Training a NLM:

1. compute loss (forward prop)
2. compute gradient via backprop
3. update params: $\theta_{new} = \theta_{old} - \eta \frac{dL}{d\theta}$

↳ coding-wise

↳ PyTorch ; autograd

↳

```
logits = model(prefix)
loss = cross_entropy(logits, target)
loss.backward()
optimizer.step()
```

logits: $z = W_o h$; L = cross entropy loss

$$\frac{dL}{dz} = q - p$$

q: model's predicted next-word dist

p: target dist

correct word	incorrect word
$p_i = 1$	$p_i = 0$
$q_i - 1 \leq 0$	$q_i - 0 \geq 0$

$$\frac{dL}{dW_0} = \frac{dL}{dz} \cdot \frac{dz}{dW_0}$$

$\frac{dL}{dz}$: $(q - p)$, $V \times 1$

$\frac{dz}{dW_0}$: h^T, $1 \cdot d_h$ $\hookrightarrow$ $V \times d_h$

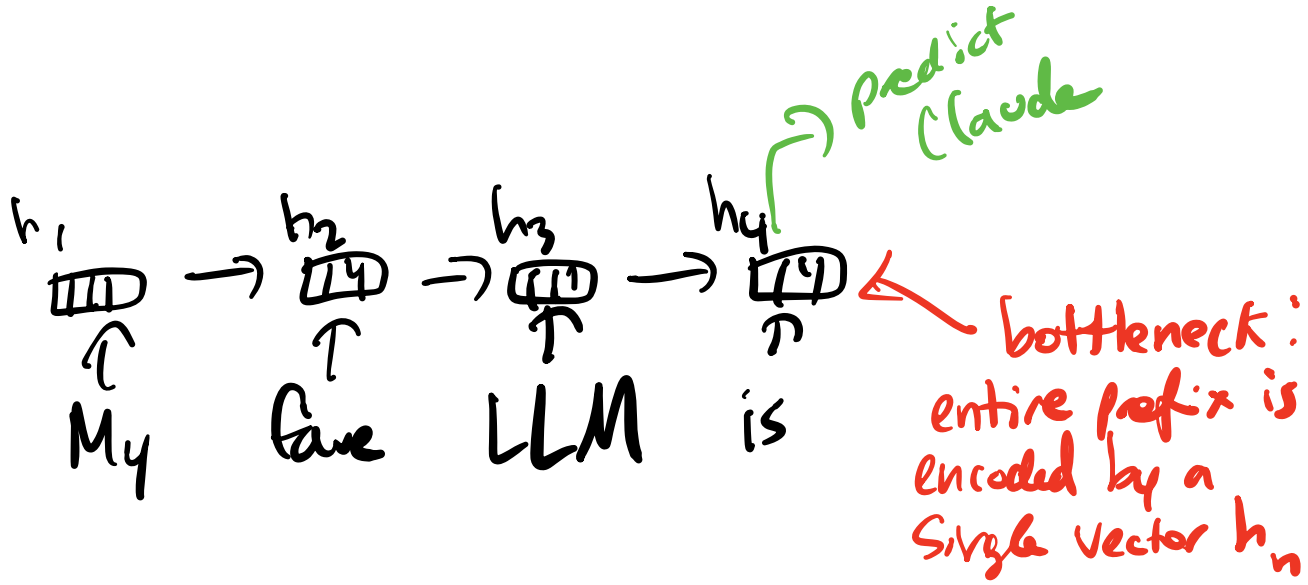
RNN:

↳ assume linear RNN

$$h_2 = W_h h_1$$
$$h_3 = W_h h_2$$
$$h_n = W_h h_{n-1}$$

$$\hookrightarrow h_n = W_h^{n-1} h_1$$

↳ vanishing gradient problem

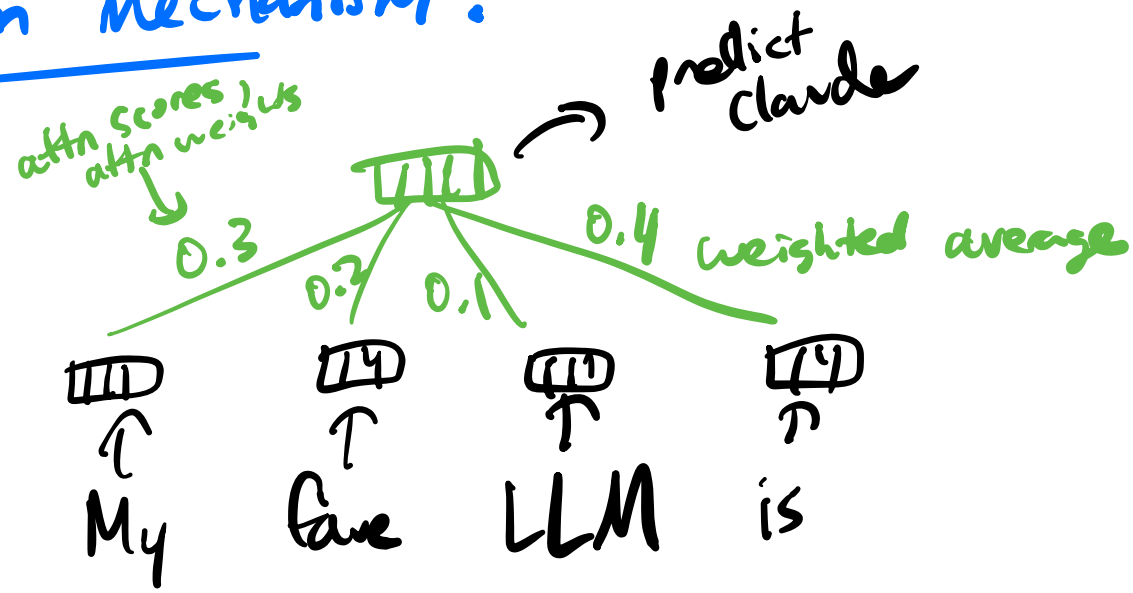


$$h_t = f(W_h h_{t-1} + W_e c_t)$$

cannot parallelize bc of sequential dependency

---

attention mechanism:

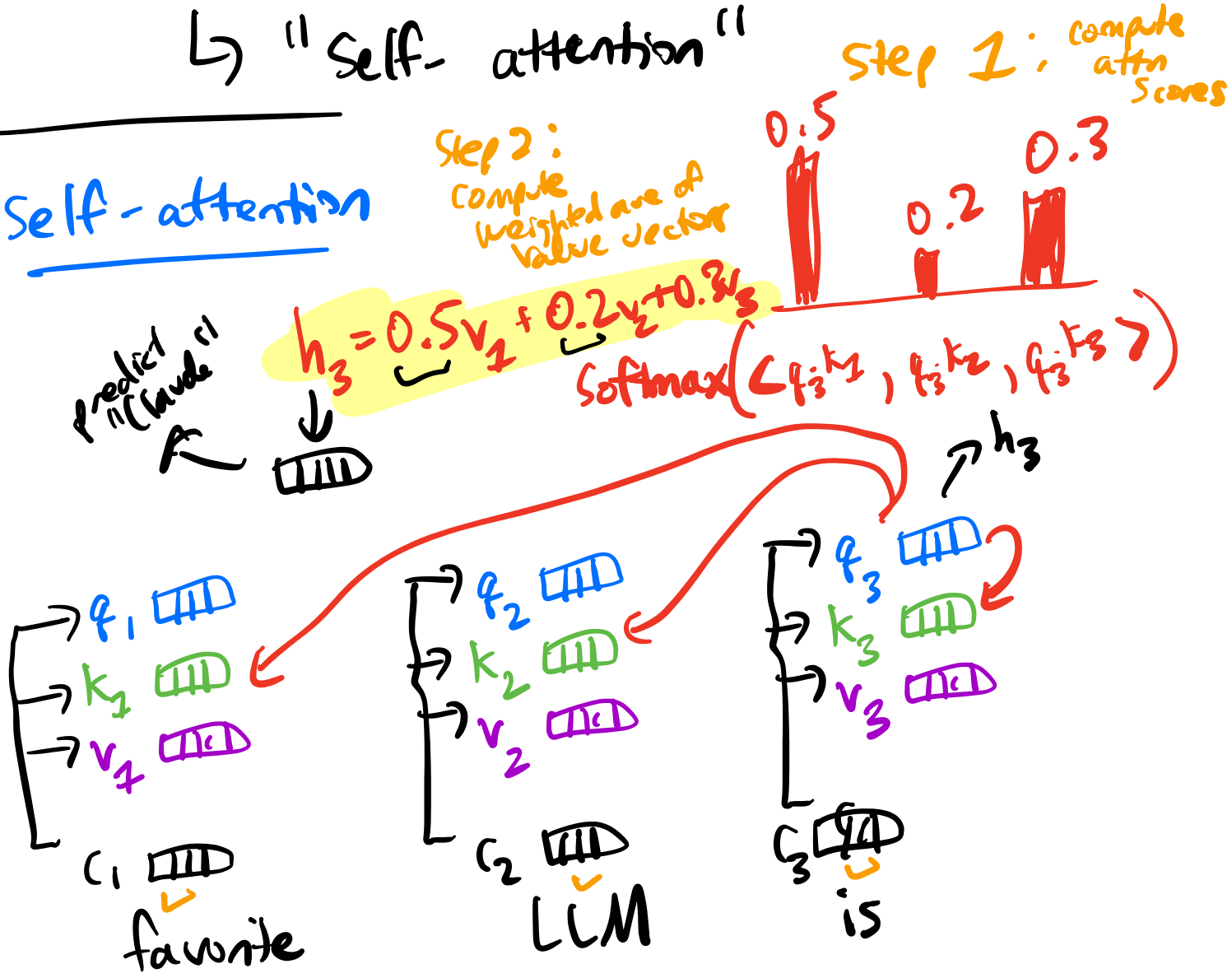


Transformer, Vaswani et al. 2017

↳ drop recurrence

↳ "self-attention"

Self-attention



query $q_1 = W_q c_1$

key $k_1 = W_k c_1$

value $v_1 = W_v c_1$

$q_2 = W_q c_2$

$q_3 = W_q c_3$

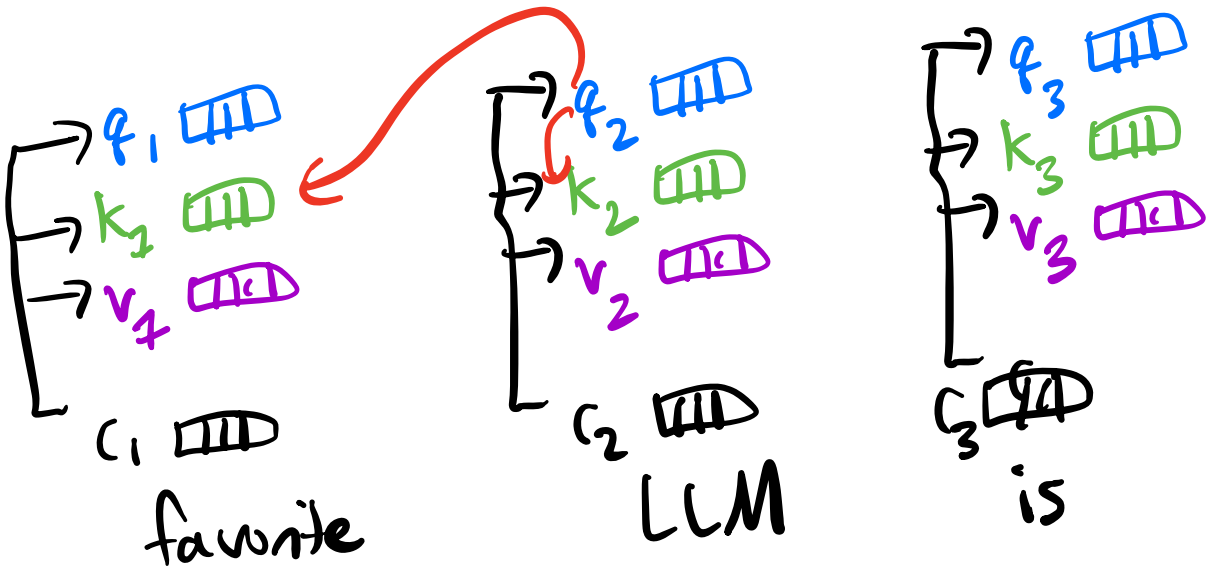
computation of h_2

predict "is"

$h_2 = 0.3v_1 + 0.7v_2$

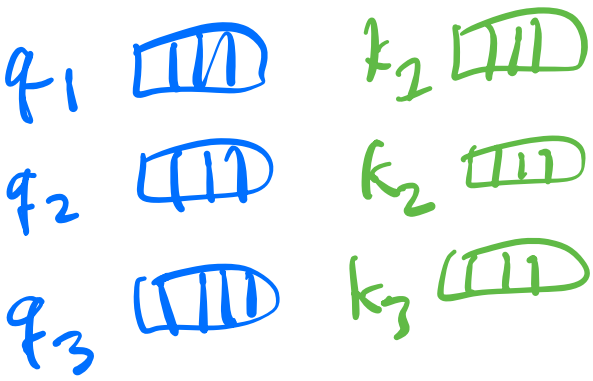


$\text{softmax}(\langle q_2 k_1, q_2 k_2 \rangle)$



↳ in self-attn, no dependency between h_t and h_{t-1}

parallelizing self-attn:



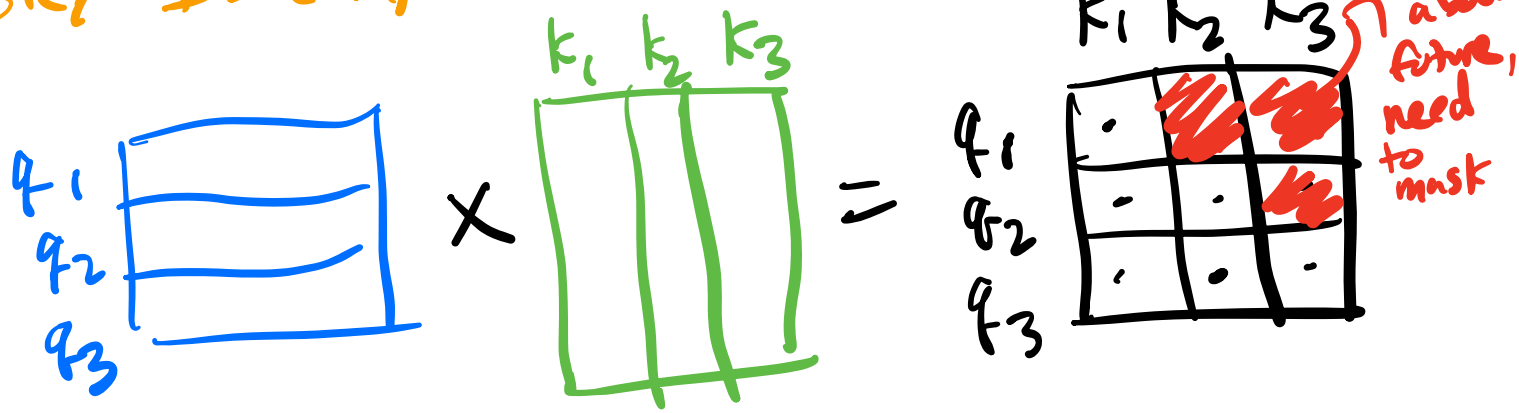
attn vectors

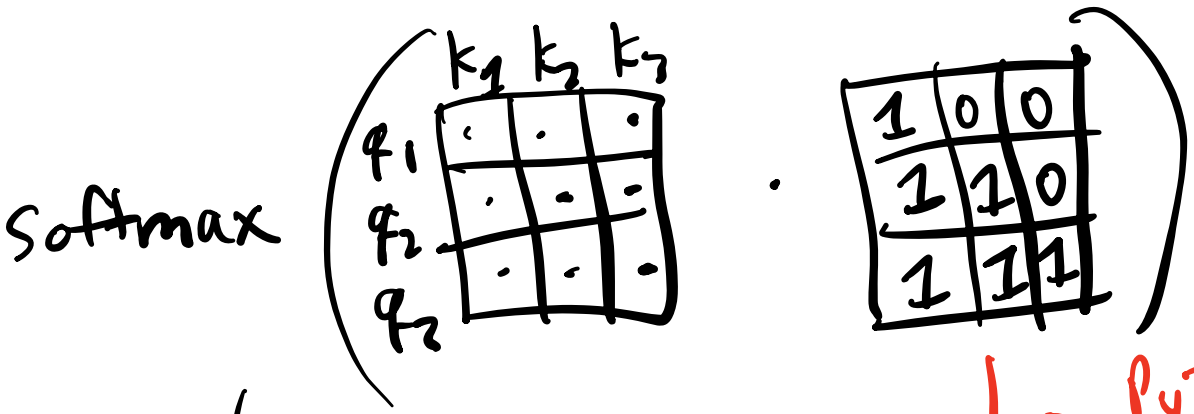
$a_1 : \langle q_1 \cdot k_2 \rangle$

$a_2 : \langle q_2 k_1, q_2 k_2 \rangle$

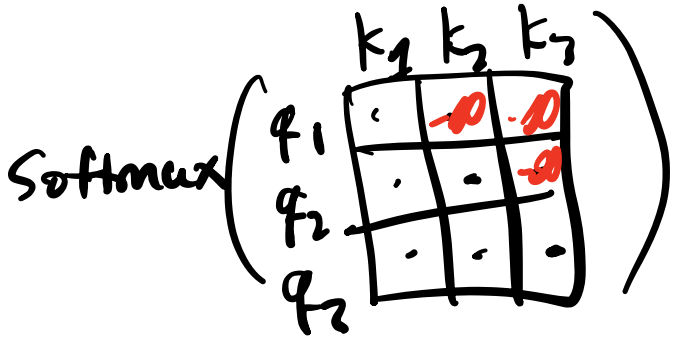
$a_3 : \langle q_3 k_1, q_3 k_2, q_3 k_3 \rangle$

step 1: compute attn scores



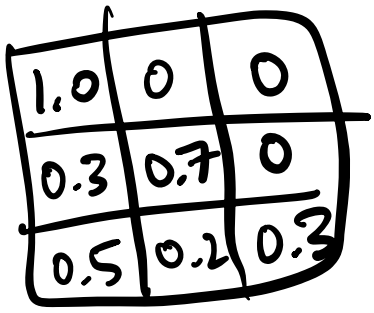


↳ PyTorch mask_fill (0, -∞)

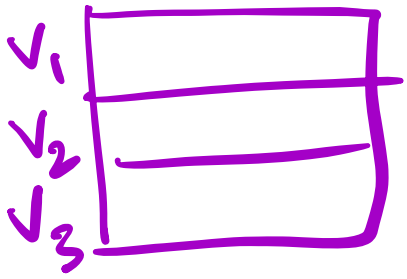


Step 2: weighted average

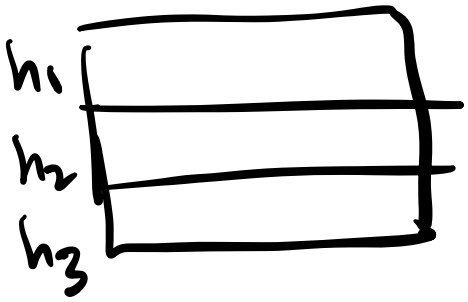
$=$



$\times$



$=$



↳ can only parallelize during training
not at test-time
decoding

---

what are the params of this self-attn model?

W_q, W_k, W_v, W_o, E