**Mạng trí nhớ ngắn hạn định hướng dài hạn**

**(LSTM - Long short term memory)**

* LSTM có cấu trúc gồm 4 tầng ẩn (3 sigmoid và 1 tanh) tương tác với nhau theo một cấu trúc đặc biệt:



****Hình 1: Sự lặp lại kiến trúc module trong mạng LSTM chứa 4 tầng ẩn (3 sigmoid và 1 tanh) tương tác****

**Các kí hiệu có thể diễn giải như sau:**

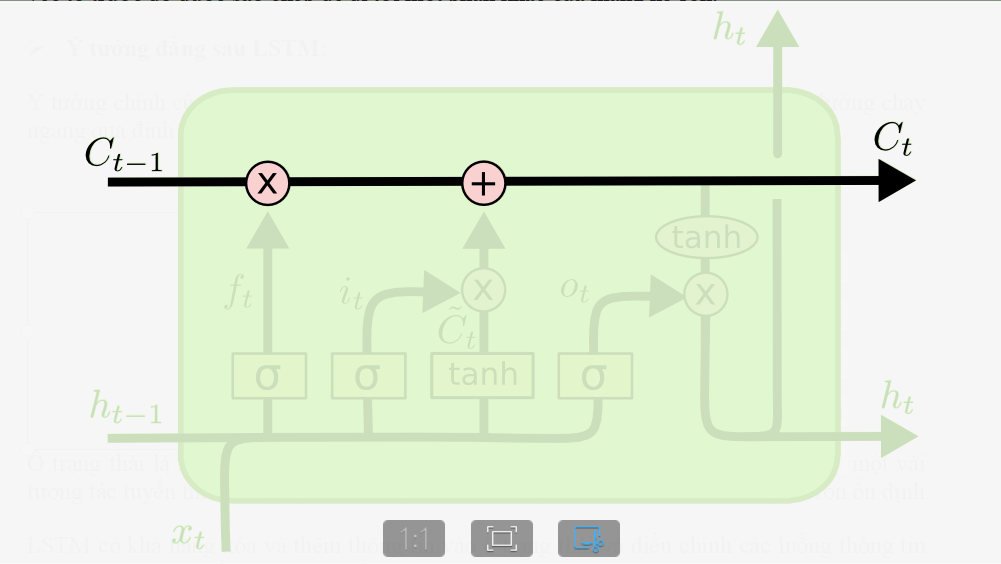


****Hình 2: Diễn giải các kí hiệu trong đồ thị mạng nơ ron (áp dụng chung cho toàn bộ bài)****

**Trong sở đồ tính toán trên, mỗi một phép tính sẽ triển khai trên một véc tơ. Trong đó hình tròn màu hồng biểu diễn một toán tử đối với véc tơ như phép cộng véc tơ, phép nhân vô hướng các véc tơ. Màu vàng thể hiện hàm activation mà mạng nơ ron sử dụng để học trong tầng ẩn, thông thường là các hàm phi tuyến sigmoid và tanh. Kí hiệu 2 đường thẳng nhập vào thể hiện phép chập kết quả trong khi kí hiệu 2 đường thẳng rẽ nhánh thể hiện cho nội dung véc tơ trước đó được sao chép để đi tới một phần khác của mạng nơ ron.**

* ****Ý tưởng đằng sau LSTM**:**

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



****Hình 3: Đường đi của ô trạng thái (cell state) trong mạng LSTM****

**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.**



****Hình 4: Một cổng của hàm sigmoid trong LSTM****

**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:****

1. 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.



****Hình 5: Tầng cổng quên (**forget gate layer**)****

1. **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**  **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.**



****Hình 6: Cập nhật giá trị cho ô trạng thái bằng cách kết hợp 2 kết quả từ tầng cổng vào và tẩng ẩn hàm tanh****

1. 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 là một giá trị mới được tính toán tương ứng với bao nhiêu được cập nhật vào mỗi giá trị trạng thái.



****Hình 7: Ô trạng thái mới****

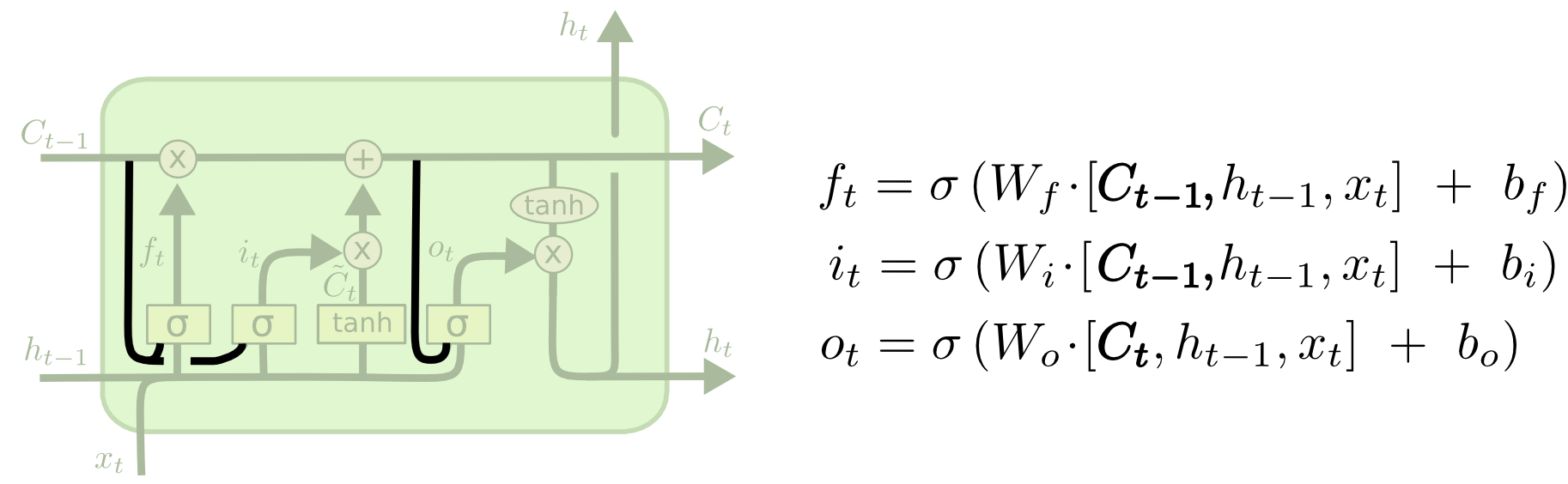
1. Cuối cùng cần quyết định xem đầu ra sẽ trả về bao nhiêu. Kết quả ở đầu ra sẽ dựa trên ô trạng thái, nhưng sẽ là một phiên bản được lọc. Đầu tiên, chúng ta chạy qua một tầng sigmoid nơi quyết định phần nào của ô trạng thái sẽ ở đầu ra. Sau đó, ô trạng thái được đưa qua hàm tanh (để chuyển giá trị về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



****Hình 8: Điều chỉnh thông tin ở đầu ra thông qua hàm tanh****

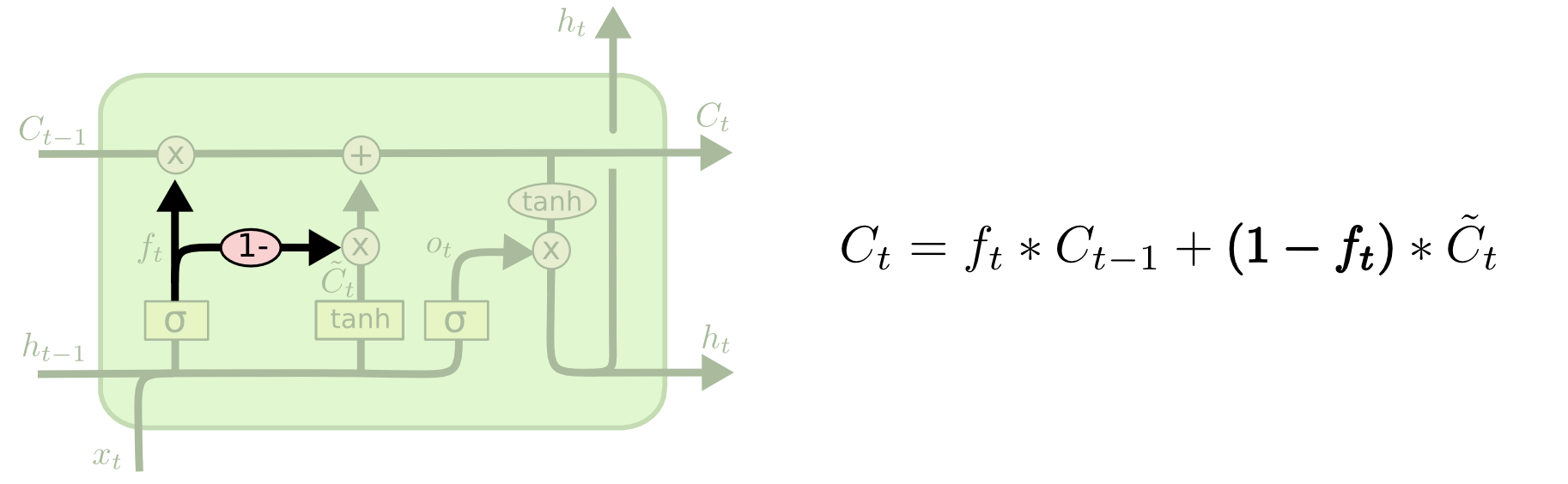
# Các biến thể của LSTM:

1. **Một trong những biến thể nối tiếng nhất của LSTM được giới thiệu bởi Gers & Schmidhuber (2000) thêm một kết nối ống tiểu (peehole connection) để các cổng có thể kết nối trực tiếp đến các ô trạng thái.**



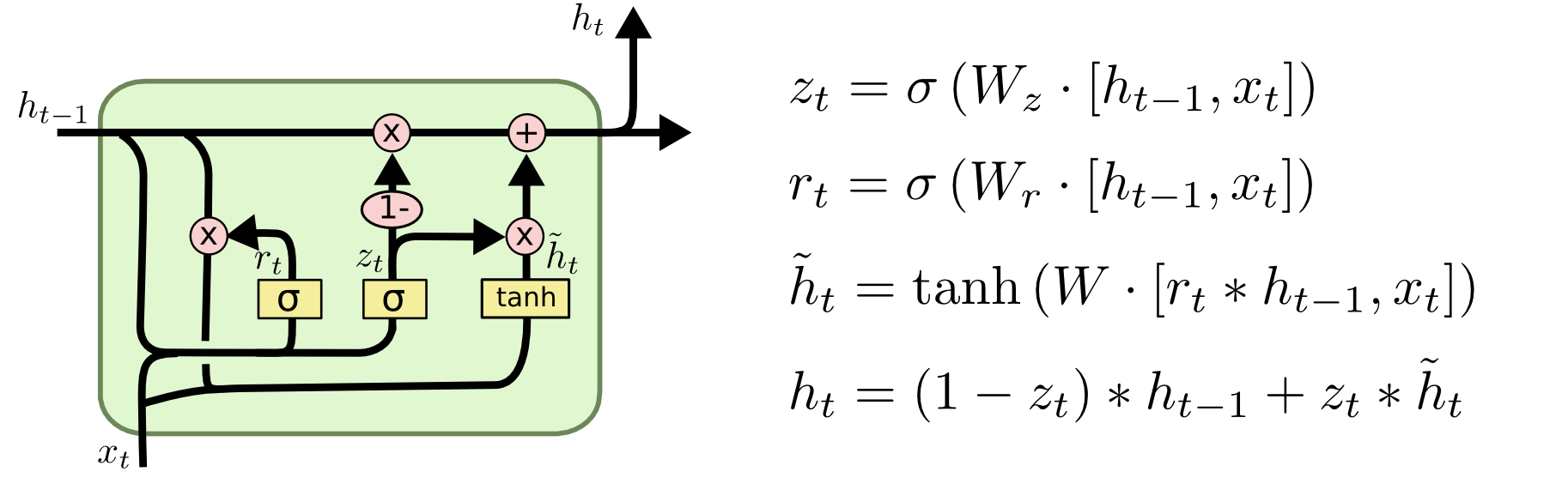
****Hình 9: Kết nối ống tiểu (**peehole**) liên kết trực tiếp ô trạng thái với các cổng****

1. **Một biến thể khác là sử dụng cặp đôi cổng vào và cổng ra. Thay vì quyết định riêng rẽ bỏ qua thông tin nào và thêm mới thông tin nào, chúng ta sẽ quyết định chúng đồng thời. Các thông tin chỉ bị quên khi chúng ta muốn cập nhập vào một vài thông tin mới.**



****Hình 10: Cấu trúc điều chỉnh thêm mới và bỏ qua thông tin đồng thời****

1. Một dạng biến thể khá mạnh khác của LSTM là cổng truy hồi đơn vị [(Gated Recurrent Unit - GRU)](https://arxiv.org/pdf/1406.1078v3.pdf) được giới thiệu bởi Cho, et al. (2014). Nó kết hợp cổng quên và cổng vào thành một cổng đơn gọi là cập nhật (update gate). Nó cũng nhập các ô trạng thái và trạng thái ẩn và thực hiện một số thay đổi khác. Kết quả của mô hình đơn giản hơn nhiều so với mô hình LSTM chuẩn, và đã trở nên khá phổ biến.



****Hình 11: Cấu trúc cổng truy hồi đơn vị (**GRU - Gated Recurrent Unit**)****

Chỉ có một số lượng nhỏ những biến thể LSTM là đáng chú ý. Rất nhiều những biến thể khác, như [kiến trúc cổng sâu RNN](https://arxiv.org/pdf/1508.03790v2.pdf) (Depth Gated RNN) của Yao, et al. (2015) hay kiến trúc [đồng hồ RNN](https://arxiv.org/pdf/1402.3511v1.pdf) (Clockword RNN) của Koutnik, et al. (2014) nhằm giải quyết vấn đề phụ thuộc dài hạn (long - term depencies).