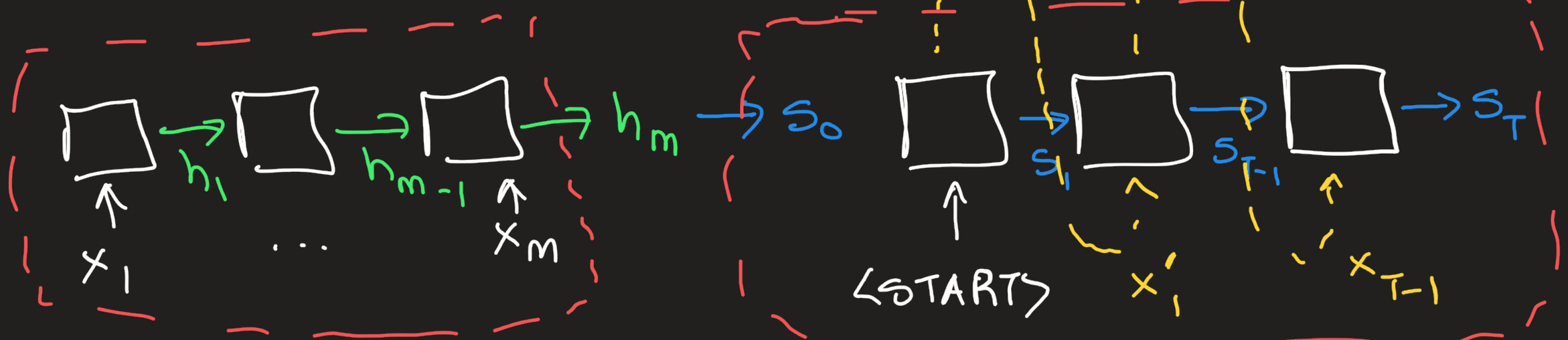
ML and Optimization Lec 25

- Seq2Seq model w/ attention
- Attention is all we need: seq2seq models w/o RNNs
- BERT: pretraining transformer models for NLP

Seq2Seq model

Input: $x_1, \cdots, x_m$

Output: $x'_1, \cdots, x_T$



Encoder Decoder

Issue: regardless of length of input sequence, all context/historical dependence of the output sequence is contained in a d-dimensional vector h_m

Leads to poor performance (BLEU decreases w/ length of input sequence for Machine translation tasks)

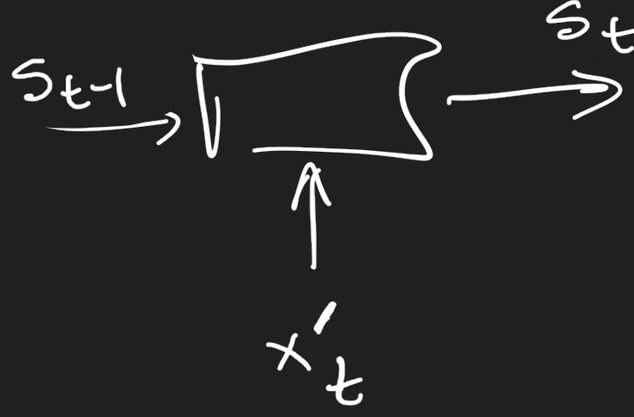
Seq2Seq with attention

- Introduced in Bahdanau et al. 2015, Neural Machine Translation by Jointly Learning to Align and Translate

- Allow each output token to attend to each individual hidden state in the input sequence. Captures idea of focusing on relevant portion of input sequence. Avoids the need to capture everything relevant to all outputs $x'_1, \ldots, x'_T$ in the one vector h_m

Seq2Seq w/ attention

Idea: replace our previous hidden state update for decoder



i.e. $s_t = g(s_{t-1}, x'_t)$

w/ first finding which historical hidden states are most relevant:

$$s_{t-1}, x'_t, [h_1, \cdots, h_m] \longmapsto \alpha_t \text{ a prob. dist over } [h_1, \cdots, h_m]$$

then form a context vector

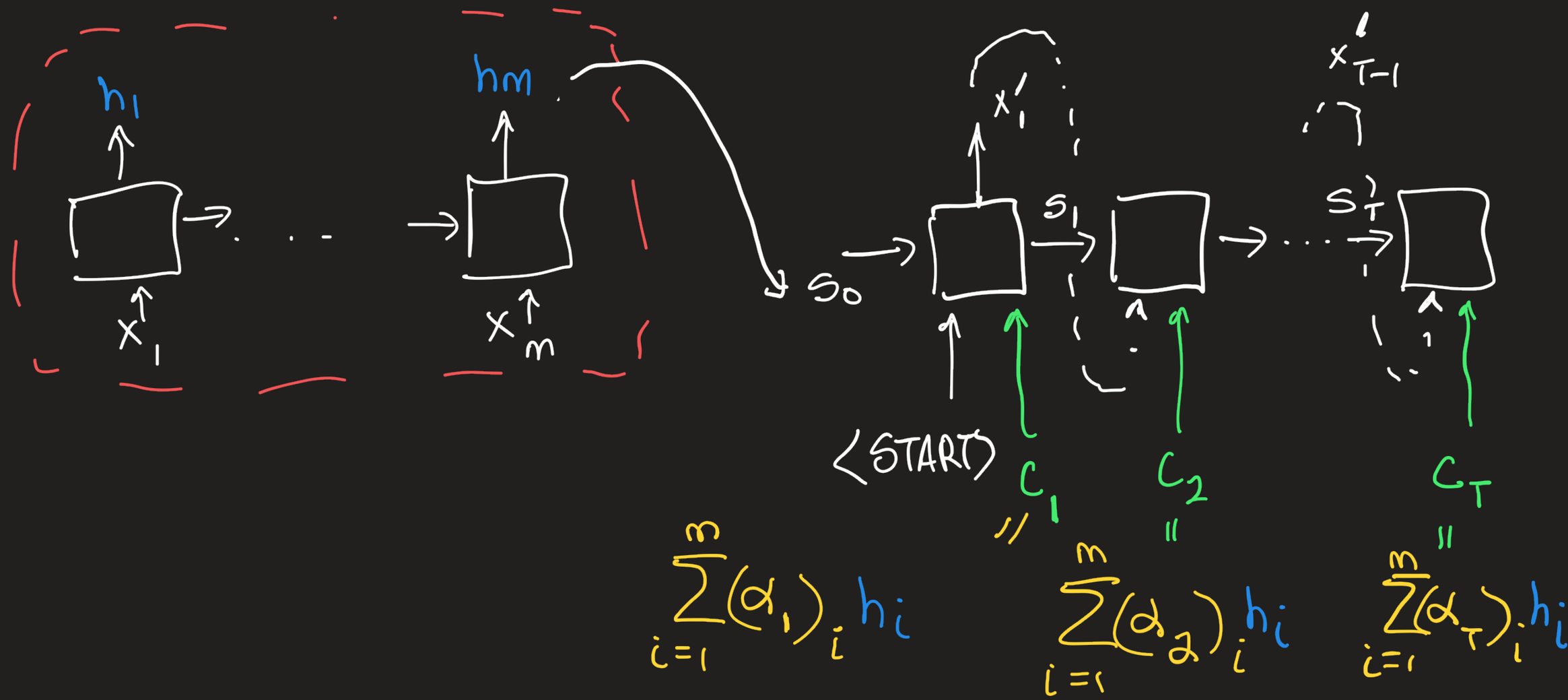
$$c_t = \sum_{i=1}^{m} (\alpha_t)_i h_i$$

and update with

$$s_t = g(s_{t-1}, c_t)$$

we removed dependence on x'_t because in the decoder x'_t isn't much more than s_{t-1}

In the context of decoding, in the decoder, use attention over the hidden states of the encoder



Options for computing the attention vectors α_t

1) Option I (used in Bahdanau et al. 2015)

Take $\tilde{\alpha}_i = \underline{v^T} \tanh\left(\underline{W} \begin{bmatrix} h_i \\ s_{t-1} \end{bmatrix}\right)$ for $i = 1, \ldots, m$

parameters to learn

Then $\alpha_t = \text{softmax}(\tilde{\alpha}) \in \mathbb{R}^m$

2) (Dot-product attention) Idea: generate keys and queries as linear transforms of h_i and s_{t-1} respectively, take logits $\tilde{\alpha}_i$ to be their inner-products

$W_K h_i = k_i$

$W_Q s_{t-1} = q_{t-1}$

$\tilde{\alpha}_i = \langle k_i, q_{t-1} \rangle$

$\alpha_t = \text{softmax}(\tilde{\alpha})$

Complexity of using attention

1) Simple RNN encoder models : $O(m + T)$

2) RNN + attention models $O(m \cdot T)$

because for each output token, must compute $\alpha_t \in \mathbb{R}^m$ by comparing to all inputs

Transformers : Attention is all we need

- 2017, Vaswani et al., Attention is all we need
- Insight: attention alone suffices to contain all historical information we need, so can eliminate recursive structure.
- Replace hidden representations in RNNs w/ contextual representations c_t computed using self-attention (for encoder)
- Use combination of self-attention in decoder and general attention against the encoder contextual representations to get contextual representations to predict output tokens

Self-Attention (replace the hidden states (learned in RNNs w/ contextual representations)

x_1 . . . x_m

$$c_i = \sum_{j=1}^{m} (\alpha_i)_j \, v_j$$

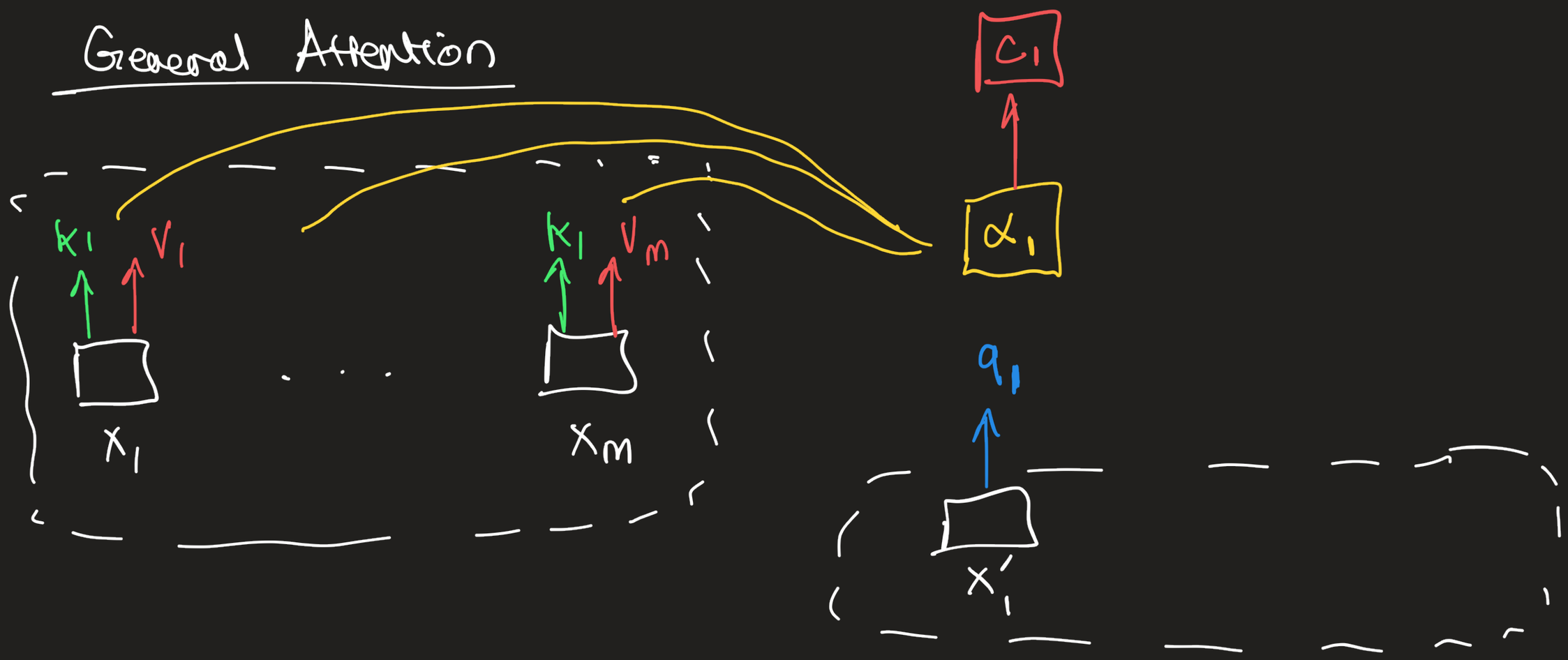
where key $k_i = W_K x_i$

query $q_i = W_Q x_i$

value $v_i = W_V x_i$

and $(\tilde{\alpha}_i)_j = \langle q_i, k_j \rangle$ and $\alpha = \text{softmax}(\tilde{\alpha})$

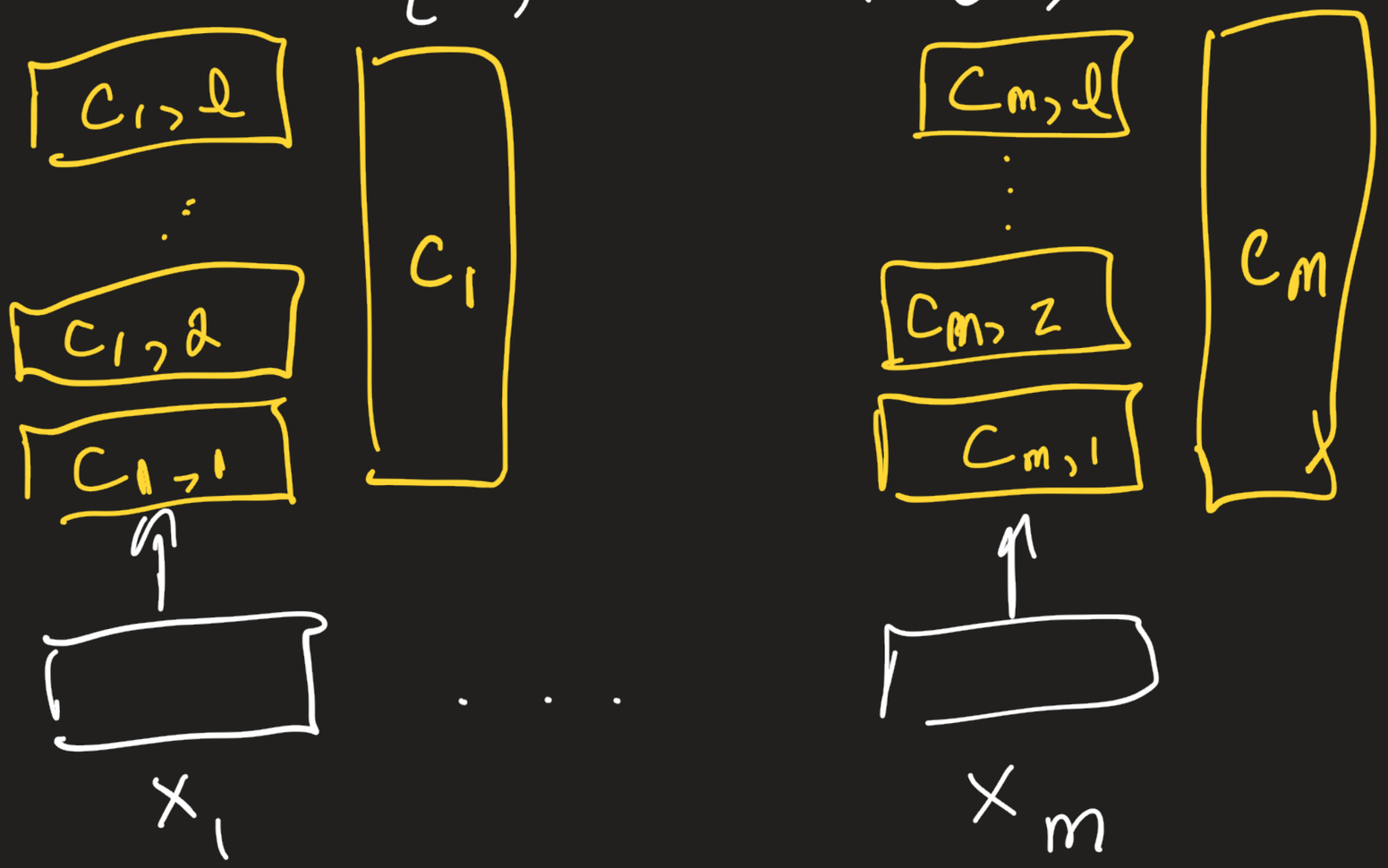
General Attention



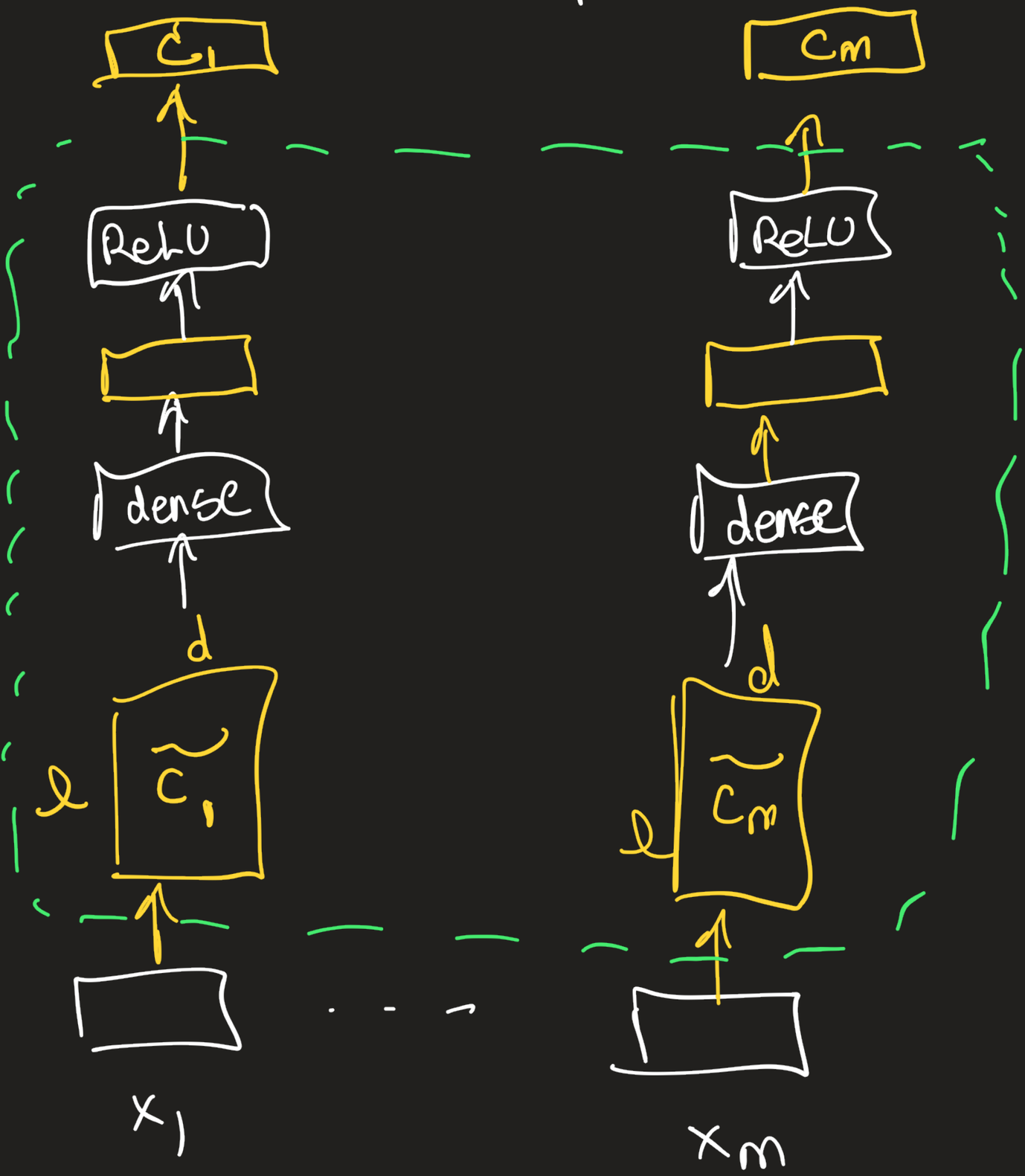
Multi-Head Self Attention

Just as with CNNs we benefit from learning multiple contextual representations:

$$(W_K)_i, (W_Q)_i, (W_V)_i \quad \text{for } i = 1, \ldots, h$$



Contextual representations using multi-head self-attention



Self-Attention + Dense Layer