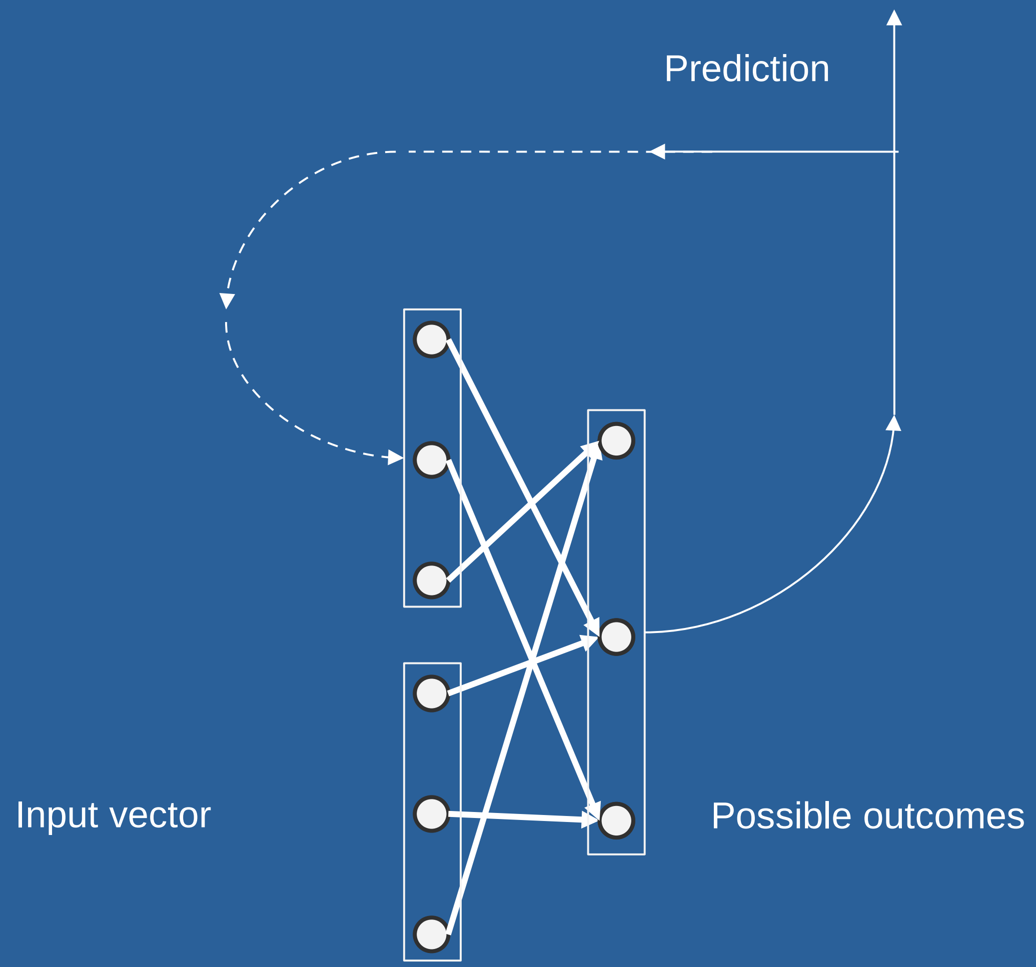
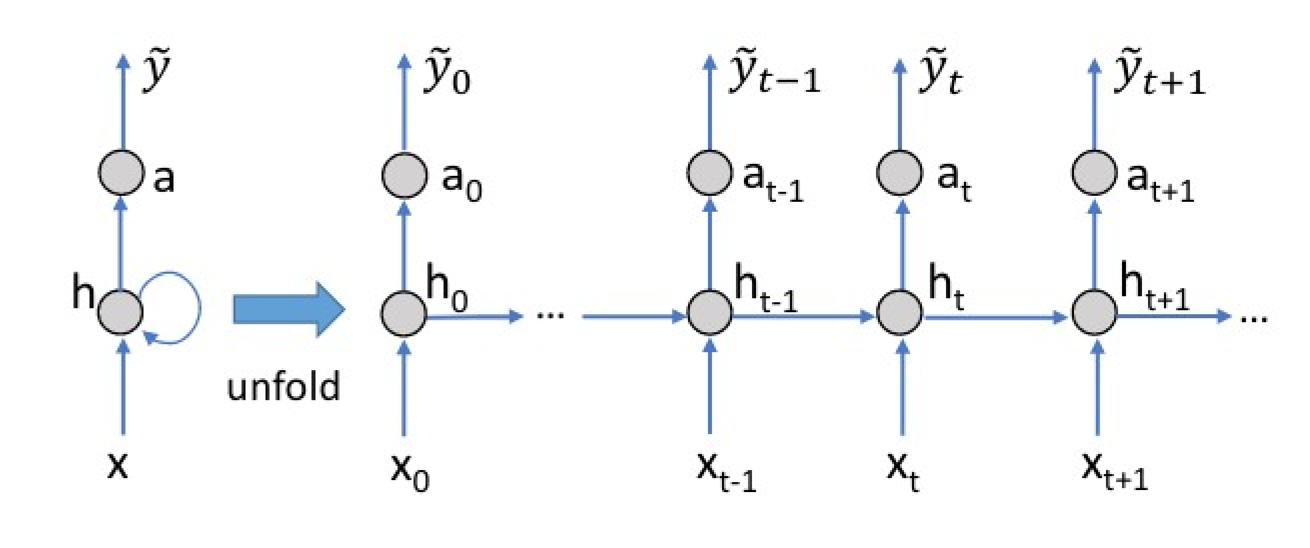
Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM)

RNN





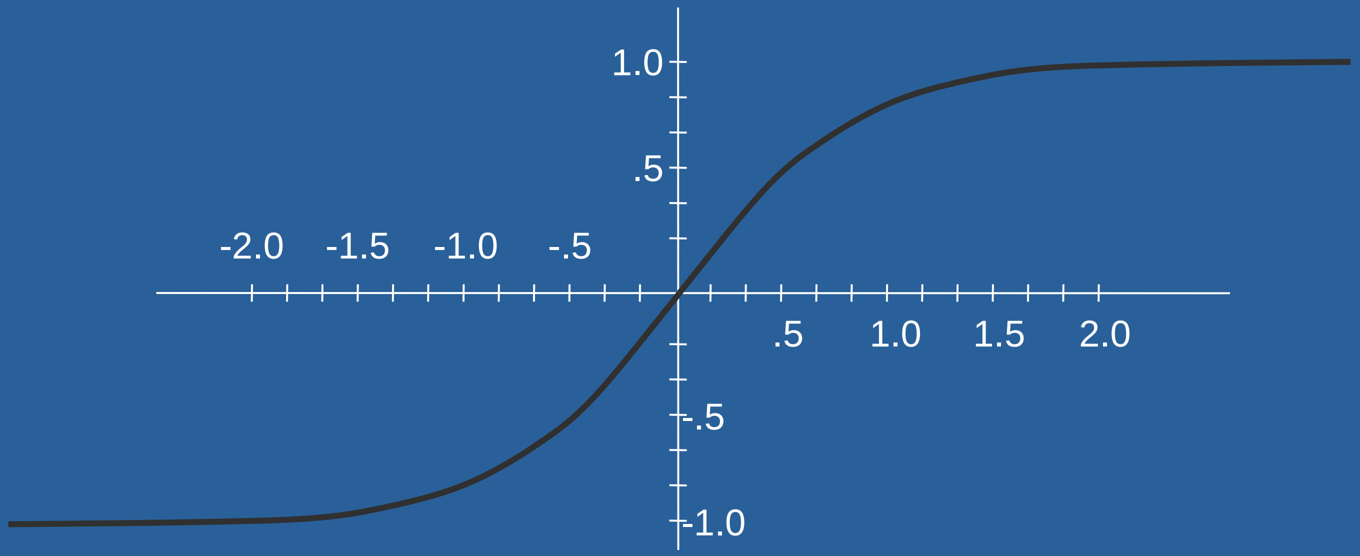
Note: Gradient is added multiplicatively for every step we trace back in our predictions.

Exploding/Vanishing Gradients

- Iterative learning process via gradient descent
- Gradient can be unstable since it is the product of earlier gradients and tends to grow/shrink exponentially (goes for all DNN, but especially RNN due to the time component)
- LSTM introduces cell states which provide ways for the gradient to flow backwards through time

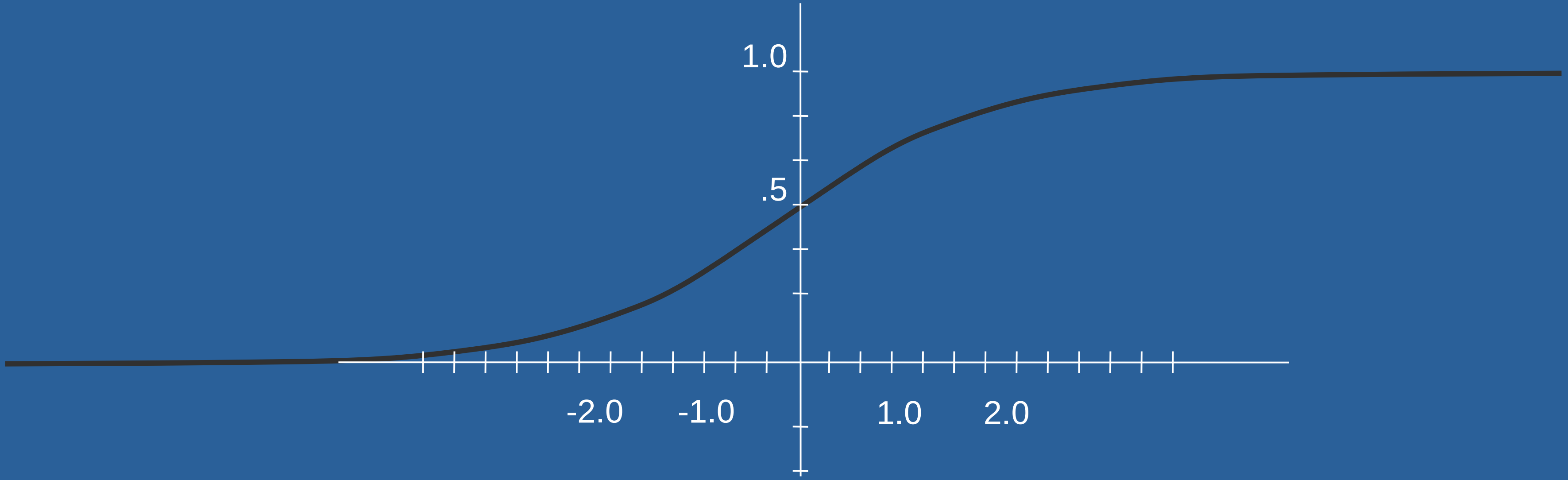
tanh squashing function

Denoted by



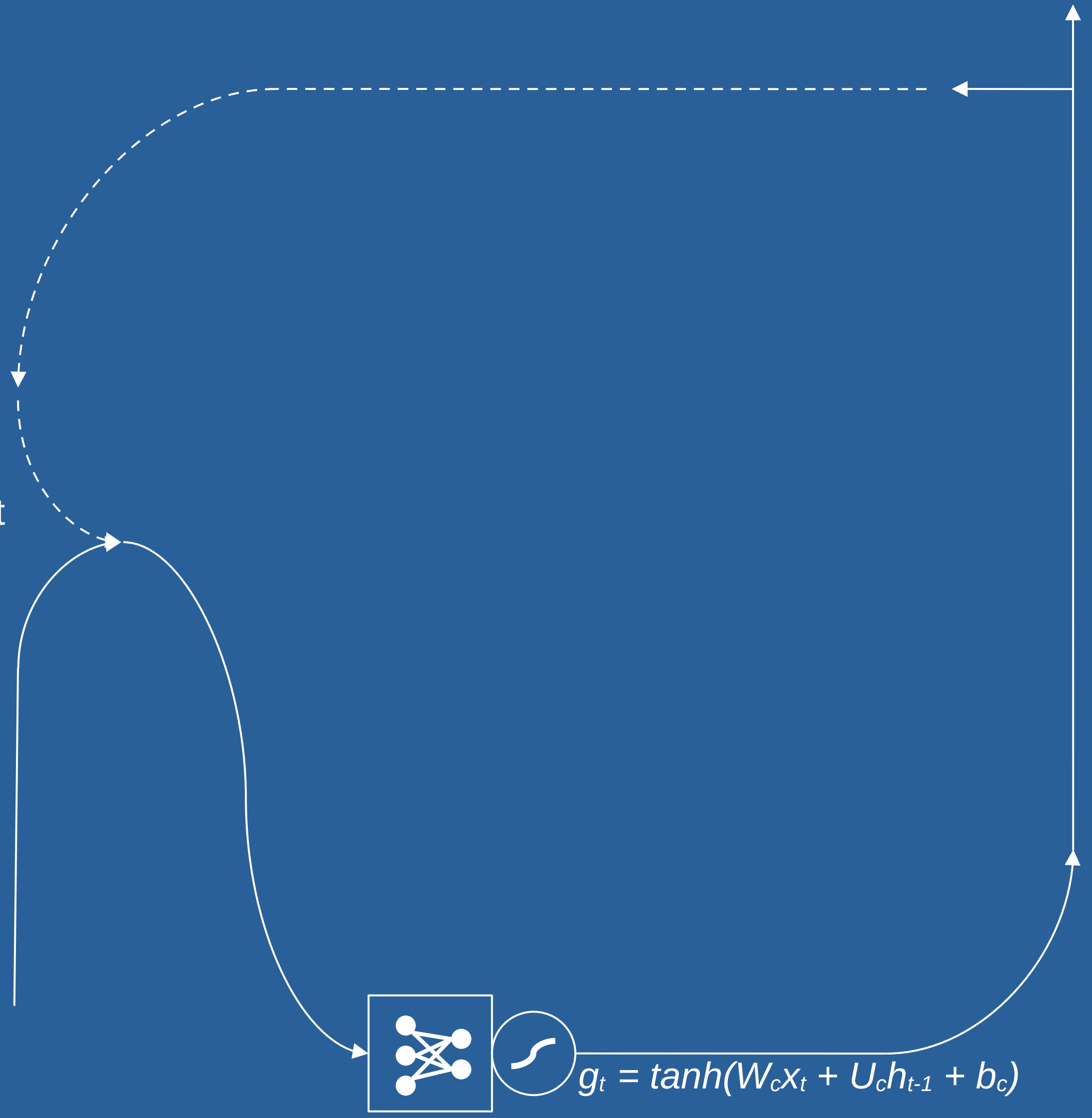
sigmoid squashing function

Denoted by 

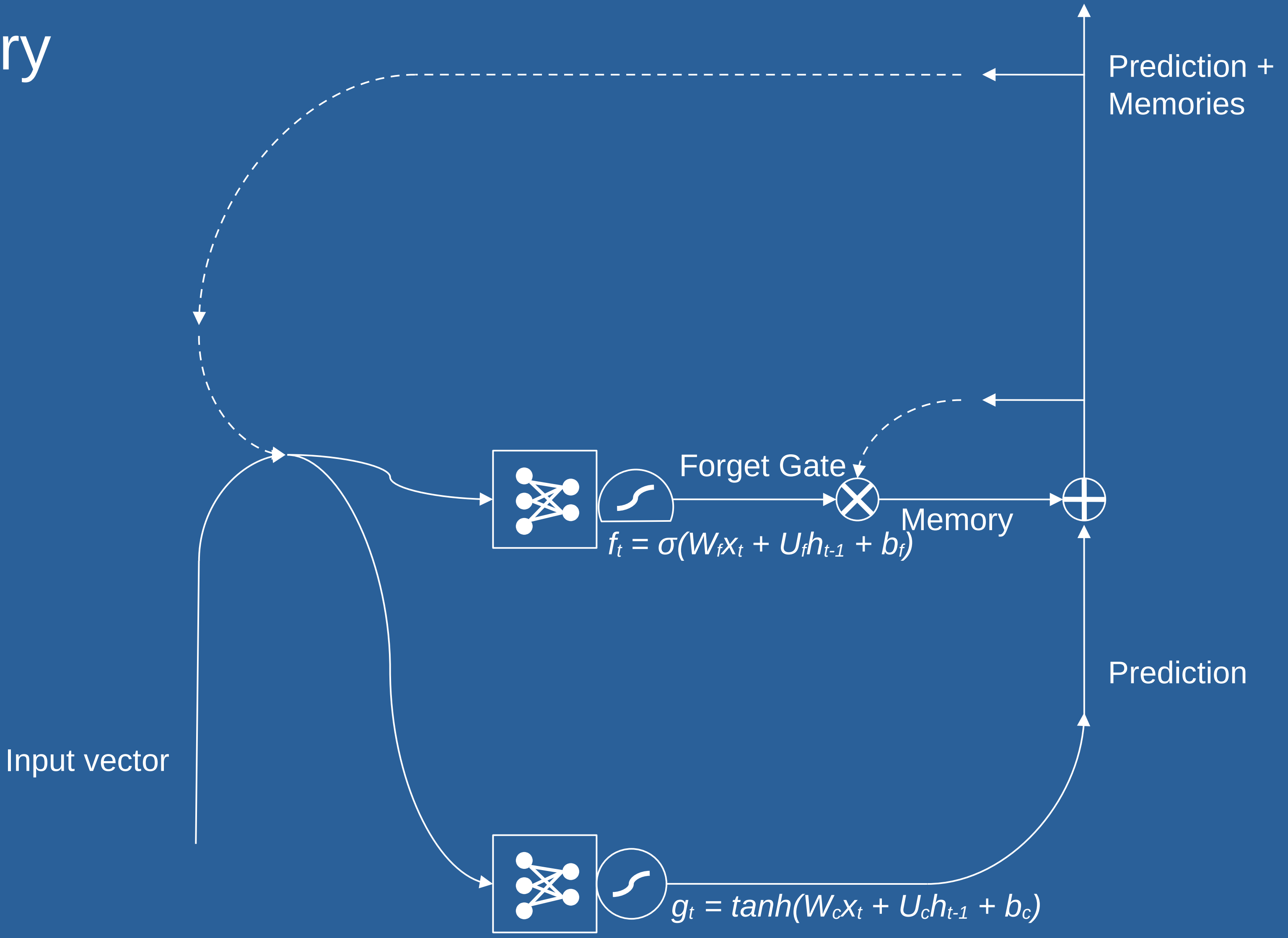


RNN

Note:
In the following slides *W*, *U*, and *b* are parameters which are learned, x_t is the input vector and h_{t-1} denotes the last prediction.

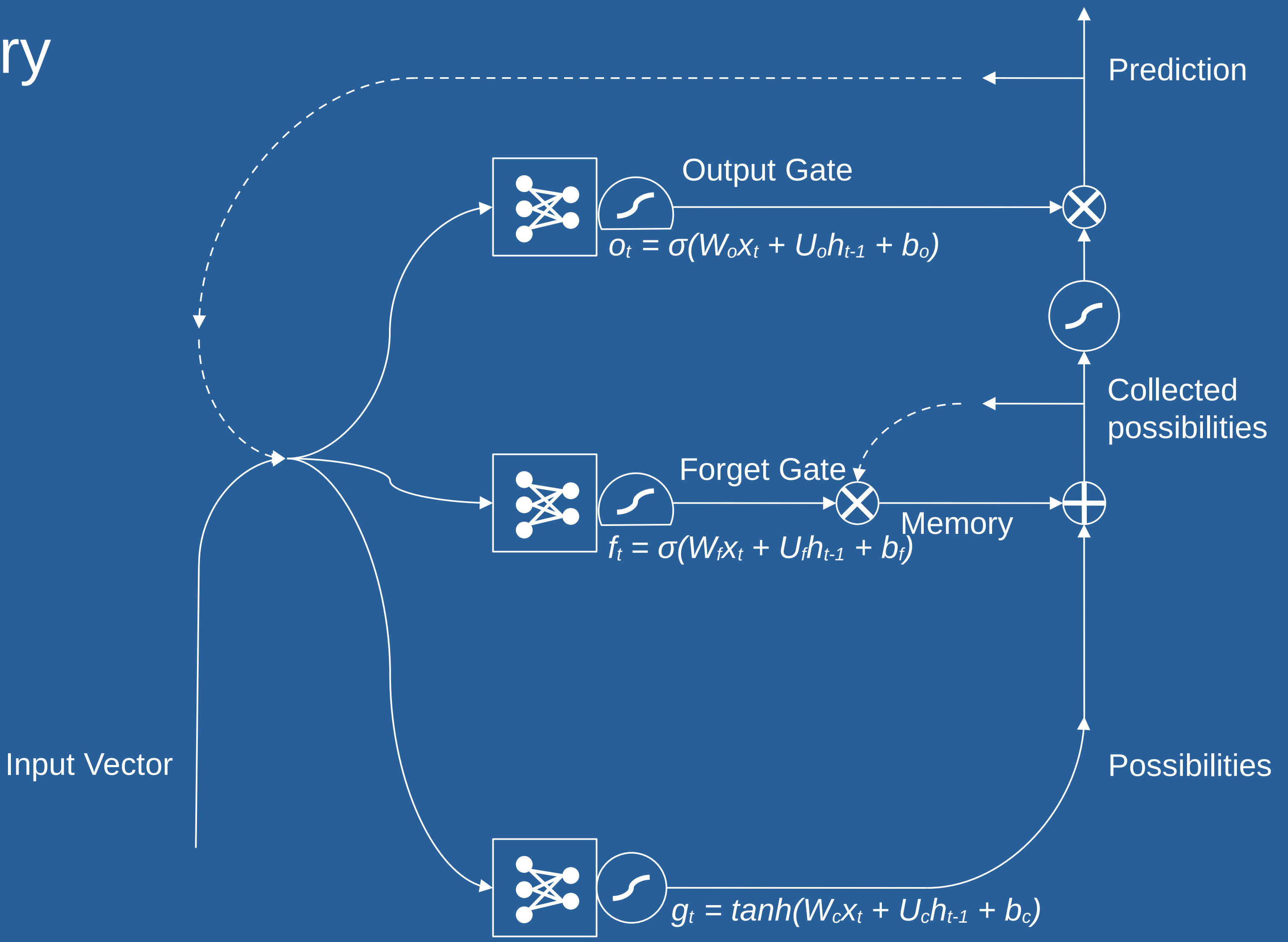


Memory



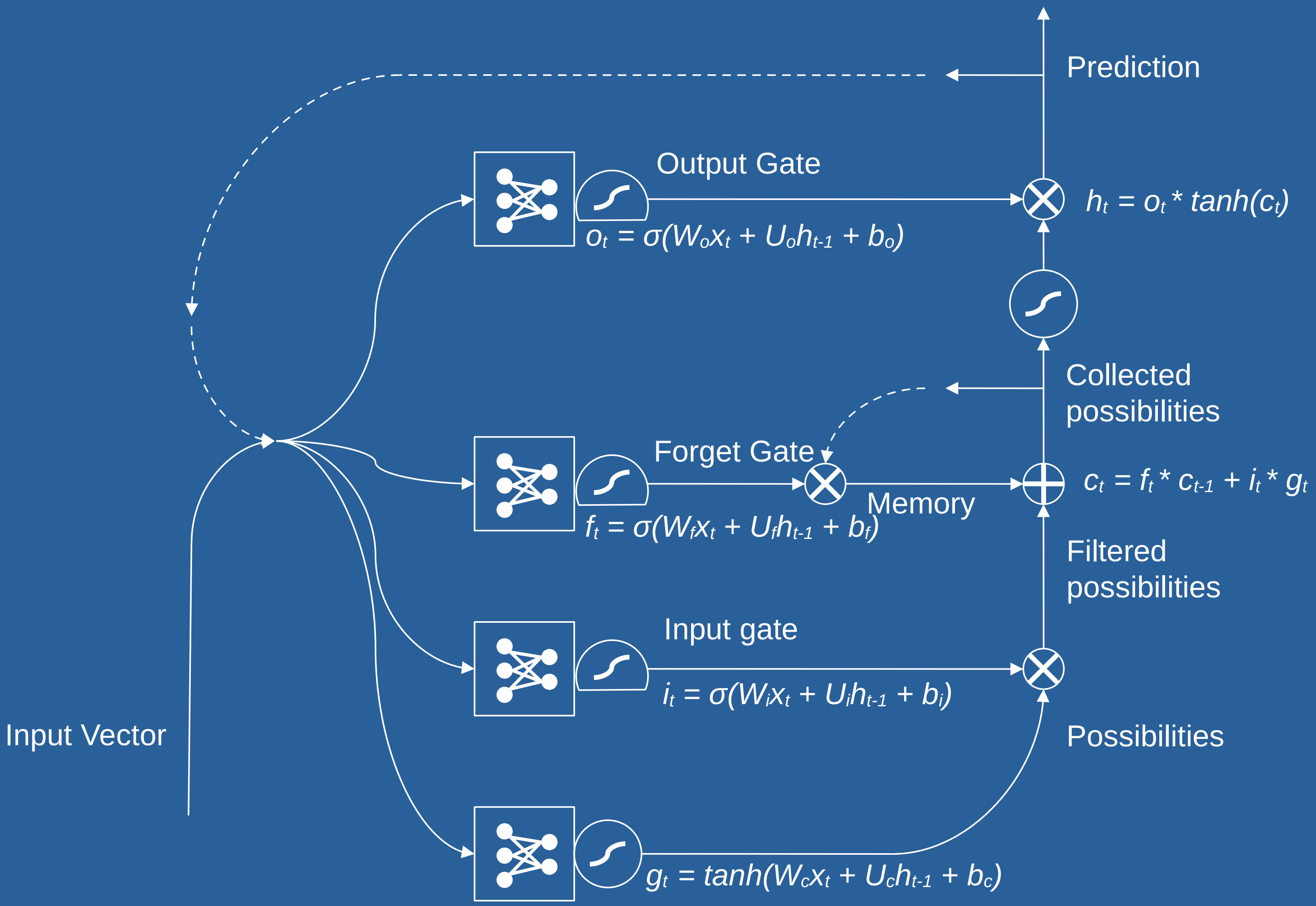
Prediction +
Memories
Forget Gate
Memory
$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$
Prediction
Input vector
$g_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$

Memory



Prediction
Output Gate
$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$
Collected
possibilities
Forget Gate
Memory
$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$
Input Vector
Possibilities
$g_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$

LSTM



Prediction
Output Gate
$h_t = o_t * tanh(c_t)$
$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$
Collected possibilities
Forget Gate
$c_t = f_t * c_{t-1} + i_t * g_t$
Memory
$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$
Filtered possibilities
Input gate
$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$
Input Vector
Possibilities
$g_t = tanh(W_c x_t + U_c h_{t-1} + b_c)$

Gates, memory cell, hidden output

- $i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$ Input Gate
- $f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$ Forget Gate
- $o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$ Output Gate
- $g_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$ Squashed Input
- $c_t = f_t * c_{t-1} + i_t * g_t$ Current Cell State
- $h_t = o_t * \tanh(c_t)$ Output

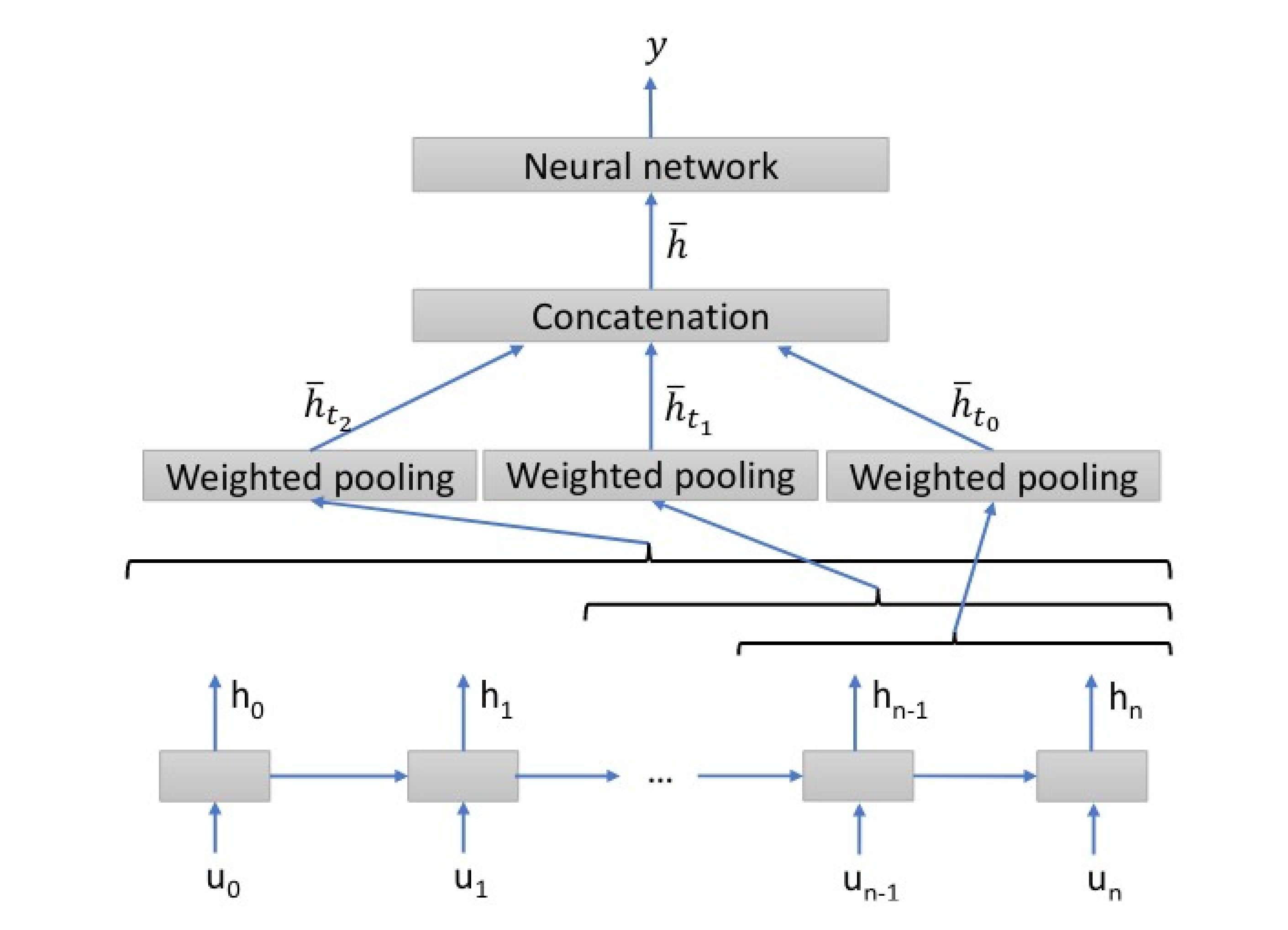
Benefits and Drawbacks of LSTM

- Much more resistant to exploding gradients
- Performs well

- High complexity
- Needs relatively large memory
- Takes longer to learn
- Overfitting

DeepCare

DeepCare Architecture



Challenges

- Variable size discrete inputs
- Confounding interactions between disease progression and intervention
- Irregular timing
- Overfitting

Representing variable-size inputs

- Input sequence $u_t = [\mathrm{x}_t, p_t, m_t, \Delta t]$
- Diagnoses and intervention codes embedded in vectors and summarised as 2M-dim admission embedding vector $[\mathrm{x}_t, p_t]$
- Max pooling admission, normalized sum pooling admission and mean pooling admission
- $i_t = (1/m_t)\sigma(W_i x_t + U_i h_{t-1} + b_i)$
- Where *$m_t = 1$ for unplanned admissions and $m_t = 2$ otherwise*

Modeling effect of interventions

- $o_t = \sigma(W_o x_t + U_o h_{t-1} + P_o p_t + b_o)$
- $f_t = \sigma(W_f x_t + U_f h_{t-1} + P_f p_{t-1} + b_f)$
- Current interventions should be considered at the output gate as it controls illness states and interventions reduce illness
- Prior interventions affect which information can be forgotten

Capturing time irregularity

- Time decay function to reduce effect of memorised acute conditions over time
- $d(\Delta_{t-1:t}) = [\log(e + \Delta_{t-t:1})]^{-1}$
- $f_t \leftarrow d(\Delta_{t-1:t}) f_t$
- More flexible forgetting to deal with chronic or worsening conditions
- $f_t = \sigma(W_f x_t + U_f h_{t-1} + Q_f q_{\Delta t-1:t} + P_f p_{t-1} + b_f)$

DeepCare Forward Pass Algorithm

Inputs: Patients' disease history records

1) For each step t do:
   1) $[x_t, p_t]$ = embed diagnoses and interventions
   2) Compute gates i_t, f_t, o_t
   3) Compute cell state and hidden state c_t, h_t
2) End for
3) Compute $\bar{h}$ the pooled illness states based on attention scheme
4) Feed to NN and compute $P(y \mid u_{0:n})$ to give prediction
5) Compute loss function L (Model learns and changes parameters based on error value)

Regularization

- Problem of overfitting
- Dropout probabilities introduced to regulate (some 1 – $p_{dropout}$)
- Targets:
  - Diagnosis and intervention vectors before pooling
  - Each value in $[x_t, p_t]$ after derivation
  - Hidden and input units after weighted pooling

References:

- S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.
- R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks,” arXiv preprint arXiv:1211.5063, 2012.
- Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” Neural Networks, IEEE Transactions on, vol. 5, no. 2, pp. 157–166, 1994.
- Brandon Rohrer, Recurrent Neural Networks (RNN) and Long, Short-Term Memory (LSTM), https://www.youtube.com/watch?v=WCUNPb-5EYI, Jun. 2017
- Recurrent neural network. (2021, July 29) In *Wikipedia.* https://en.wikipedia.org/wiki/Recurrent_neural_network
- Long short-term memory. (2021, July 27) In *Wikipedia*. https://en.wikipedia.org/wiki/Long_short-term_memory