$$\text{Softmax}(Wx)$$

W: weight matrix, $V \times d_h$

x: prefix representation d_h

$$P(w_n \mid w_{<n})$$

$w_{<n}$: prefix

What can we do with $\text{Softmax}(Wx)$?

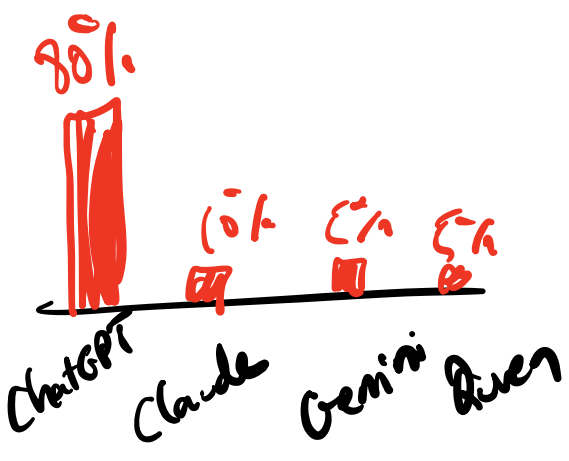
↳ Score a LM

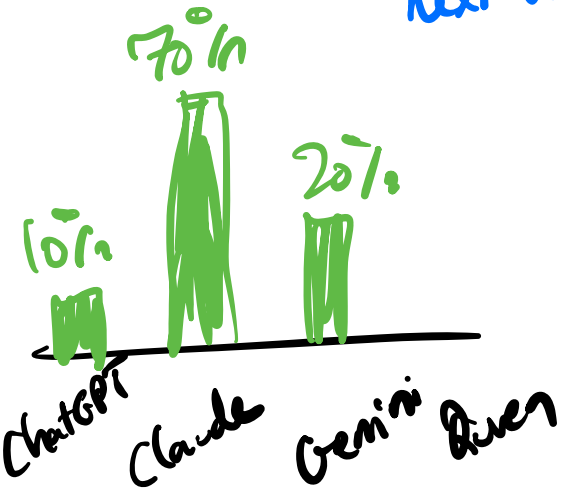
↳ assume we know ground-truth next word

My favorite LLM is Claude

"My favorite LLM is": prefix

"Claude": ground-truth next word





↳ training

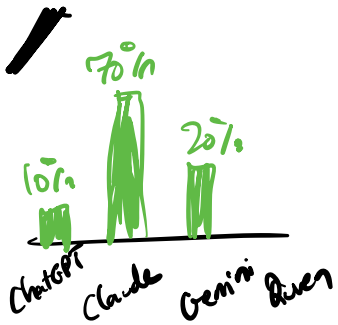
↳ if LM puts low prob on correct next word, big update to model parameters

↳ evaluation

↳ compute PPL (test)

↳ use softmax (W_k) to generate new text (decoding)

$$P(w_n | w_{<n})$$



My favorite LLM is Claude

prefix$_1$

Prefix$_2$

Prefix$_3$

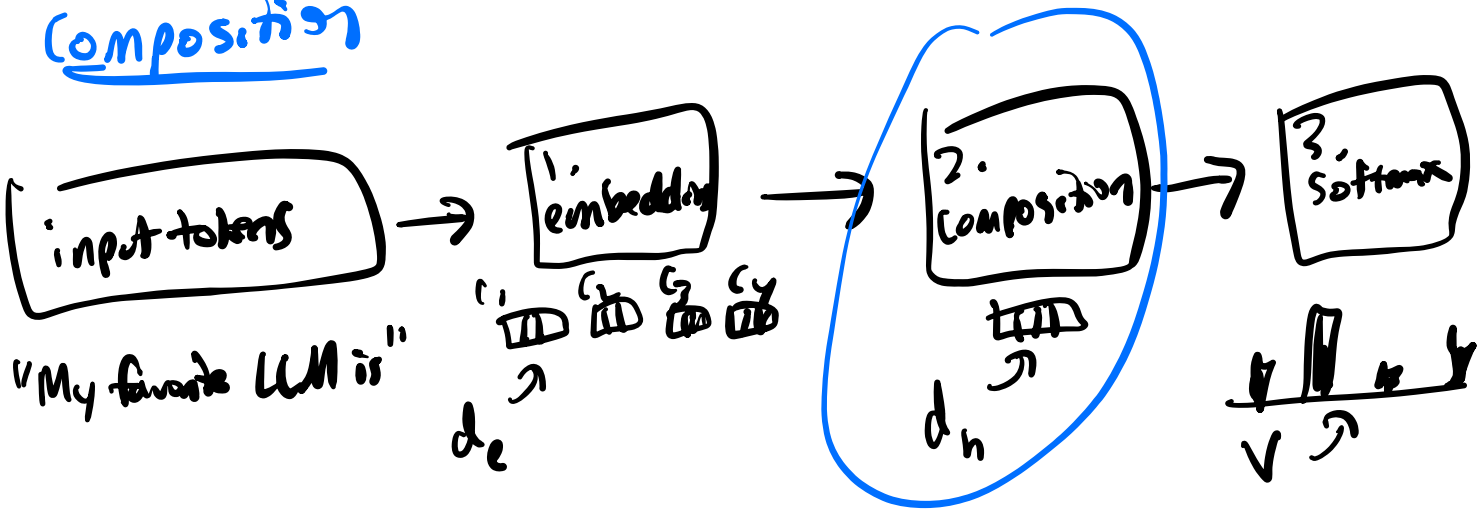
↳ greedy decoding
  ↳ argmax $P(w_t | w_{<t})$ at each timestep
  ↳ end decoding at <EOS> token
    or if some fixed number of timesteps
    has gone by

↳ ancestral sampling
  ↳ draw a word from $P(w_t | w_{<t})$
    proportional to its probability

↳ truncated sampling
  ↳ only sample from top-k most
    probable words
    ↳ "top-k" sampling
      k is fixed for all prefixes
  ↳ choose k depending on the
    entropy of the next word distribution
    ↳ pick a cum. prob threshold
      e.g. 90%

↳ top-p sampling, "nucleus sampling"

Composition



↳ input: sequence of d_e-dim word embeddings, one per input token

output: single d_h-dim vector representing the entire prefix

Possible composition functions:

1. element-wise function (sum, mean)
2. fixed-window neural LM
3. recurrent LM
4. Transformers (self-attention)

elementwise fn:

$$X = \frac{1}{n}\sum_{i=1}^{n} c_i$$

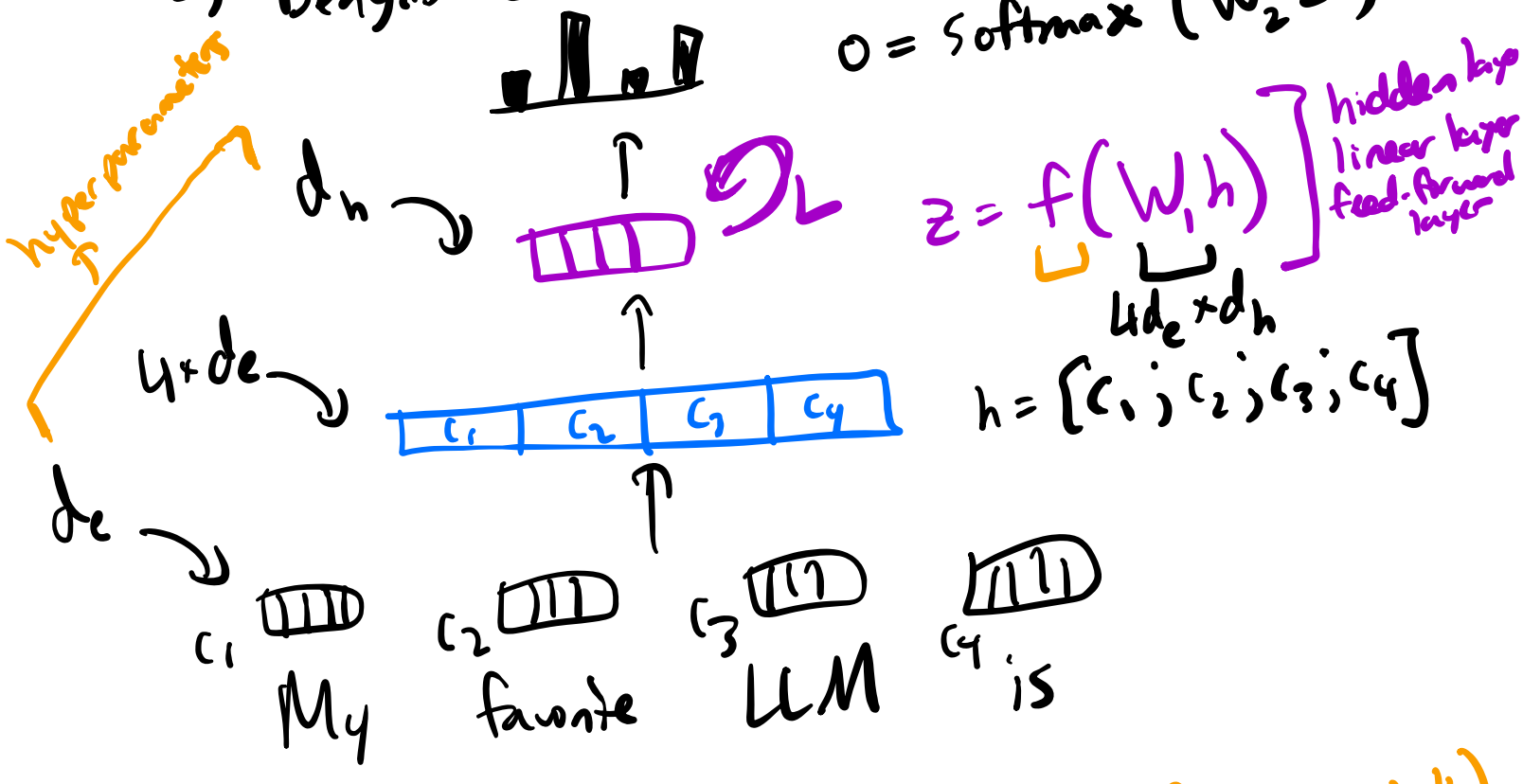
c_i → word embeddings

↳ order of words is lost

↳ $X_{\text{"my fave LLM is"}} = X_{\text{"LLM is fave my"}}$

fixed-word neural LMs:

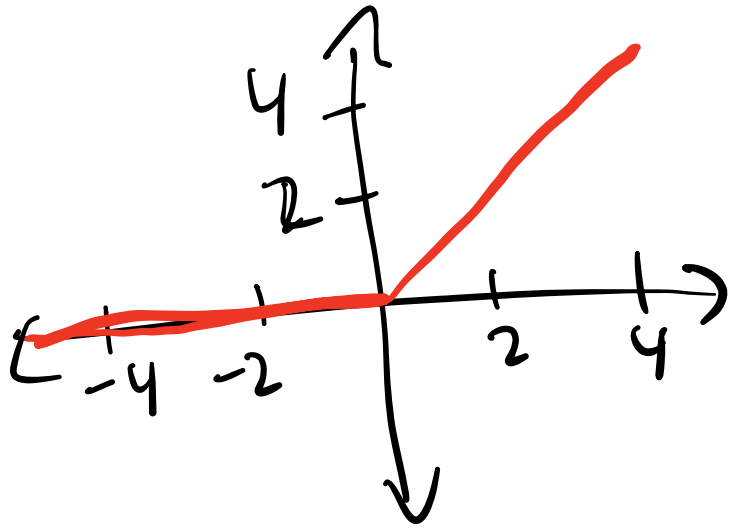
↳ Bengio et al. 2003



f is a nonlinear elemwise fn ("nonlinearity")

f: rectified linear unit ("ReLU")

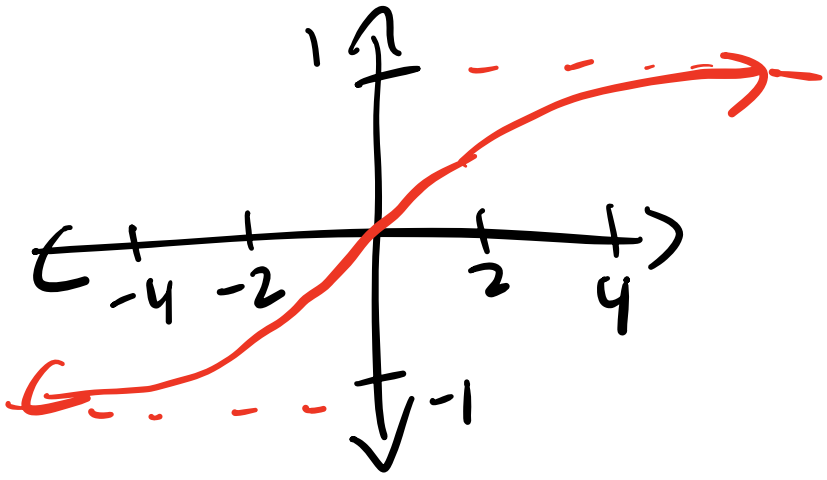
$$ReLU(x) = \max(0, x)$$



$$z_1 = W_1 h$$
$$z_2 = W_2 z_1$$
$$z_3 = W_3 z_2$$

$$z_3 = W h$$
$$W = W_1 W_2 W_3$$

$\tanh(x)$

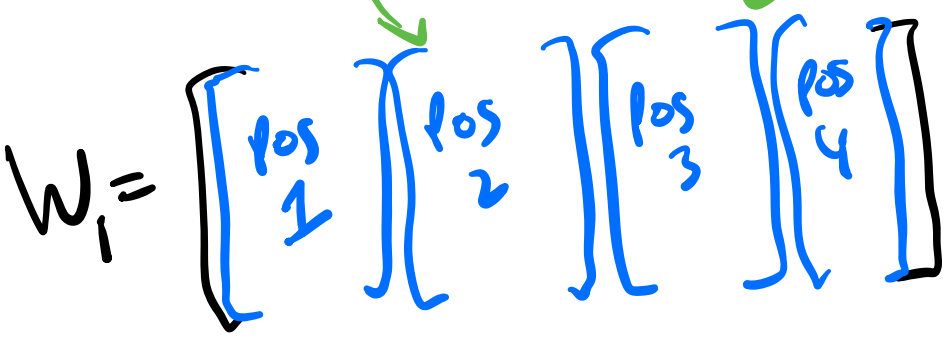


issue w/ fixed window LM:

↳ context size: no info beyond window

↳ "My favorite LLM is"
c_1 c_2 c_3 c_4

"favorite LLM really is"
c_1 c_2 c_3 c_4



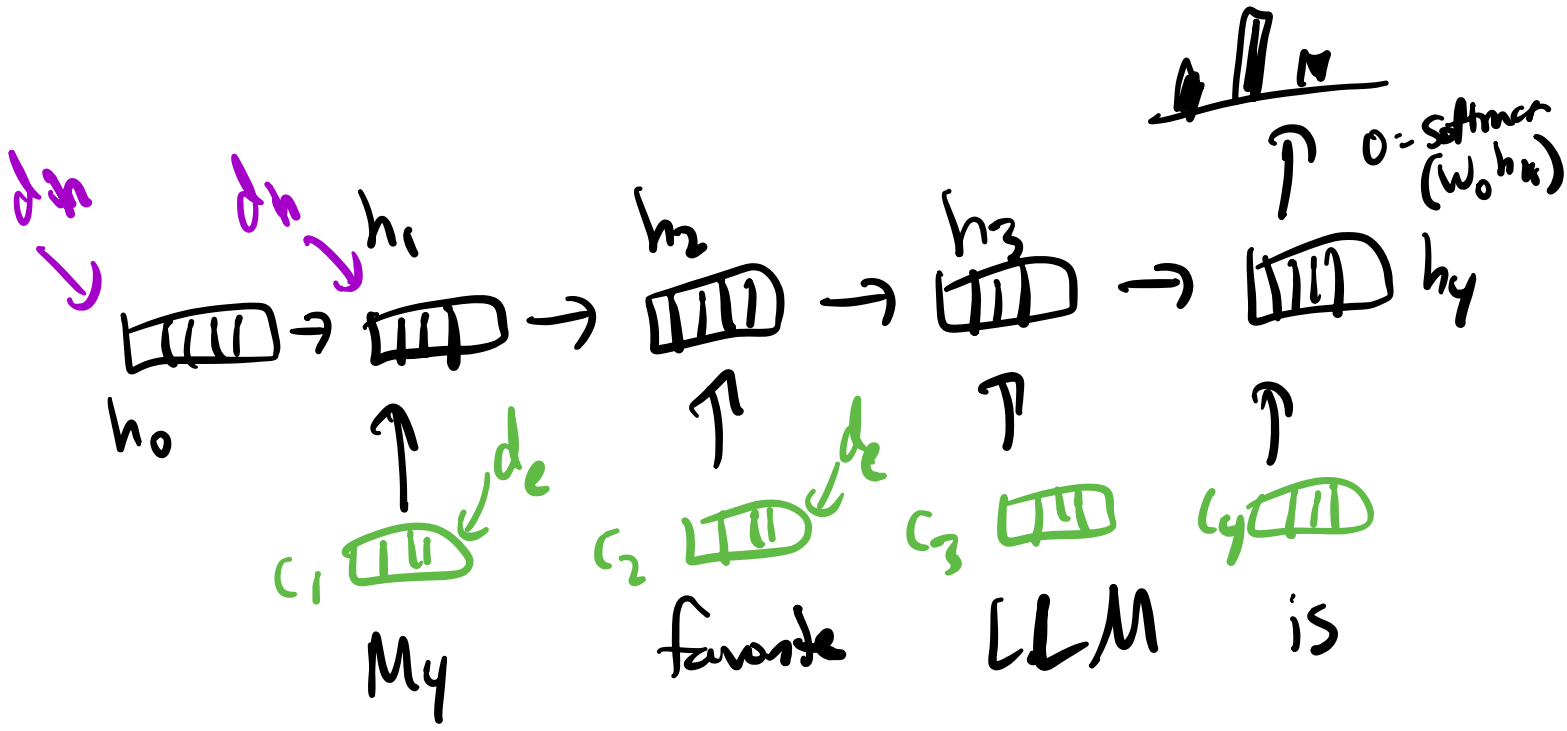
↳ no weight sharing among positions

↳ model has to learn what "LLM" means separately for each position in the window

---

Recurrent neural network

↳ "read" the prefix left-to-right one word at a time



recurrent update:

$$h_t = f(W_h h_{t-1} + W_c c_n)$$

W_h: $d_h \times d_h$

W_c: $d_h \times d_e$