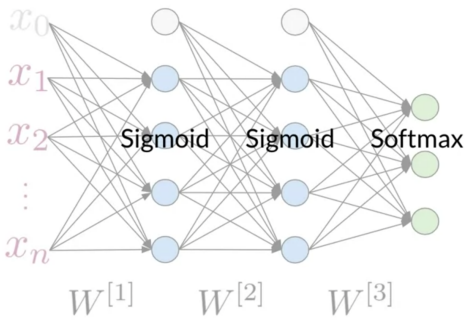
Week 1 - Neural Networks for Sentiment Analysis

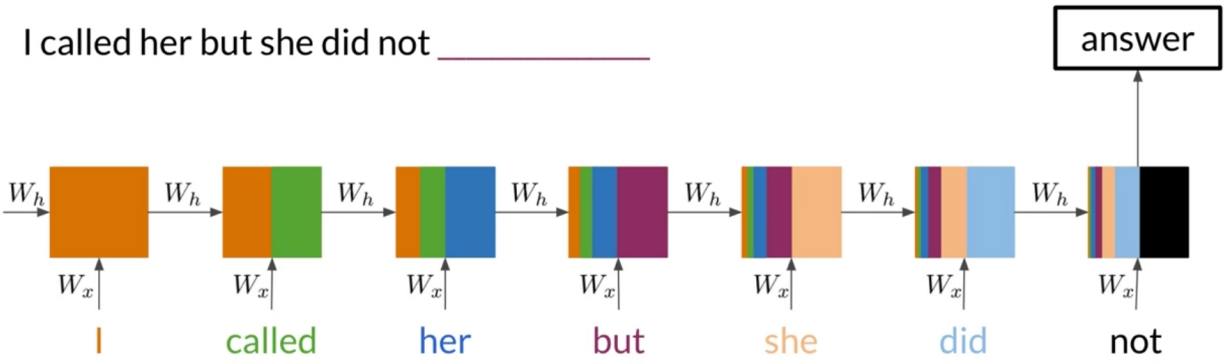
Neural Networks in Trax




```
from trax import layers as tl
Model = tl.Serial(
        tl.Dense(4),
        tl.Sigmoid(),
        tl.Dense(4),
        tl.Sigmoid(),
        tl.Dense(3),
        tl.Softmax())
```

Week 2 - Recurrent Neural Networks for Language Modeling

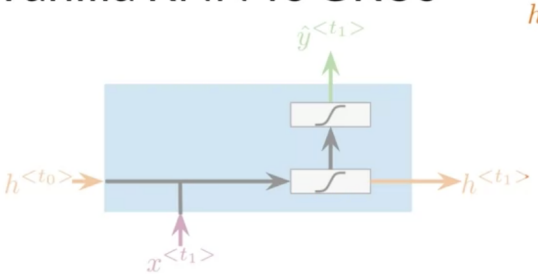
RNNs Basic Structure



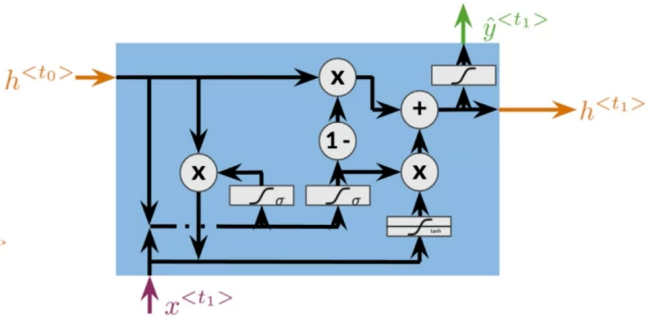
- RNNs can be implemented for a variety of NLP tasks
- Applications include Machine translation and caption generation

1.25

Vanilla RNN vs GRUs



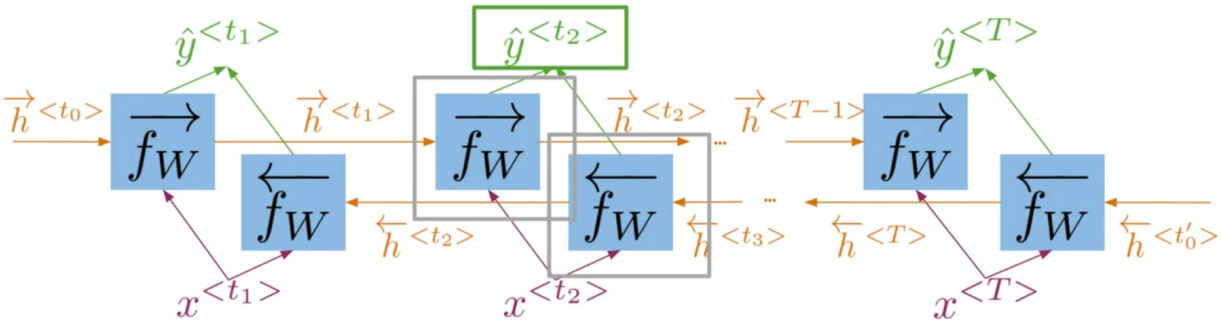
$$h^{<t>} = g(W_h[h^{<t-1>}, x^{<t>}] + b_h)$$
$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b_y)$$



$$\Gamma_u = \sigma(W_u[h^{<t_0>}, x^{<t_1>}] + b_u)$$
$$\Gamma_r = \sigma(W_r[h^{<t_0>}, x^{<t_1>}] + b_r)$$
$$h'^{<t_1>} = \tanh(W_h[\Gamma_r * h^{<t_0>}, x^{<t_1>}] + b_h)$$
$$h^{<t_1>} = \Gamma_u * h^{<t_0>} + (1 - \Gamma_u) * h'^{<t_1>}$$
$$\hat{y}^{<t_1>} = g(W_y h^{<t_1>} + b_y)$$

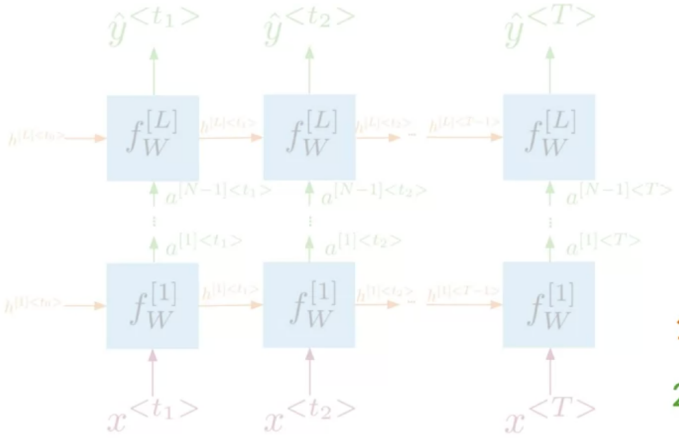
- GRUs "decide" how to update the hidden state
- GRUs help preserve important information

Bi-directional RNNs



$$\hat{y}^{<t>} = g(W_y[\overrightarrow{h}^{<t>}, \overleftarrow{h}^{<t>}] + b_y)$$

Deep RNNs



$$h^{[l]<t>} = f^{[l]}(W_h^{[l]}[h^{[l]<t-1>}, a^{[l-1]<t>}] + b_h^{[l]})$$

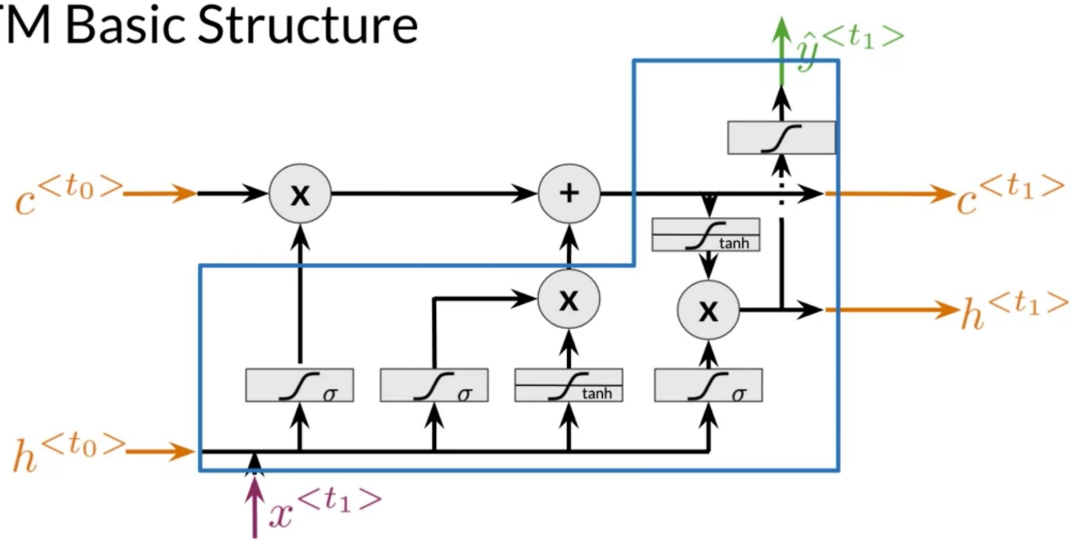
$$a^{[l]<t>} = f^{[l]}(W_a^{[l]} h^{[l]<t>} + b_a^{[l]})$$

1. Get hidden states for current layer
2. Pass the activations to the next layer

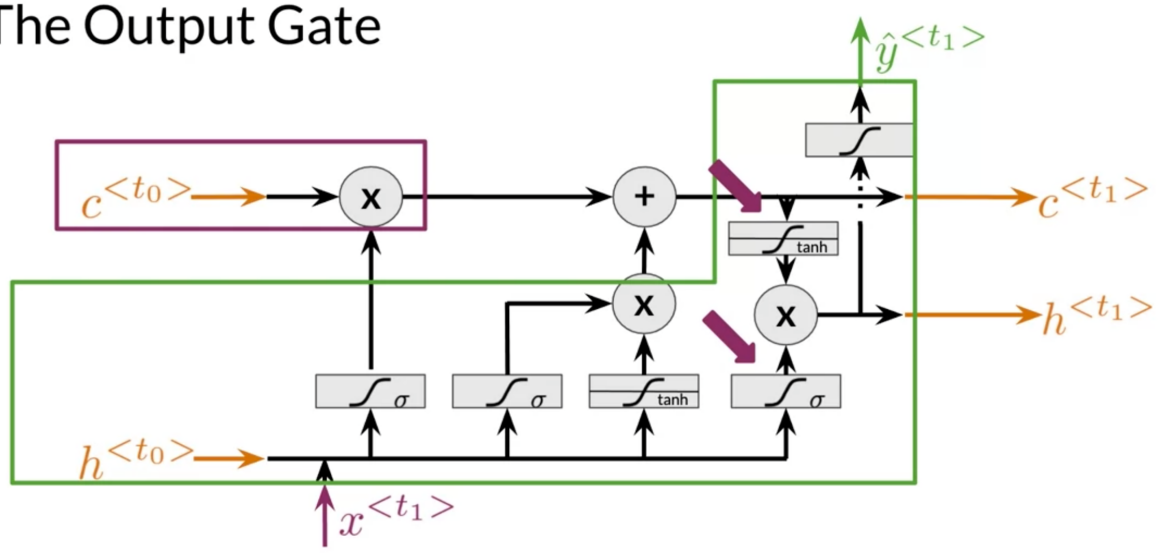
- In bidirectional RNNs, the outputs take information from the past and the future
- Deep RNNs have more than one layer, which helps in complex tasks

Week 3 - Natural Language Processing with Sequence Models

LSTM Basic Structure



The Output Gate



Summary

- LSTMs use a series of gates to decide which information to keep:
  - Forget gate decides what to keep
  - Input gate decides what to add
  - Output gate decides what the next hidden state will be

- One time step is completed after updating the states