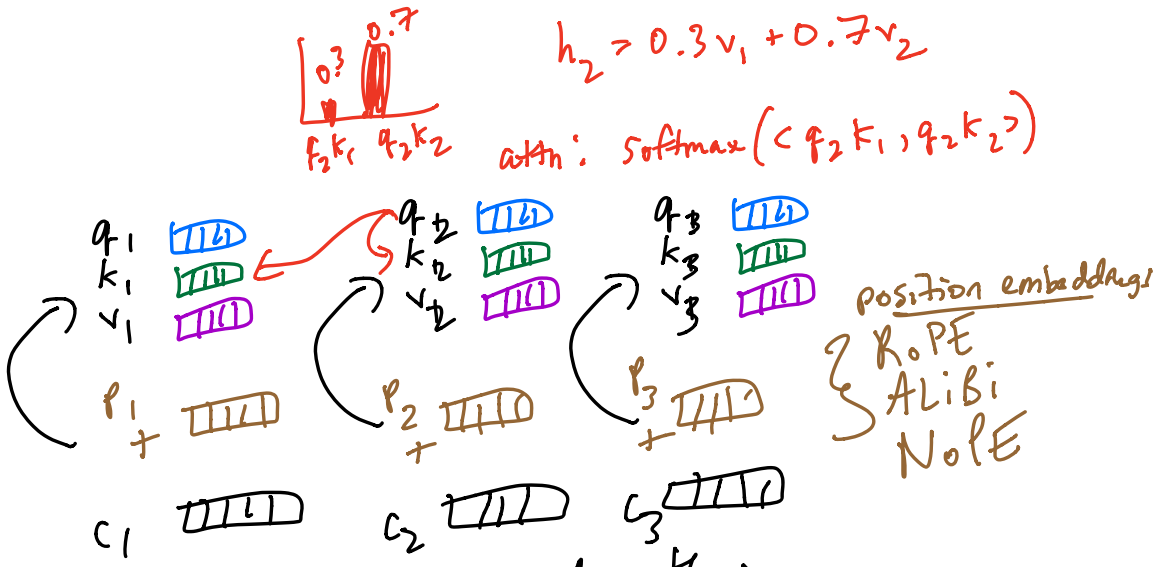


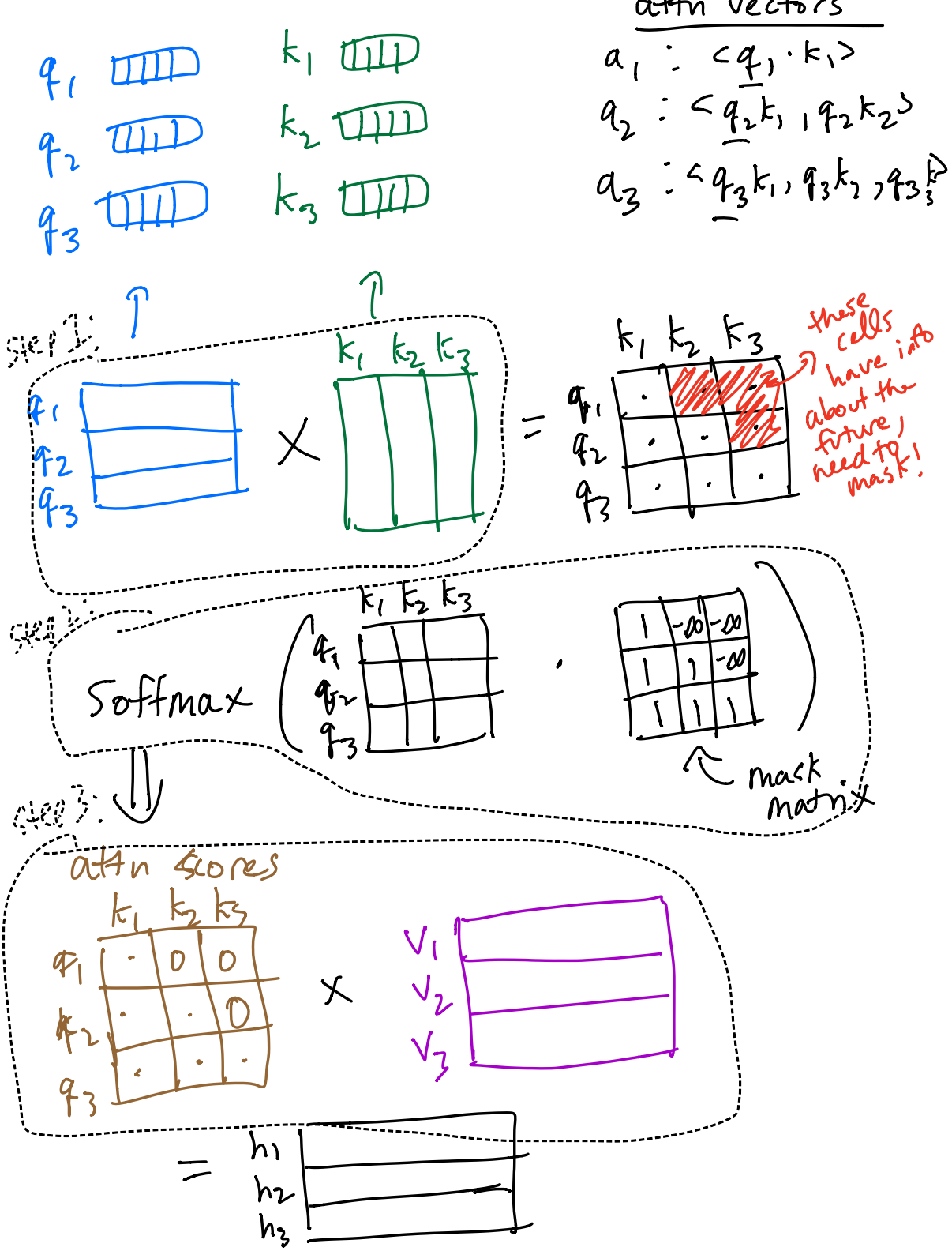
$$q_1 = f(W_q [c_1 + p_1])$$

$\Rightarrow$ no dependencies between h_1, h_2, h_3

↳ parallelize

↳ reduce bottleneck

how to parallelize a self-attn computation:



Self-attention:

↳ training time: parallelizable

↳ test-time: sequentially

⇒ we can only parallelize computations of hidden states when we know the full sequence ahead of time

test-time

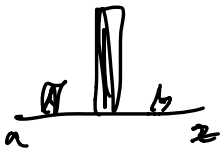
q_1, k_1, v_1 q_2, k_2, v_2 ?

↑ ↑

Students opened

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

multiply by value vecs ↳ h_2 → Softmax layer



↑

Sample "laptops"

q_1, k_1, v_1 . . . -

↑

Students opened laptops ?

KV cache

↳ storing prev. computed keys/values so you don't have to recompute every time step

Transformer

↳ neural LM built on multi-head self-attn

↳ "deep", stacked layers (or "blocks") of MH self-attn

↳ Vaswani et al. 2017

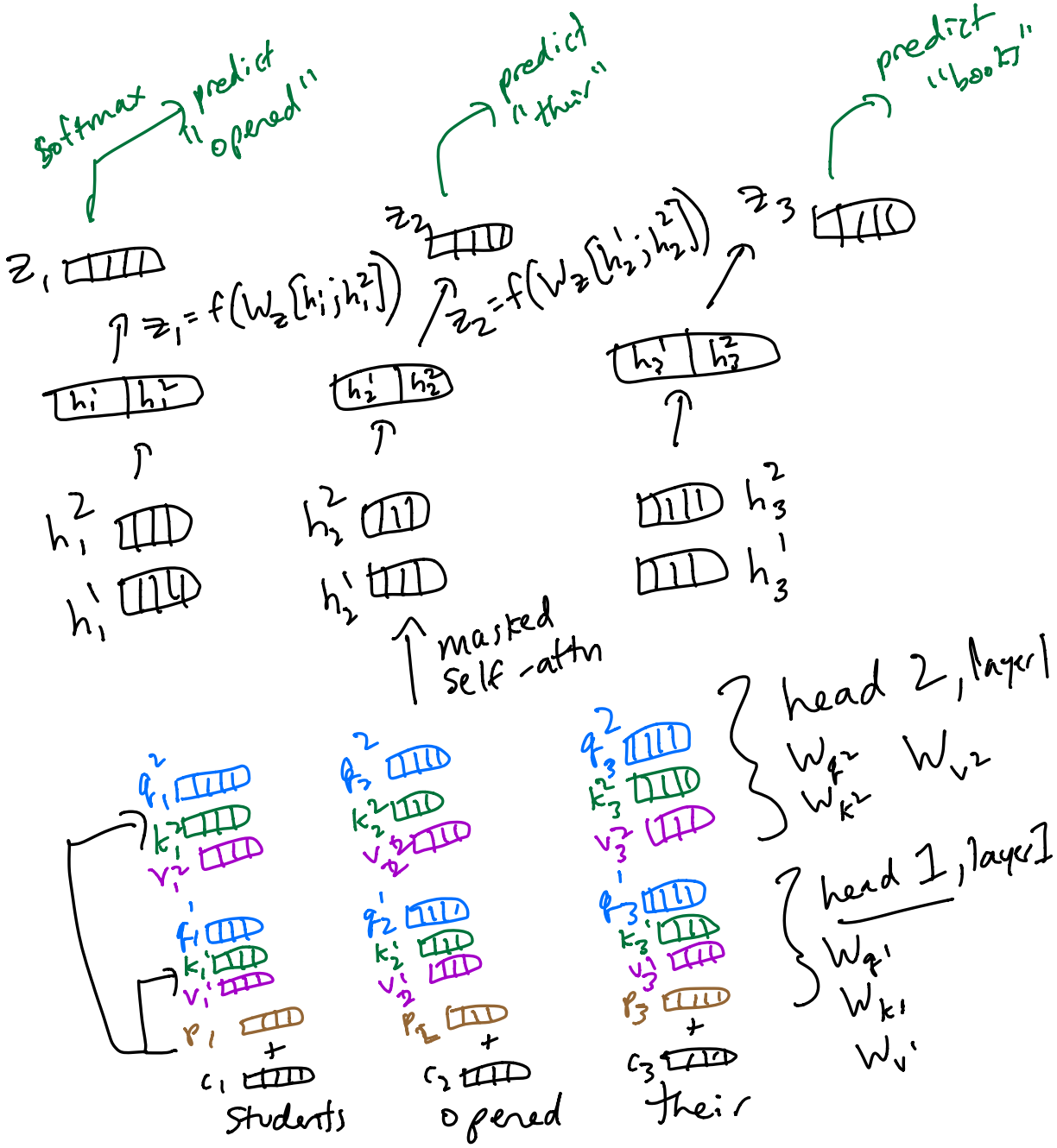
↳ intuition:

→ let's have multiple sets of q, k, v vectors for each token, then each "head" can focus on a specific linguistic property

↳ syntax (e.g. all verbs in prefix)

↳ activate on specific words/phrases

↳ entities/dates


softmax
predict "opened"
predict "their"
predict "books"
z_1
z_2
z_3
$z_1 = f(W_z[h_1^1; h_1^2])$
$z_2 = f(W_z[h_2^1; h_2^2])$
h_1^1 h_1^2
h_2^1 h_2^2
h_3^1 h_3^2
h_1^1
h_1^2
h_2^1
h_2^2
h_3^1
h_3^2
masked self-attn
q_1^2 k_1^2 v_1^2
q_1^1 k_1^1 v_1^1
p_1
c_1
students
q_2^2 k_2^2 v_2^2
q_2^1 k_2^1 v_2^1
p_2
c_2
opened
q_3^2 k_3^2 v_3^2
q_3^1 k_3^1 v_3^1
p_3
c_3
their
head 2, layer 1
W_{q^2} W_{v^2} W_{k^2}
head 1, layer 1
W_{q^1} W_{k^1} W_{v^1}

Adding depth: