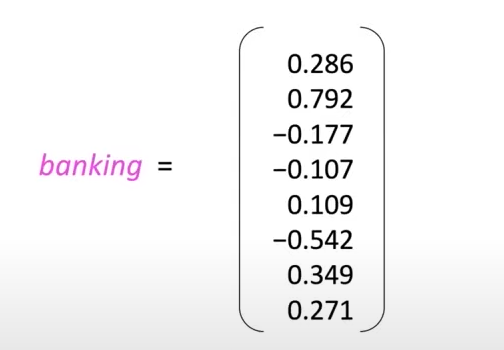
**Báo cáo tiến trình học tập tuần 5**

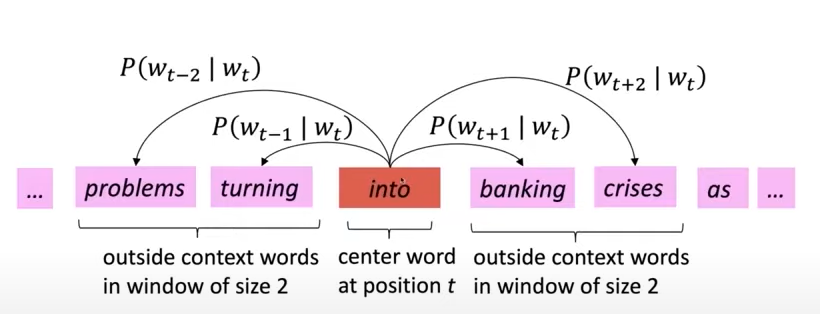
1. **Word2vec**
2. **Word embedding (word vectors)**

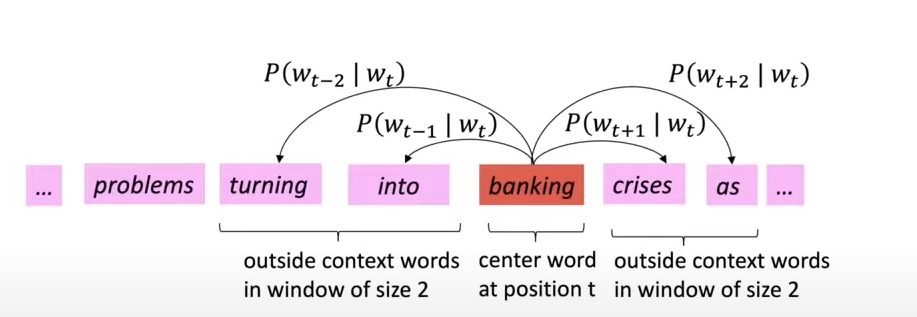
* Chúng ta sẽ xây dựng một vector đặc trưng cho mỗi từ, và vector này được chọn (học ra) sao cho nó **giống** với vector của những từ thường xuất hiện trong **ngữ cảnh tương tự.**



1. **Word2vec:**
2. **Tổng quan**

* Chúng ta có một **corpus lớn (“**tập văn bản)
* Mỗi từ trong đó được biểu diễn là một vector (khởi tạo random)
* Duyệt qua từng vị trí ***t*** trong văn bản, tại đó có một từ trung tâm ***c*** và các từ ngữ cảnh (“outside”) ***o***.
* Sử dụng **độ tương đồng của** vector từ cho c và o **để tính xác suất** của o khi biết c (hoặc ngược lại).
* **Tiếp tục điều chỉnh các vector từ** để **tối đa hóa xác suất này.**





1. **Objective function (hàm mục tiêu)**

* Với mỗi vị trí t = 1,…,T, dự đoán các từ ngữ cảnh trong một cửa sổ có kích thước cố định là m, với từ trung tâm là wt​. Khả năng xảy ra của dữ liệu:

Note:

* Mục tiêulà chọn sao cho đạt max, làm cho cặp từ gần nhau sau nhiều lần đào tạo gần nhau trong không gian vector.
* Việc maximize có ý nghĩa là***làm cho dữ liệu thật trở nên ít bất ngờ nhất đối với mô hình.***

Hàm objective function là trung bình âm của log likelihood:

Note:

* Việc lấy giá trị trung bình loss bằng cách chia cho T (nghiệm tối ưu không đổi), giúp cho loss có giá trị ổn định và so sánh được giữa các corpus có kích thước khác nhau.
* Tránh làm cho giá trị hàm loss chỉ phản ánh dữ liệu nhiều hay ít thay vì ***mức độ khớp dữ liệu.***

1. **Cách tính xác suất**

* Chia thành 2 vector cho mỗi từ **w:**
* khi **w** là một center word
* khi **w** là một context word
* Với center word c và context word o, ta có:

Note:

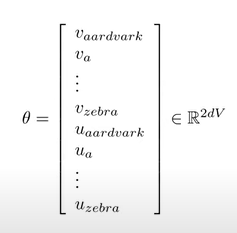
* Tích vô hướng giúp đo mức độ tương đồng giữa 2 vector (Dot product càng lớn → 2 từ càng có mối quan hệ ngữ cảnh mạnh → xác suất càng cao)
* Lấy hàm mũ để biến mọi giá trị thành dương.

1. **Theta (**

* Tham số trong Word2vec chính là **vector embedding.**

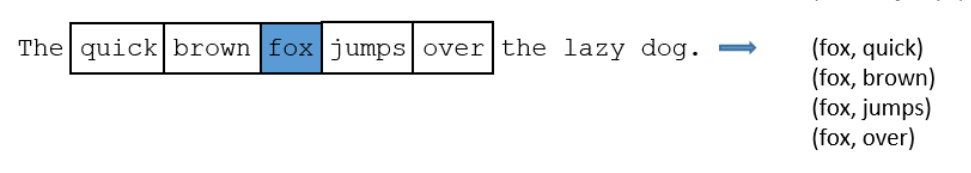
Cụ thể:

* Mỗi từ có 2 vector, tập vocab có V từ => Tổng số vector là 2V, mỗi vector có d chiều, nên tổng tham số là:

****

1. **Mô hình Skip-Gram**

* Mô hình skip-gam giả định rằng một từ có thể được sử dụng để sinh ra các từ xung quanh nó trong một chuỗi văn bản.

****

* Xét ví dụ bên trên với từ đích là “fox” và các từ ngữ cảnh là “quick”, “brown”, “jumps” và “over”. Việc dự đoán xác suất xảy ra các từ ngữ cảnh khi biết từ đích được mô hình hóa bởi:



* Ta có thể giả sử rằng sự xuất hiện của một từ ngữ cảnh khi biết từ đích *độc lập* với các từ ngữ cảnh khác để xấp xỉ xác suất trên đây bởi:



* Tổng quát:
* Để tránh sai số khi tích xác suất quá nhỏ:
* Với center word c và context word o, ta có:

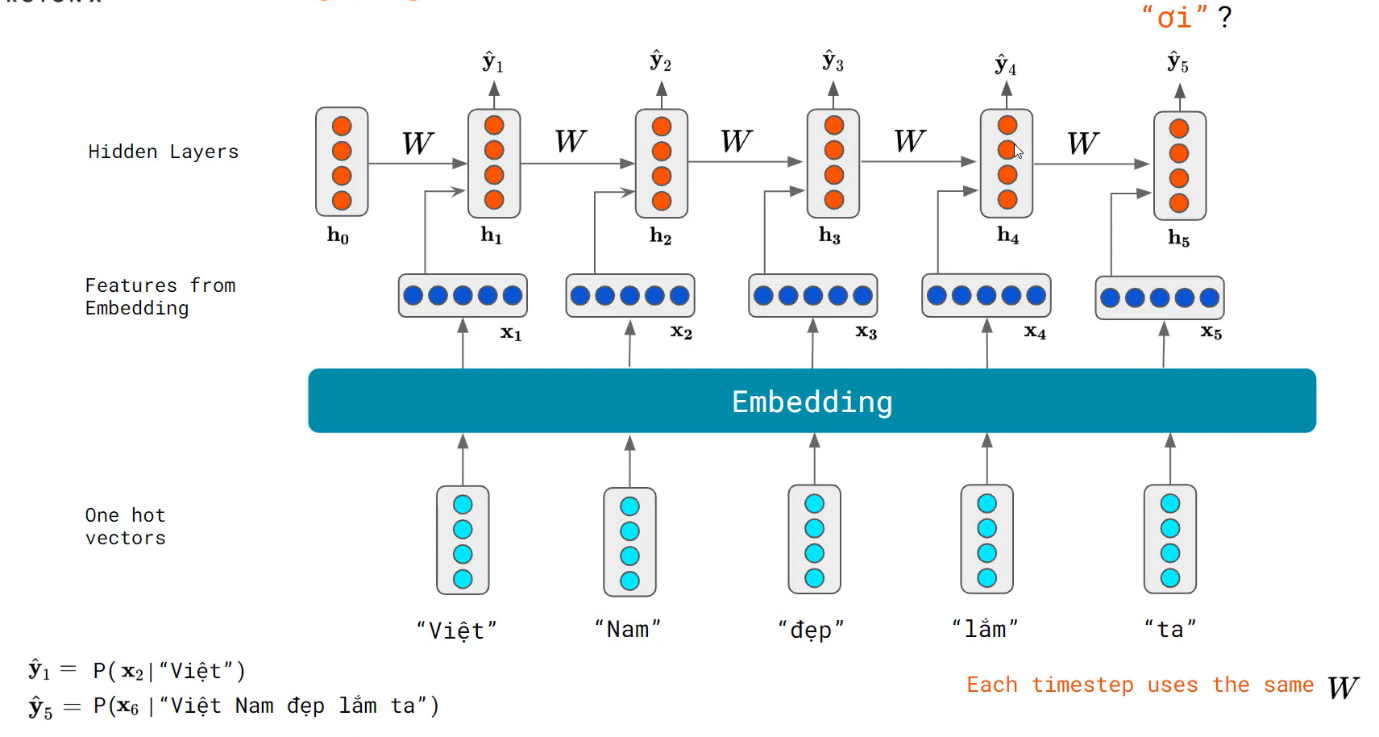
1. **Mô hình CBOW (*Continuous bag of words)***

* Khác biệt lớn nhất là mô hình CBOW giả định rằng từ đích trung tâm được tạo ra dựa trên các từ ngữ cảnh phía trước và sau nó trong một chuỗi văn bản.

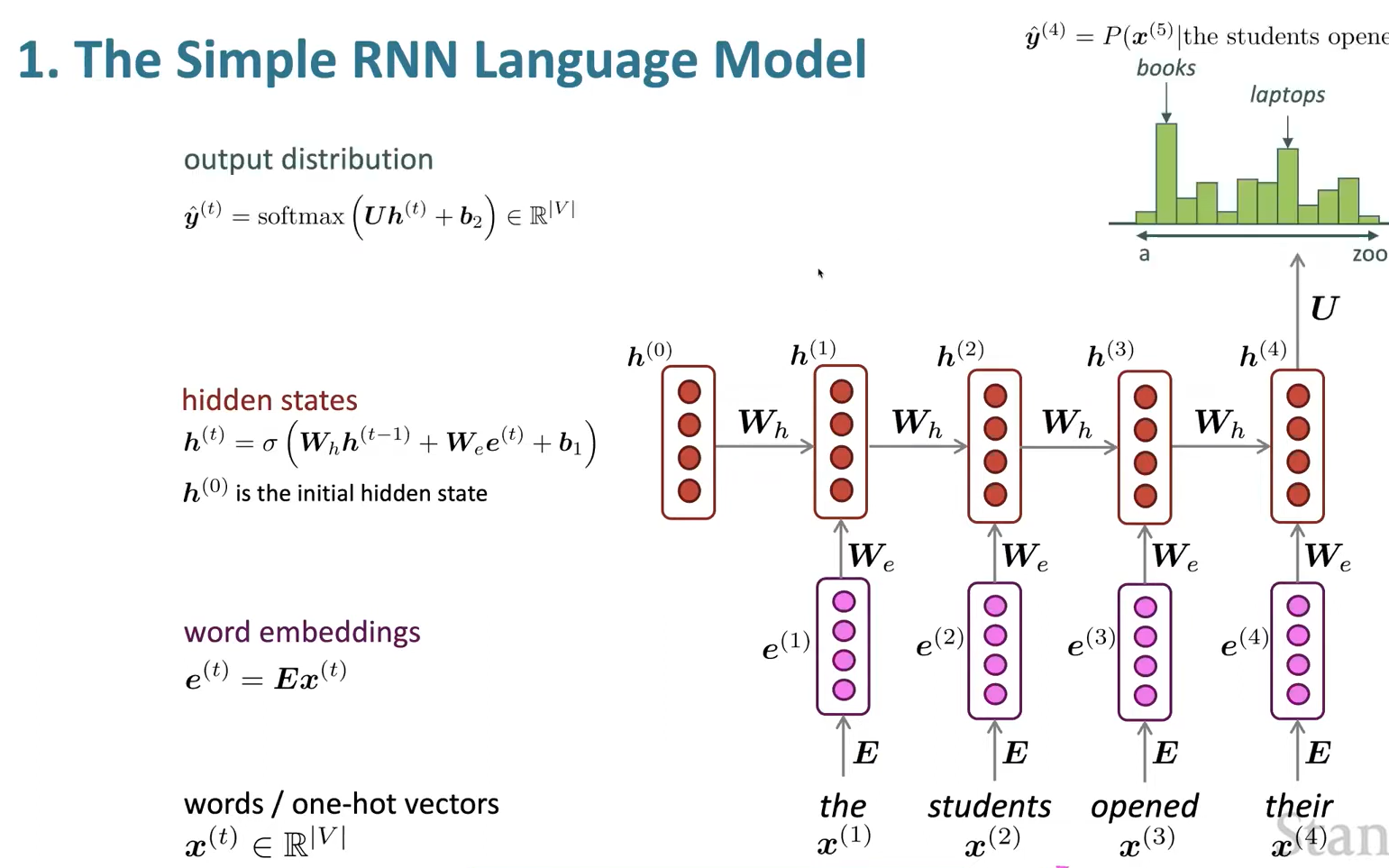


* Vì có nhiều từ ngữ cảnh trong điều kiện, chúng thường được đơn giản hóa bằng cách lấy một từ “trung bình” làm đại diện.
* Với center word c và là chỉ số các từ ngữ cảnh, ta có:

1. **RNN (Recurrent Neural Network)**
2. **Giới thiệu**



* Tạo các vector embedding từ các vector one-hot thông qua lớp embedding.
* Các vector sẽ lưu thông tin của và , và sẽ xem là từ nào có xác suất cao nhất khi có các từ đứng trước như đã được lưu trong .

****

1. **Loss function (Cross-Entropy)**

* Ta có một chuỗi từ:
* Sau đó RNN đọc lần lượt từng từ, sinh ra phân phối trên toàn bộ vocab, đó là phân phối xác suất của tất cả các từ có thể xuất hiện ở vị trí t+1.
* Loss tại bước t:
* Vì được biểu diễn bằng vector one-hot, mà vị trí đúng tại t+1, nên ta có thể rút gọn:
* Lấy trung bình toàn chuỗi:

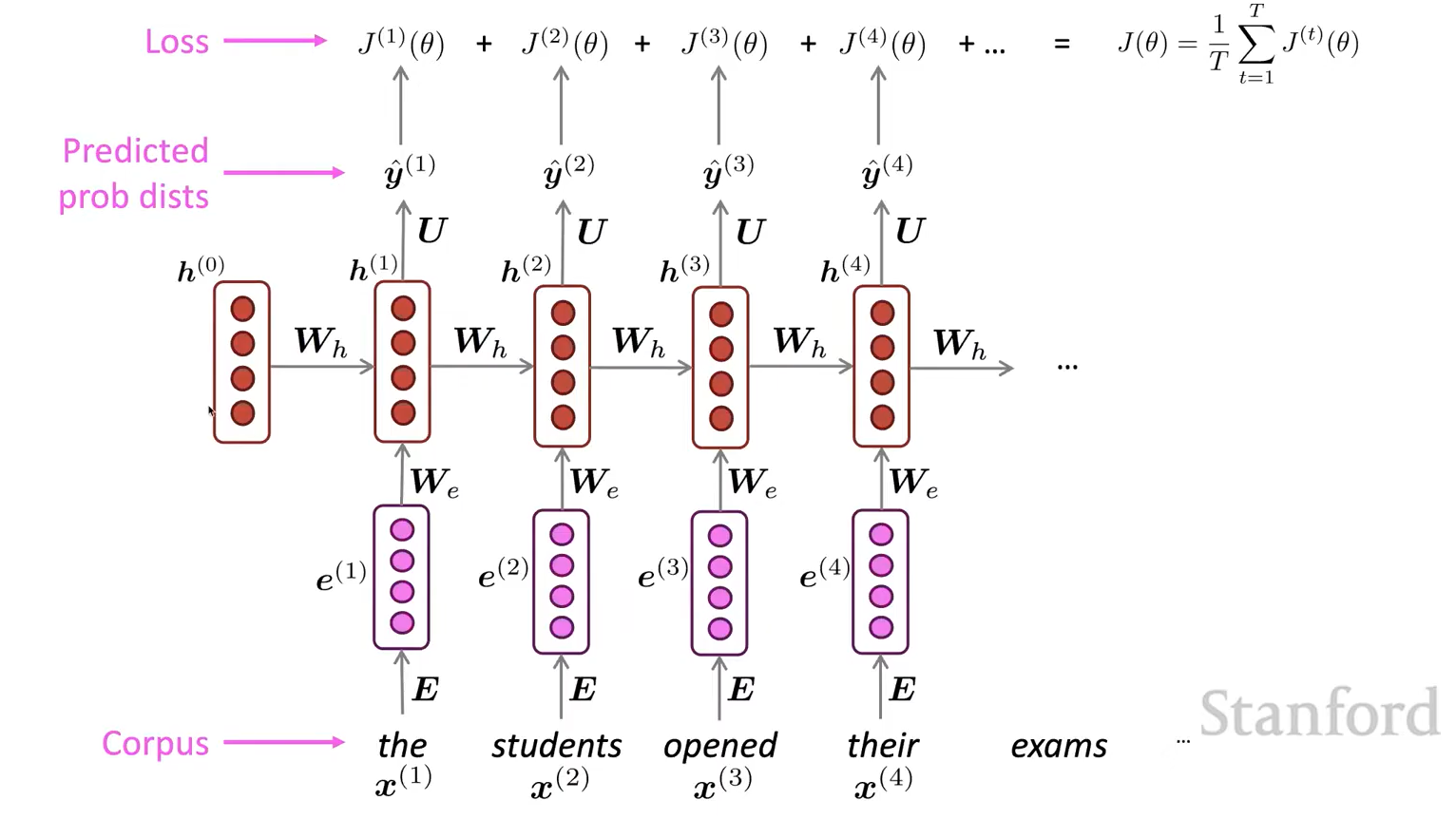
1. **Ưu điểm và nhược điểm**

Ưu điểm:

* Có thể làm việc với các đoạn văn bản dài (vì tham số không phụ thuộc vào độ dài văn bản)

Nhược điểm:

* Vì là tính tuần tự, nên RNN sẽ chạy khá chậm
* Khi một câu quá dài thì dễ dẫn đến vanishing (dẫn về 0) và exploding(rất lớn) gradient.

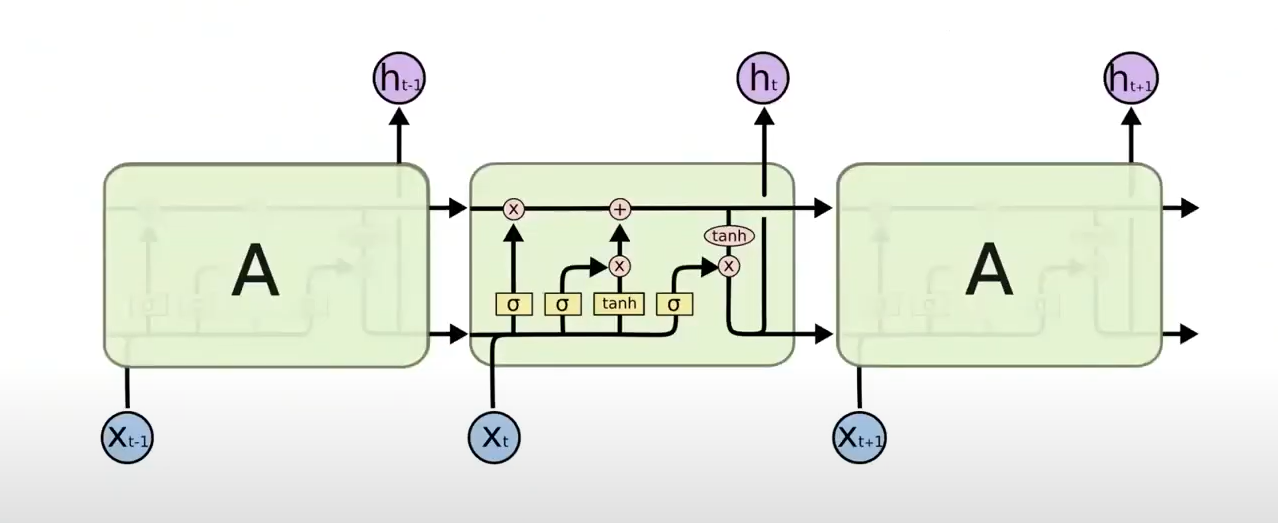
****

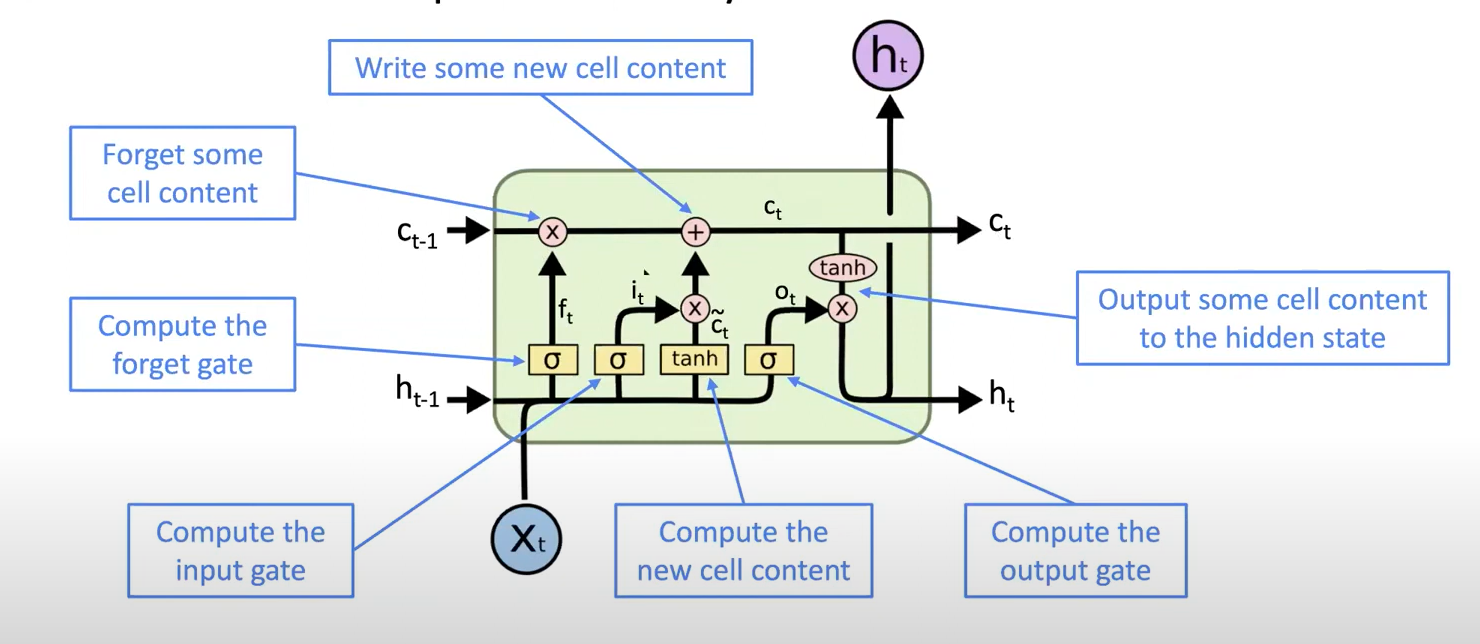
1. **Long Short-Term Memory (LSTM)**
2. **Giới thiệu**

* LSTM là một loại RNN đặc biệt để giải quyết vấn đề vanishing gradient.
* Ở mỗi bước thời gian t, có hai loại trạng thái:
  1. Hidden State
* Mang thông tin ngắn hạn
  1. Cell state
* Mang thông tin dài hạn (long-term memory).
* LSTM có thể đọc, xóa và ghi thông tin bởi cell state với các cổng tương ứng là Output gate, Forget gate và Input gate.
* Mỗi phần tử của gate có thể nhận giá trị giữa 0 và 1:
  + - 0 = đóng
    - 1 = mở
    - Trung gian = cho qua một phần.

1. **Các phương trình trong LSTM**

* **Forget gate :** Quyết định giữ bao nhiêu % thông tin cũ từ cell state .
* **Input gate :** Quyết định thêm bao nhiêu % thông tin mới vào cell state.
* **Output gate :** Quyết định xuất ra bao nhiêu % từ cell state thành hideen state
* Trước khi viết vào cell state, LSTM tạo ra một vector ứng viên, đây là thông tin mới được thêm vào cell:
* Cập nhật cell state: cell state kết hợp phần giữ lại từ quá khứ và thêm mới từ hiện tại:
* Hidden sate: từ cell state mới, chọn phần thông tin để xuất ra:

****



1. **LSTM giải quyết vanishing gradient như thế nào?**

* LSTM có thêm cell state, các gates trong cell sate kiểm soát việc giữ lại, thêm vào, hay loại bỏ thông tin.
* Dù W có xu hướng làm gradient quá to hoặc quá nhỏ, thì gates và cell state giúp **ổn định hơn**.
* Điều này giúp cho thông tin được kiểm soát không bị “pha loãng” hay quá “đặc”.