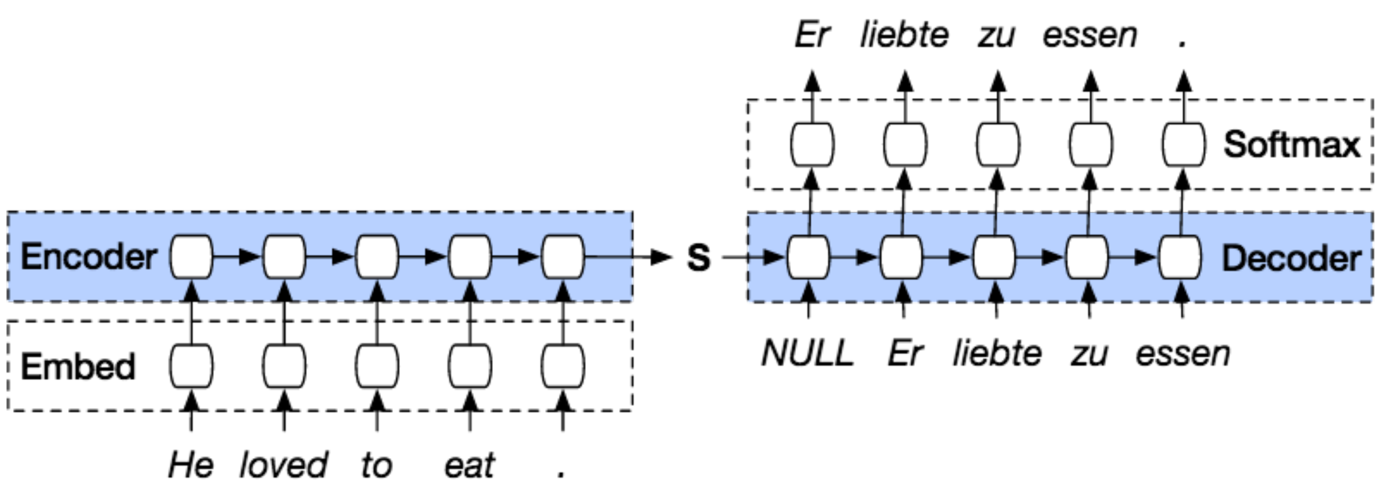
UC San Diego

CSE 151B: Deep Learning

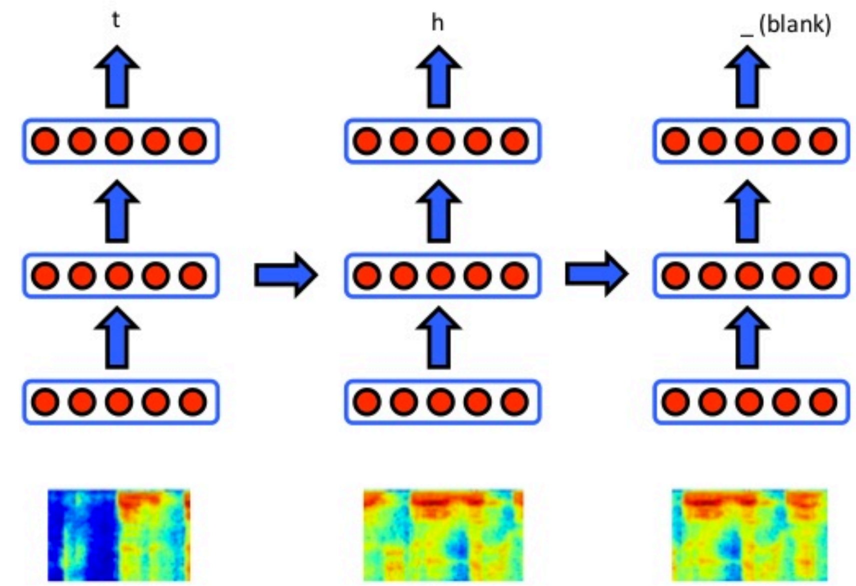
Rose Yu

SEQUENCE MODELING

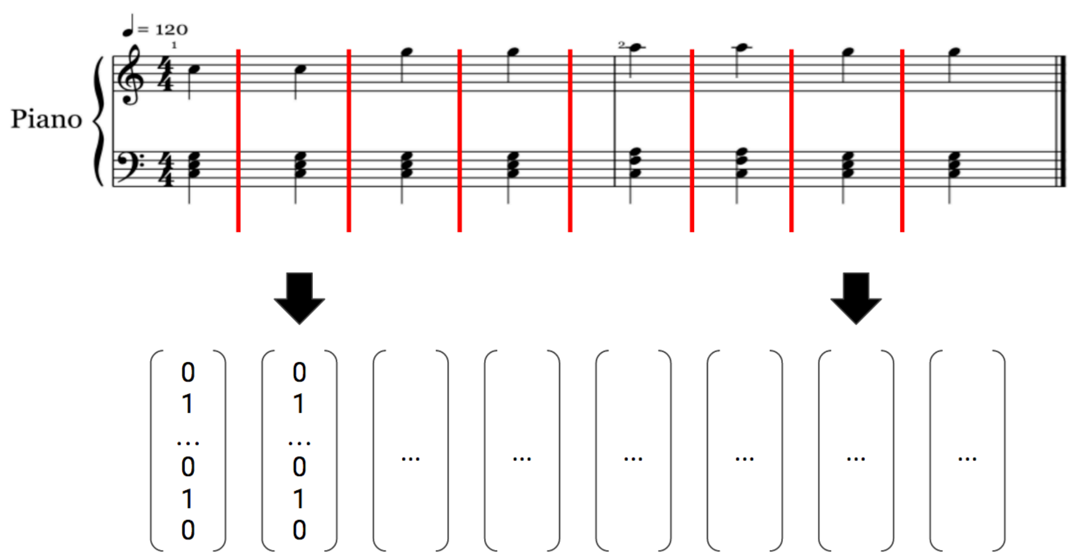
Sequences are everywhere



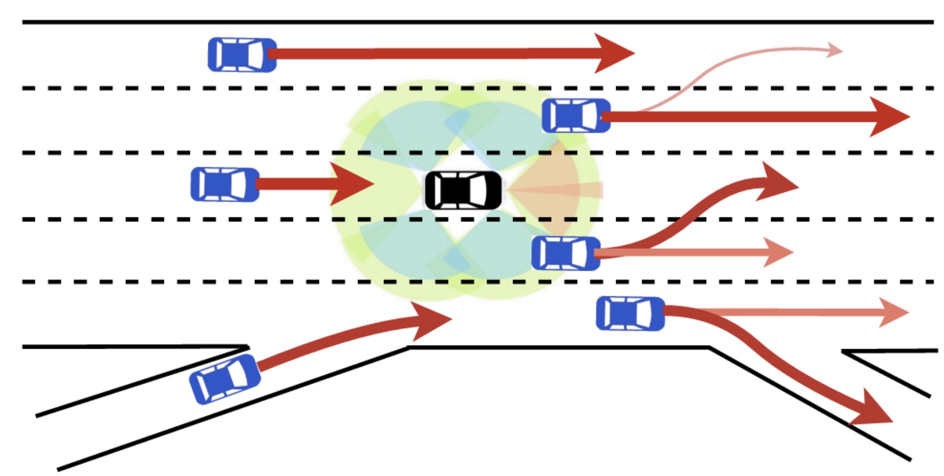
machine translation



speech recognition

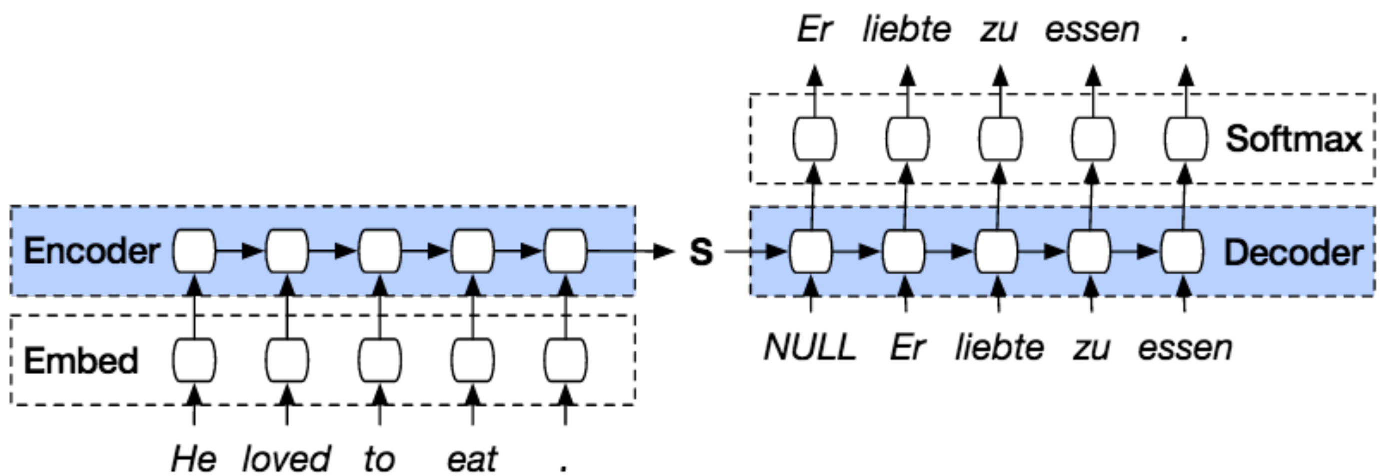


music composition



trajectory prediction

Mathematical Formulation



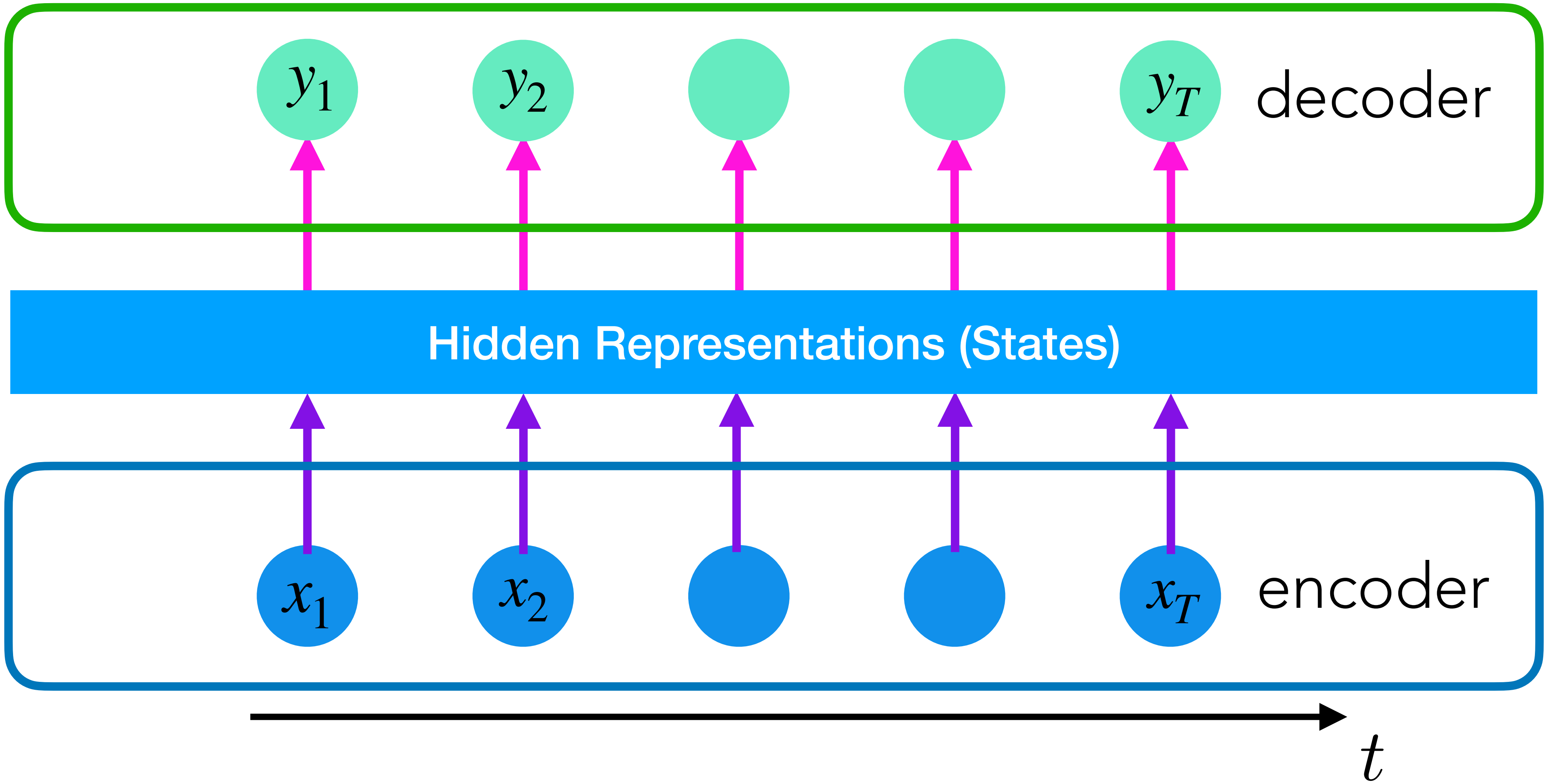
machine translation

- Learn the probability
  p(Er, Liebte, Zu, Essen| He, loved, to, eat)

- In general, given input sequence $(x_1, x_2, \cdots, x_T)$, output sequence $(y_1, y_2, \cdots, y_T)$

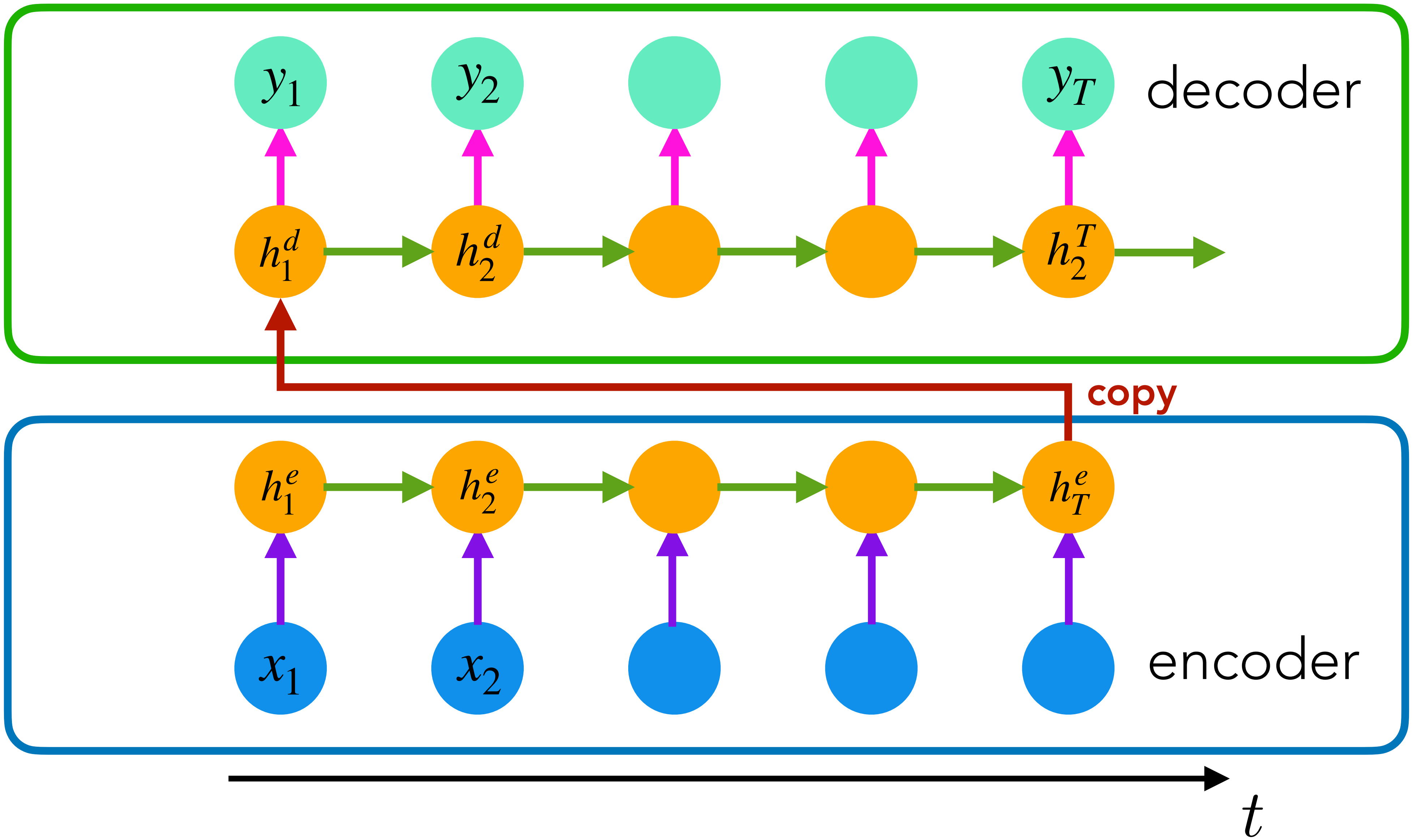
- Learn the probability $p(y_1, \cdots, y_T | x_1, \cdots, x_T)$

Encoder-Decoder



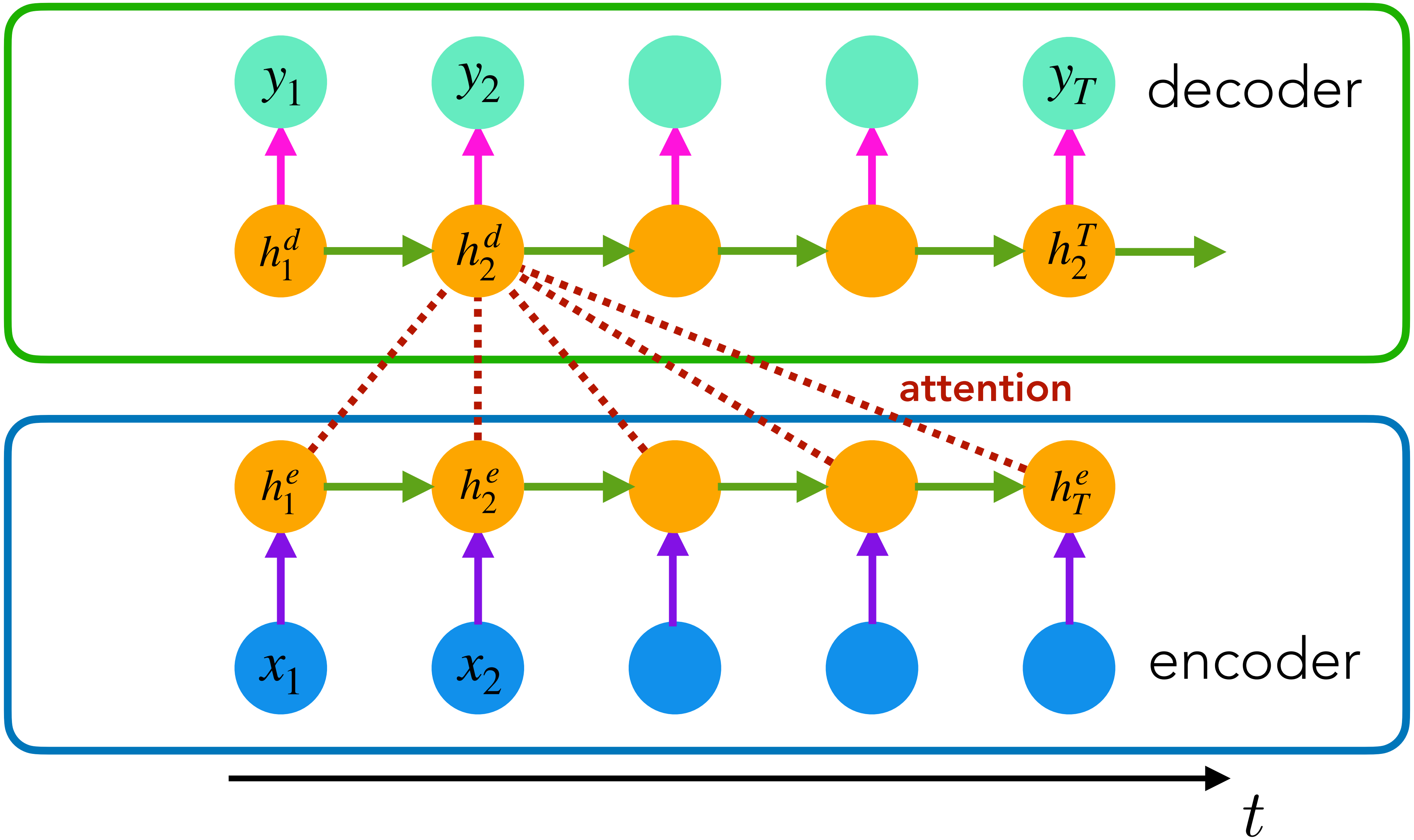
- Approximate $p(y_1, \cdots, y_T | x_1, \cdots, x_T)$ with neural networks
- Encoder: $(x_1, \cdots, x_T) \to h$ maps the input sequence into hidden states
- Decoder: $h \to (y_1, \cdots, y_T)$ maps the hidden states to output sequence

Seq2Seq



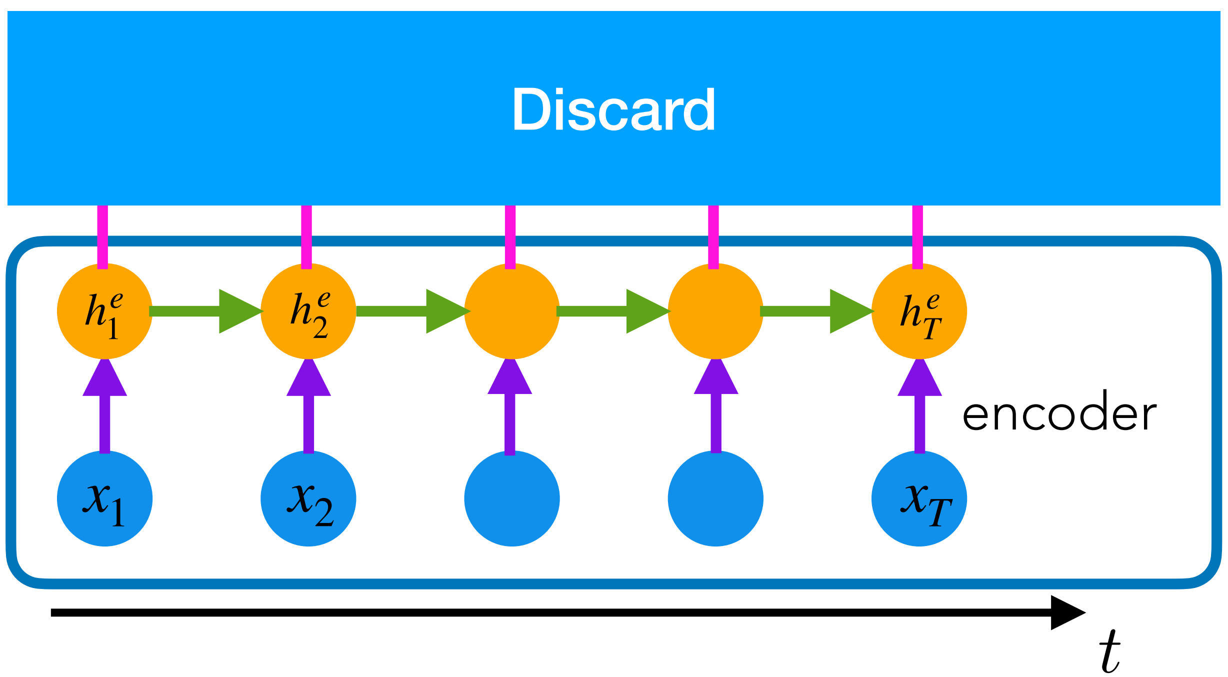
- Use (multi-layer) RNNs for both encoder and decoder
- Copy the last hidden state of the encoder to the initial state of the decoder

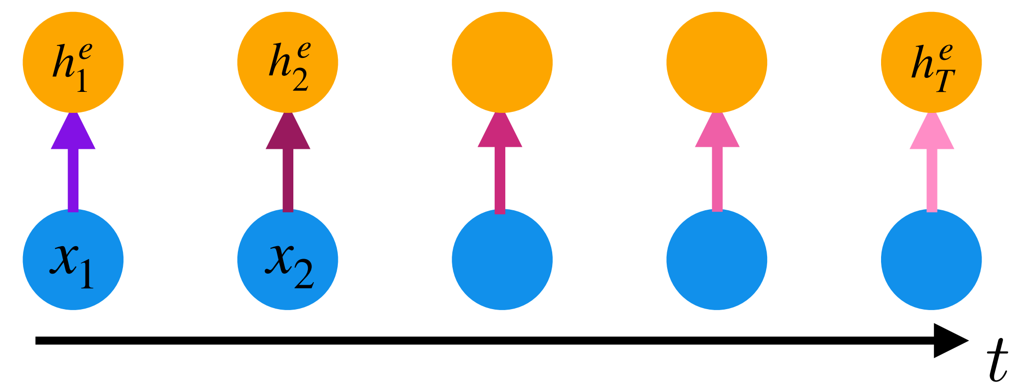
Attention/Transformer

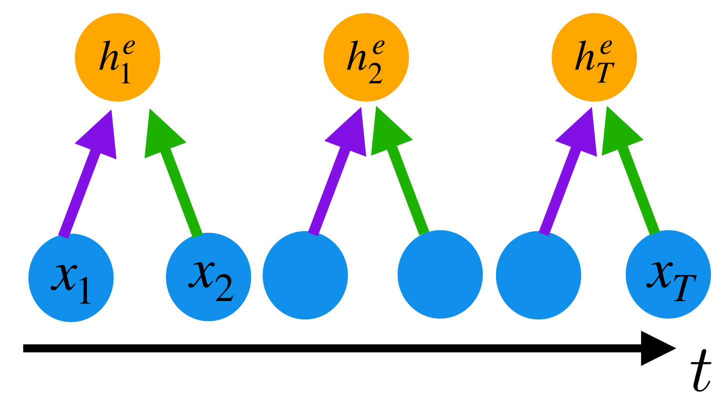


- Compute the alignment between encoder-decoder hidden states
- Decoder decides part of the source (input) to pay attention to

Encoder

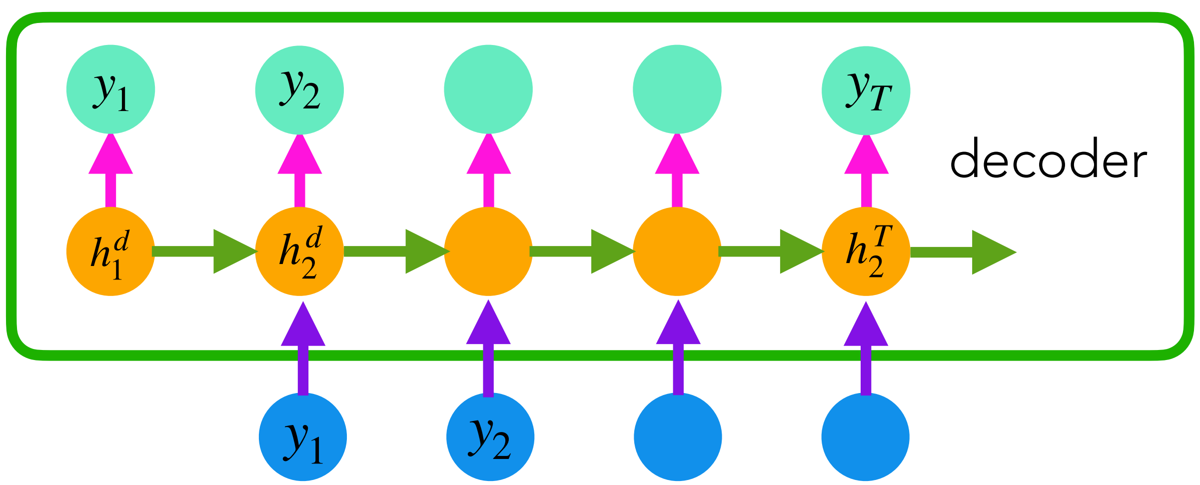


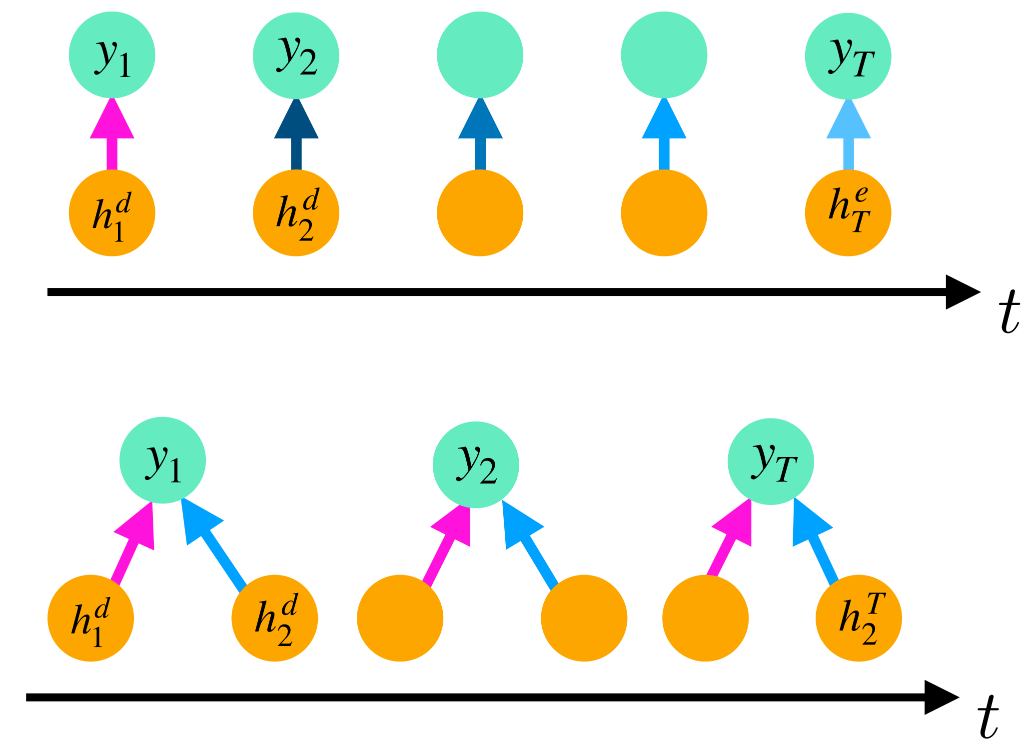




- Discard the outputs in the encoder RNN and only keep the hidden states
- Encoder architecture is flexible: RNN, FC, CNN, as long as it gives the map from input sequence to hidden states

Decoder





- Decoder: RNN with no input? Not really
- **Auto-regressive**: use the model's own prediction from previous step
- **Teacher-forcing**: use the ground truth output sequence from previous step
- Decoder architecture is flexible: RNN, FC, CNN, as long as it gives the map from hidden states to output sequence