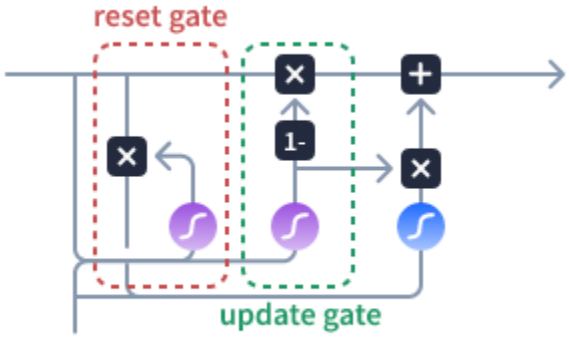
UIT - CS431.O12.KHCL 14/11/2023 Task
Nguyễn Hoàng Tân. (21521413)

Vanishing Gradient là hiện tượng gradient bị "bốc hơi" trong quá trình lan truyền ngược khi huấn luyện các mạng *deep neural network*, do đó sẽ không có sự cập nhật tham số nào ở đây cả. Vấn đề này gây ra trở ngại rất lớn cho các mạng muốn học các long-range dependencies. Gated Recurrent Units (GRU) và Long Short-Term Memory (LSTM) là các nhánh của *recurrent neural network (RNN)* được sinh ra để giải quyết vấn đề này.

Problem 1. *How GRU solves vanishing gradient ?*

Solution. Trước hết hãy xem qua cấu trúc của GRU. Một GRU có 2 cổng, cổng reset r_t và cổng update z_t, 2 cổng này phụ trách việc kiểm soát luồng thông tin ra vào ô nhớ. Hidden state h_t được update theo các cổng này và input ở mỗi step.



Hình 1: GRU Architecture

Chúng ta có các công thức của GRU như sau:

1. Cổng reset: $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$
2. Cổng update: $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$
3. Ô nhớ hiện tại: $\tilde{h_t} = tanh(W \cdot [r_t \bigodot h_{t-1}, x_t])$
4. Hidden state: $h_t = (1 - z_t) \bigodot h_{t-1} + z_t \bigodot \tilde{h_t}$

Dưới đây là cách mà GRU có thể giảm tác động của vanishing gradient:

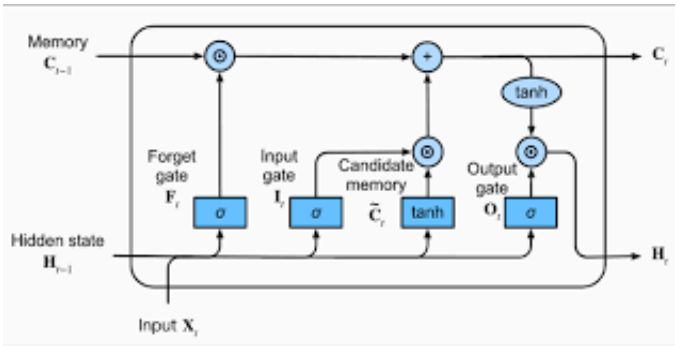
- Cổng Update (z_t): Cổng update z_t là nơi quyết định xem trạng thái trước đó h_{t-1} sẽ được chuyển đến trạng thái hiện tại h_t như thế nào. Khi z_t gần bằng 1, nó cho phép thông tin được chuyển qua mà không cần thay đổi nhiều. Khi z_t gần bằng 0, thì trạng thái mới $\tilde{h_t}$ quan trọng hơn nhiều. Vì vậy, kể cả khi z_t gần bằng 0, gradient vẫn có thể đi qua cổng z_t trong quá trình lan truyền ngược.

- Cổng Reset (r_t): Ngược lại, cổng reset r_t kiểm soát trạng thái trước đó h_{t-1} nên bị bỏ qua bao nhiêu khi tính toán $\tilde{h_t}$. Nếu r_t gần bằng 0, nó cho phép model hoàn toàn bỏ qua trạng thái trước, từ đó học được các short-term dependencies một cách hiệu quả.

Sự phối hợp giữa cổng reset và update cho phép GRU update bộ nhớ một cách có chọn lọc, cho phép nó học cả short-term and long-term dependencies một cách hiệu quả hơn mạng RNN truyền thống. □

Problem 2. *How does LSTM prevent the vanishing gradient problem ?*

Solution. Long Short-Term Memory networks (LSTMs) được thiết kế để bao quát các long-term dependencies dùng trong dữ liệu dạng chuỗi. Để làm được điều này, LSTM sử hữu một thiết kế ô nhớ phức tạp hơn nhiều so với mạng RNN truyền thống.



Hình 2: LSTM Architecture

Các thành phần quan trọng của LSTM bao gồm, *cell state, input gate, forget gate* and *output gate*:

1. Input Gate: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t])$
2. Forget Gate: $f_t = \sigma(W_f \cdot [h_{t-1}, x_t])$
3. Cell State Update: $\tilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t])$
4. Output Gate: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t])$
5. Hidden State Update: $h_t = o_t \bigodot tanh(C_t)$

Dưới đây là cách mà LSTM có thể giảm bớt ảnh hưởng của vanishing gradient:

- Forget Gate(f_t): Cổng Forget quyết định bao nhiêu phần của Cell State trước đó nên được giữ lại. Nếu f_t gần bằng 1, LSTM sẽ giữ lại phần lớn cell state trước đó, cho phép ghi nhớ các long-term dependencies. Điều này hạn chế vanishing gradient bằng cách maintain thông tin suốt các time steps.
- Cell State Update ($\tilde{C_t}$): Cell này sẽ bao quát các thông tin mới từ input x_t, và hidden state trước đó h_{t-1}. Hàm activation *tanh* đảm bảo rằng các giá trị sẽ được scaled để nằm trong khoản -1 và 1, từ đó việc lan truyền gradient sẽ trở lên dễ dàng hơn trong quá trình *backpropagation.*

□