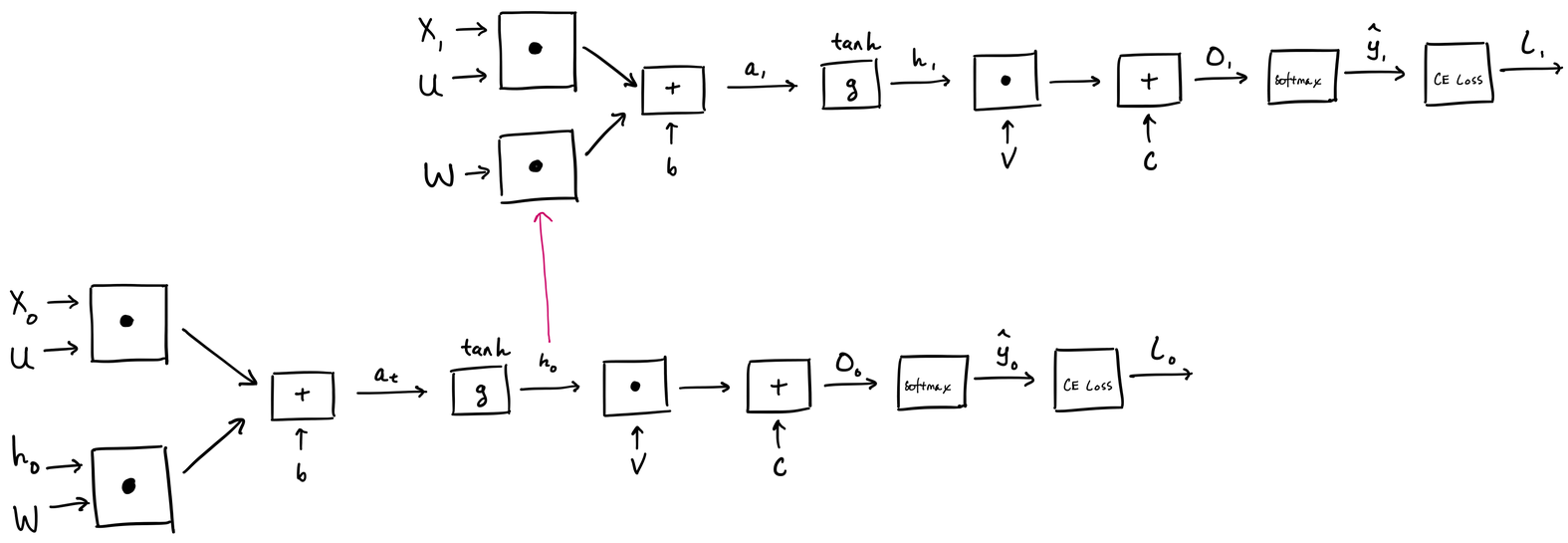
RNN computation graph

Tuesday, October 24, 2023 11:45 AM



Feed Forward

- Dot product between input X and input weight U
- Dot product between initial hidden state h_0 & weight matrix W
- Summation between output of both dot product operations & bias b, resulting in logits a_t
- Pass a_t into the activation node g which uses the hyperbolic tangent (tanh) as activation function, resulting in h_t
- Dot product of h_t & V, added with C (this represents a fully connected layer after the new hidden state is calculated), outputting o_t
- Pass o_t to softmax to output $\hat{y}_t$
- Pass $\hat{y}_t$ to CE_{loss} node to get L_t, which represents the loss for the branch at time t

$$a_t = Ux_t + Wh_{t-1} + b$$
$$h_t = \tanh(a_t)$$
$$o_t = Vh_t + c$$
$$\hat{y}_t = \text{softmax}(o_t)$$
$$L_t = CE(\hat{y}_t, y_t)$$

Dimensionality

Input

X_t : [batch_size, input_dim]

Weight matricies & Biases

U : [input_dim, hidden_dim]

W : [hidden_dim, hidden_dim]

b : [1, hidden_dim]

V : [hidden_dim, output_dim]

c : [1, output_dim]

Hidden State

h_t : [batch_size, hidden_dim]

Output Before Softmax

o_t : [batch_size, output_dim]

Predicted output & True output

$\hat{y}_t$ & y_t : [batch_size, output_dim]

Loss

L_t : [batch_size, 1]