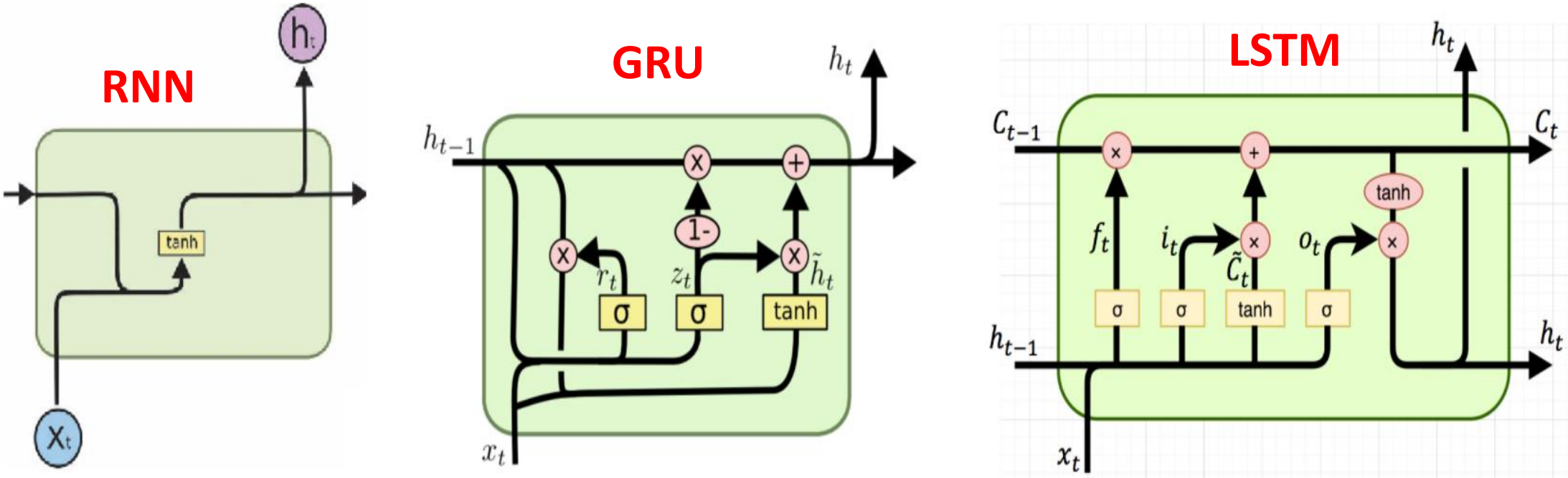
Gated Recurrent Units - GRU

Variants on Long Short Term Memory

Basic architectures of RNN, GRU and LSTM cells



Simple RNN :- Here there is simple multiplication of Input (x_t) and Previous Output (h_{t-1}). Passed through Tanh activation function. No Gates present.

Gated Recurrent Unit (GRU) :- Here a **Update gate** is introduced, to decide whether to pass Previous O/P (h_{t-1}) to next Cell (as h_t) or not.

Long Short Term Memory Unit (LSTM) :- Here 2 more Gates are introduced (Forget and Output) in addition to Update gate of GRU.

Gated Recurrent Units (GRU)

- Proposed by Cho et al. in 2014 as a simpler alternative to the LSTM.
- On each timestep *t* we have input $\boldsymbol{x}^{(t)}$ and hidden state $\boldsymbol{h}^{(t)}$ (no cell state).

Update gate: controls what parts of hidden state are updated vs preserved

Reset gate: controls what parts of previous hidden state are used to compute new content

New hidden state content: reset gate selects useful parts of prev hidden state. Use this and current input to compute new hidden content.

Hidden state: update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

$$\boldsymbol{u}^{(t)} = \sigma\left(\boldsymbol{W}_u \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_u \boldsymbol{x}^{(t)} + \boldsymbol{b}_u\right)$$

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{W}_r \boldsymbol{h}^{(t-1)} + \boldsymbol{U}_r \boldsymbol{x}^{(t)} + \boldsymbol{b}_r\right)$$

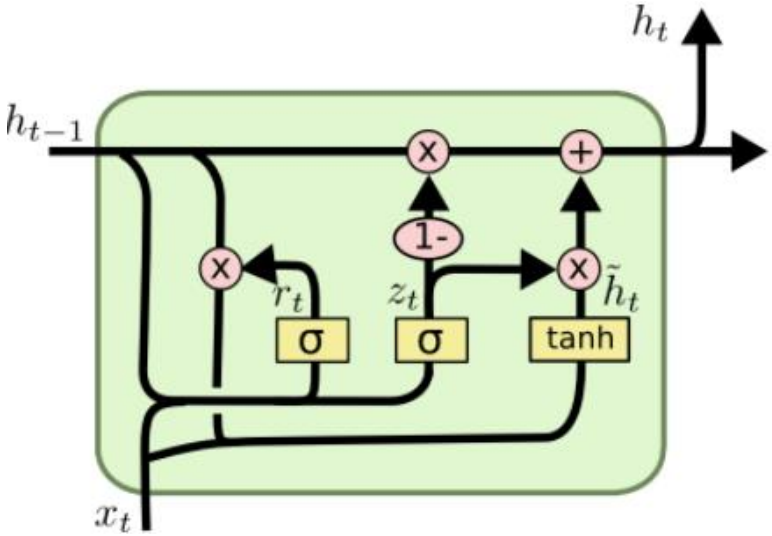
$$\tilde{\boldsymbol{h}}^{(t)} = \tanh\left(\boldsymbol{W}_h(\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}) + \boldsymbol{U}_h \boldsymbol{x}^{(t)} + \boldsymbol{b}_h\right)$$

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{u}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{u}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)}$$

How does this solve vanishing gradient?
Like LSTM, GRU makes it easier to retain info long-term (e.g. by setting update gate to 0)

Gated Recurrent Units- GRU

- **LSTMs uses lots of additional parameters to RNN's**
- **LSTM has 8 sets of weights to learn, simple RNN units had 2**
- Training these additional parameters imposes a much significantly higher training cost.
- GRU eases this by dispensing the use of a separate context vector, and by reducing the number of gates to 2
- GRU got rid of the cell state and used the hidden state to transfer information.
- **The two gates are a reset gate and update gate.**
- **Update Gate z : The update gate acts similar (combines) to the forget and input gate of an LSTM. It decides what information to throw away and what new information to add.**
- **Reset Gate r : The reset gate is another gate is used to decide how much past information to forget.**
- GRU's has fewer tensor operations; therefore, they are a little speedier to train then LSTM's. There isn't a clear winner which one is better.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

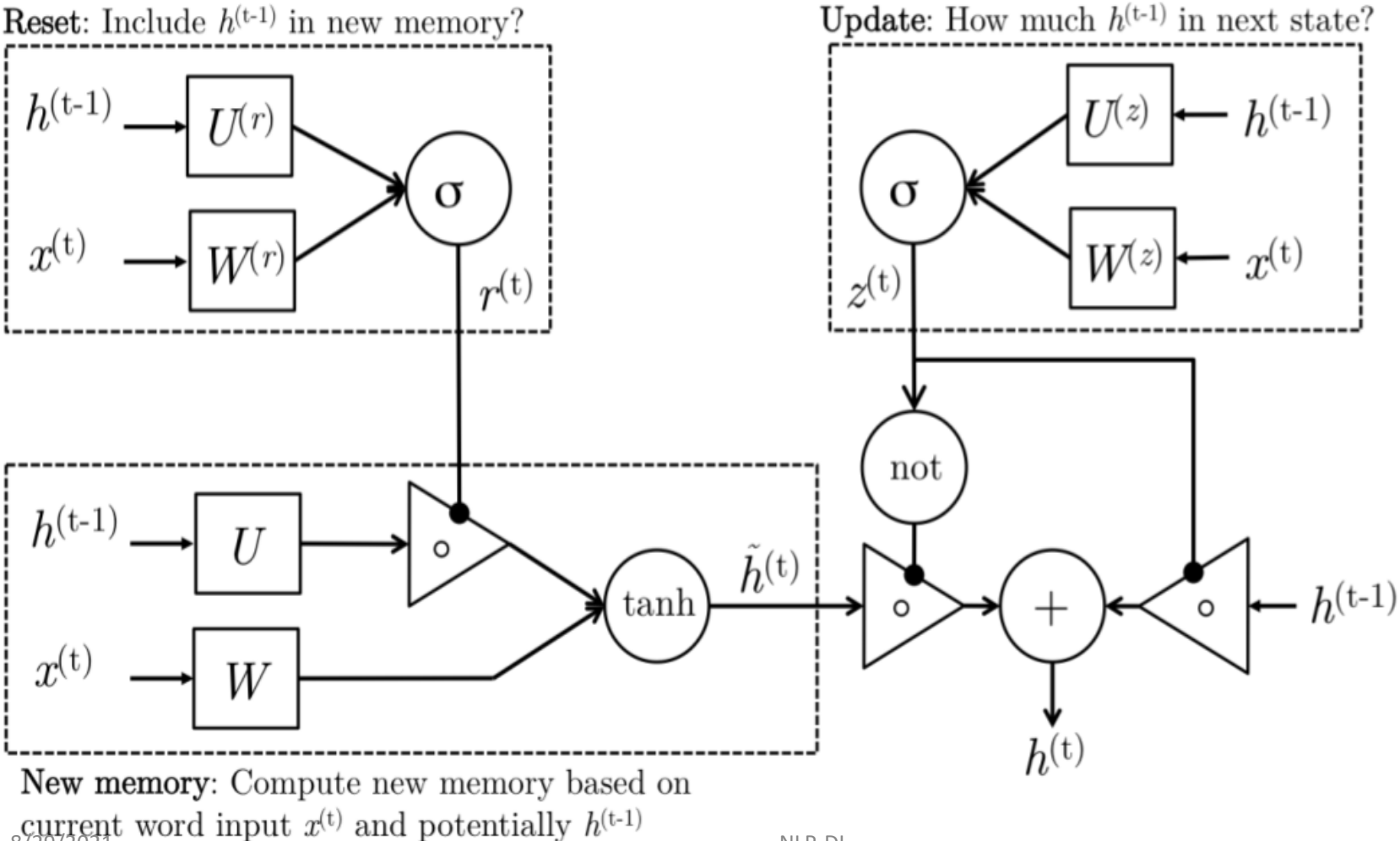
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Gated Recurrent Units- GRU

- As with LSTMs, the use of the sigmoid in the design of these gates results in a binary-like mask that either blocks information with values near zero or allows information to pass through unchanged with values near one.
- The purpose of the reset gate is to decide which aspects of the previous hidden state are relevant to the current context and what can be ignored.
- This is accomplished by performing an element-wise multiplication of **r** with the value of the previous hidden state.
- We then use this masked value in computing an intermediate representation for the new hidden state at time t.
- The job of the update gate z is to determine which aspects of this new state will be used directly in the new hidden state and which aspects of the previous state need to be preserved for future use.
- This is accomplished by using the values in z to interpolate between the old hidden state and the new one.

The detailed internals of a GRU



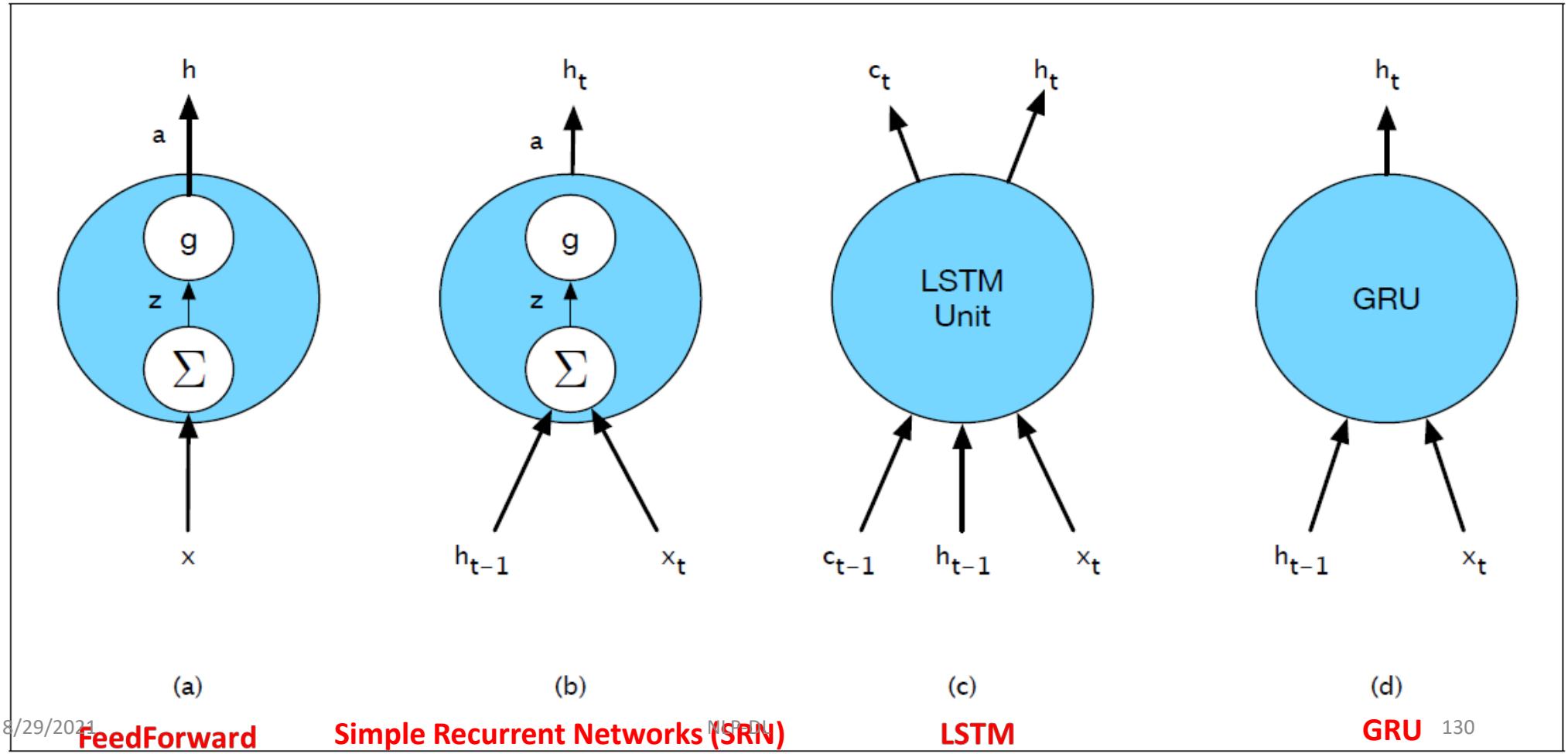
LSTM vs GRU

- Researchers have proposed many gated RNN variants, but LSTM and GRU are the most widely-used
- The biggest difference is that GRU is quicker to compute and has fewer parameters
- There is no conclusive evidence that one consistently performs better than the other
- LSTM is a good default choice (especially if your data has particularly long dependencies, or you have lots of training data)
- Rule of thumb: start with LSTM, but switch to GRU if you want something more efficient

The increased complexity of the LSTM (c) and GRU (d) units on the right is encapsulated within the units themselves. The only additional external complexity for the LSTM over the basic recurrent unit (b) is the presence of the additional context vector as an input and output.

The GRU units have the same input and output architecture as the simple recurrent unit.

Basic neural units used



FeedForward **Simple Recurrent Networks (SRN)** **LSTM** **GRU**