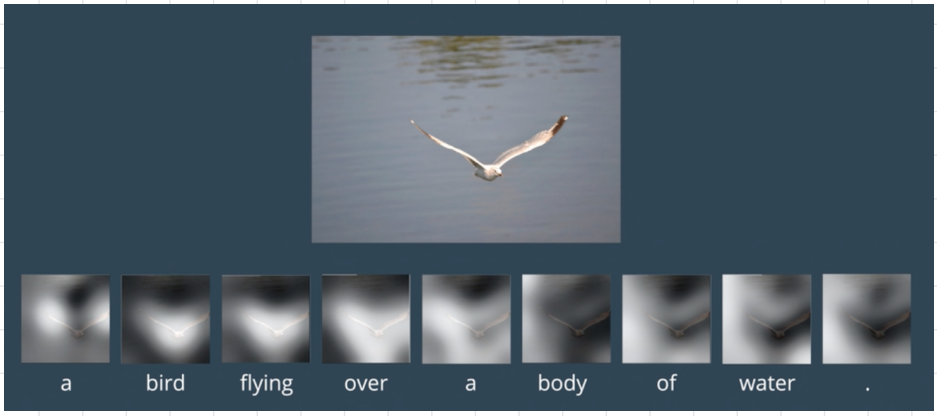
Attention

Introduction to Attention

"One important property of human perception is that one does not tend to process a whole scene in its entirety at once. Instead humans focus attention selectively on parts of the visual space to acquire information when and where it is needed, and combine information from different fixations over time to build up an internal representation of the scene, guiding future eye movements and decision making."

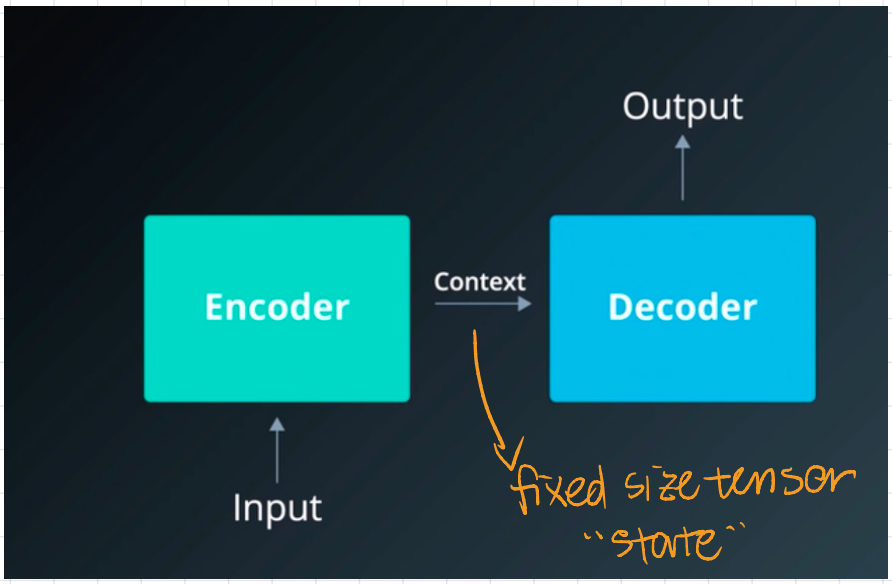
— Recurrent Models of Visual Attention, 2014

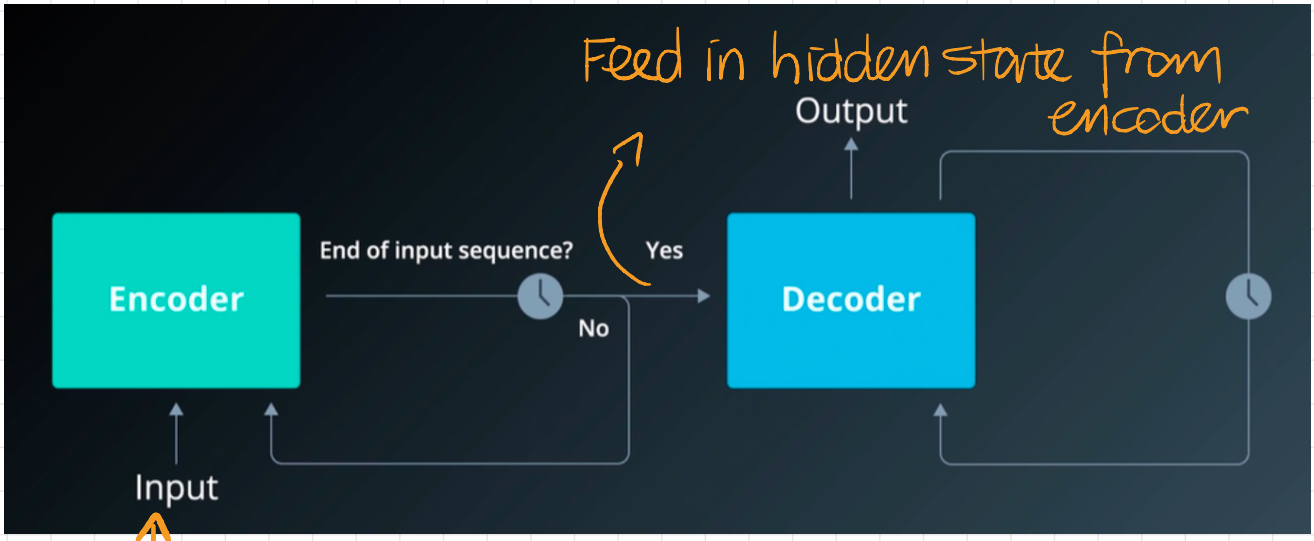
- Eyetracking device
- Classifrcatrion of images
  - CNN looks at entire image to be able to class -ify.



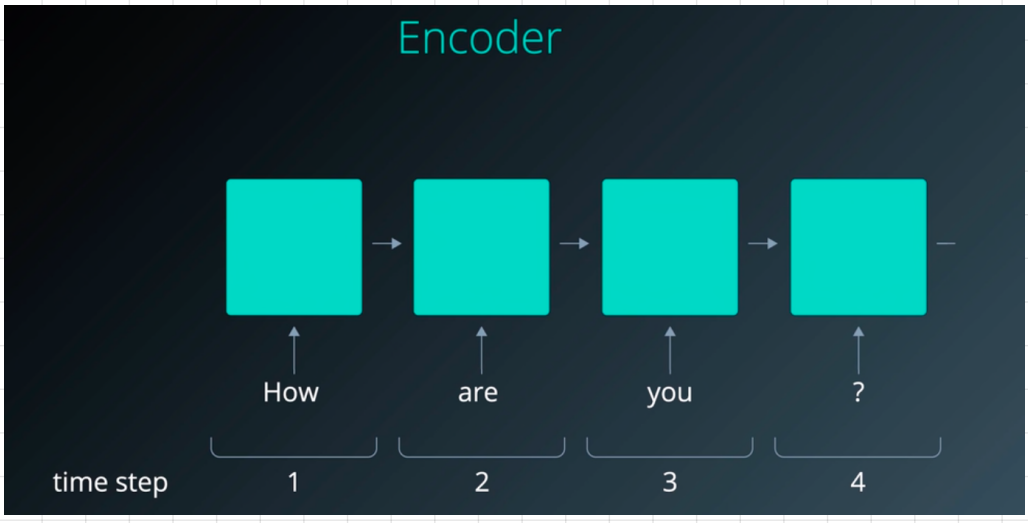
- Machine neural translation
  - Classic seq to seq models uses the entire sentence input to produce every single small output
  - Attention can use small parts of input to start translate.

Encoders & Decoders

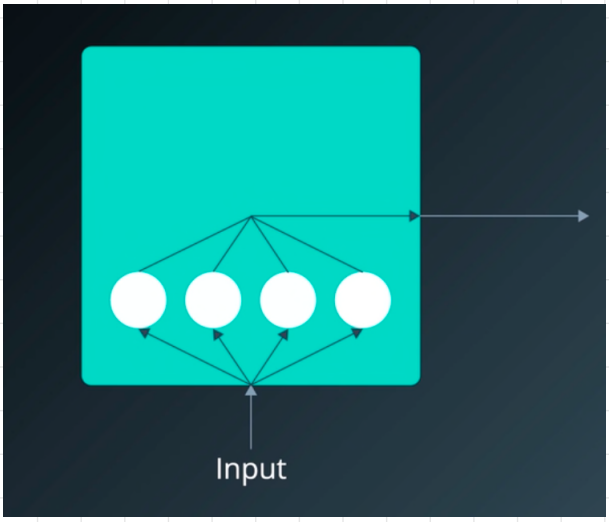




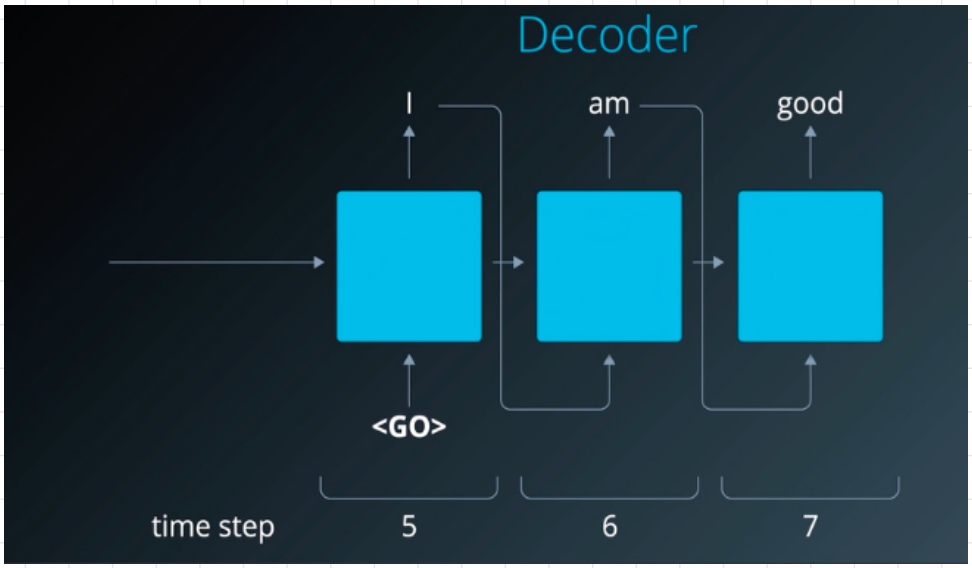
How are you? ⟩— 4 tokens, will take the RNN 4 timesteps to read



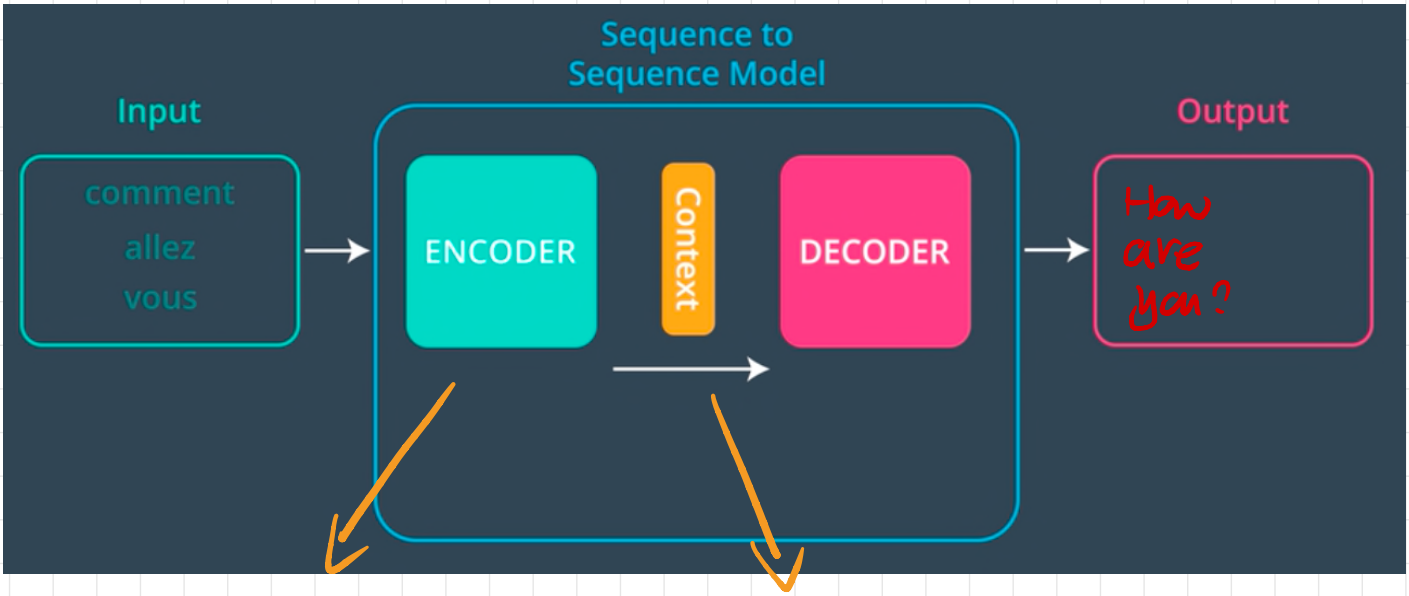
Hidden state

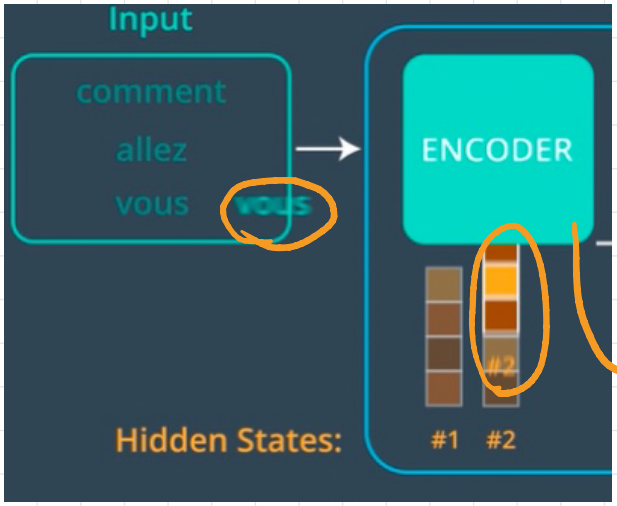


Hidden_dim?



Seq to seq Model





$h_t = f\{h_{t-1}, x_t\}$

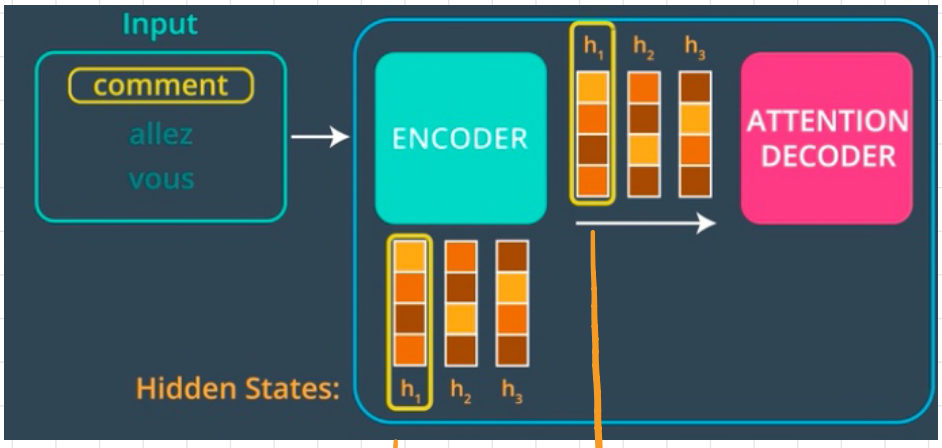
The last h generated from the input ≡ the context vector

limitation:

- The encoder can only send a vector to decoder
- Overfitting if dim of context vector too large when dealing w/ short sequences.

Sequence to Sequence, with Attention

Encoder:

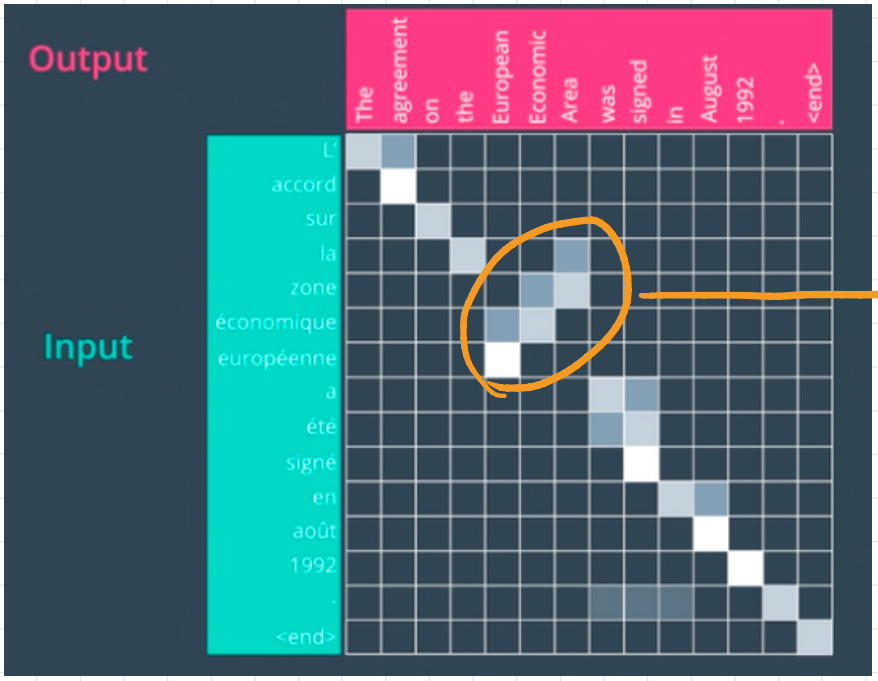


Context vector
=
all the hidden states

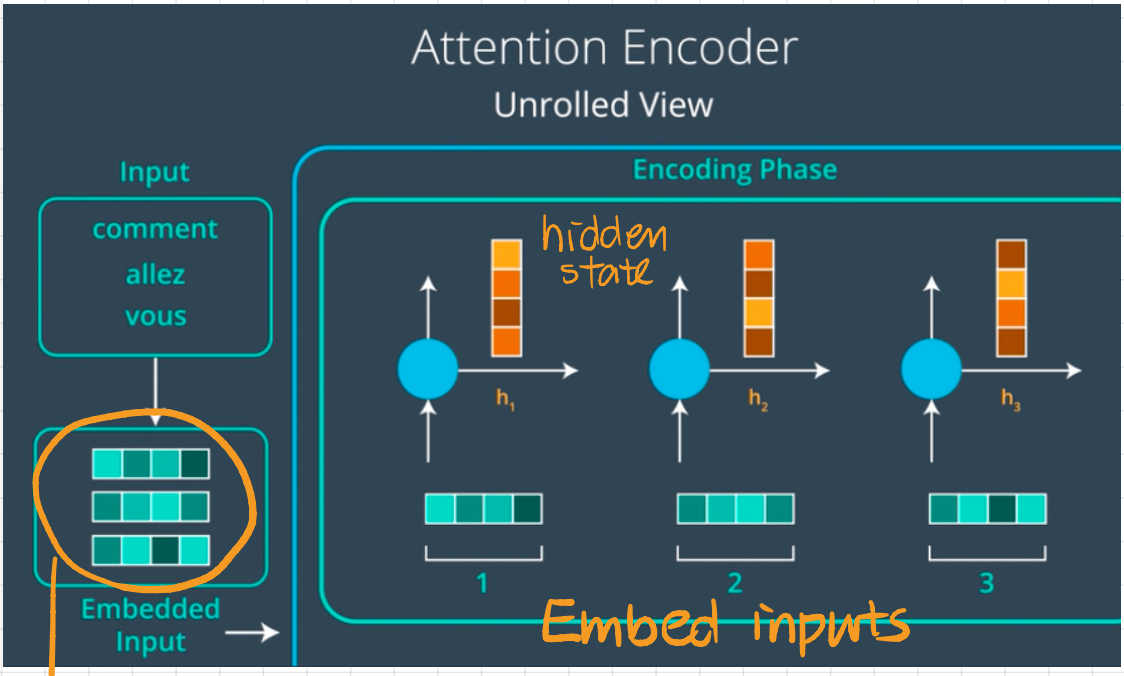
Longer sequence
= longer context vec

Each of own focus

Decoder:

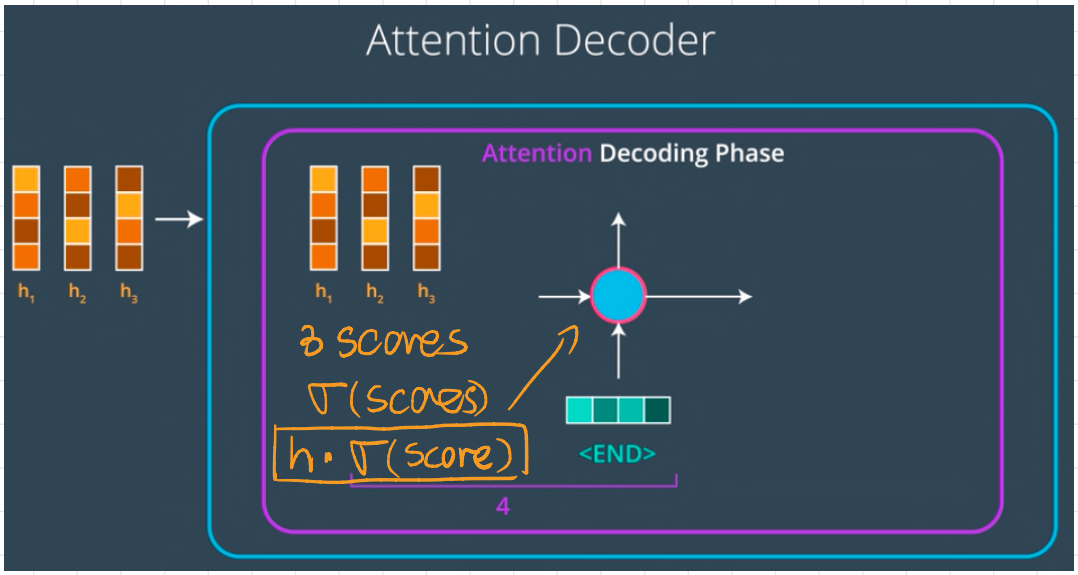


After training.
Adapt to the grammar of English



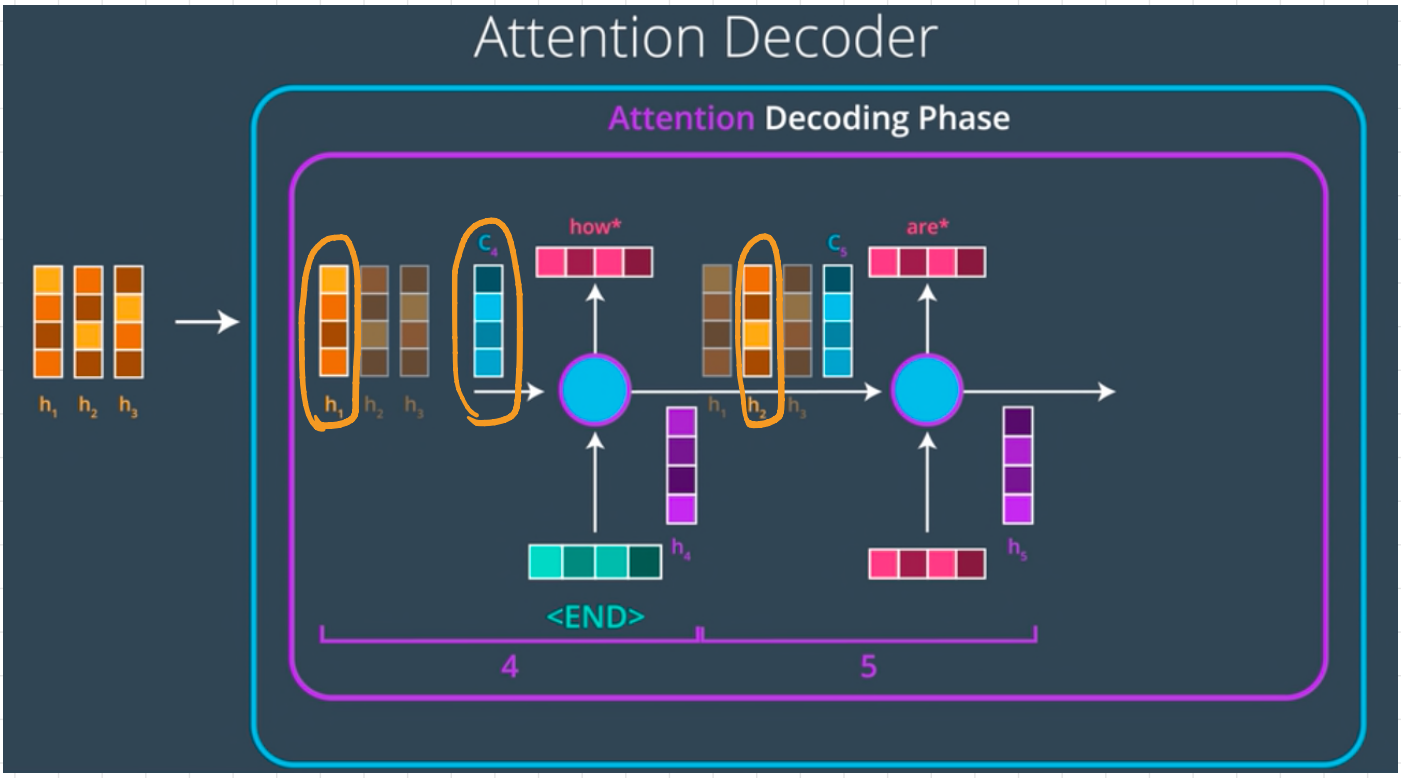
→ Decoder

columns: embedding dim
(In real world, this will be much higher)



Before feeding hidden states into decoder:

1. Each hidden state applied with a scoring function.
2. Obtain a score for each h. n scores total
3. Apply softmax to scores
4. ∇(Score of each) x each h = context vector to enter decoder



To translate:

- hidden state of each word from encoder + hidden state of decoder / output from prev. decoder cell + context word => output

Additive & Multiplicative Attention

Bahdanau Attention

$$e_{ij} = v_a^\top \tanh(W_a s_{i-1} + U_a h_j)$$

A scoring function

s_{i-1}: hidden state from decoder in prev timestep

h_j: hidden state from encoder

$$a_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

Softmax

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

Weighted Sum

Luong Attention

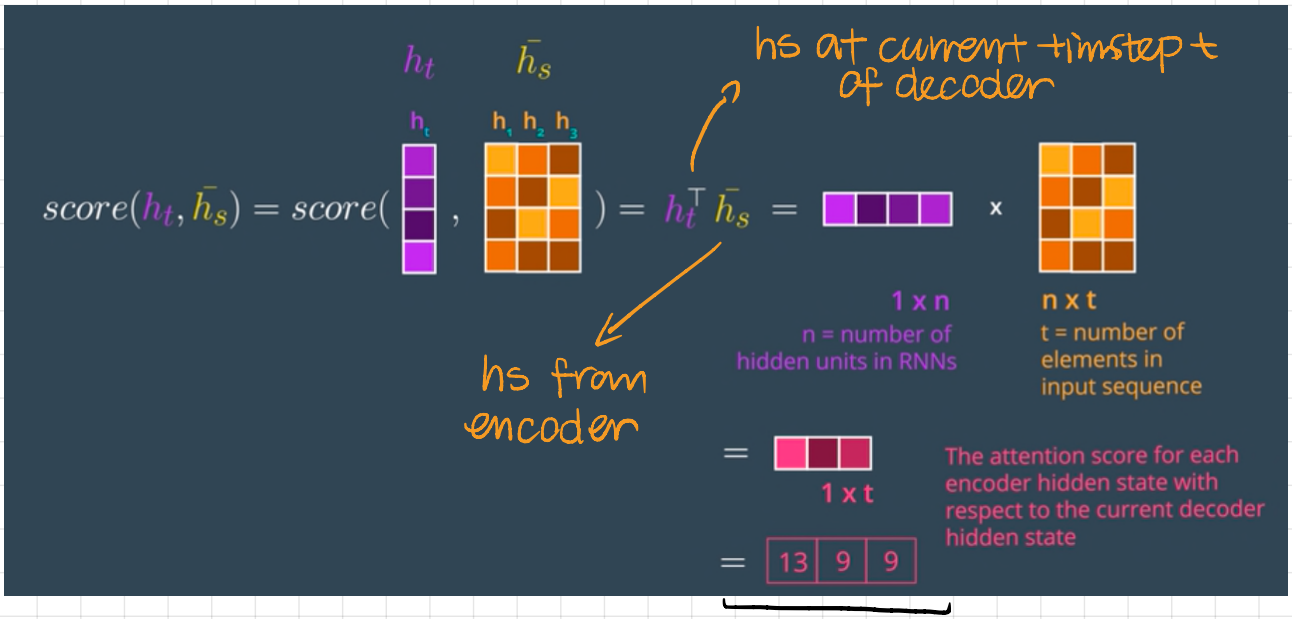
$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \bar{\boldsymbol{h}}_s & \textit{dot} \\ \boldsymbol{h}_t^\top \boldsymbol{W}_a \bar{\boldsymbol{h}}_s & \textit{general} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W}_a [\boldsymbol{h}_t; \bar{\boldsymbol{h}}_s]\right) & \textit{concat} \end{cases}$$

$$\begin{aligned} \boldsymbol{a}_t(s) &= \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \\ &= \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \end{aligned}$$

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_c [\boldsymbol{c}_t; \boldsymbol{h}_t])$$

Multiplicative

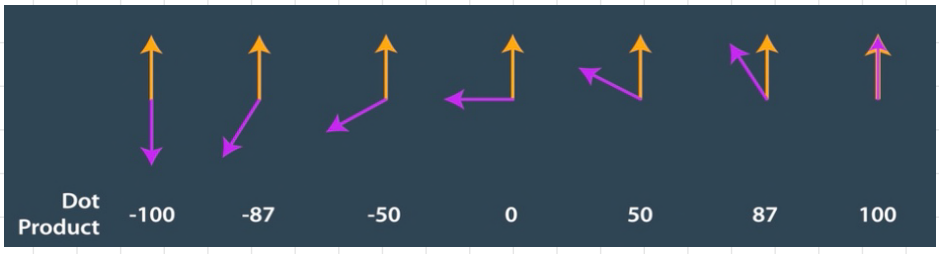
Scoring method #1. Simply dot product



Score for each hs in columns!

Intuition:

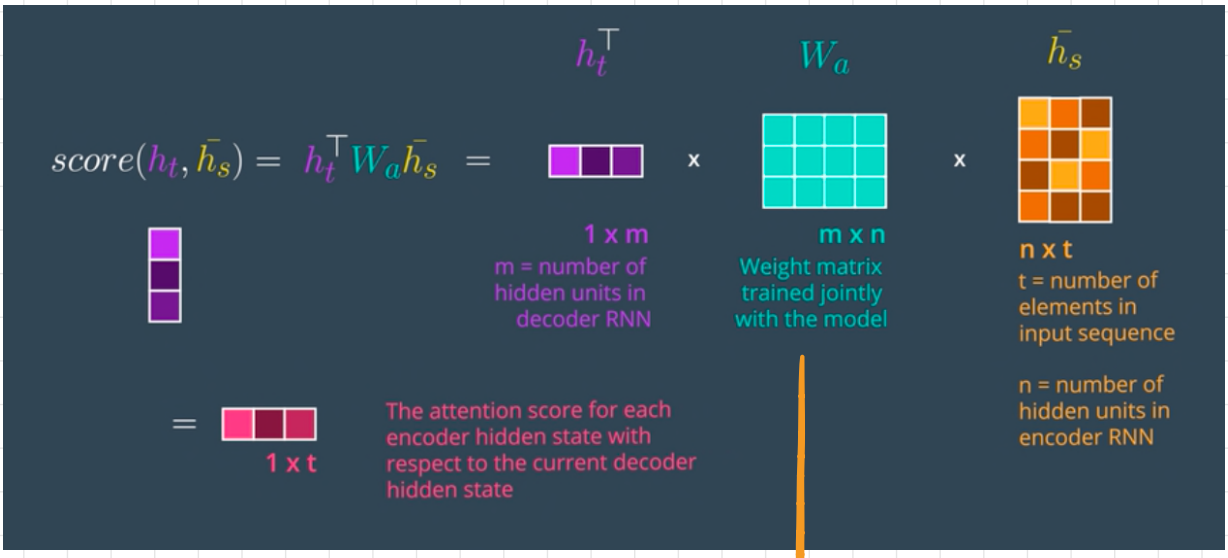
- One scoring method would be the dot product
  - DP returns a single number
  - DP ≡ similarity measurement: ↓ angle ↑ value / sim.



Drawback:

- Assume that encoder & decoder have same embeod space
- Diff languages have diff embed space.

Scoring Method #2 : General



Unite the embed dim