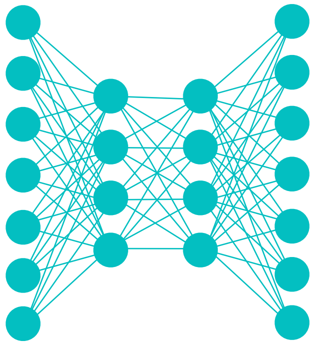
Lecture Notes for

Neural Networks and Machine Learning

Transformers and Vision Transformers



Logistics and Agenda

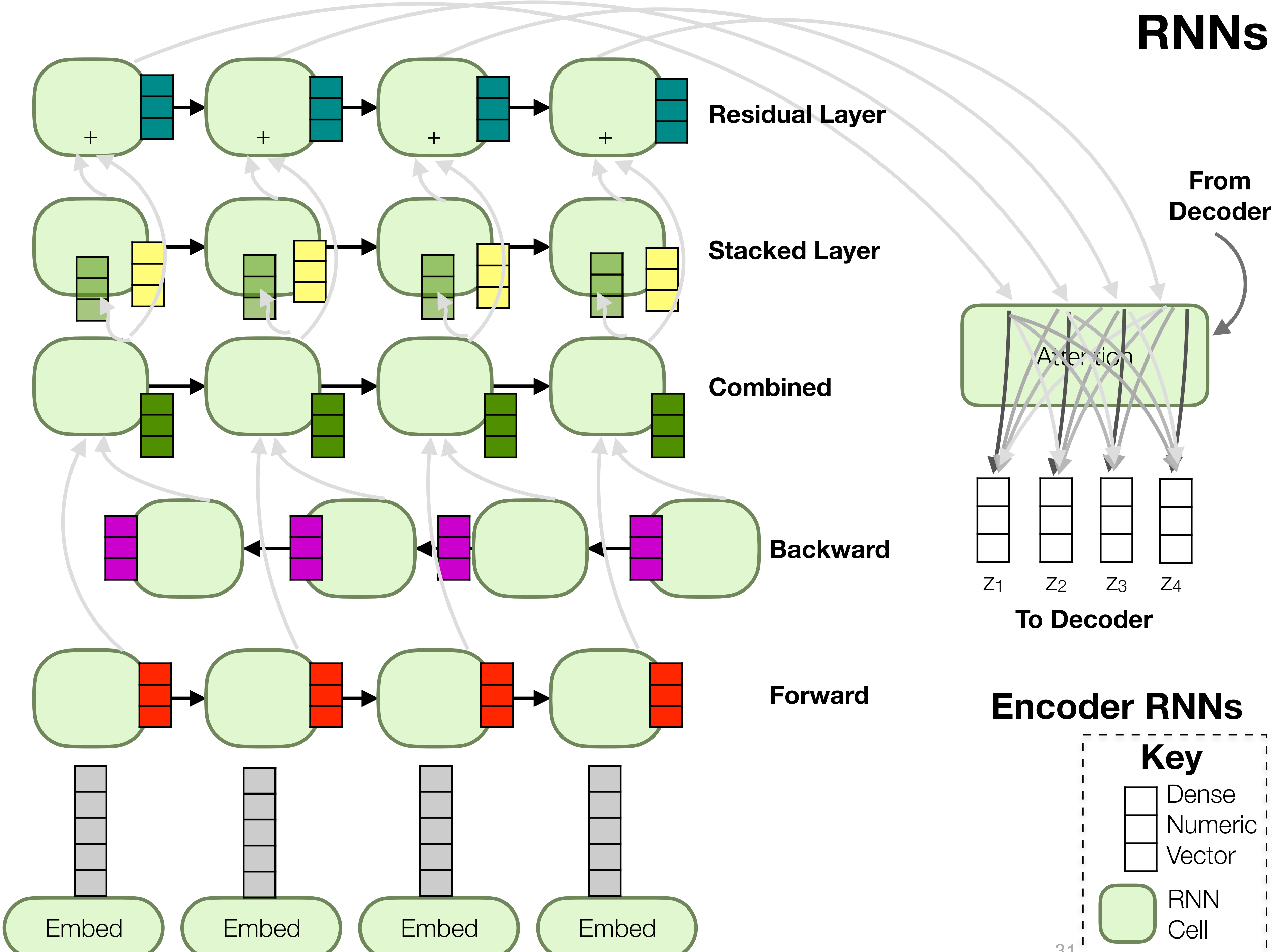
- Logistics
  - Paper presentations (Thurs)
- Agenda
  - Transformers
- Next Time:
  - Vision Transformers
  - Paper Presentation
  - Consistency losses

Transformers

Dr Simone Stumpf @DrSimoneS... · 13h ···

God grant me the confidence of an average machine learning expert.

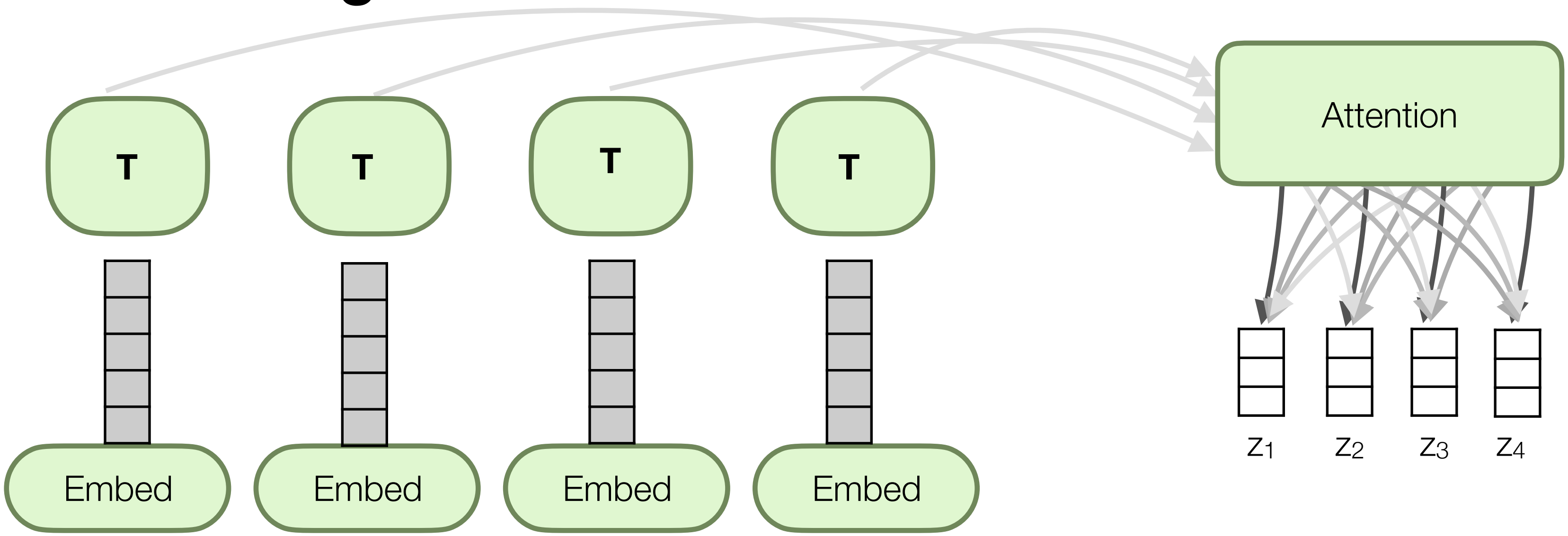
RNNs



Encoder RNNs

Transformers Intuition

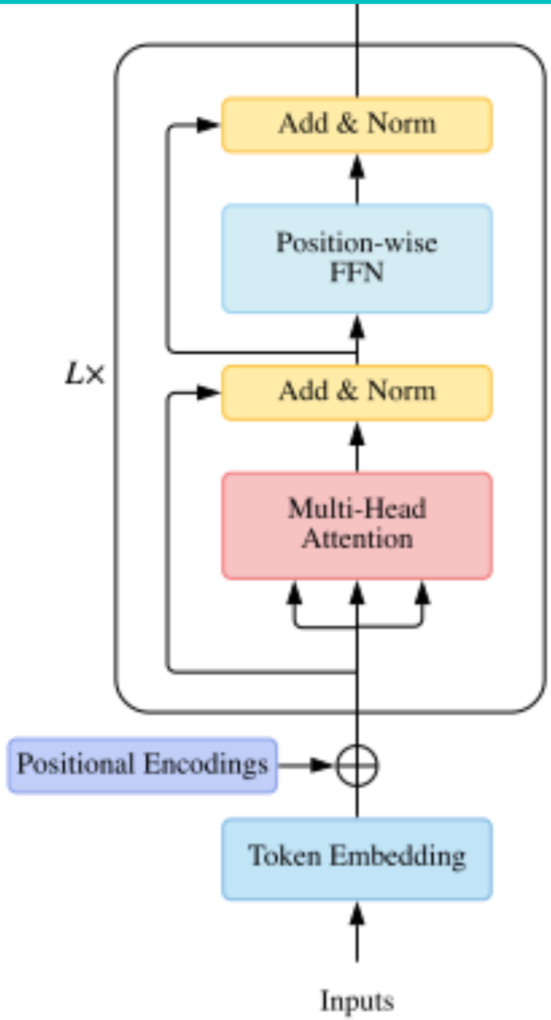
- Recurrent networks track state using an “updatable” state vector, but this takes lots of processing to across sequence
- Attention mechanism (in RNNs) already takes a weighted sum of state vectors to generate new token in a decoder
- … so why not just use attention on a transformation of the embedding vectors? **Do away with the recurrent state vector all together?**

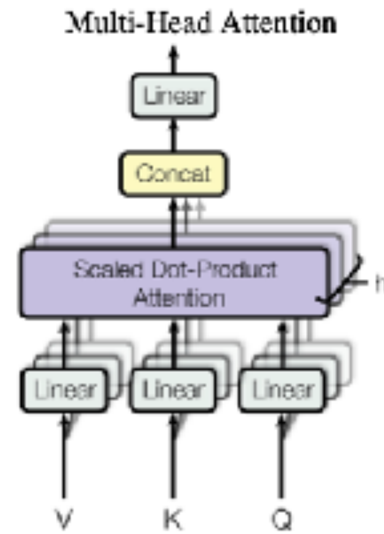


Attention is All You Need

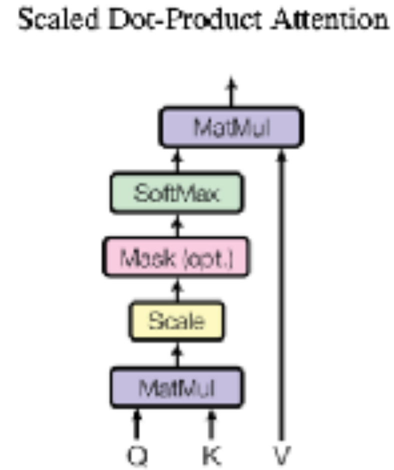
- **Continued Motivation**:
  - RNNs are not inherently parallelized or efficient at remembering based on state vector
  - CNNs are not resilient to long-term word relationships, limited by filter size
- **Transformer Solution:**
  - Build attention into model from the **beginning**
  - Compare all words to each other through **self-headed** attention
  - Define a notion of “**position**”in the sequence
  - ***Should be resilient to long term relationships and be highly parallelized for GPU computing!!***

Transformer Overview





more than one
Q,K,V use in document



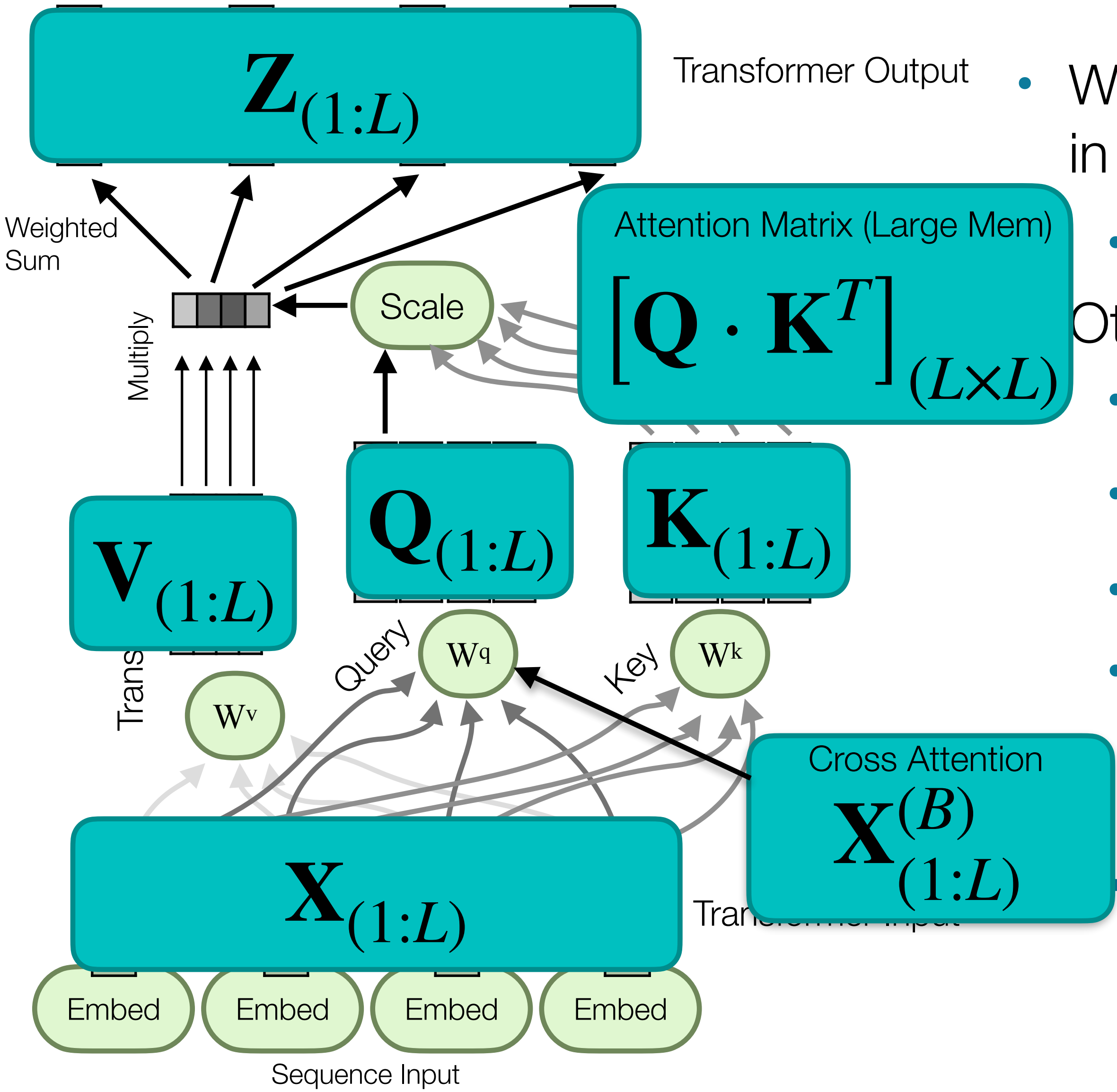
for each word

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$\mathrm{MultiHead}(Q,K,V) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)W^O$$
$$\text{where } \mathrm{head}_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

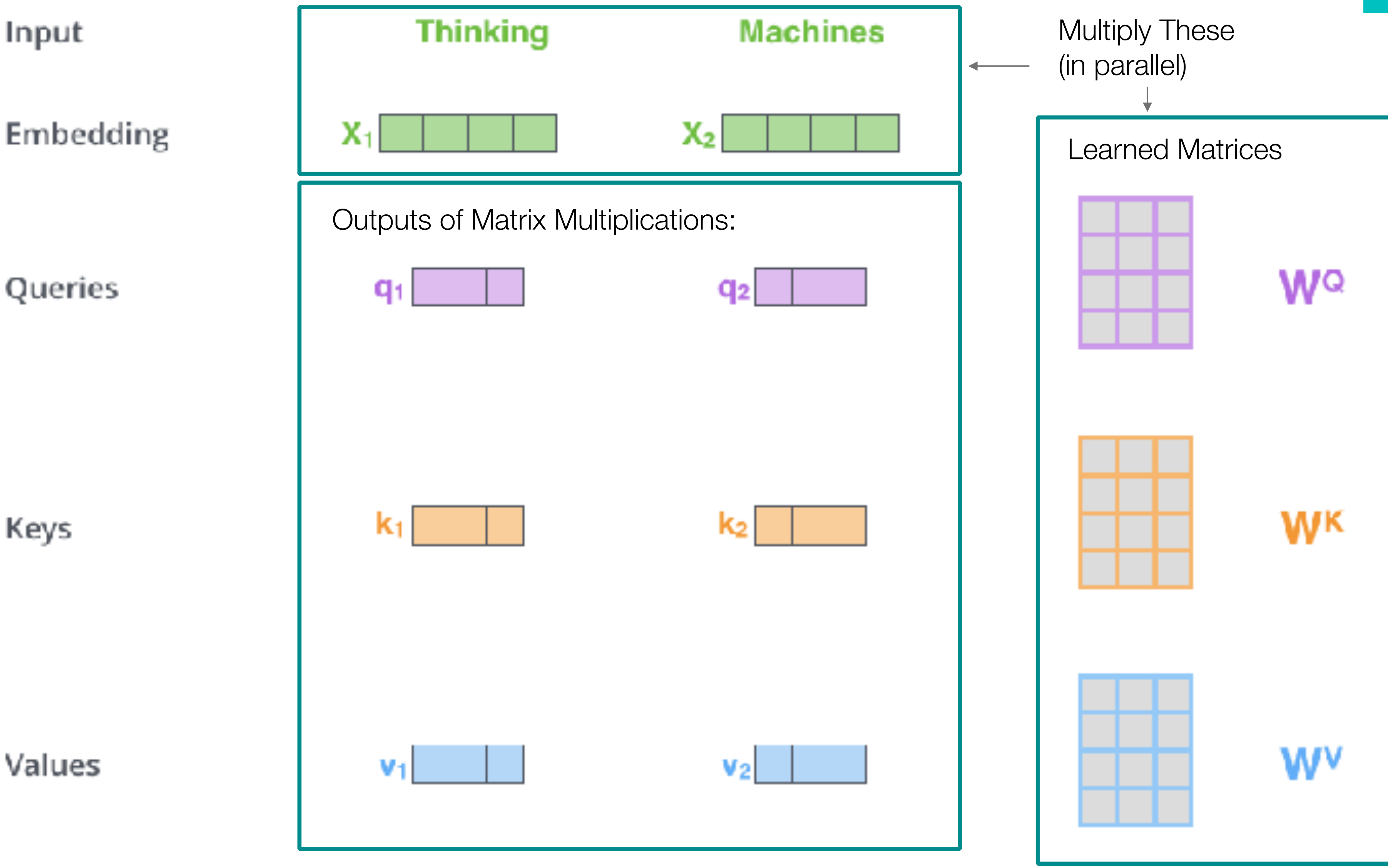
https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

Self Attention Overview



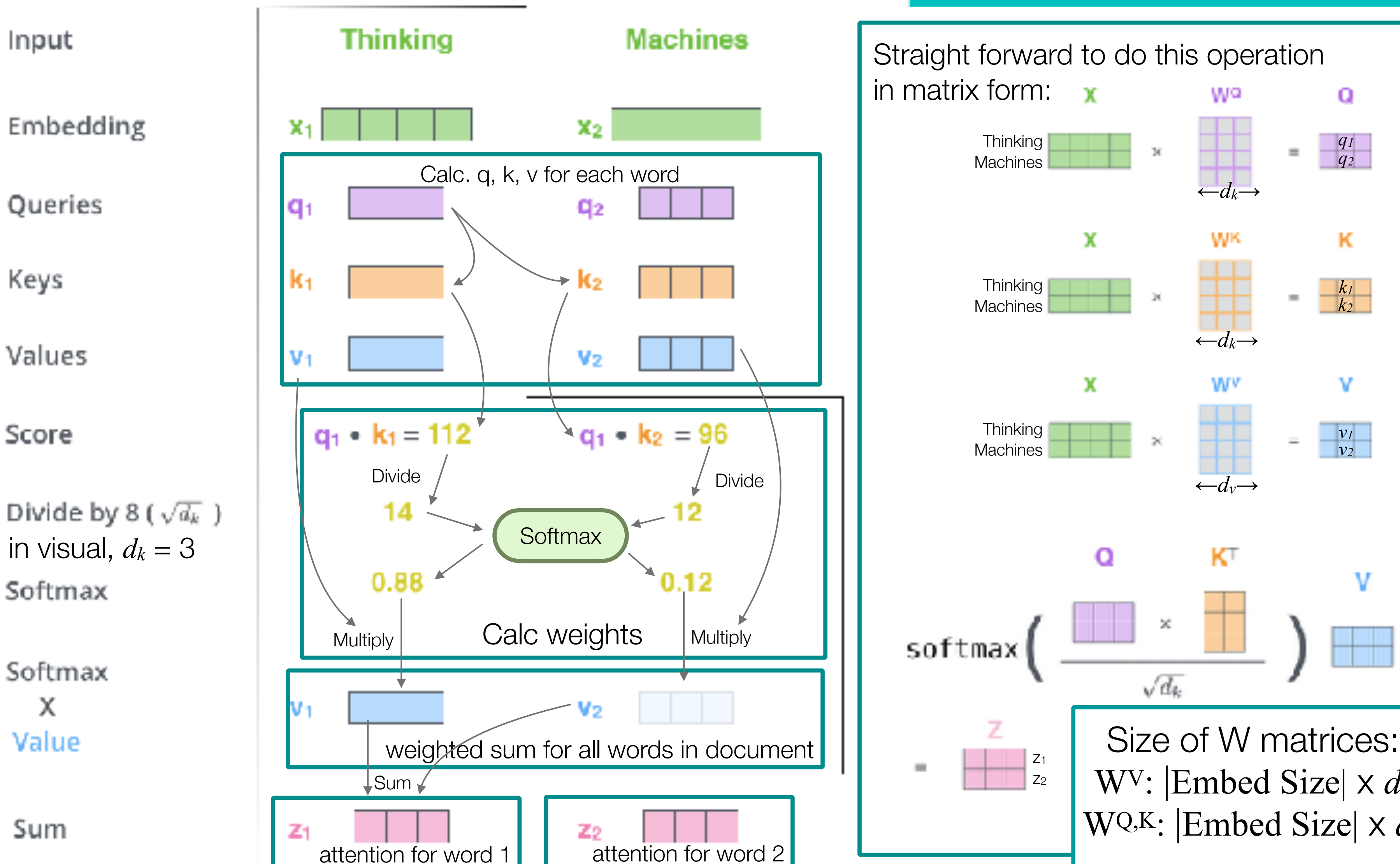
- What parameters are trained in diagram?
  - $\mathbf{W}^v, \mathbf{W}^q, \mathbf{W}^k$
- Other Parameters:
  - L: length of sequence
  - Query/Key dimension, d_k
  - Value dimension, d_v
  - How many times to apply attention (i.e., number of heads)
  - Type of positional encoding (more later)

Transformer: in more detail



Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Transformer: in more detail

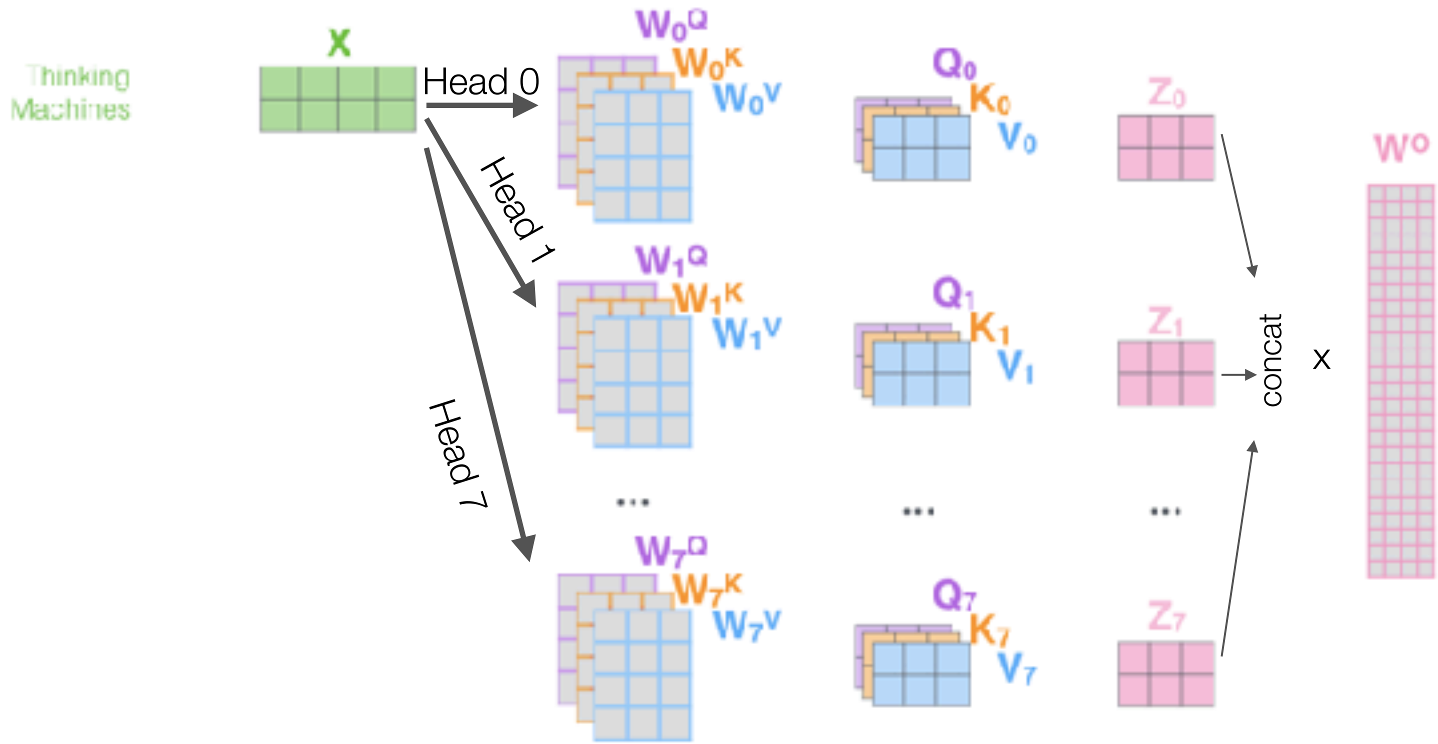


Size of W matrices:
W^V: |Embed Size| x d_v
$W^{Q,K}$: |Embed Size| x d_k

Size of Q,K,V:
|Seq Len| x d_v or d_k

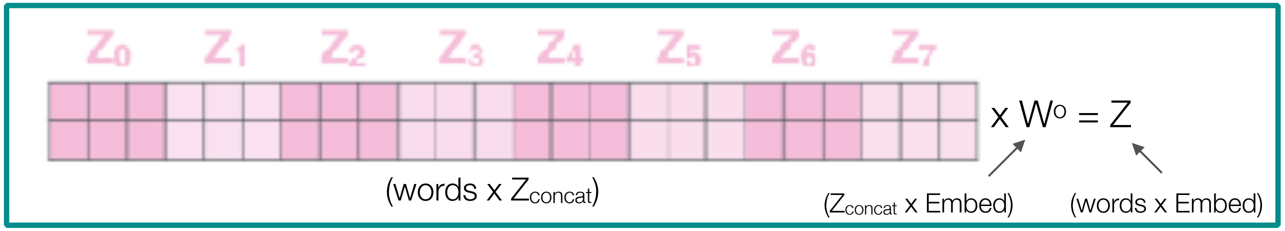
Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Transformer: Multi-headed Attention



one row for each word

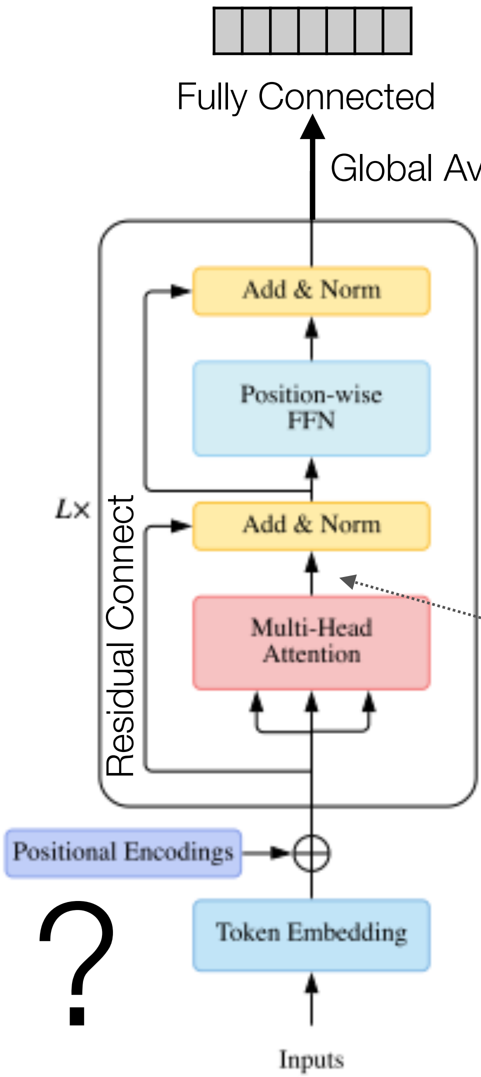
Num columns determined by W^O



Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Putting It Together

Prediction

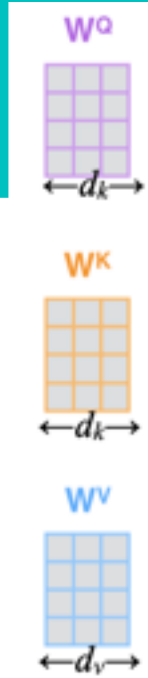


Layer Normalization

Fully Connected

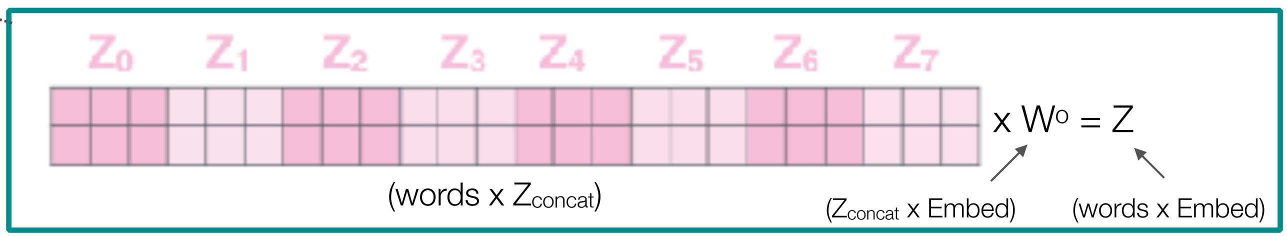
Layer Normalization

```
tf.keras.layers.MultiHeadAttention(
    num_heads,      (Number of heads Z1-Z7)
    key_dim,        (size of query/key dk)
    value_dim,      (size of each dv)
    output_shape,   (Embed size of Z)
    …
```



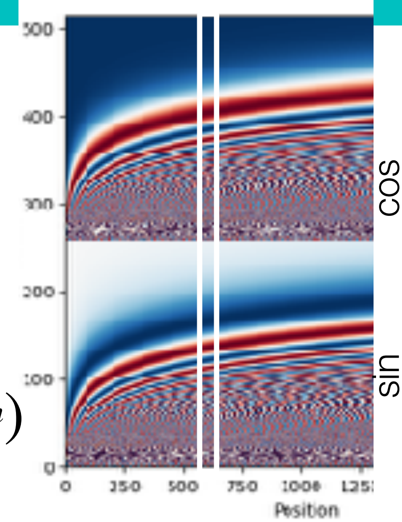
$$LN(Z_{row}) = \alpha \frac{Z_i - \mu_Z}{\sqrt{\sigma_Z^2 + \epsilon}} + \beta$$

Learn to normalize the rows of Z



Transformer: Positional Encoding

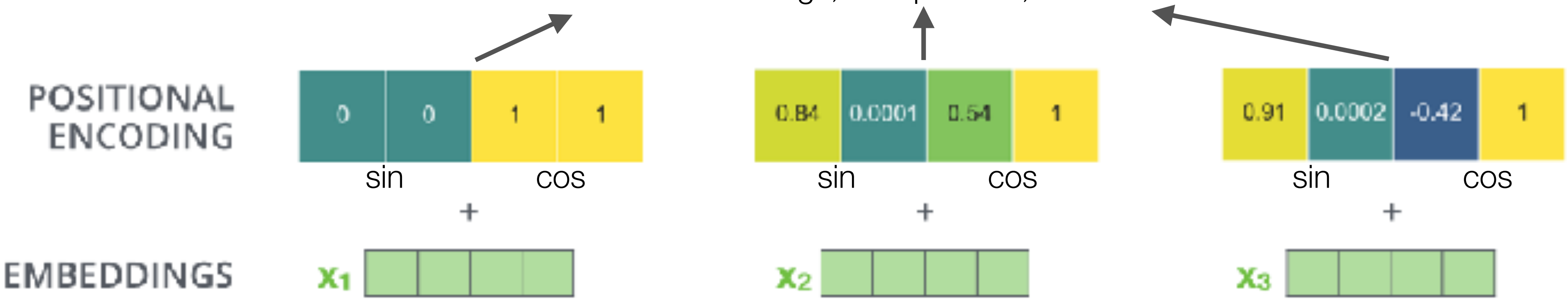
- Objective: add notion of position to embedding
- Attempt in paper: add sin/cos to embedding



p: in sequence
d_m: 1/2 dim of embed
i = index in vector

$$PE_{(p,i\in 0\ldots d_m-1)} = \sin(p/10000^{i/d_m})$$
$$PE_{(p,i\in d_m\ldots 2d_m)} = \cos(p/10000^{(i-d_m)/d_m})$$

Now use the new embeddings, with position, into transformer architecture

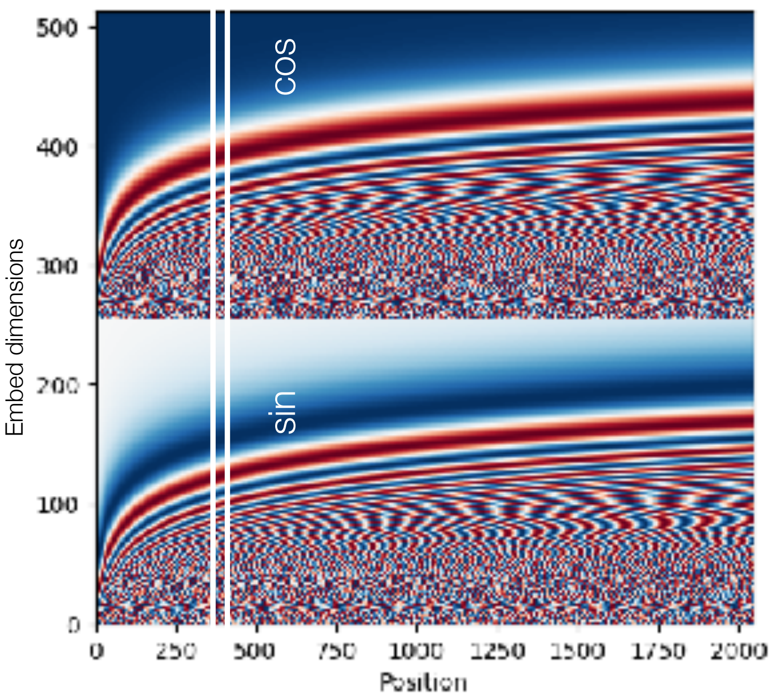


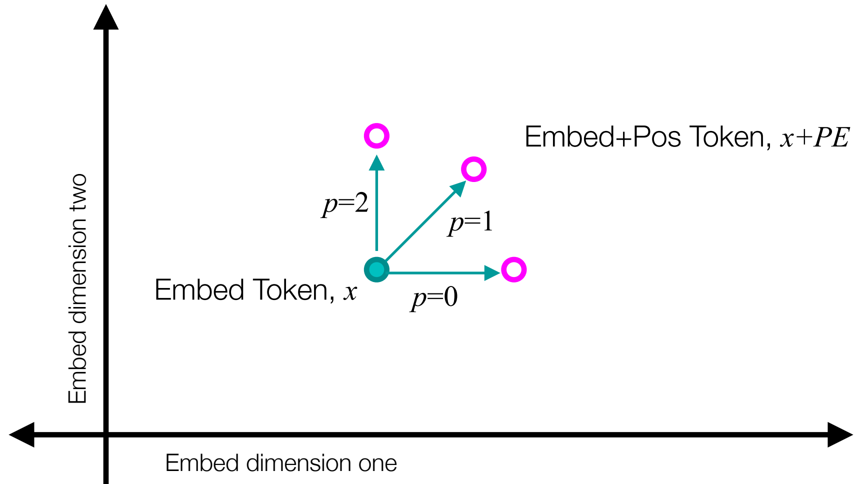
p=1, $i \rightarrow$ p=2, $i \rightarrow$ p=3, $i \rightarrow$

Hypothesis: Now the word proximity is encoded in the embedding matrix, with other pertinent information. Well, it does help… so it could be true that this is a good way to do it.

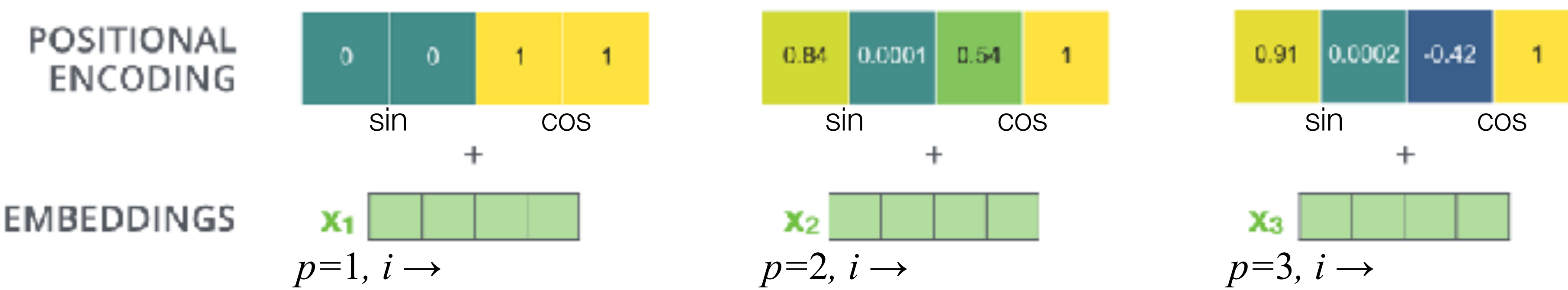
Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Positional Intuition, Geometrically



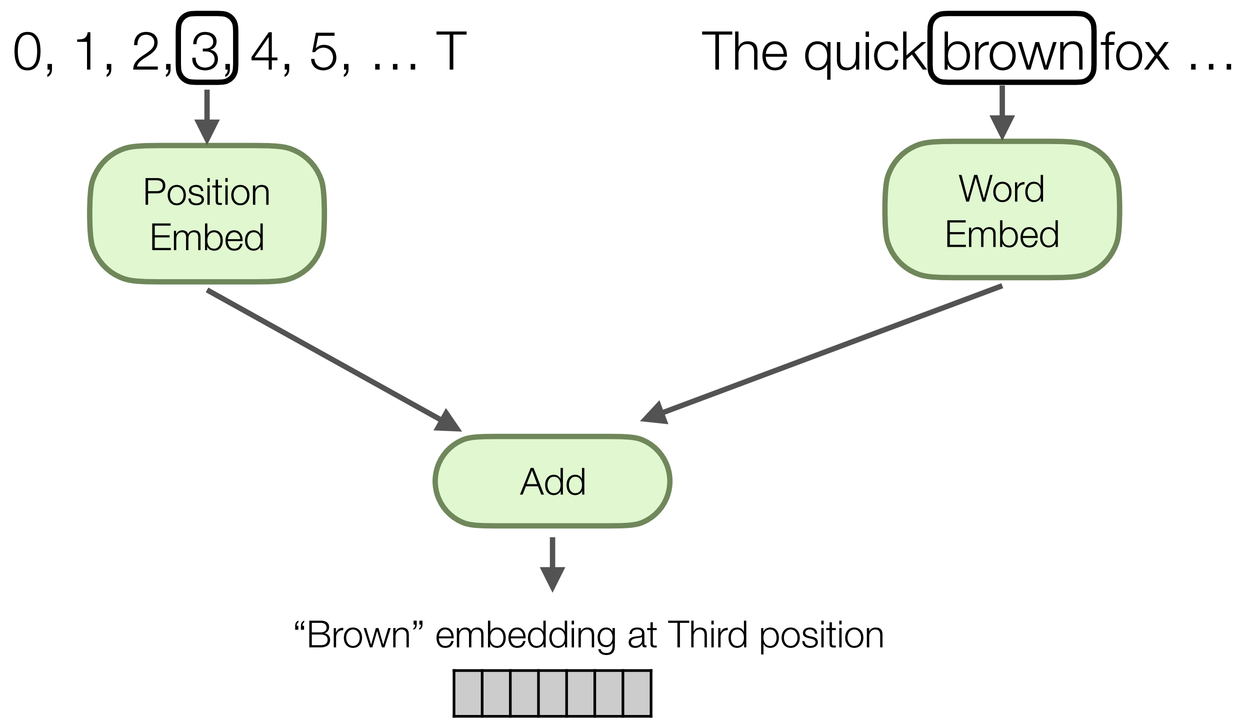


Assume that transformer model can learn the small perturbations in positional encoding, and how to use them in the context of the word.



Transformer: Positional Encoding

- Objective: add notion of position to embedding
- Attempt in original paper: add sin/cos to embedding
- **But could be anything that encodes position, like:**



Excellent Blog on Transformers: http://jalammar.github.io/illustrated-transformer/

Lecture Notes for

Neural Networks and Machine Learning

Transformers

Next Time:

SSL, Vision Transformers

Reading: None