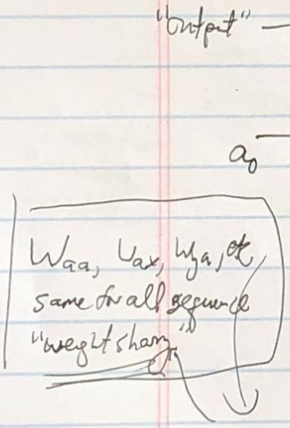
pretrain + finetune
Also outperforms classifiers trained just on this datasets!

Although transformers have "taken" over, prior approaches to sequence modeling still useful to have in toolbox (as in ASTROMER example) — RNN, LSTM, GRU

Ref: stanford CS-230 "cheatsheet" for RNNs

Brief overview of RNN, LSTM, GRU architectures

- very sequential, not at all perm equiv.
- not feed forward per se — or like FF w/ # layers depends on length of sequence

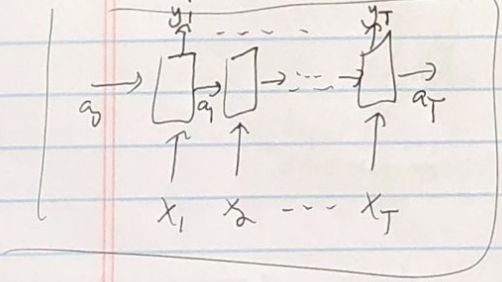


depends on prev hidden state + current input

$$\begin{cases} a_1 = g_1(W_{aa} a_0 + W_{ax} x_0 + b_a) \\ y_0 = g_2(W_{ya} a_1 + b_y) \end{cases}$$

depends on current hidden state

repeat for each element of sequence

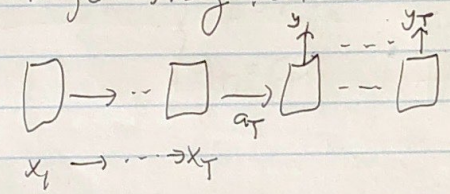


produces transformed sequence + hidden state that encodes full sequence

- Can also use as 1 → many

$x_1 \rightarrow y_1 \rightarrow y_2 \rightarrow \cdots$ (music, text gen)

- many → 1 : just keep y_T → classification
- many → many : name entity recog (classify each token according to type)
  esp named entities (person, corp., ...)
- many → many : translation



→ also: slower to train since sequential

- vanilla RNNs struggle w/ vanish/expl gradients — long sequences, information is lost

short term memory problem

→ introduce "gated" RNNs (analog of residual/skip connections!)

Two popular examples: Gated Rec. unit (GRU)

Long Short-Term Memory (LSTM)

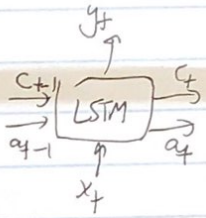
GRU ⊂ LSTM (special case, LSTM more general)

hidden + cell state

↙ local short term info

↳ global info long term

$$\begin{cases} c_t = \Gamma_u \tilde{c}_t + \Gamma_f c_{t-1} \\ \tilde{c}_t = \tanh\left(W_c\left(\Gamma_r a_{t-1}, x_t\right)+b_c\right) \\ a_t = \Gamma_o c_t \end{cases}$$



$\Gamma_u, \Gamma_f, \Gamma_r, \Gamma_o$ "gates" $\qquad \Gamma = \sigma\left(Wx_t + Ua_{t-1} + b\right)$

update, forget, relevance, output

GRU: $\begin{cases} \Gamma_f = 1 - \Gamma_u \\ a_t = c_t \end{cases}$ simplification!
