MultiHead Attention

Attention Is All You Need

- Most competitive neural sequence transduction models have an encoder-decoder structure [1]
- Transformer : Solely based on Attention (No Recurrent / Convolutional connections)
- Recurrent model : Can't parallelize within an example
- Attention is almost always used with Recurrent Networks (before)
- Transformer : Relying entirely on Attention
- Self Attention / Intra Attention

[1] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," Sep. 2014.

Attention Is All You Need - Important topics

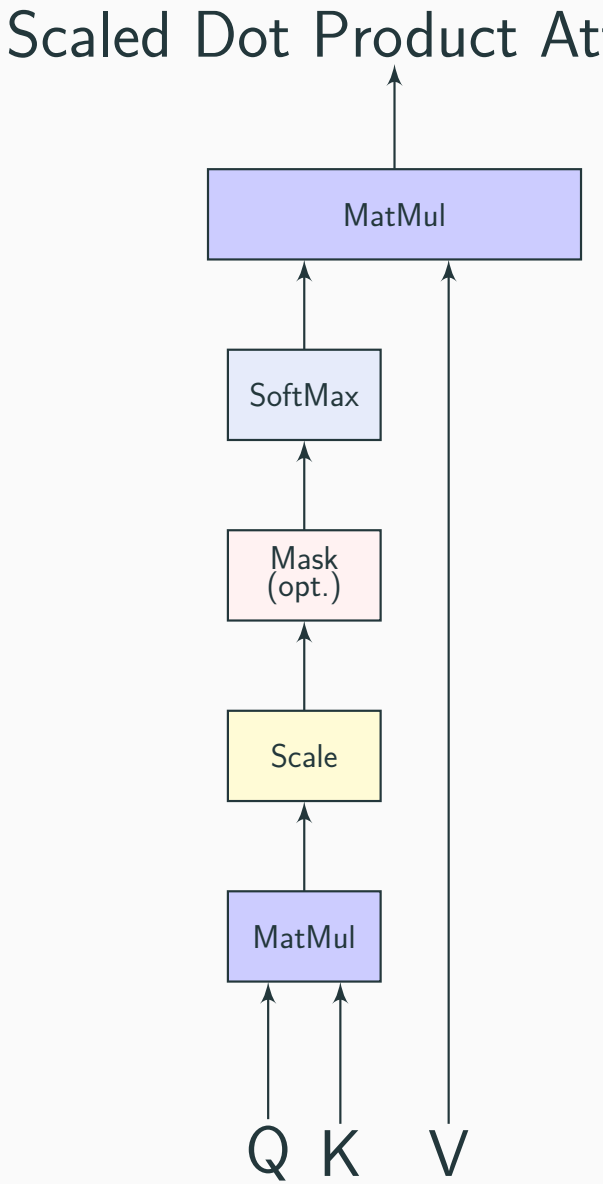
- Self Attention / Intra Attention
- Scaled Dot Product Attention
- Mutli-Head Attention
- Attention - K,V and Q
- Position wise FeedForward Networks
- Positional Encoding
- Residual Connections
- Learning rate scheduling

Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output.

- The query, keys, values, and output are all vectors
- The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.
- Query : $dec_z \in \mathcal{R}^{d_k}$
- Key : $f(h) \in \mathcal{R}^{d_k}$
- Value : $g(h) \in \mathcal{R}^{(}d_v)$
- $Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V$

Scaled Dot product



Scaled Dot Product

$$Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V$$

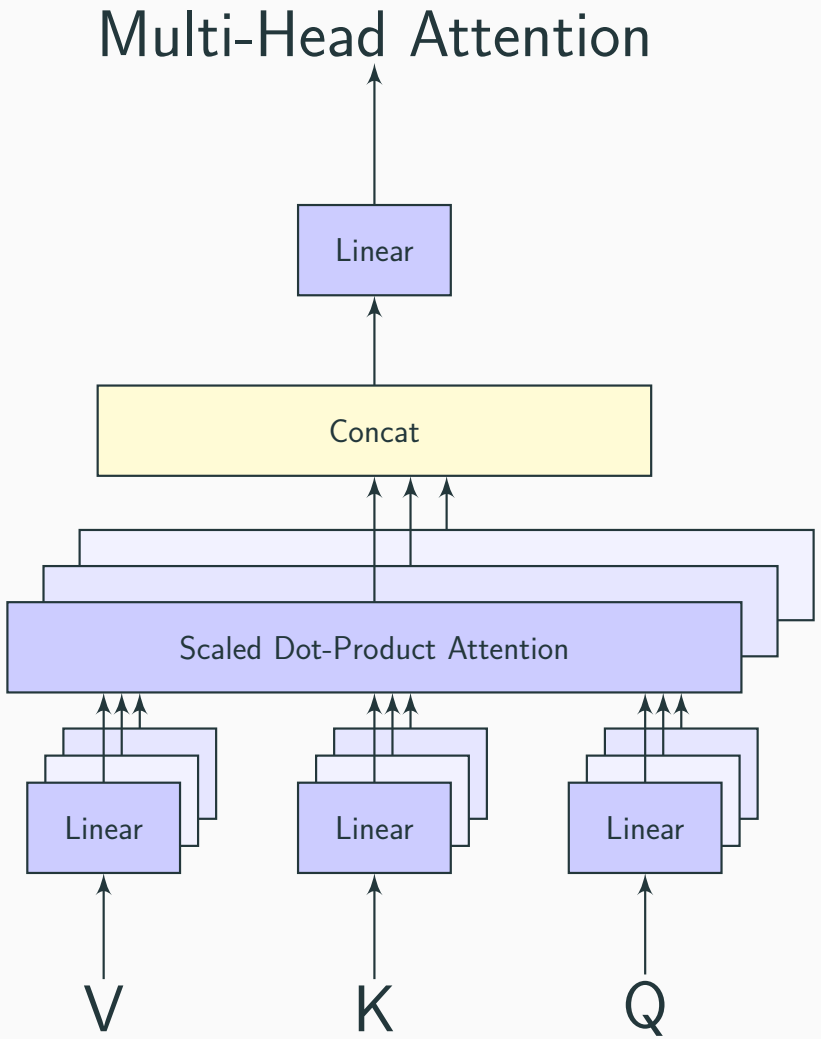
- compute the dot products of the query with all keys
- divide each by $\sqrt{d_k}$
- apply a softmax function to obtain the weights on the values

Dot Product Attention

Dot Product Attention is similar to this except for the scaling factor $1/\sqrt{d_k}$

- For large dk AttAdd $>$ MultiHeadDot [3]
- The dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients -> Scale the dot product

Multi-Head Attention



Multi Head Attention

Instead of a Single Attention with Q, K, V :

- Linearly Project Q,K,V to d_k, d_k, d_v dimensions - h times!
- Perform Attention on each of these Q,K,V in parallel to yield d_v dimensional values
- d_v^i where $i = 1 \ldots h$
- All are concatenated and projected to get the final value

Advantage over Single Head attention

This allows the model to jointly attend to information from different representation at different positions

Single Head Attention

Averaging inhibits this behavior

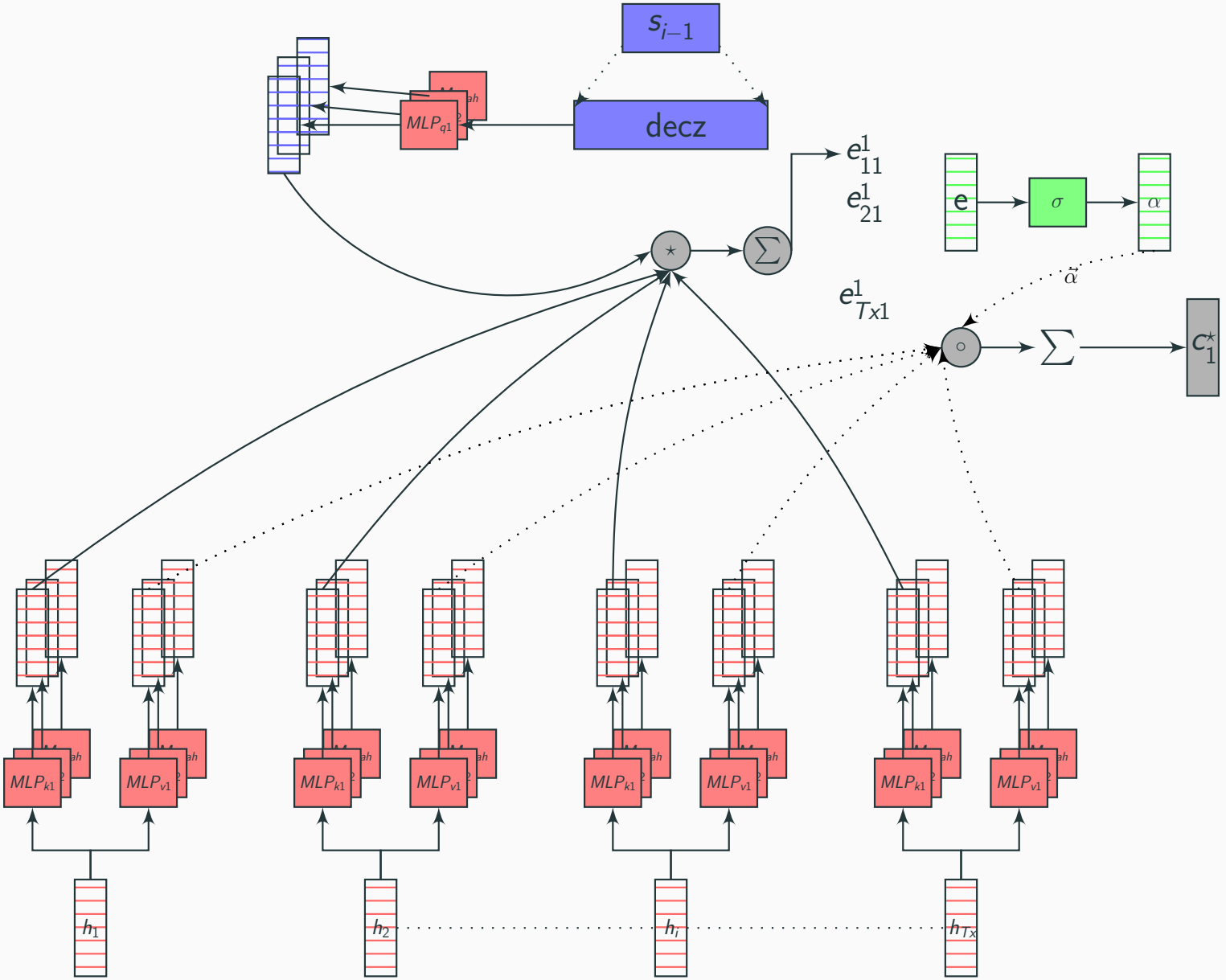
Projections of Q,K,V

- $W_i^Q \in \mathcal{R}^{d_{model} \times d_k}$
- $W_i^K \in \mathcal{R}^{d_{model} \times d_k}$
- $W_i^V \in \mathcal{R}^{d_{model} \times d_v}$
- $W_i^O \in \mathcal{R}^{hd_v \times d_{model}}$
- Here, h=8 parallel attention layers or heads
- $d_k = d_v = d_{model}/h = 64$

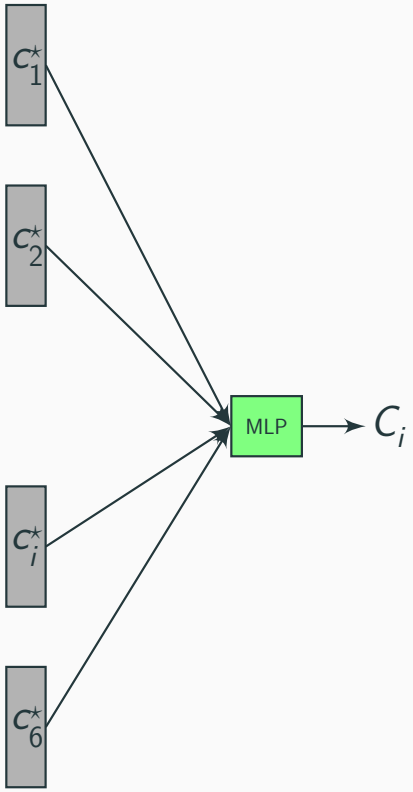
Computational Cost

Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

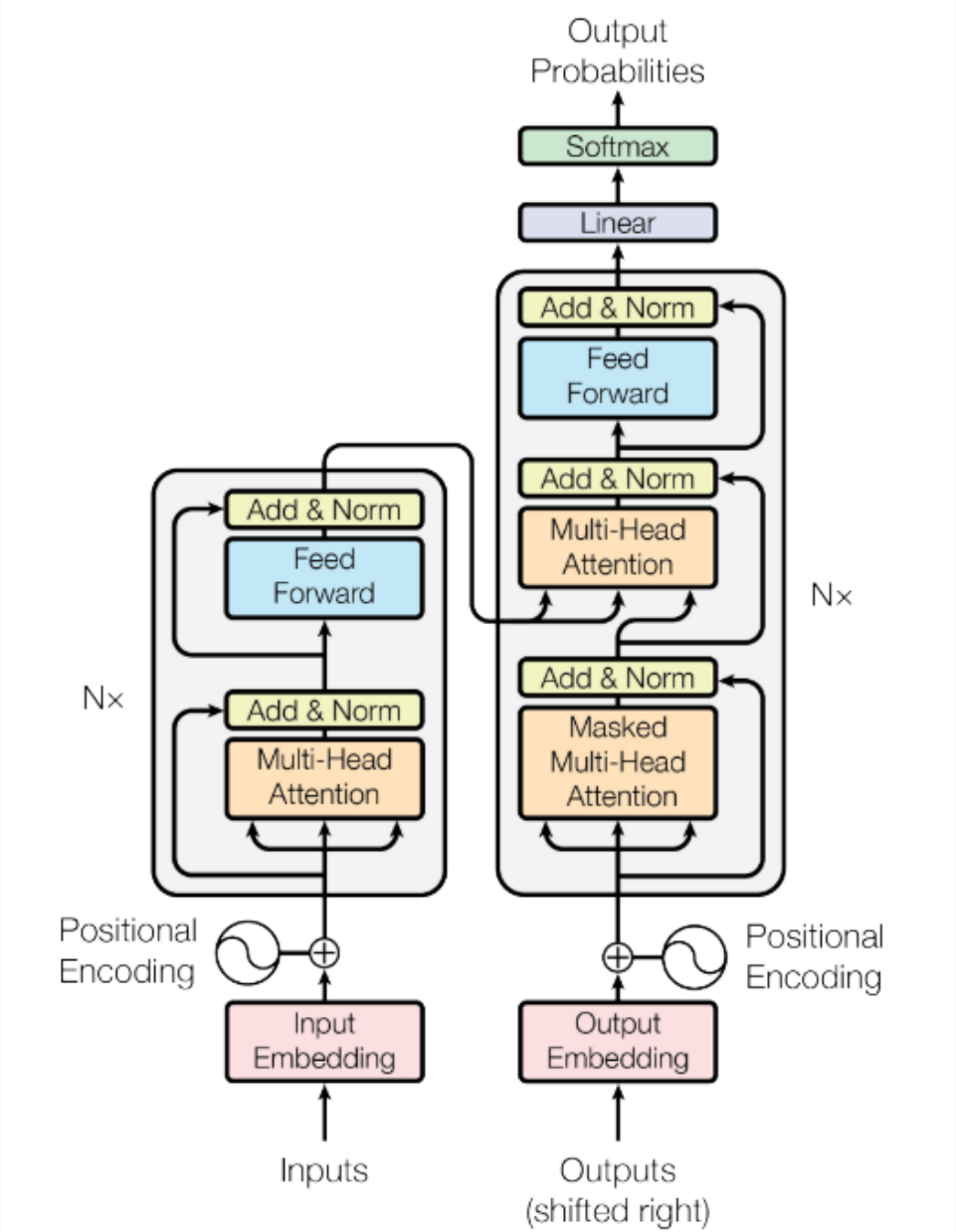
MultiHead Dot Product Attention - Full picture



MultiHead Dot Product Attention - Full picture



The Transformer - model architecture



The Transformer

- In "encoder-decoder attention" layers, the queries come from the previous decoder layer, and the memory keys and values come from the output of the encoder.
- Allows every position in the decoder to attend over all positions in the input sequence.
- Encoder has self-attention layers - Q,K,V come from same place. : Output of previous layer in the Encoder
- Decoder also has self-attention layers - allows each position in the decoder to attend to all positions in the decoder up to and including that position.
- Preserver auto regressive [2] property - mask out (by setting to $-\infty$) to all input values of Softmax corresponding to illegal positions.

[2] A. Graves, "Generating Sequences With Recurrent Neural

Position-wise Feed-Forward Networks

In addition to attention sub-layers, each of the layers in our encoder and decoder contains a fully connected feed-forward network - applied to each position separately and identically.

$$FFN(x) = max(0, xW_1 + b_1)W_2 + b_2$$

Weights are shared across positions, not across layers. [4]

[4] O. Press and L. Wolf, "Using the Output Embedding to Improve Language Models," Aug. 2016.

Learning rate scheduling

Adam optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$

Varied learning rate over training time according to:

$$lrate = d_{model}^{-0.5} \times min(step\_num^{-0.5}, step\_num \times warmup\_steps^{-1.5})$$

Positional Encoding

- Transformer contains no Recurrence or Convolution
- We have to inject some information about the relative or absolute position
- Added at the bottom of both encoder and decoder
- same dimension (d_{model}) as the embeddings, so they can be summed
- Many choices of positional encoding - learned or fixed (ConvS2S [5])

$$PE_{(pos,2_i)} = sin(pos/1000^{2i/d_{model}})$$

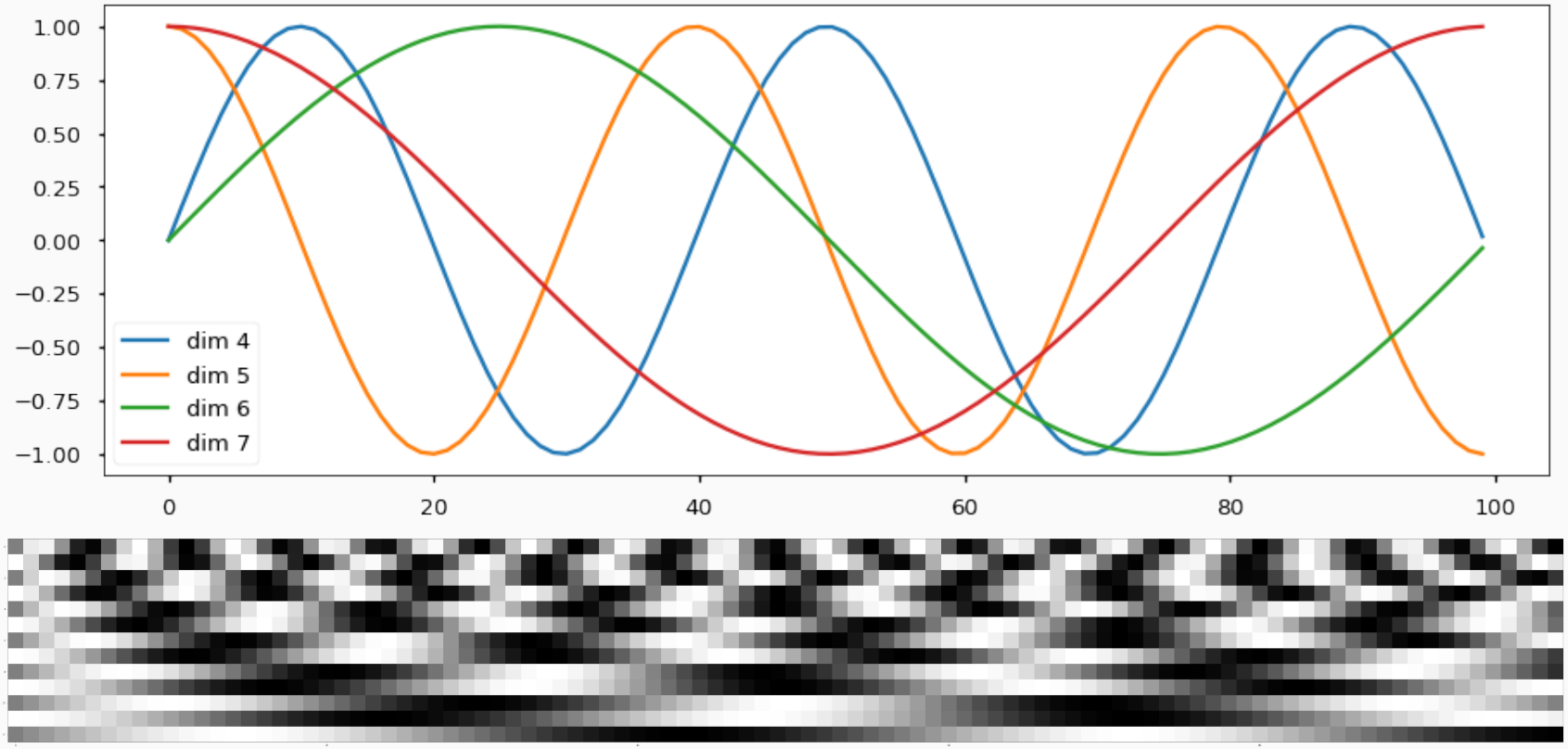
$$PE_{(pos,2_{i+1})} = cos(pos/1000^{2i/d_{model}})$$

[5] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional Sequence to Sequence Learning," May 2017.

Positional Encoding

- Each dimension of the positional encoding corresponds to a sinusoid
- The wavelengths form a geometric progression from 2π to 10000.2π
- Hypothesis of the authors : allows the model to easily learn to attend by relative positions
- for any fixed offset k PE_{pos+k} can be represented as a linear function of PE_{pos}

Positional Encoding - Example



Other variants

- MultiHead Additive Attention - Replace dot product by sum and an MLP
- MultiHead locatiob aware attebtuib - Consider attention weights from previous positions and use a CNN (like Location aware Attention we discussed)
- MultiHead Multi Resolution Attention - Use different filter size for each head!

Thank you for your Attention.

Questions?