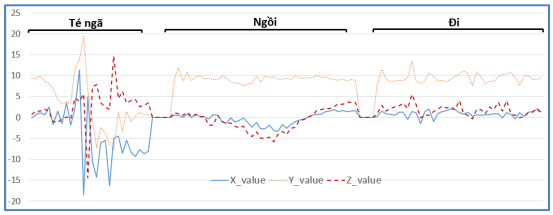
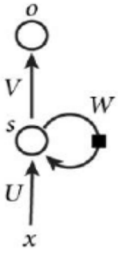


Hình 5: Các giá trị x, y, z tương ứng với các hành vi té ngã, ngồi, đi

Đối với thao tác té ngã thì sự biến thiên của *x*, *y*, và *z* là rõ rệt nhất. Đối với người cao tuổi thì các thao tác thường chậm rãi nên sự biến thiên của *x*, *y* và *z* khi ngồi xuống là khá ít. Đối với hành vi đi lại thì *x* và *z* ít biến động trong khi *y* là biến động rõ ràng nhất.

3.2 Xây dựng mô hình nhận dạng bằng mạng Long Short-Term Memory

Long Short-Tem Memory (LSTM) là một kiến trúc học sâu được coi như là một sự mở rộng của Mạng neural hồi qui (Recurrent Neural Network, RNN) với khả năng học các phụ thuộc dài hạn (long-term dependencies) (Schmidhuber and Hochreiter, 1997; Mikolov *et al.*, 2014; Rojas, 2013). Đây là một mô hình mạng có trí nhớ, có khả năng "nhớ " được các thông tin đã tính toán trước đó. Kết quả tại thời điểm hiện tại không những phụ thuộc vào đầu vào tại thời điểm hiện tại mà còn phụ thuộc vào kết quả tính toán của các thành phần ở những thời điểm trước. Do đó, kiến trúc mạng này được gọi là mạng "hồi qui" (recurrent).



Hình 6: Quá trình xử lý thông tin trong RNNs

Quá trình xử lý thông tin trong mạng hồi qui được mô tả trong Hình 6. Trong đó, x_t là input tại thời điểm thứ *t*, o_t là output tại thời điểm thứ *t*, S_t là trạng thái ẩn (hiden state) tại thời điểm thứ *t*, chính là "bộ nhớ" của mạng. S_t được tính dựa trên các trạng thái ẩn trước kết hợp với input tại thời điểm thứ *t*. Quá trình này có thể được biểu diễn bằng mô hình toán sau (Mikolov *et al.*, 2014):

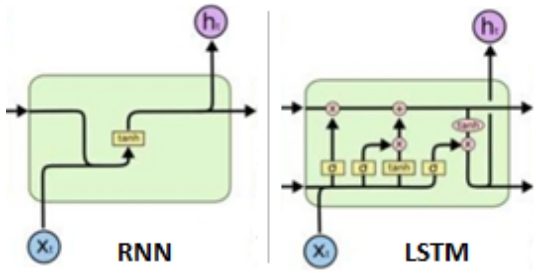
$$S_t = f(Ux_t + Ws_{t-1})$$

với (*U, V, W*) là ba tham số của mạng.

Hàm *f* thường được sử dụng nhất là hàm tanh hoặc hàm RELU (Rojas Raúl, 2013).

Với khả năng "nhớ" được, đặc điểm chung của mạng neural hồi qui là có khả năng xử lý thông tin dạng chuỗi (sequential data) và các dữ liệu thời gian (temporal data).

Về lý thuyết, RNNs có thể nhớ được thông tin của chuỗi có chiều dài bất kỳ nhưng trong thực tế thì mô hình này chỉ có khả năng nhớ được thông tin ở một vài bước trước đó (Schmidhuber and Hochreiter, 1997). Do đó, để tăng khả năng nhớ thì bước xử lý lặp của LSTM sử dụng 4 lớp thay vì một lớp như RNN. Hình 7 mô tả sự khác nhau giữa cấu trúc 1 cell (module lặp) trong RNN và LSTM. Mấu chốt của khả năng "nhớ lâu" của LSTM là cấu trúc "trạng thái nhớ" (cell state), là đường kẻ ngang phía trên trong module lặp. Các thông tin có thể được thêm hoặc bớt vào cell state, dựa trên qui định của các cổng (gate), là các phép toán được đặt trong vòng tròn trong cell state.



Hình 7: Cấu trúc module lặp của RNN và LSTM

Trong nghiên cứu này, chúng tôi sử dụng kiến trúc học sâu LSTM với 64 lớp ẩn. Hàm mất mát (loss funtion) được sử dụng trong nghiên cứu này là hàm L2-norm (Least Squared Error, LSE). Hàm này cực tiểu hóa sự chênh lệch giữa giá trị mục tiêu y_i và giá trị dự đoán $f(x_i)$ theo công thức sau:

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Ouput tại thời điểm thứ *t* được tính dựa theo hàm softmax theo công thức sau:

$$O_t = \text{softmax}(VS_t)$$

Để tối ưu hóa mô hình, chúng tôi sử dụng thuật toán Adam Optimization (Kingma and Ba, 2014). Đây là sự mở rộng của thuật toán Stochastic Gradient Descent. Thuật toán này sử dụng 1 learning rate cho cả quá trình học và đường trung bình (moving average) của các dữ liệu gần nhất sẽ có trọng số cao hơn các dữ liệu đã xử lý xa hơn trước đó. Phương pháp này được gọi là đường trung bình siêu nhanh (exponential moving average), cho phép xử lý tốt hơn đối với các thay đổi gần nhất trong mô hình (Wu and Wei, 1989).