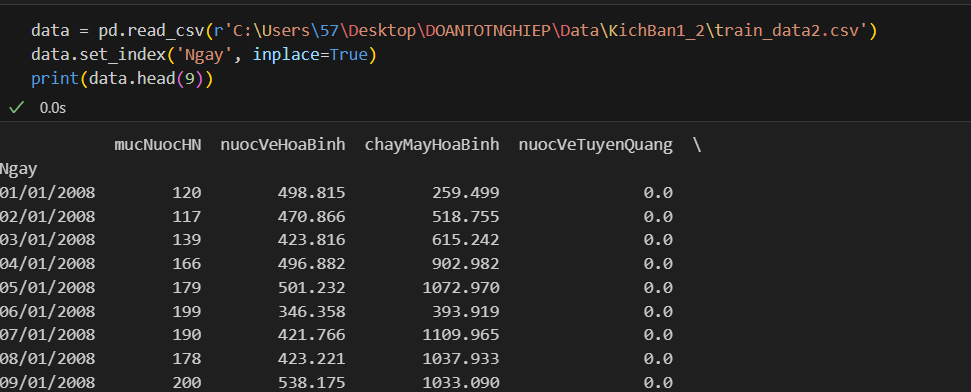
* Dùng mô hình với thuật toán LSTM
* Kết hợp LSTM với DropAttention
* Kết hợp LSTM với Multi-Head Attention
* Kết hợp LSTM với Global Self-Attention

Chuẩn bị dữ liệu mực nước

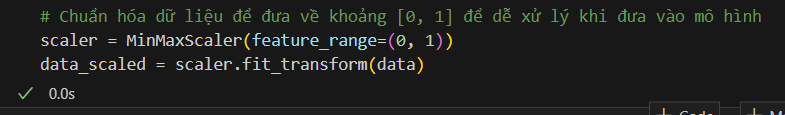


Bắt đầu huấn luyện mô hình trước tiên ta cần xử lý dữ liệu huấn luyện trước :

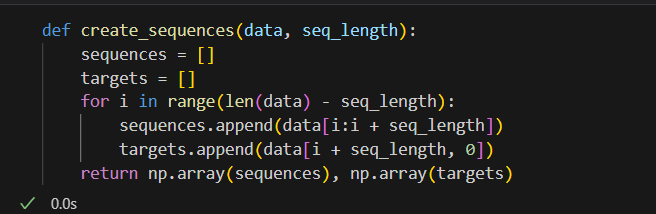




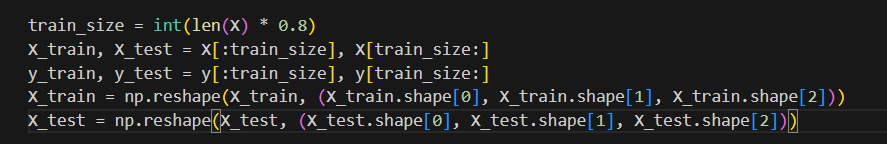
Sau đó ta chuẩn hóa những dữ liệu số này về dạng 0 đến 1 nhằm cho dễ dàng làm giá trị đầu vào dùng cho huấn luyện



Thiết lập những giá trị đầu vào và đầu ra cho mô hình cần để huấn luyện do đã có dữ liệu có sẵn:

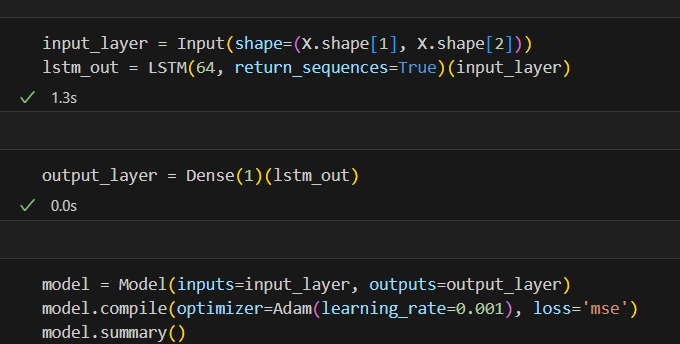




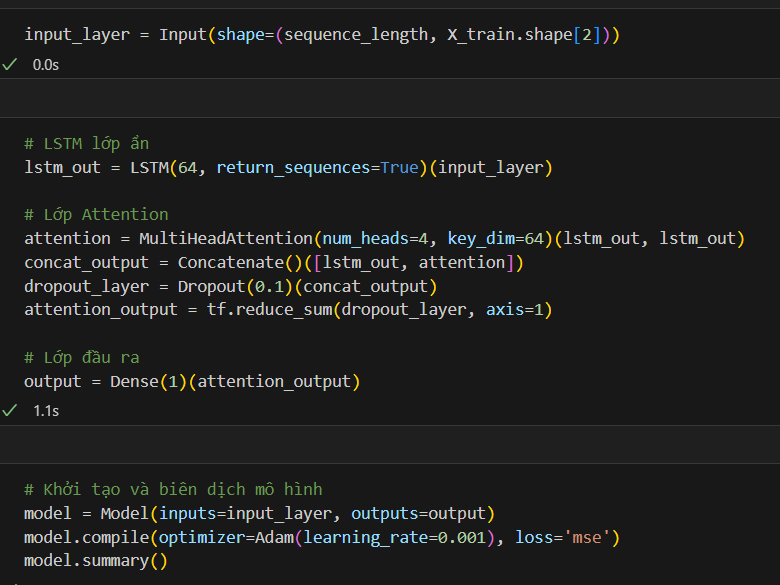


Sau đó bắt đầu xây dựng mô hình :

Mô hình LSTM :

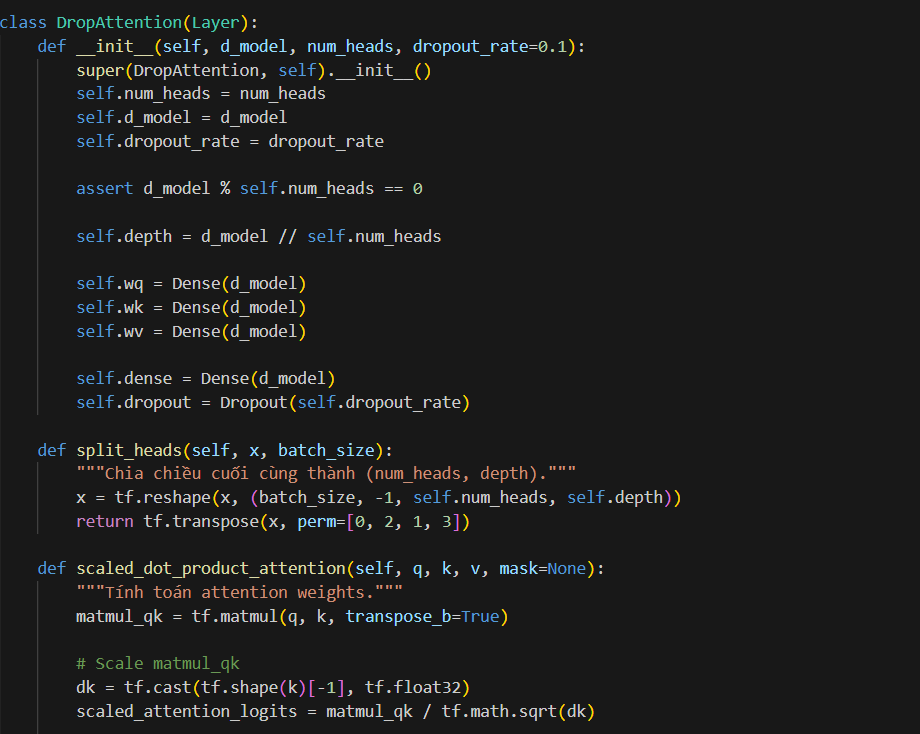


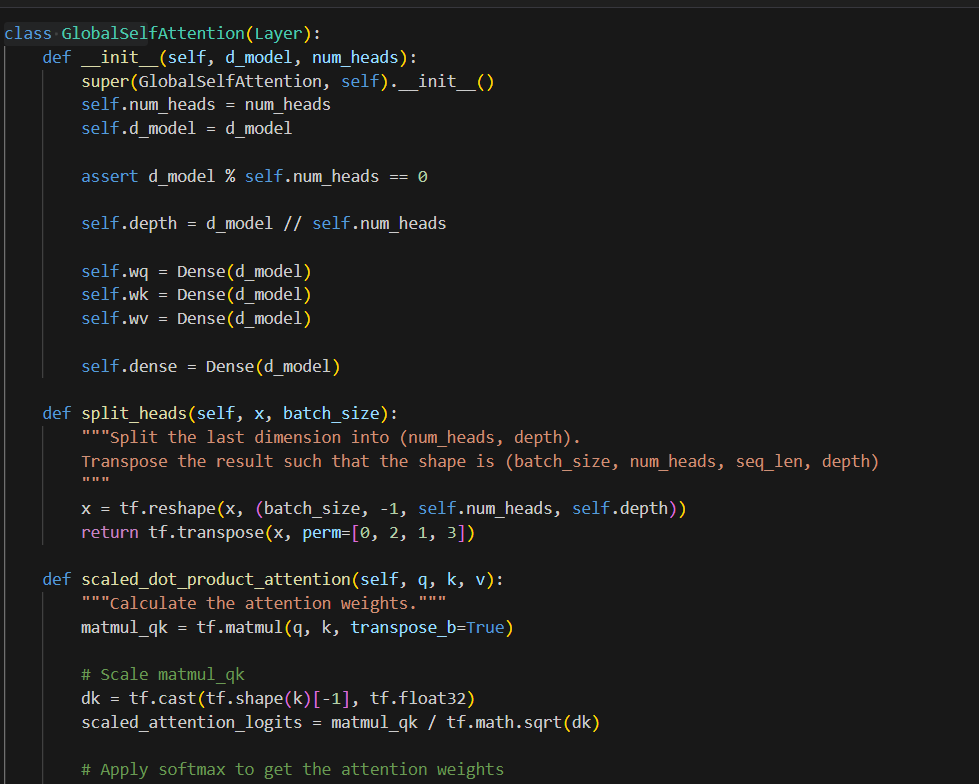
Mô hình kết hợp giữa LSTM với Multi-Head Attention



Mô hình kết hợp LSTM với DropAttention hoặc Global Self-Attention

Trong quá trình nghiên cứu tôi thấy 2 cơ chế này của Attention không có thư viện hỗ trợ làm do đấy phải tự cấu hình cho nó và đây là code cấu hình của cả 2 :



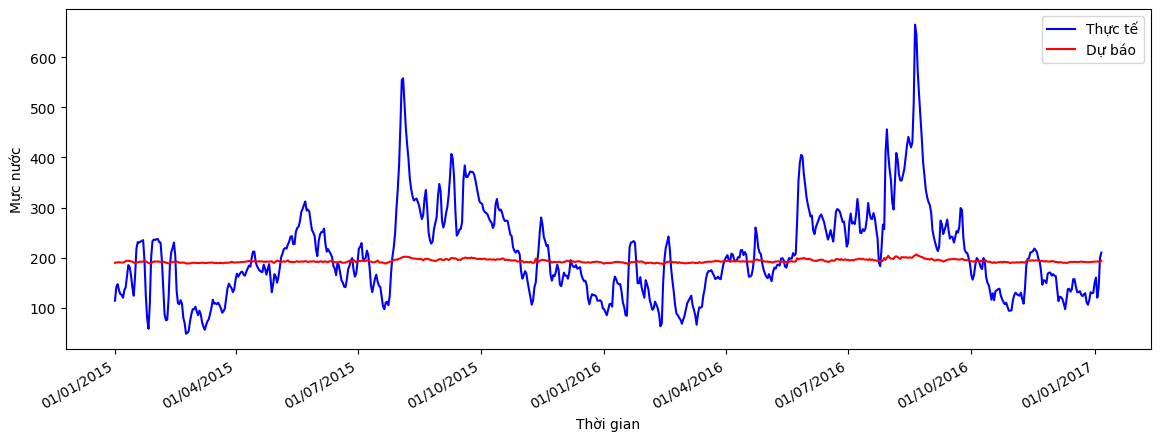


Sau đó huấn luyện mô hình theo từng mô hình và sau quá trình đấy tôi thu được độ chính xác của mỗi mô hình theo các cách khác nhau :

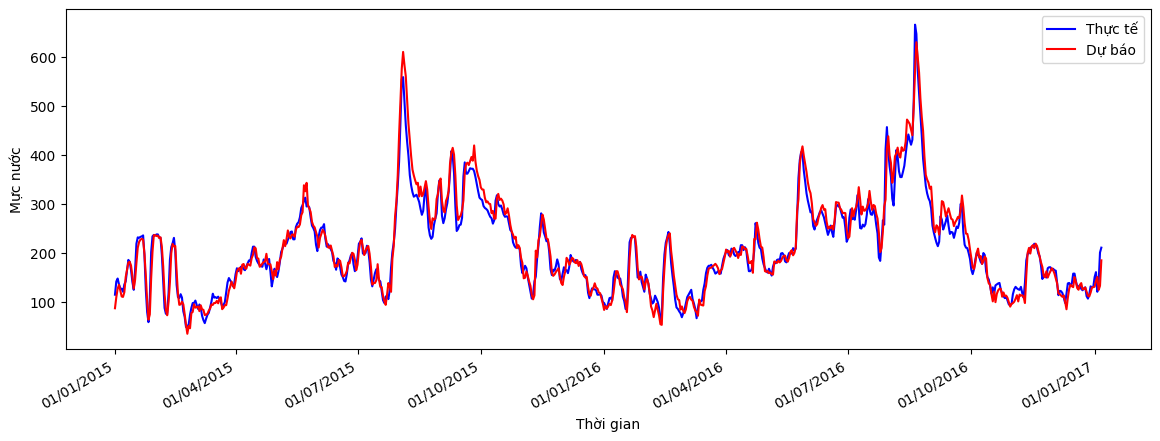
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | R² (R-squared) | NSE | MAE | RMSE |
| LSTM | 0.012210 | 0.0122102 | 0.10009 | 0.122471 |
| LTSM Drop | 0.96232 | 0.9623 | 0.017987 | 0.02391 |
| LSTM Multi | 0.9701 | 0.970 | 0.01524 | 0.02128 |
| LSTM Global | 0.9573 | 0.9573 | 0.01971 | 0.0254 |

Sau khi huấn luyện xong mô hình ta có thể test mô hình này với dữ liệu đã chuẩn bị từ trước và đây là kết quả :

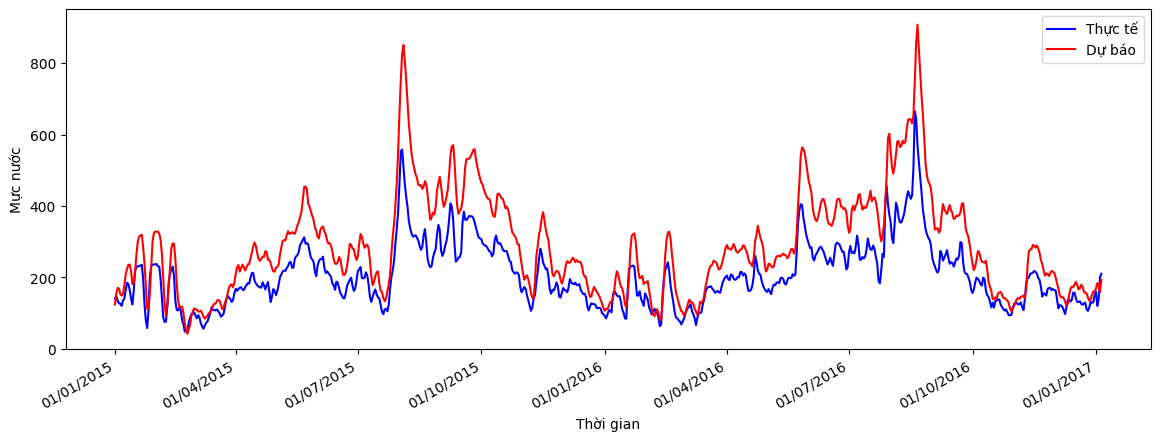
Mô hình LSTM :



Mô hình LSTM với DropAttention:



Mô hình LSTM với Global Self-Attention:



Mô hình LSTM với Multi-Head Attention:

