Today

- Re-visiting "learning to add"; incorporating **attention**
- Generalizations of enc-dec
- (Time permitting) **bonus** t-SNE

From last Time

"7 + 15" $\rightarrow$ "22"
\[ \times \quad y \]

\[ E = \begin{bmatrix} \vdots \end{bmatrix} \]

\[ Z = h_t \]

\[ \hat{y}_1, \hat{y}_2 \]

\[ \text{If Teacher forcing, swap in } \hat{y}_1. \]
One drawback: The encoder must pack all relevant info into $z$.

Consider machine translation. It is natural to align outputs with inputs.

$C_t$ is a context vector induced at decoding step $t$.

This is a weighted sum over hidden states $h_i$ from enc.

Where do the weights $\alpha$ come from?

As in self-attention these come from a learned module!

$$\alpha_t = \text{SoftMax} (S_t)$$

Vector of Scores at Time $t$

Attention over hidden states at time $t$

$S_{ti}$ should capture the relevance of state $h^e_i$ with respect to the current decoder hidden state $h^d_t$.

\[ C_t \rightarrow \sum \alpha_{ti} h_i \]

\[ L_e \rightarrow \text{chat} \rightarrow \text{noir} \]
One version ("additive attention")

\[ S_{ti} = \sqrt{T} \tanh(W[ h_i^e \odot h_t^d]) \]

\[ S_{ti} = \text{CosineSim}(h_i^e, h_t^d) \]
\[ S_{ti} = h_i^e \odot h_t^d \]

Other options

Coming back to our example...

"10 + 15" \rightarrow "25" might hope for something where we attend to first digits, then second...

Recall that for Transformers we thought about Queries, Keys, Values. What are these here?

- Queries \quad \text{decoder hidden states}
- Keys, Values \quad \text{encoder hidden states}

Let's see in Colab!

So this is attention for Seq2Seq / enc-dec architectures; can also use for unstructured tasks
We have introduced Seg2Seq models with RNNs, but we can also use Transformers as our encoder and decoder.

**BART** (Lewis et al., 2019)

This helps generalize pre-training as we can define "noising" tasks!
Decoding

For generation tasks we have been assuming greedy decoding: argmax at every step, based on $h_t$.

Another strategy is beam search in which we keep the top-k hypotheses $H_k$. 