The Transformer

Attention Based Sequence Transduction

Andreas Köpf, andreas.koepf@provisio.com

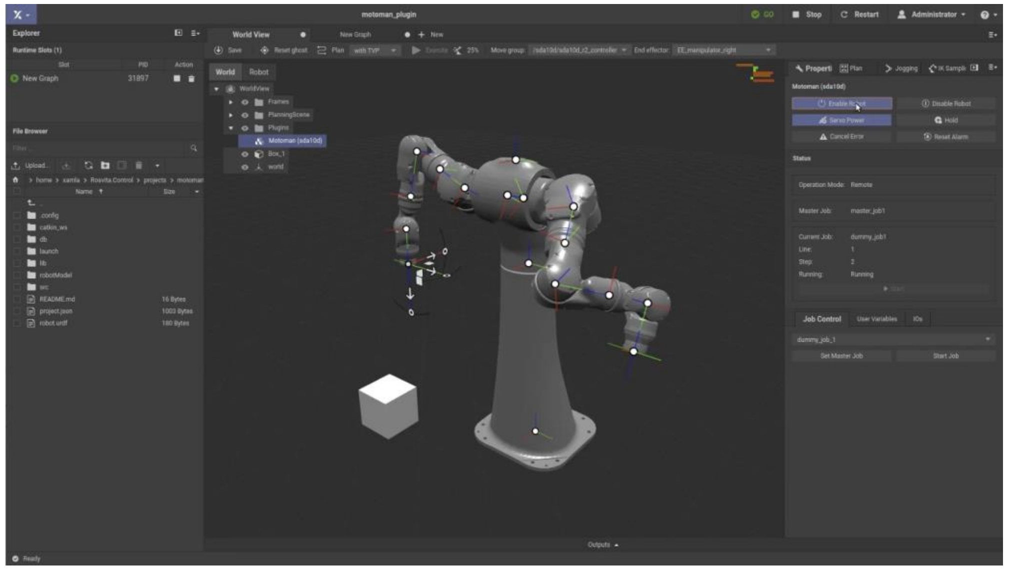
Sep 2019

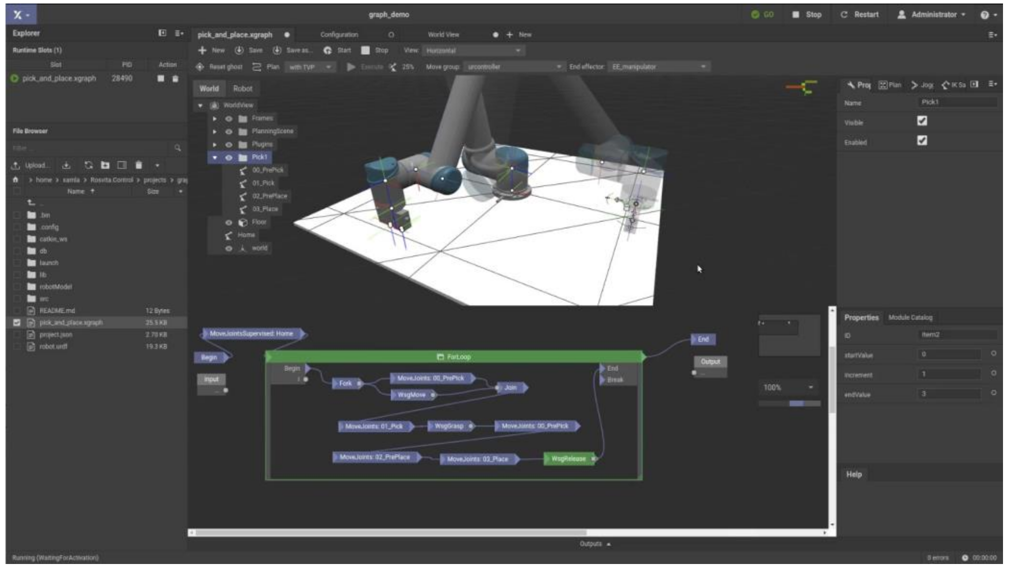
About Me

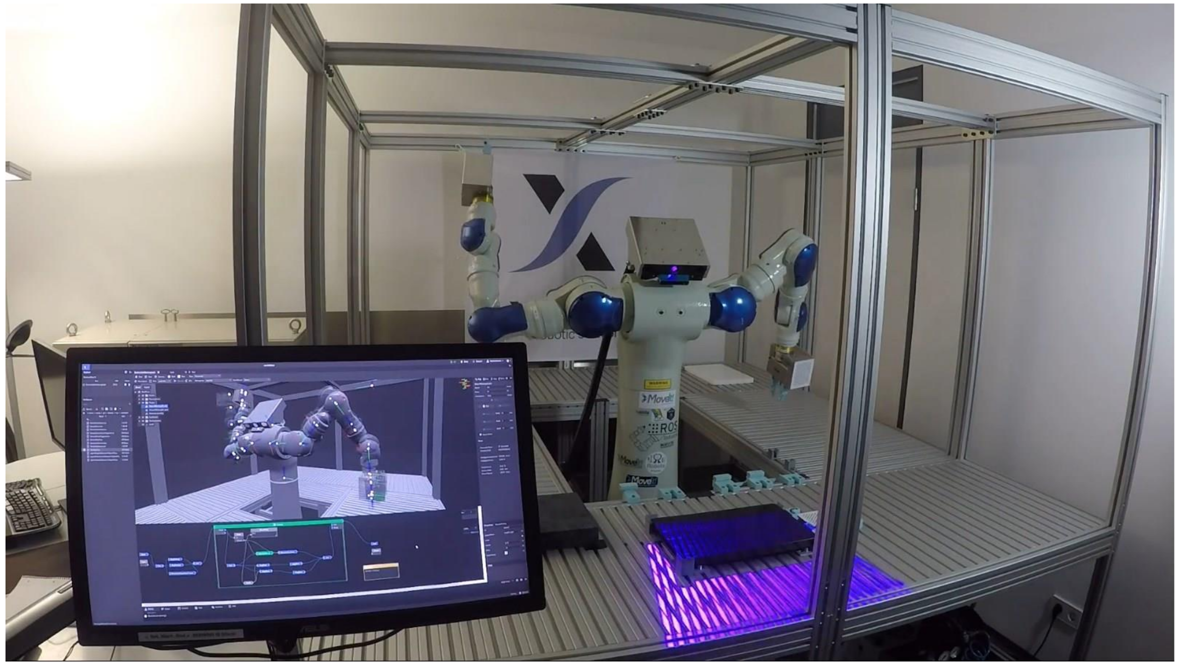
- Andreas Köpf, software developer, age 39
- 10+ years: hands-on experience in AI/ML, neural networks

- My company: **PROVISIO GmbH**, creator of self-service and digital signage software **SiteKiosk**, offices in Münster & Miami, USA.
- Last 4 years: Team leader at **Xamla** – adaptive robotics, developed **Rosvita** Robot Programming IDE & industrial robotics projects
- Currently exploring: Video prediction, model based RL

Xamla & Rosvita







The Transformer

- A sequence-to-sequence neural network architecture
- Paper „Attention is all you need“, by Google 2017

- **Simple:** Attention, feed-forward & normalization, no recurrent structure
- **Fast:** Whole sequence processed at once -> less training time
- **Powerful:** Improved translation quality over previous models

- Basis for newer pre-trained models: BERT & GPT-2

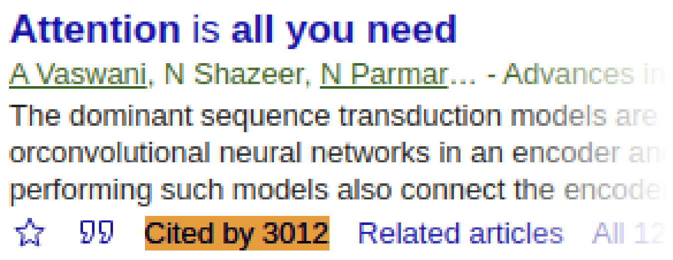
Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com
Noam Shazeer* Google Brain noam@google.com
Niki Parmar* Google Research nikip@google.com
Jakob Uszkoreit* Google Research usz@google.com
Llion Jones* Google Research llion@google.com
Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu
Łukasz Kaiser* Google Brain lukaszkaiser@google.com
Illia Polosukhin* ‡ illia.polosukhin@gmail.com

Abstract

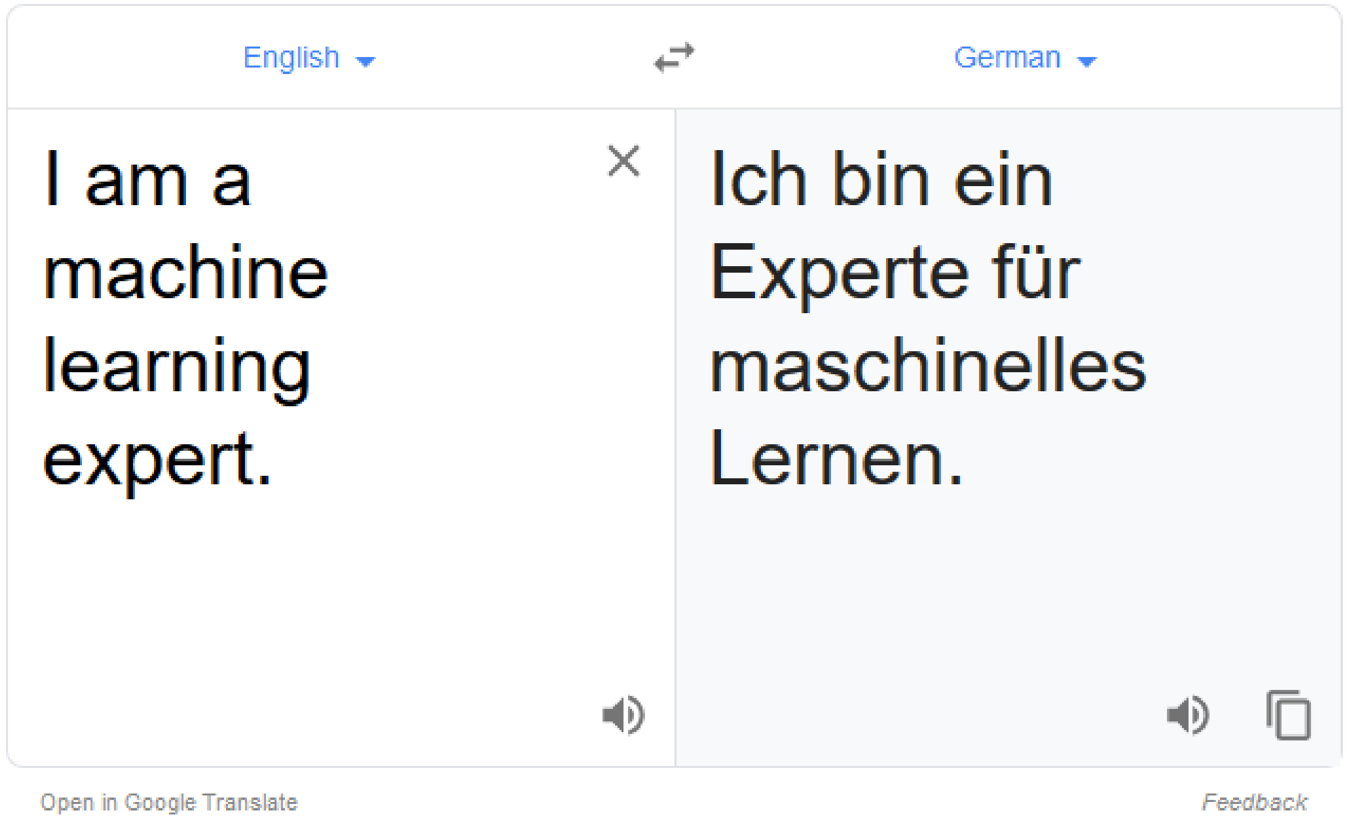
The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.8 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

https://arxiv.org/abs/1706.03762



Attention is all you need
A Vaswani, N Shazeer, N Parmar... - Advances i
The dominant sequence transduction models are
orconvolutional neural networks in an encoder an
performing such models also connect the encode
☆ 99 Cited by 3012 Related articles All 12

Google Scholar, 2019-09-06

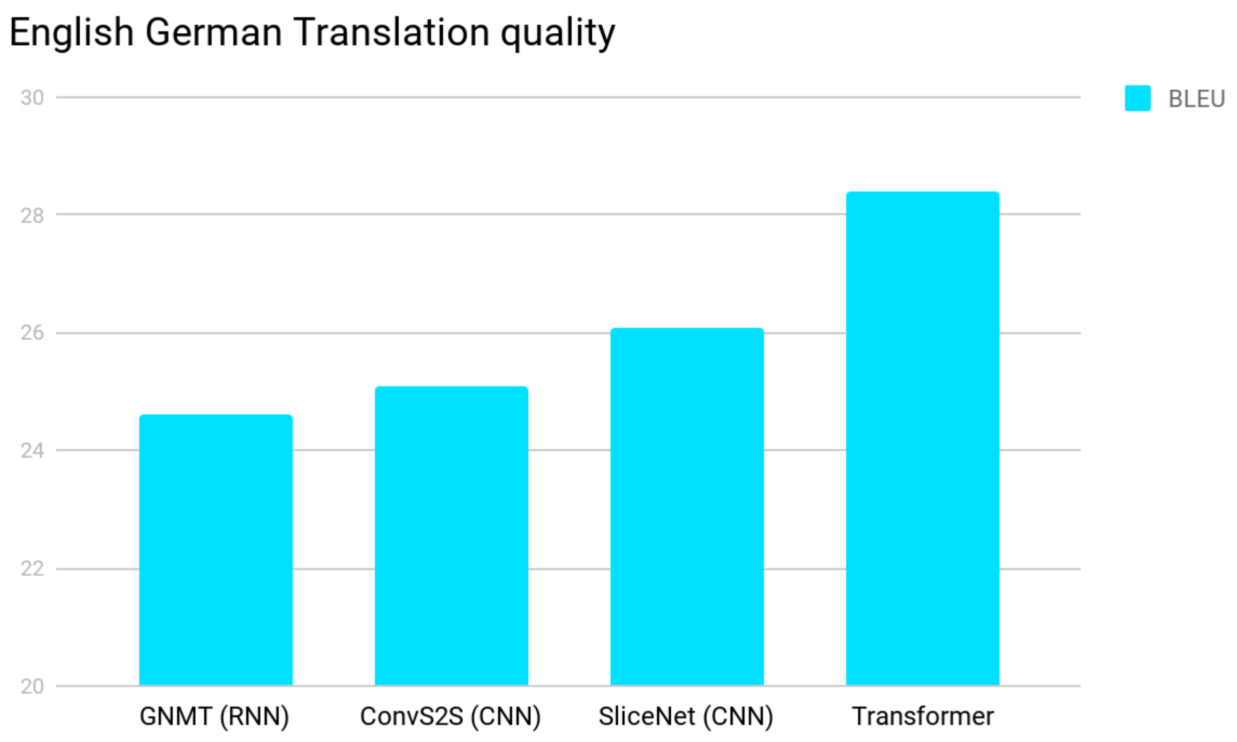

English German

I am a machine learning expert.

Ich bin ein Experte für maschinelles Lernen.

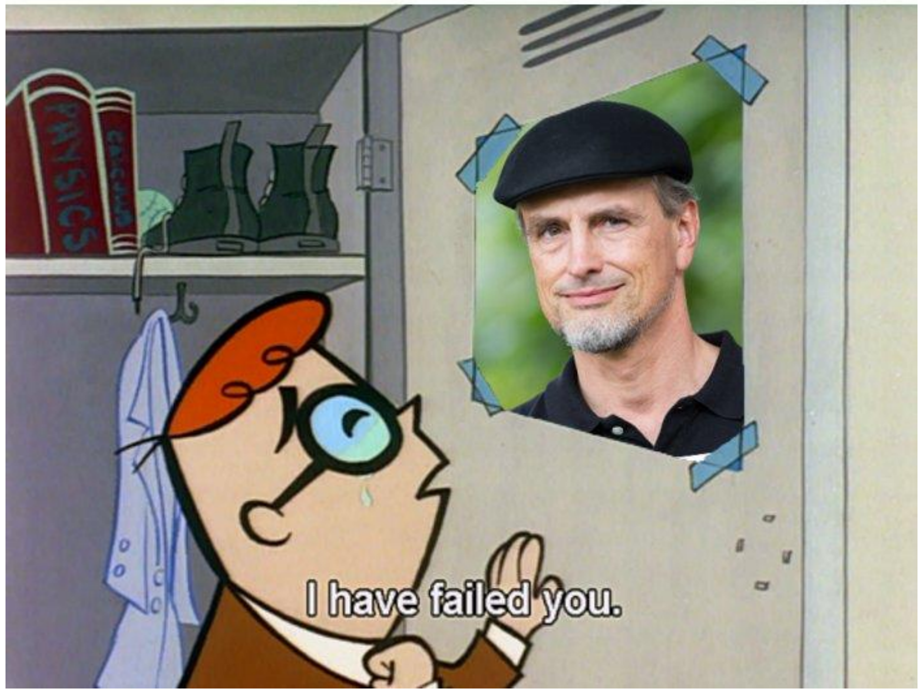
Open in Google Translate Feedback

Transformer vs LSTM



[source]

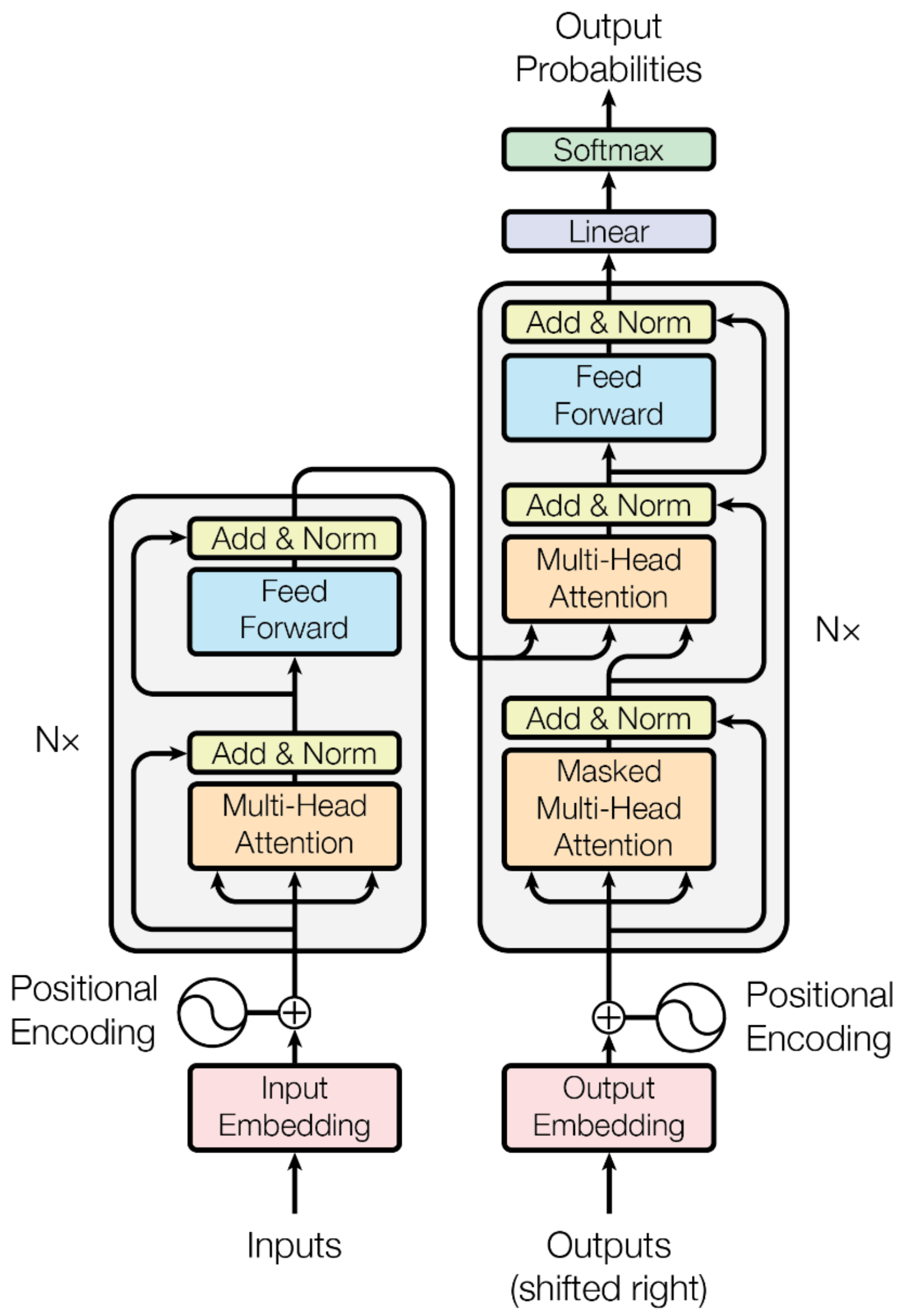
When I use a transformer instead of an LSTM



Baseline: GNMT (2016) “LSTM network with 8 encoder and 8 decoder layers using attention and residual connections”
[source]

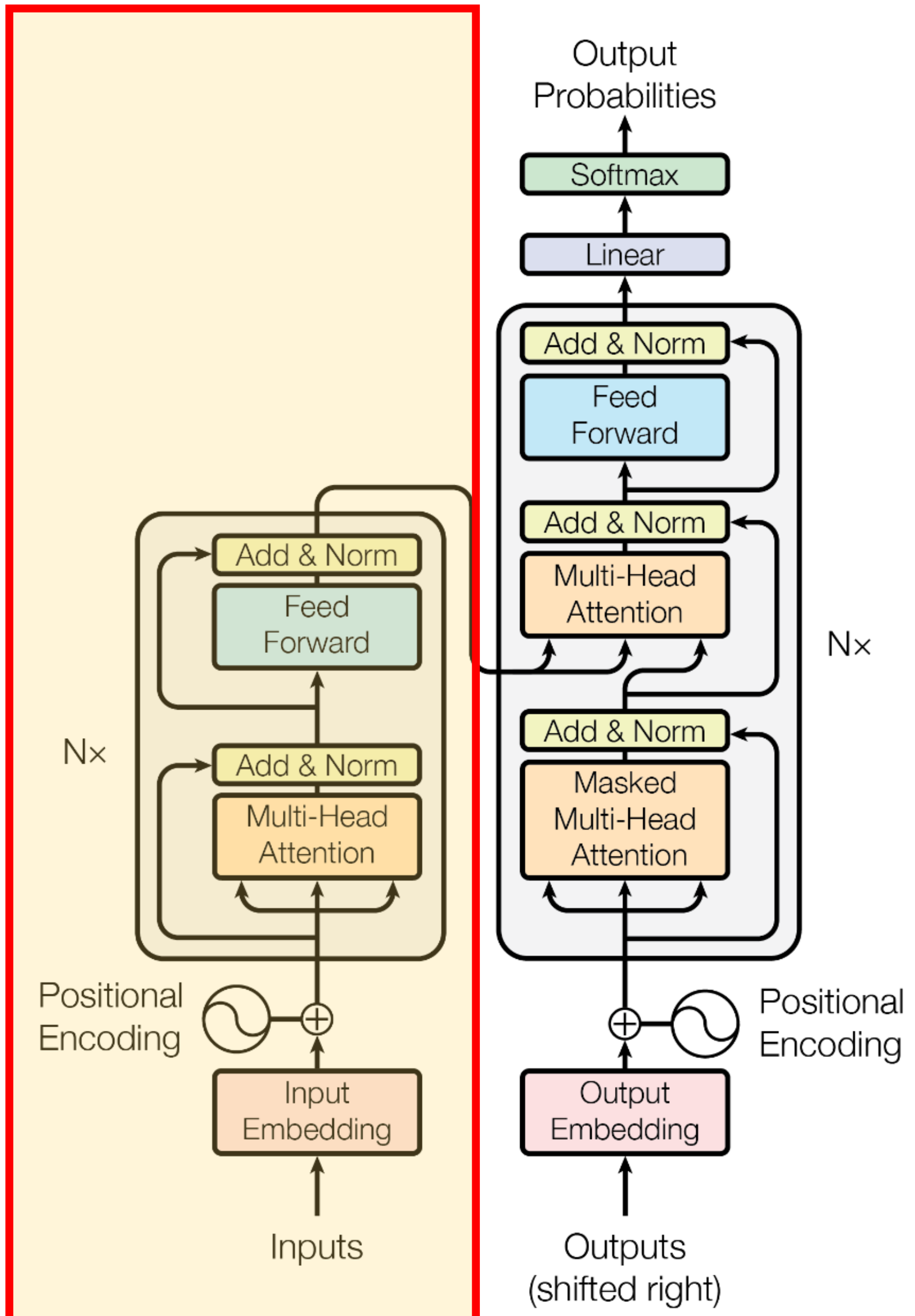
Key Components

- Encoder & Decoder
- Input Embedding
- Next Token Classifier
- Auto-Regressive Decoder
- Encoder-Decoder Attention
- Self-Attention
- Feed-Forward Layer
- Residual & Normalization
- Positional Encoding



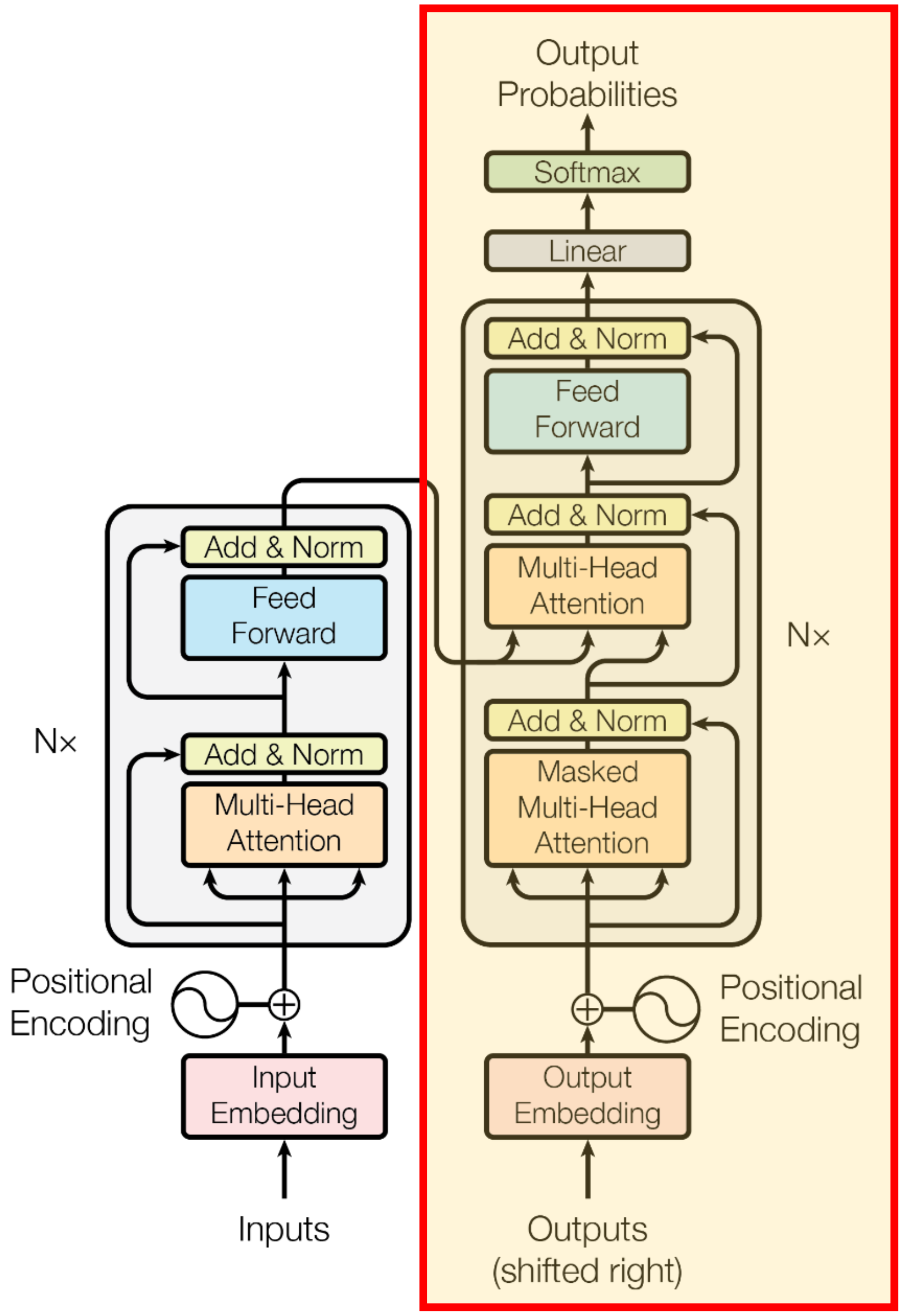
Key Components

- **Encoder** & Decoder
- Input Embedding
- Next Token Classifier
- Auto-Regressive Decoder
- Encoder-Decoder Attention
- Self-Attention
- Feed-Forward Layer
- Residual & Normalization
- Positional Encoding

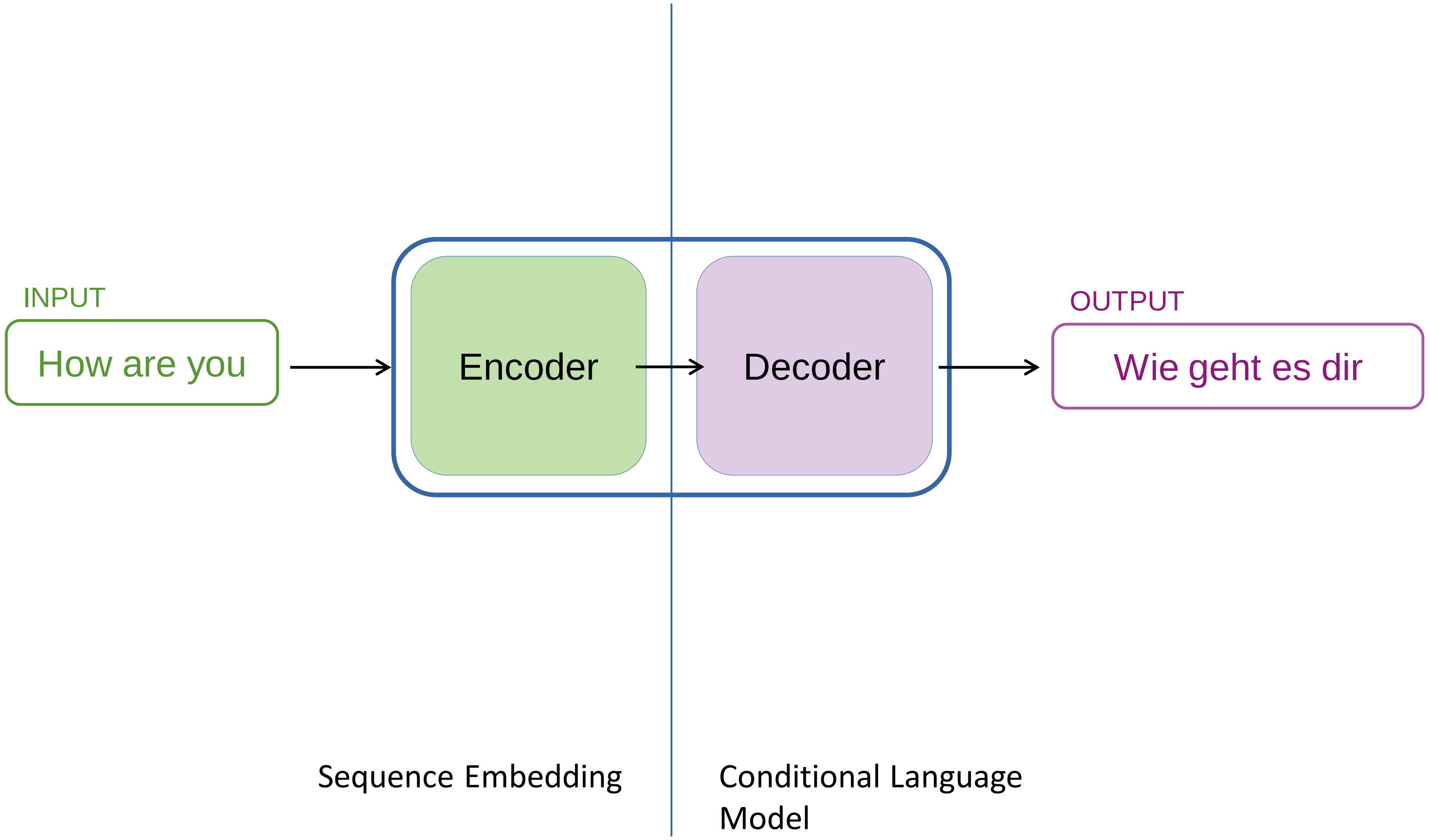


Key Components

- Encoder & **Decoder**
- Input Embedding
- Next Token Classifier
- Auto-Regressive Decoder
- Encoder-Decoder Attention
- Self-Attention
- Feed-Forward Layer
- Residual & Normalization
- Positional Encoding



Encoder & Decoder

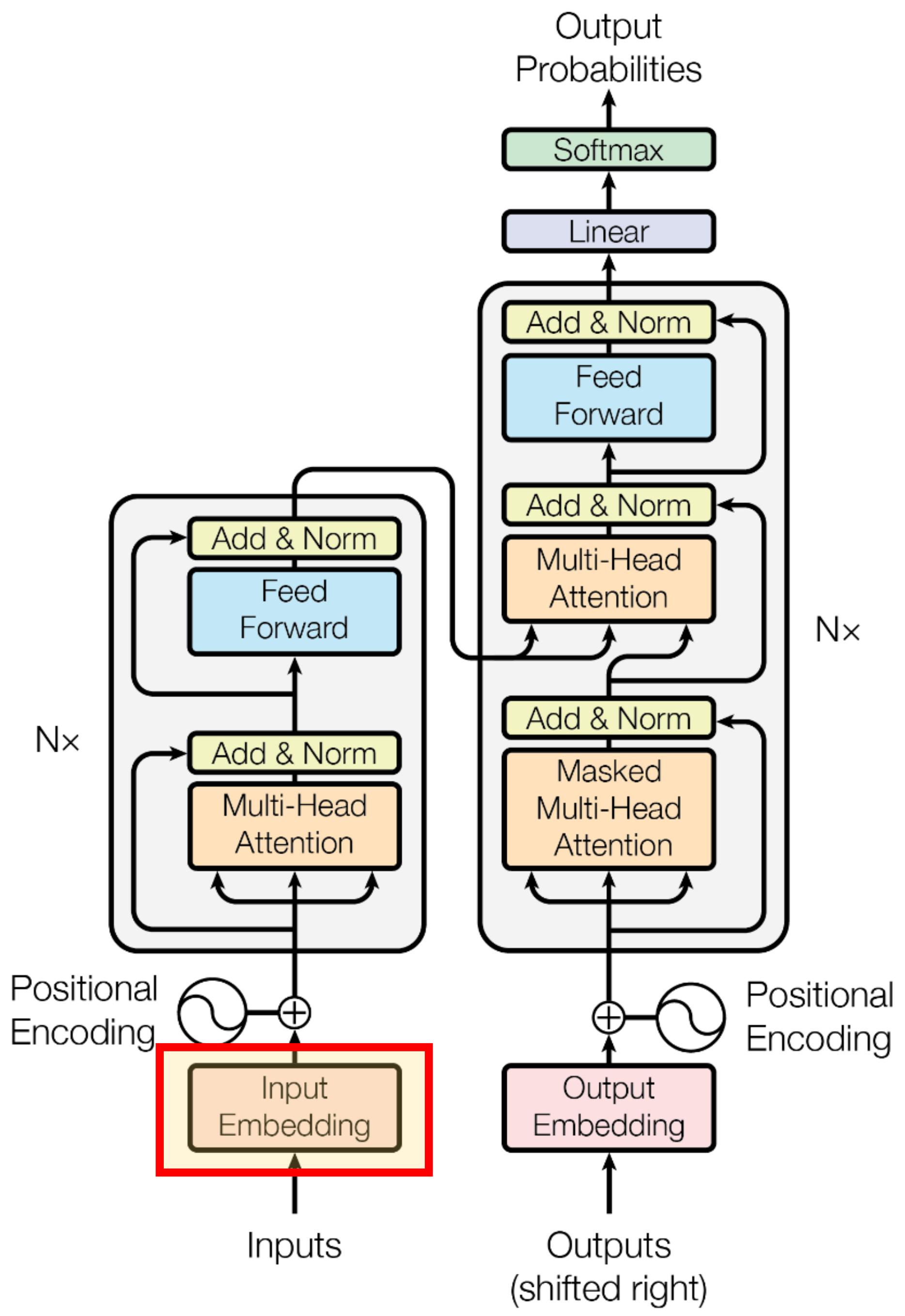


Sequence Embedding

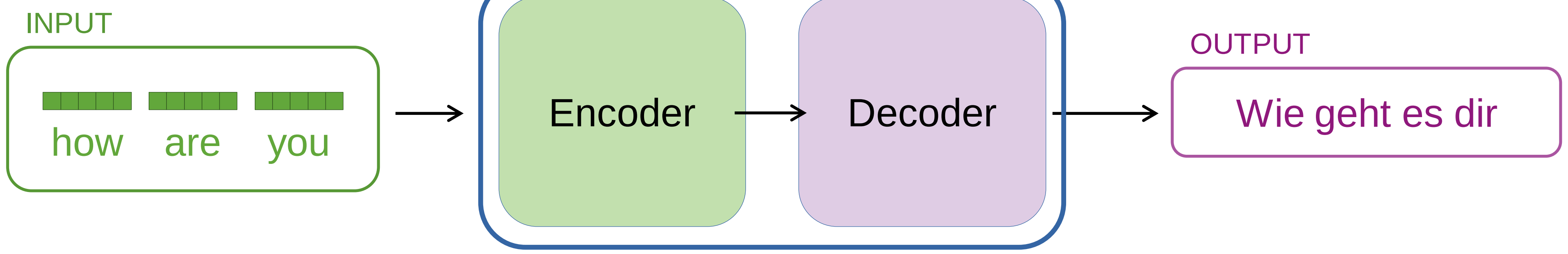
Conditional Language Model

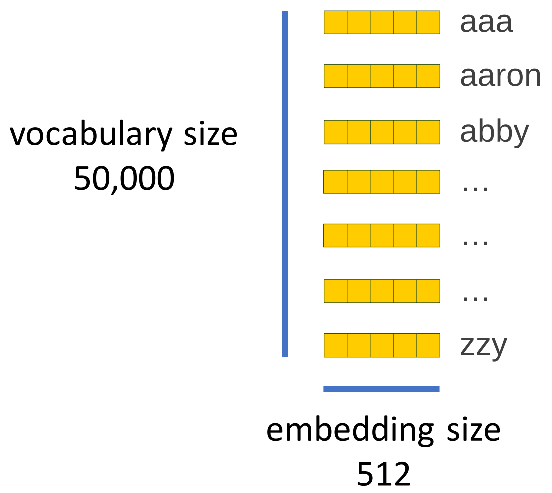
Key Components

- Encoder & Decoder
- **Input Embedding**
- Next Token Classifier
- Auto-Regressive Decoder
- Encoder-Decoder Attention
- Self-Attention
- Feed-Forward Layer
- Residual & Normalization
- Positional Encoding



Input Embedding

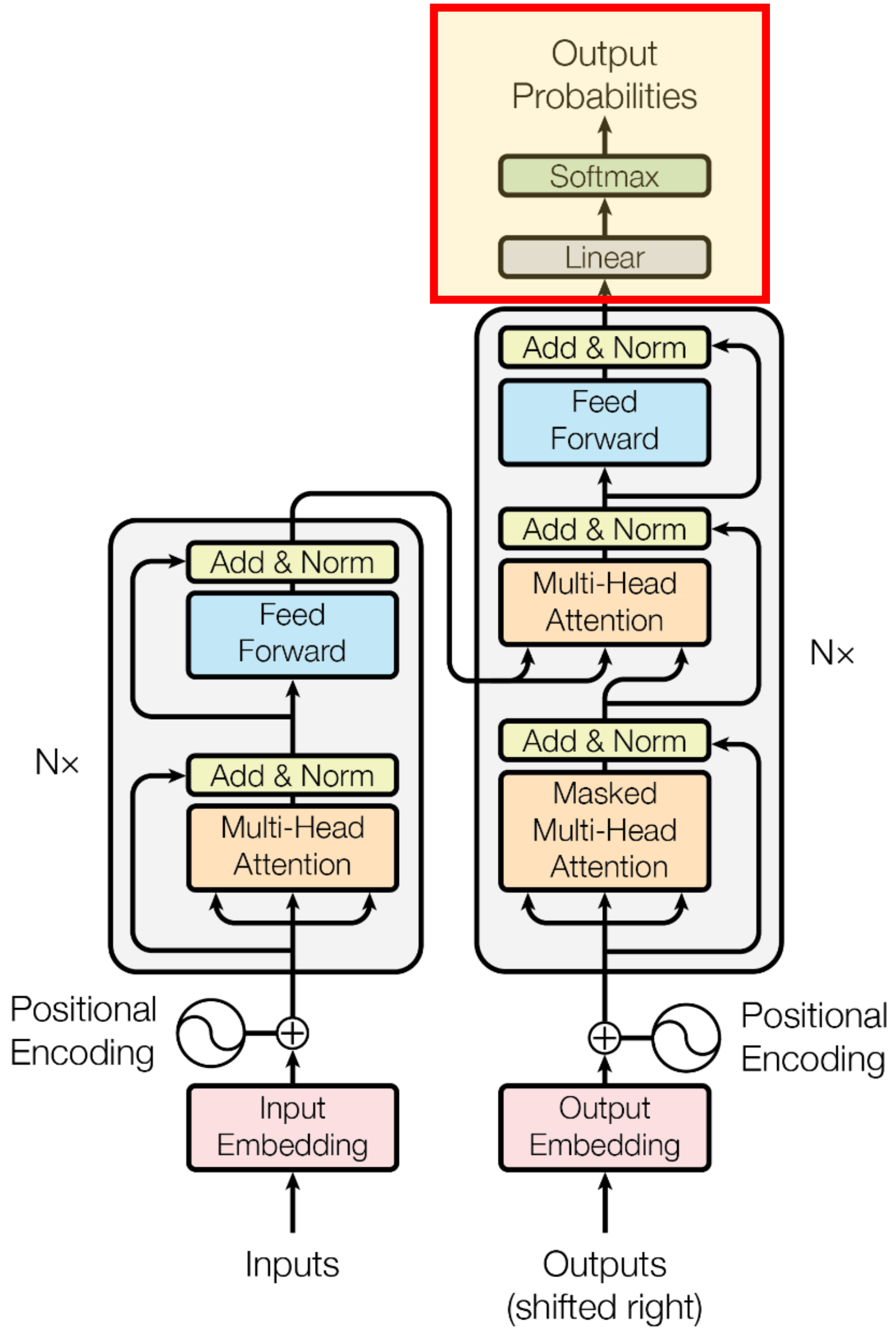




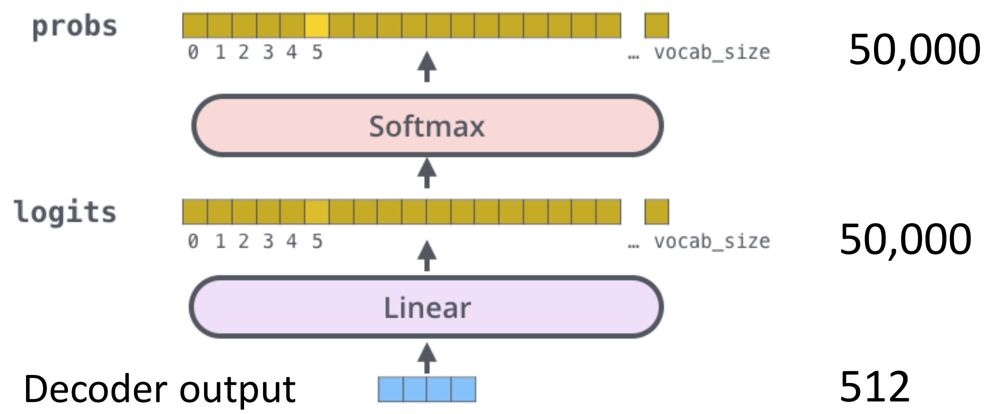
- Split text into tokens, e.g. words, sub-words (BPE)
- Convert tokens into vectors using a lookup-table
- Learn representation during training

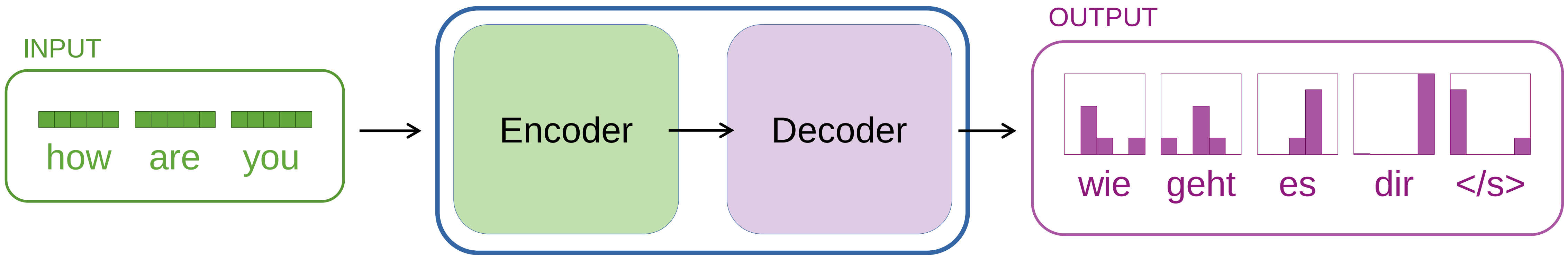
Key Components

- Encoder & Decoder
- Input Embedding
- **Next Token Classifier**
- Auto-Regressive Decoder
- Encoder-Decoder Attention
- Self-Attention
- Feed-Forward Layer
- Residual & Normalization
- Positional Encoding



Next Token Classifier

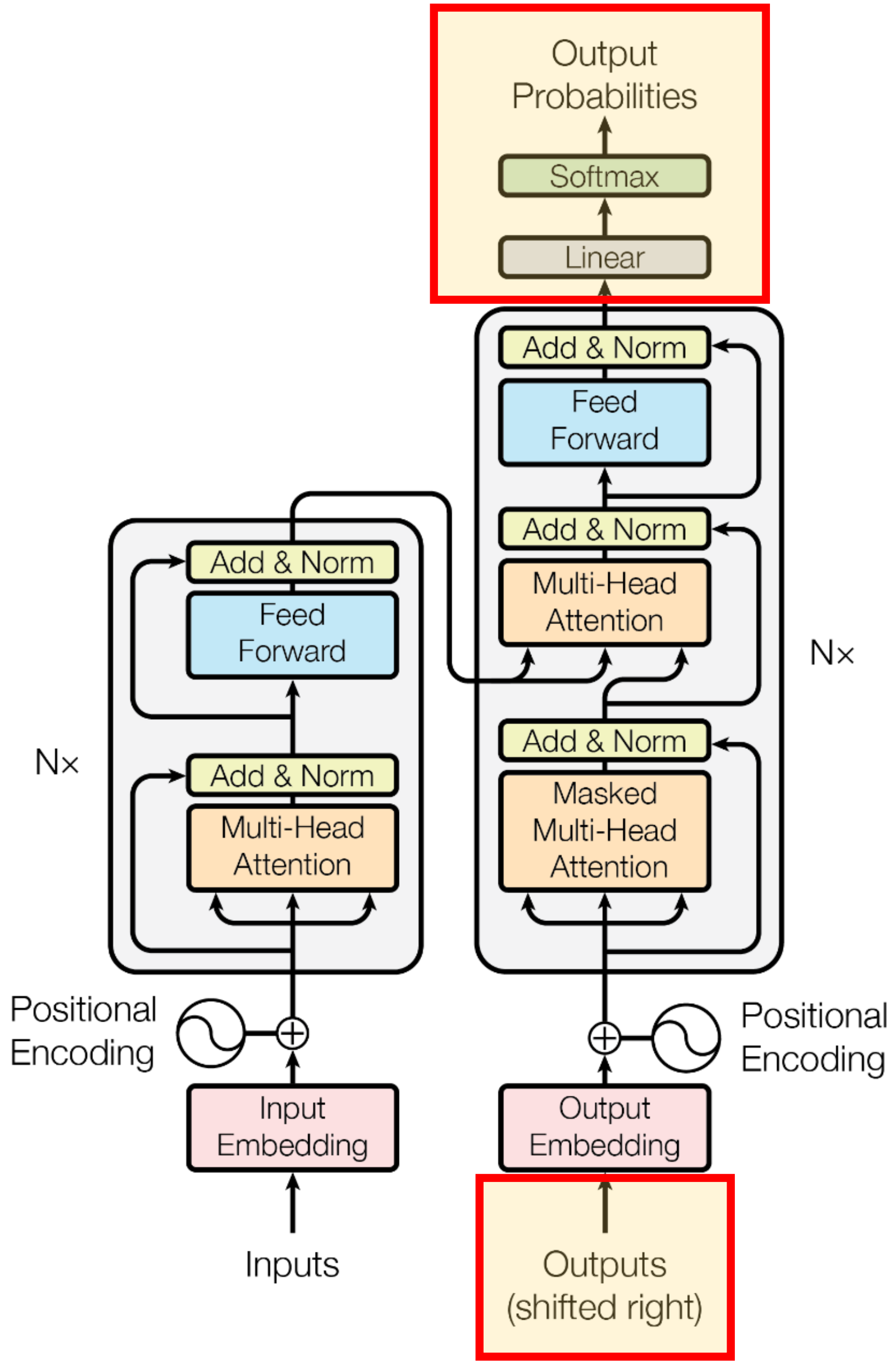




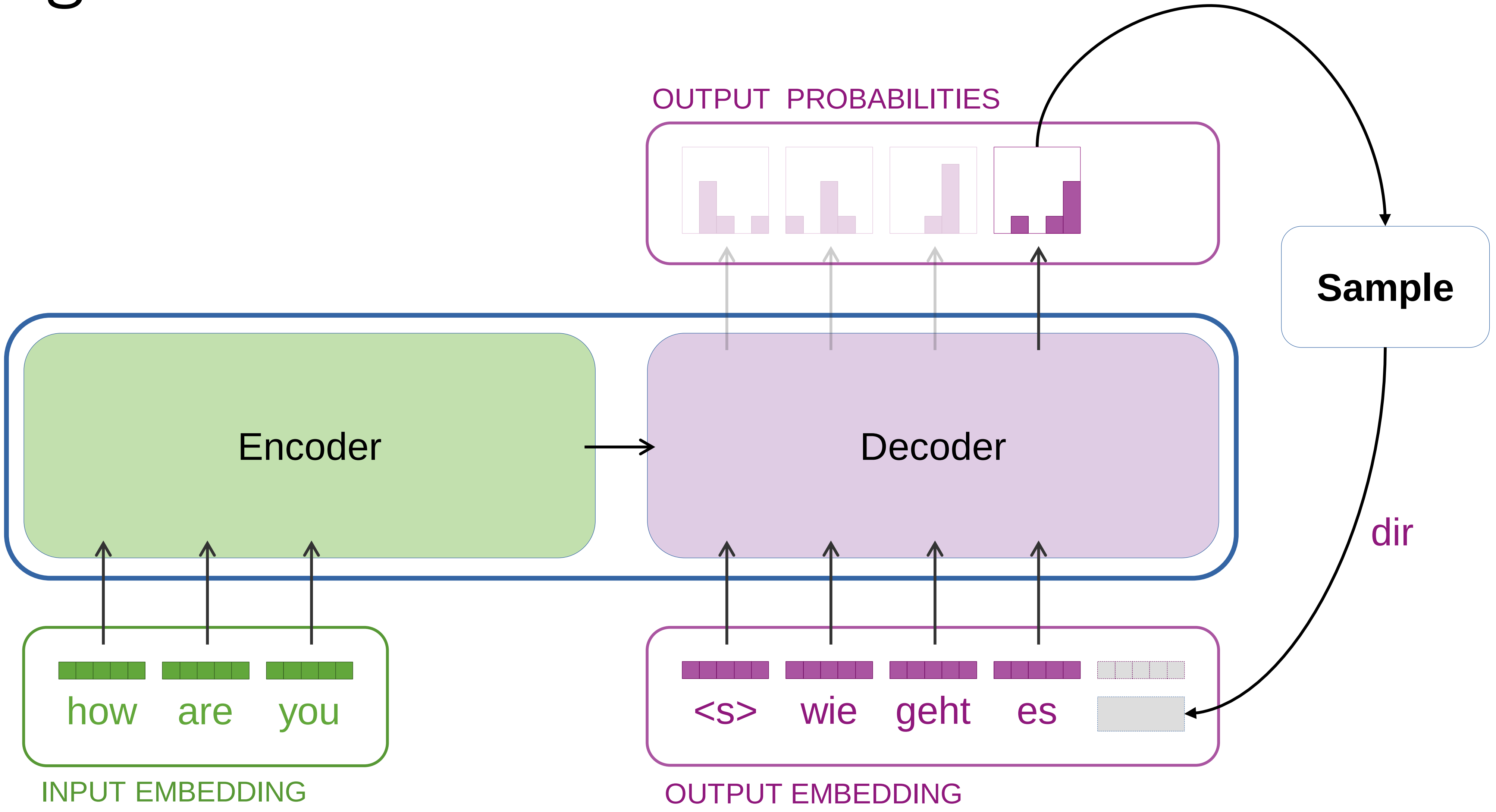
- Decoder output is projected to vocabulary size logits vector
- Softmax step gives token probabilities
- Weight matrix of embeddings & pre-softmax output projection may be tied

Key Components

- Encoder & Decoder
- Input Embedding
- Next Token Classifier
- **Autoregressive Decoder**
- Encoder-Decoder Attention
- Self-Attention
- Feed-Forward Layer
- Residual & Normalization
- Positional Encoding

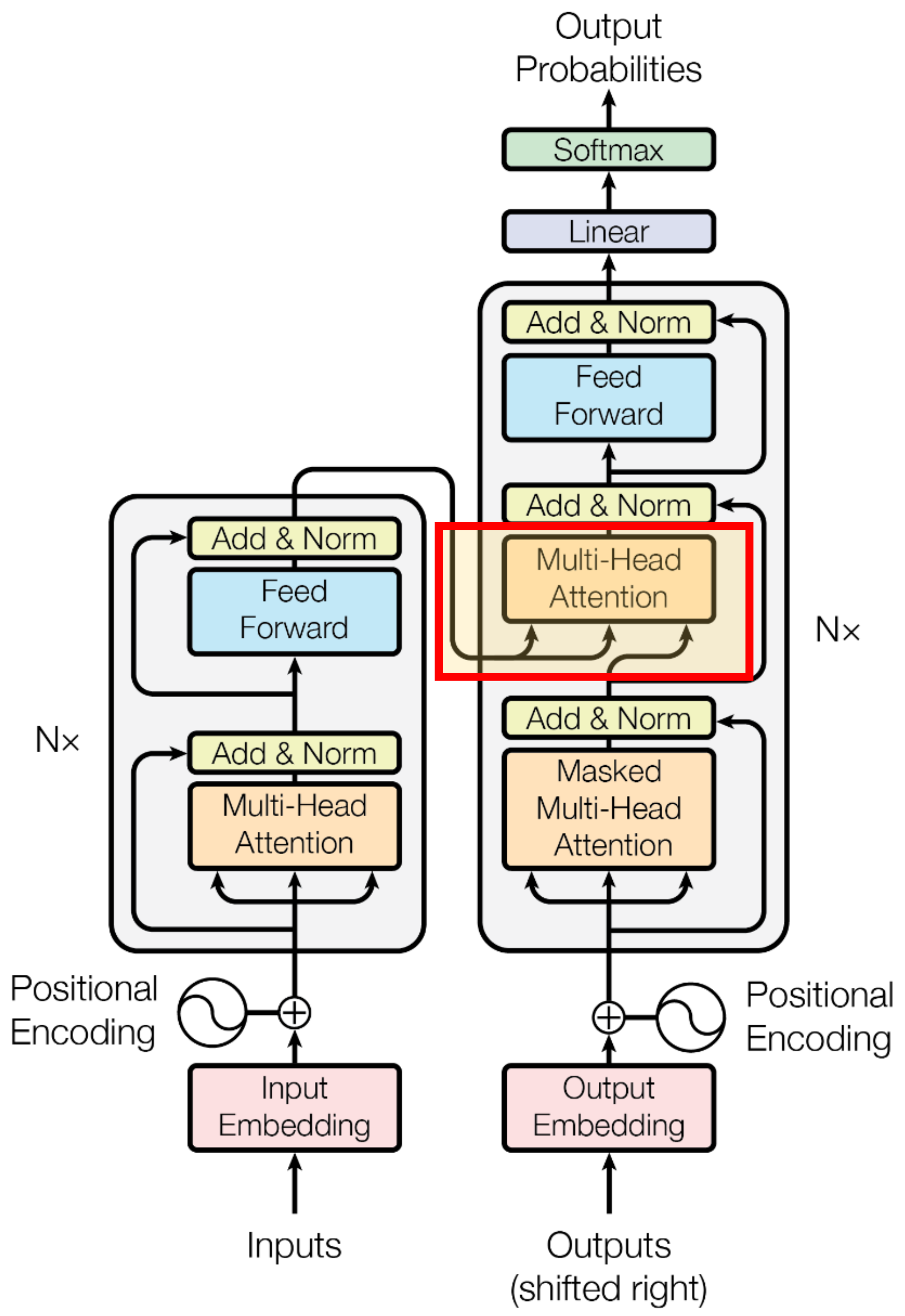


Autoregressive Decoder



Key Components

- Encoder & Decoder
- Input Embedding
- Next Token Classifier
- Autoregressive Decoder
- **Encoder-Decoder Attention**
- Self-Attention
- Feed-Forward Layer
- Residual & Normalization
- Positional Encoding



Encoder-Decoder Attention Mechanism

Intuition

- Focus on relevant source words while generating a target word
- Decide which information enters "working memory"
- Allow the decoder to attend over all positions in the input sequence (in one step)

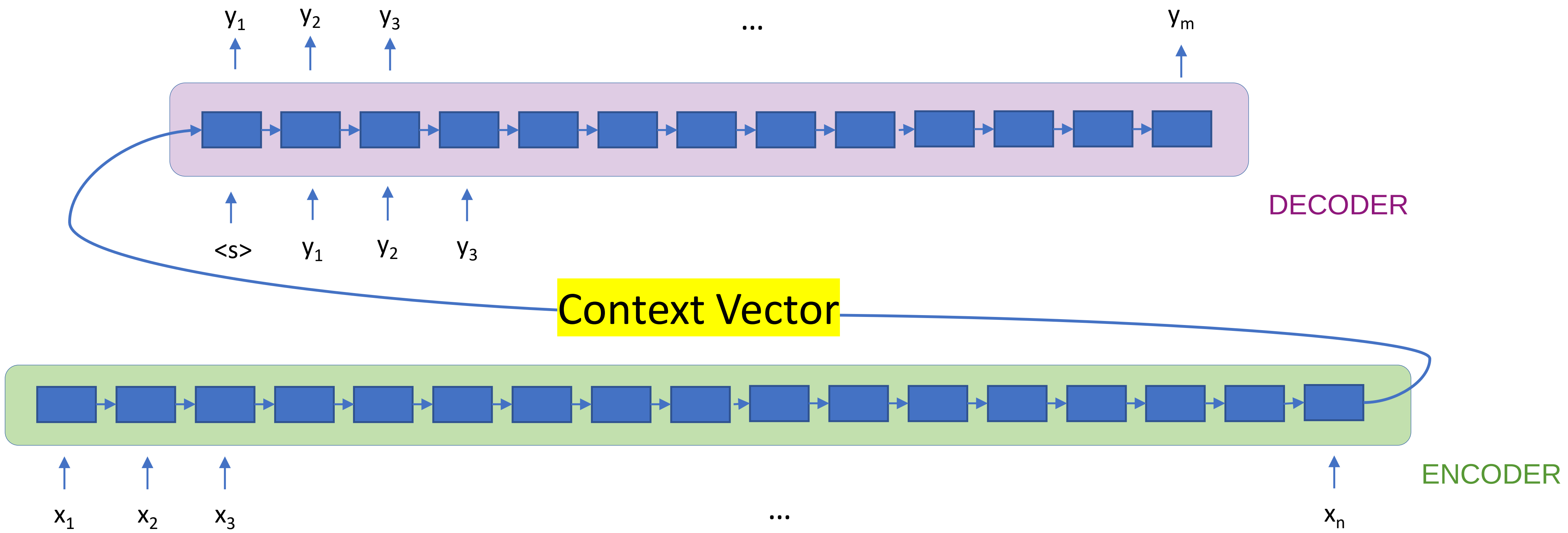
Math View

- A form of differentiable soft memory read and transform operation
- Extraction of fixed sized vectors from variable length memory

History of Encoder-Decoder Attention

Sequence-to-Sequence RNN without attention:

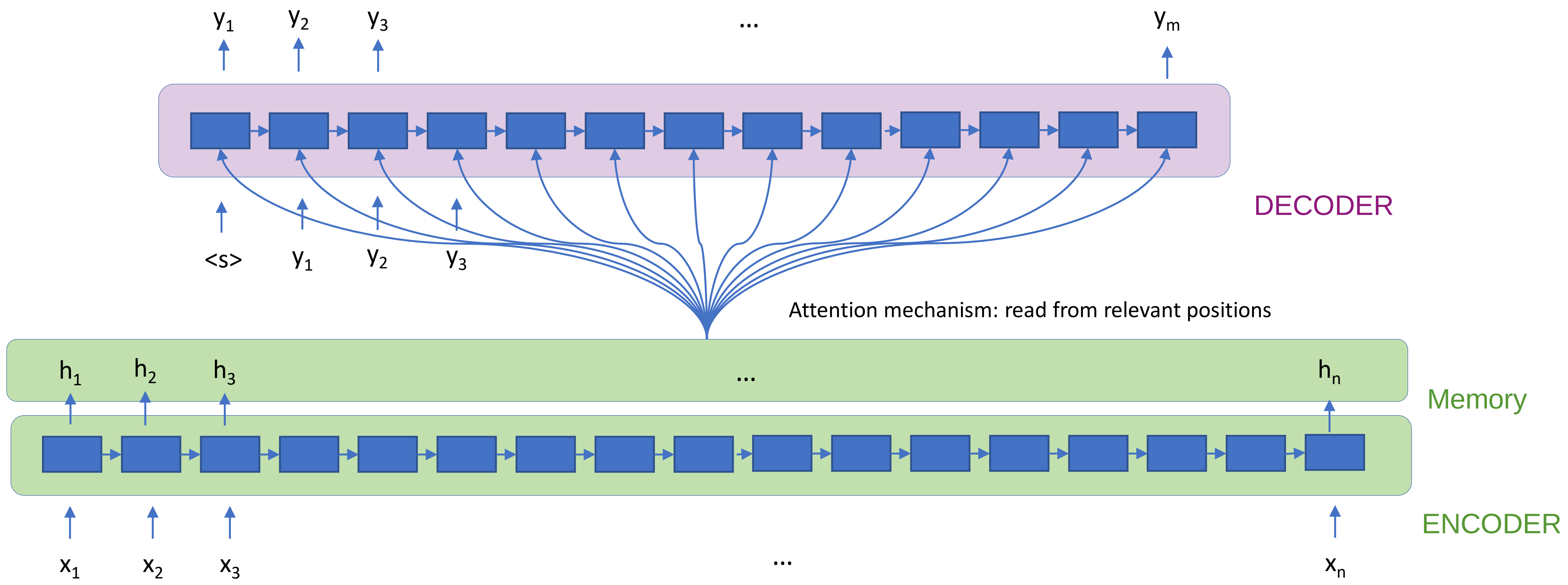
- Input sequence had to be "compressed" into context vector
- Problem: Long computational path between input & output

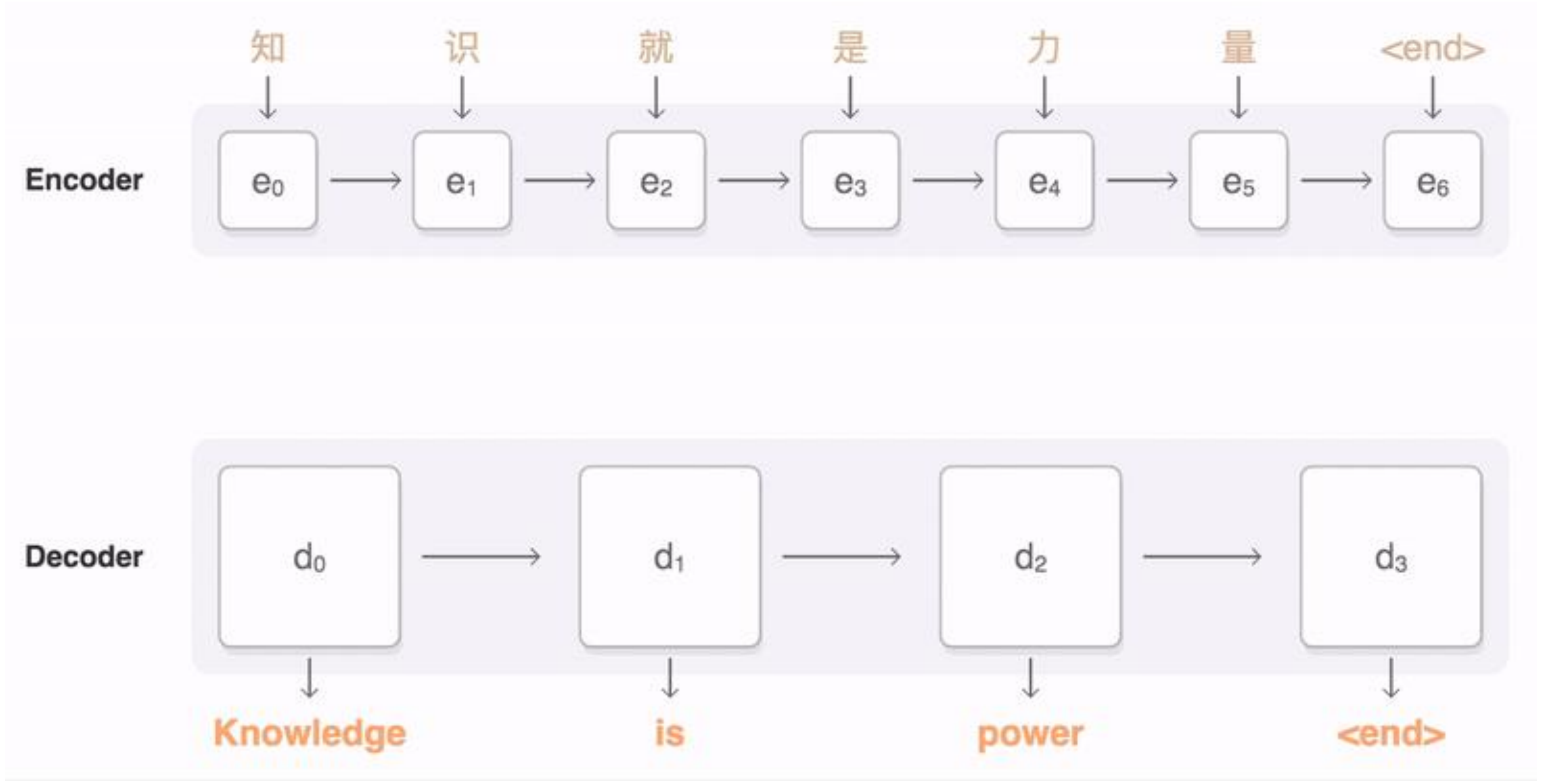


History of Encoder-Decoder Attention

RNN with attention:

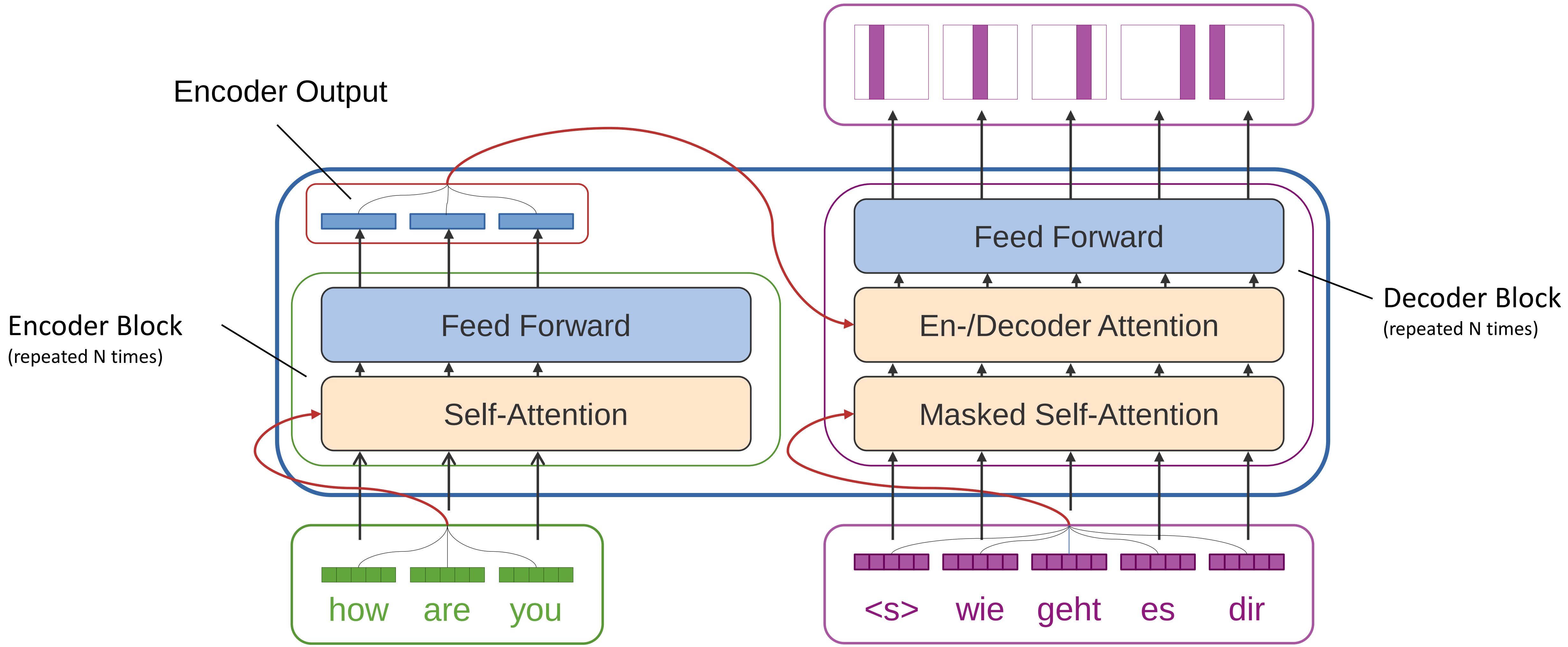
- Attention layer “augments” hidden states of the decoder
- Decoder “reads” values from the encoder (based on its state)





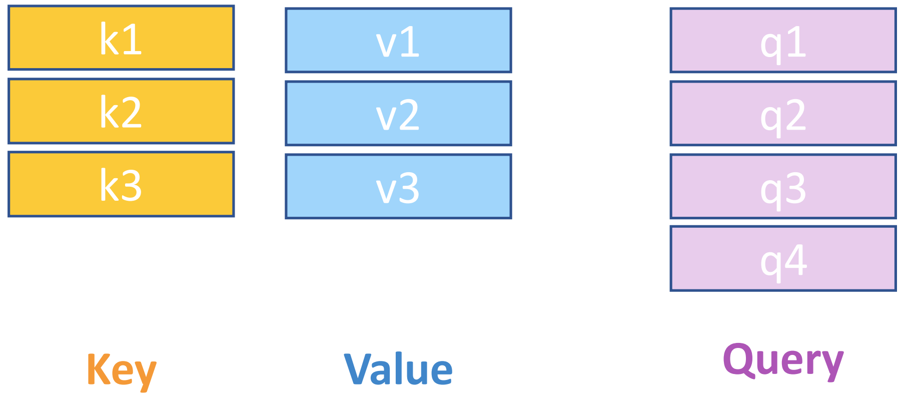
Source: https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html

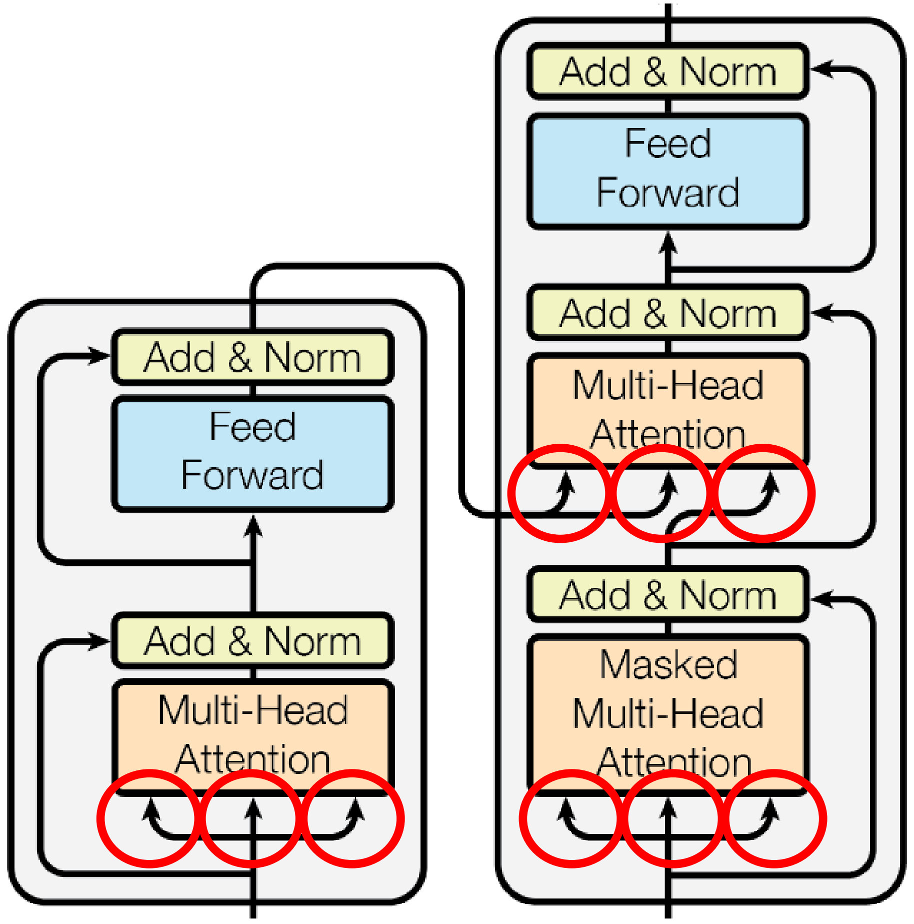
Encoder-Decoder Attention



Attention

- Attention layers have three inputs:
- Key, Value & Query (K, V, Q)



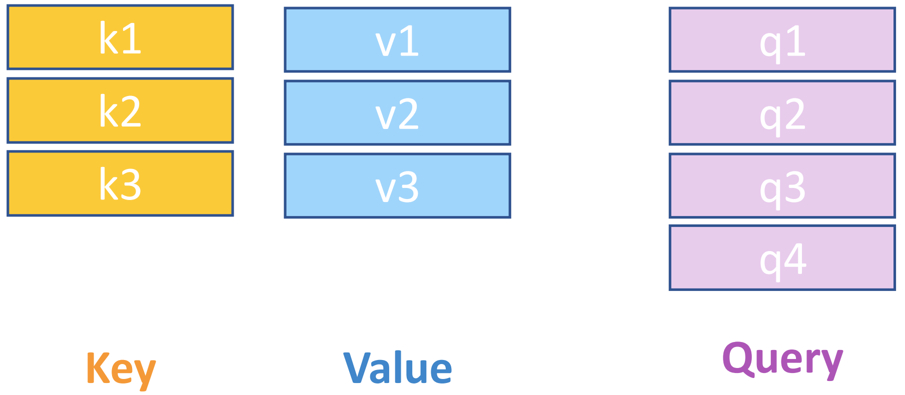


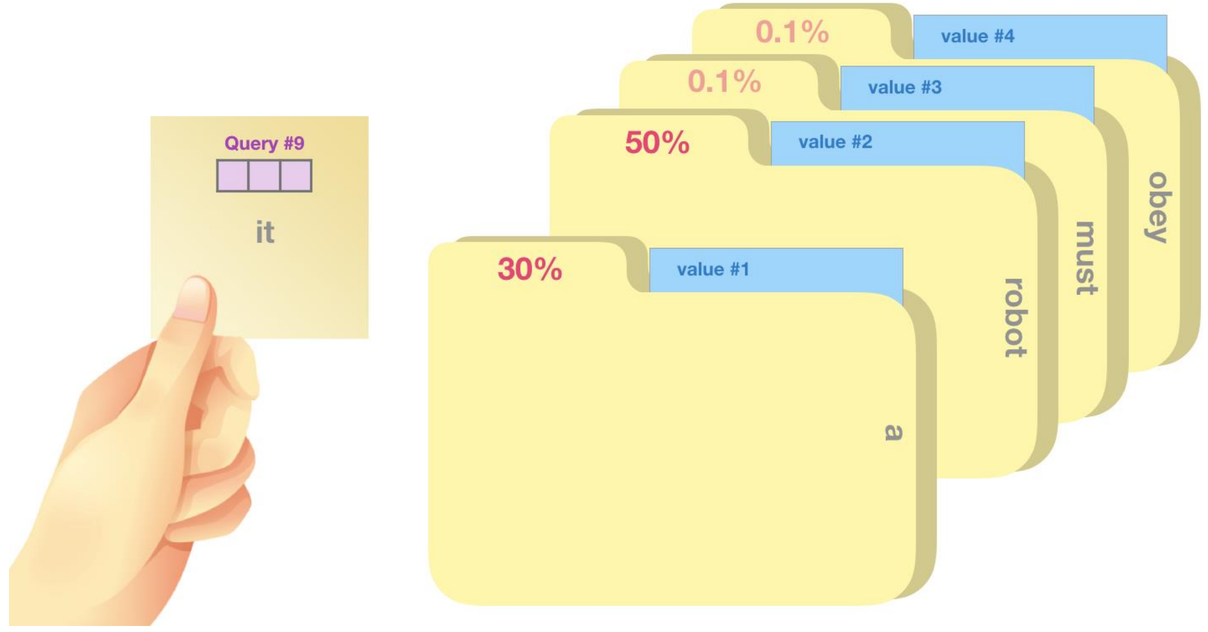
Attention

- Attention computes a soft **Key** - **Value** based mapping of **Query** elements.
- Each **Key** $\mathbf{k}_i$ corresponds to **Value** $\mathbf{v}_i$, as in a lookup-table / dictionary
- Attention is like a memory read op:

Mem content:	**Value**
Mem addresses:	**Keys**
Read addresses:	**Query**

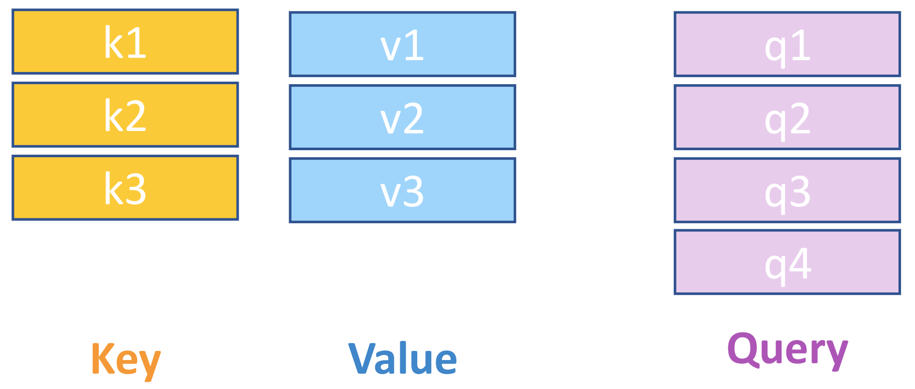
-> Transformer uses a soft, differentiable Attention mechanism!

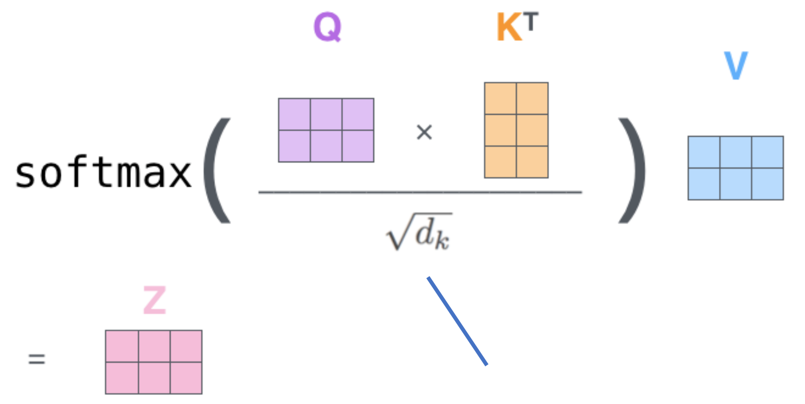




Scaled Dot Product Attention

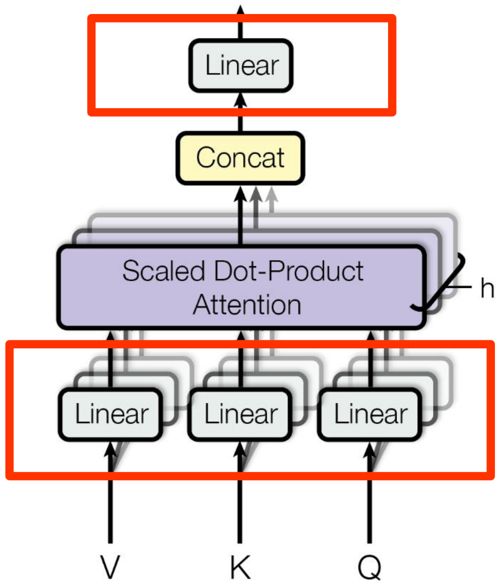
1. The similarity of each **Query** vector to all **Key** vectors is measured via dot-product.
2. These similarity-scores are passed through softmax to determine the contribution of each **Value** element to the output.
3. A weighted average of **Value** rows is computed based on the softmax output.

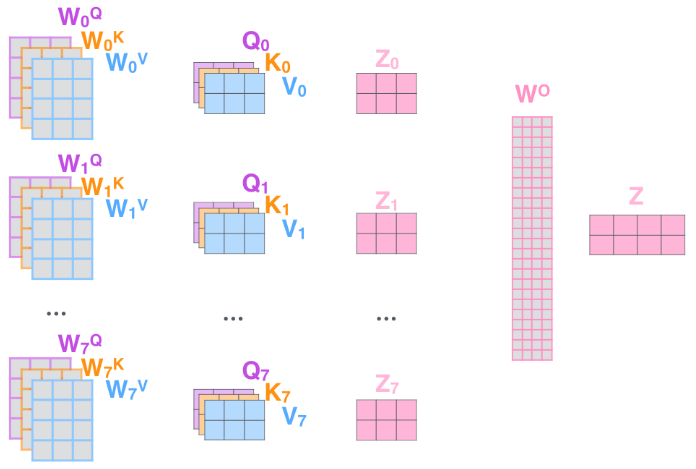




Scaling compensate "variance gain", lowers softmax temperature, helps gradient flow

Multi-head Attention



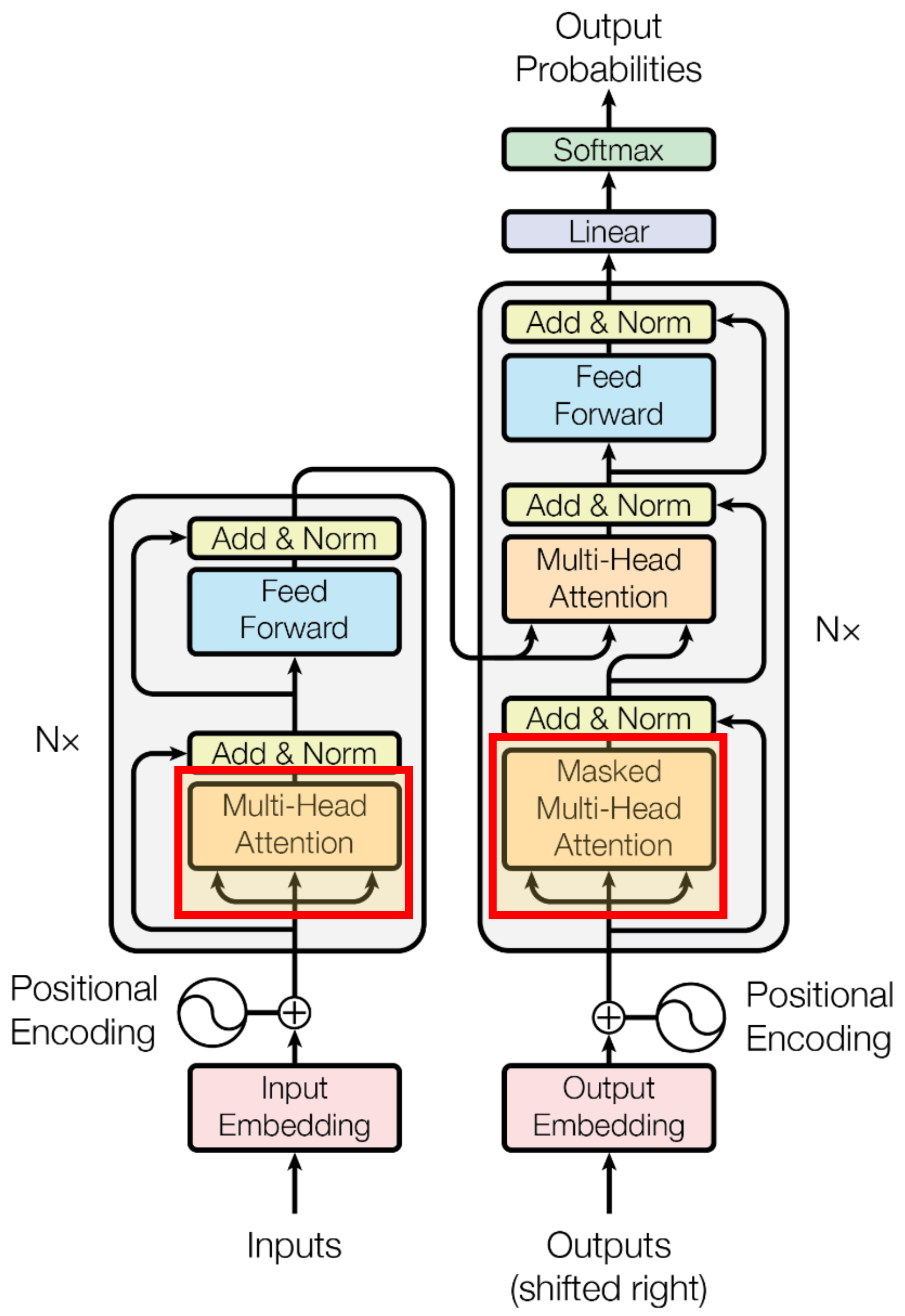


- Perform multiple attention functions in parallel: “attention heads”
- Linearly project **Key**, **Value** and **Query** multiple times (with different weights) for each “head”
- Finally concat all outputs and project once again

- Low overhead if **V**, **K**, **Q** are projected to lower dimensionality (e.g. $d_k = d_{model} / h$)

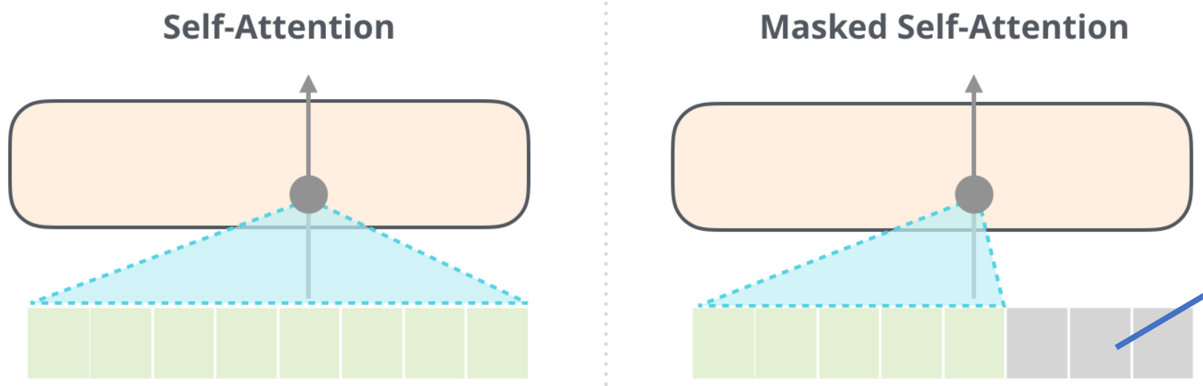
Key Components

- Encoder & Decoder
- Input Embedding
- Next Token Classifier
- Autoregressive Decoder
- Encoder-Decoder Attention
- **Self-Attention**
- Feed-Forward Layer
- Residual & Normalization
- Positional Encoding



Self-Attention

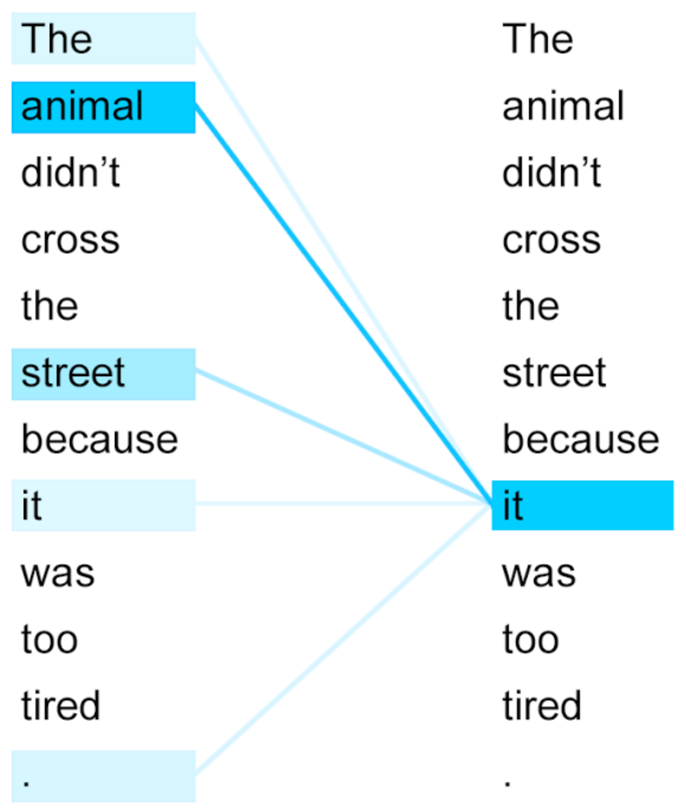
- **Key**, **Value** and **Query** inputs of a Self-Attention layer are all the same: `multi_head_attention(x, x, x)`
- Create context dependent representations: word -> thought vector
- Relate with all positions of the input sequence in one step
- Masked Self-Attention: Let decoder only attend to past outputs (masks out all (future) sequence elements to the right of a token

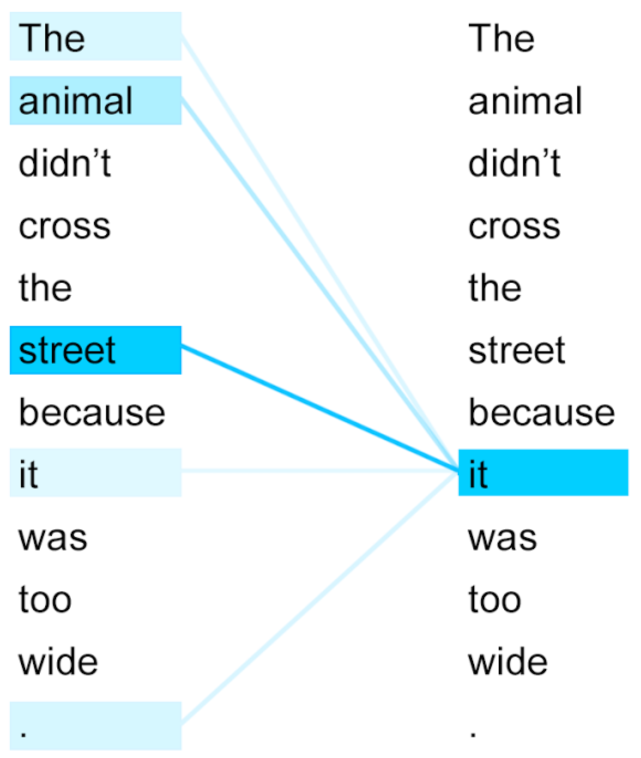


e.g. fill with `-1e9` before softmax

[image source]

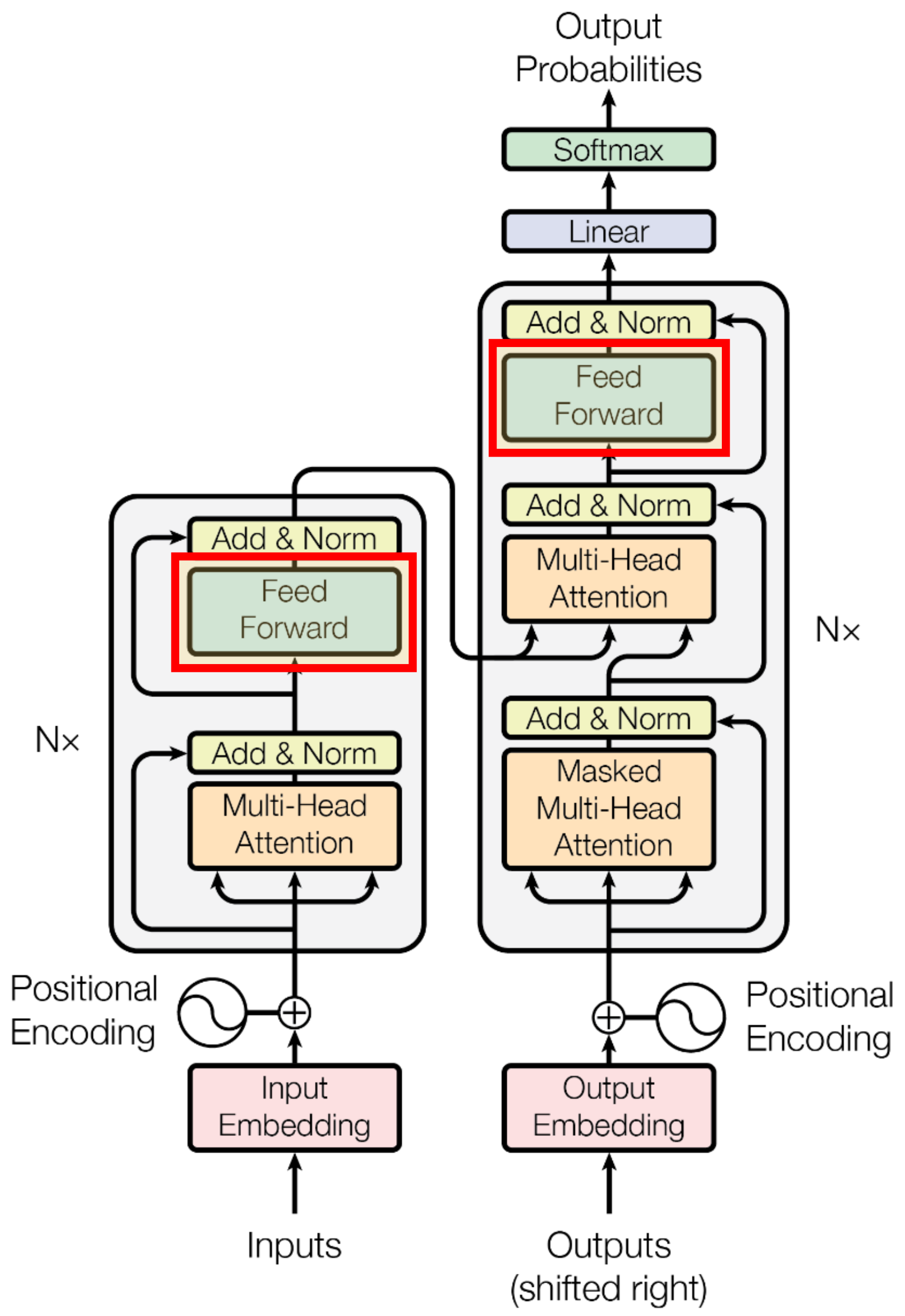
Self-Attention





Key Components

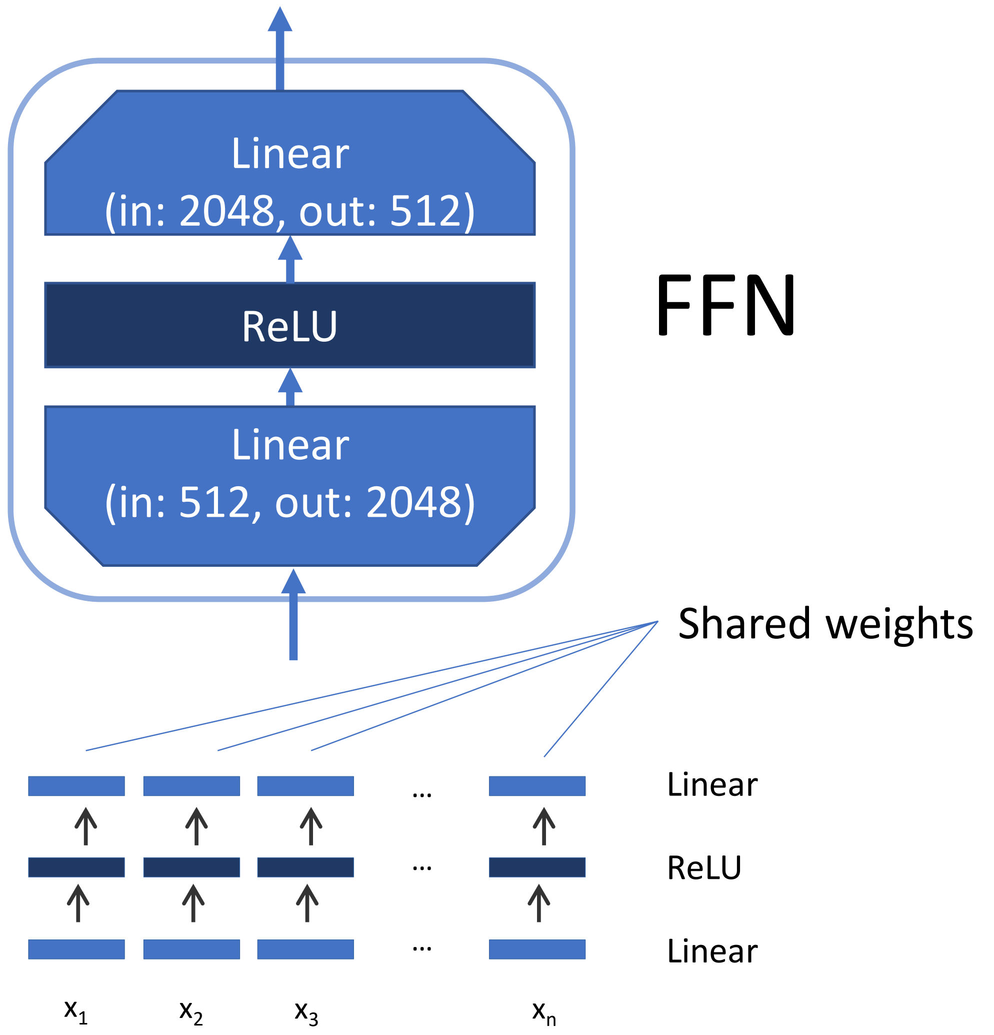
- Encoder & Decoder
- Input Embedding
- Next Token Classifier
- Autoregressive Decoder
- Encoder-Decoder Attention
- Self-Attention
- **Feed-Forward Layer**
- Residual & Normalization
- Positional Encoding



Position-wise Feed-Forward Network

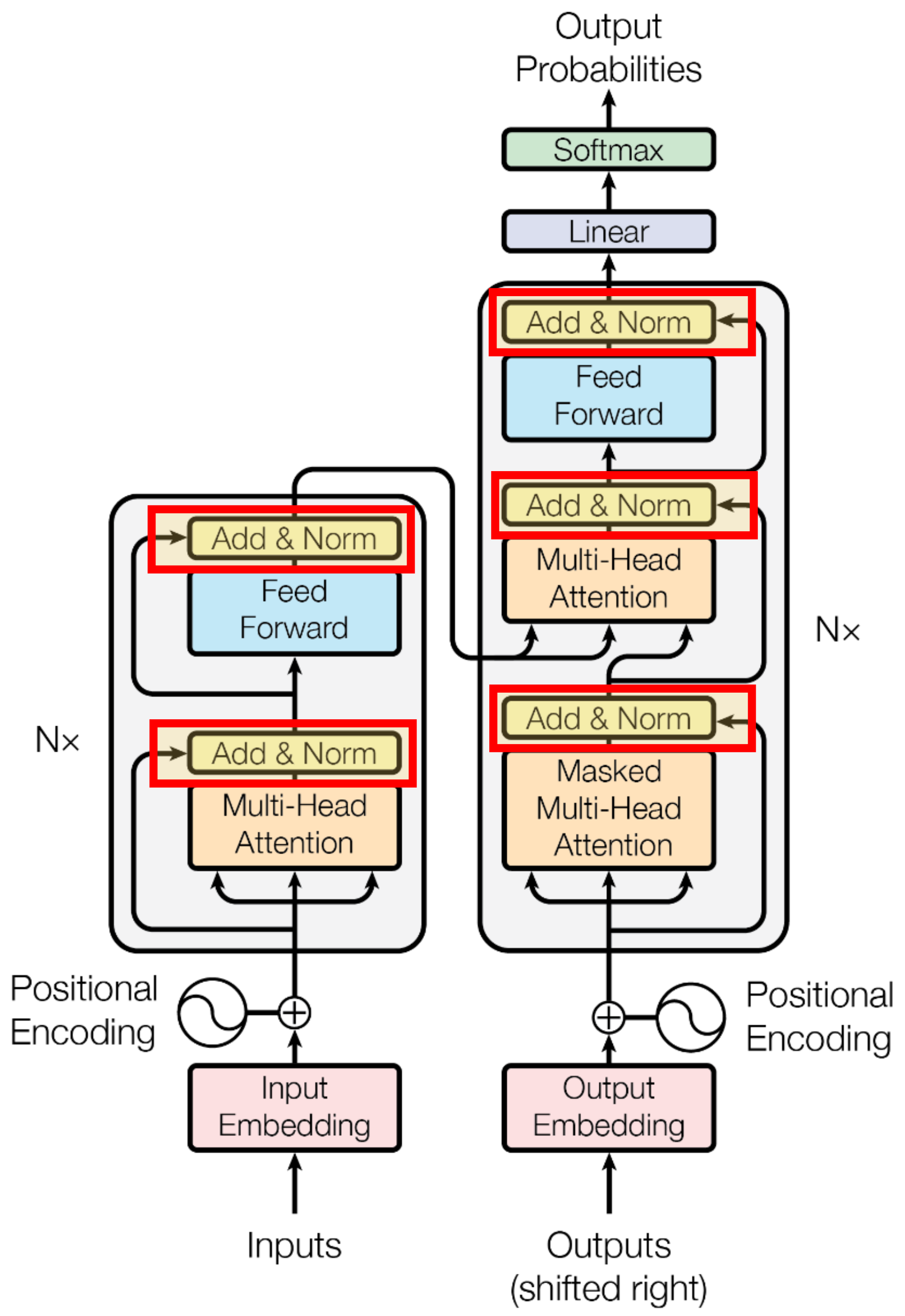
- Two linear transformation with a ReLU activation function
- Shared weights for all positions in a layer, like Conv with kernel-size 1
- Different weights layer to layer
- Inner layer dimensionality: $d_{ff} = 4 * d_{model}$

- Contains ~50% of the transformer parameters



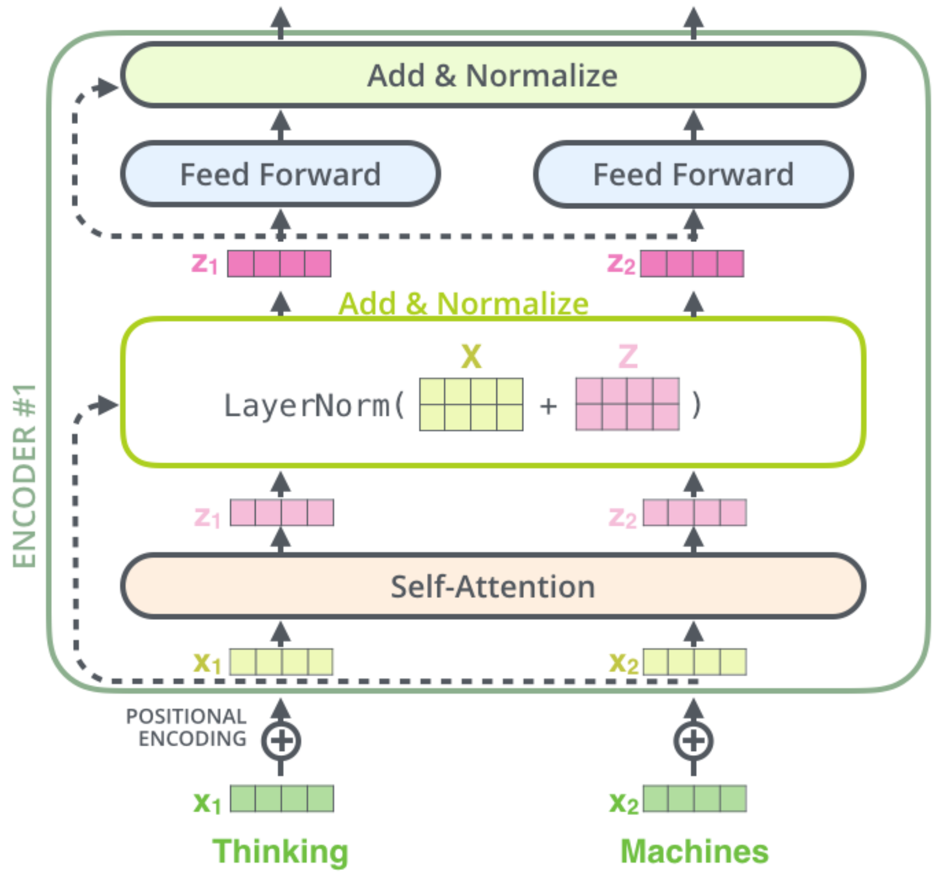
Key Components

- Encoder & Decoder
- Input Embedding
- Next Token Classifier
- Autoregressive Decoder
- Encoder-Decoder Attention
- Self-Attention
- Feed-Forward Layer
- **Residual & Normalization**
- Positional Encoding



Residual & Normalization

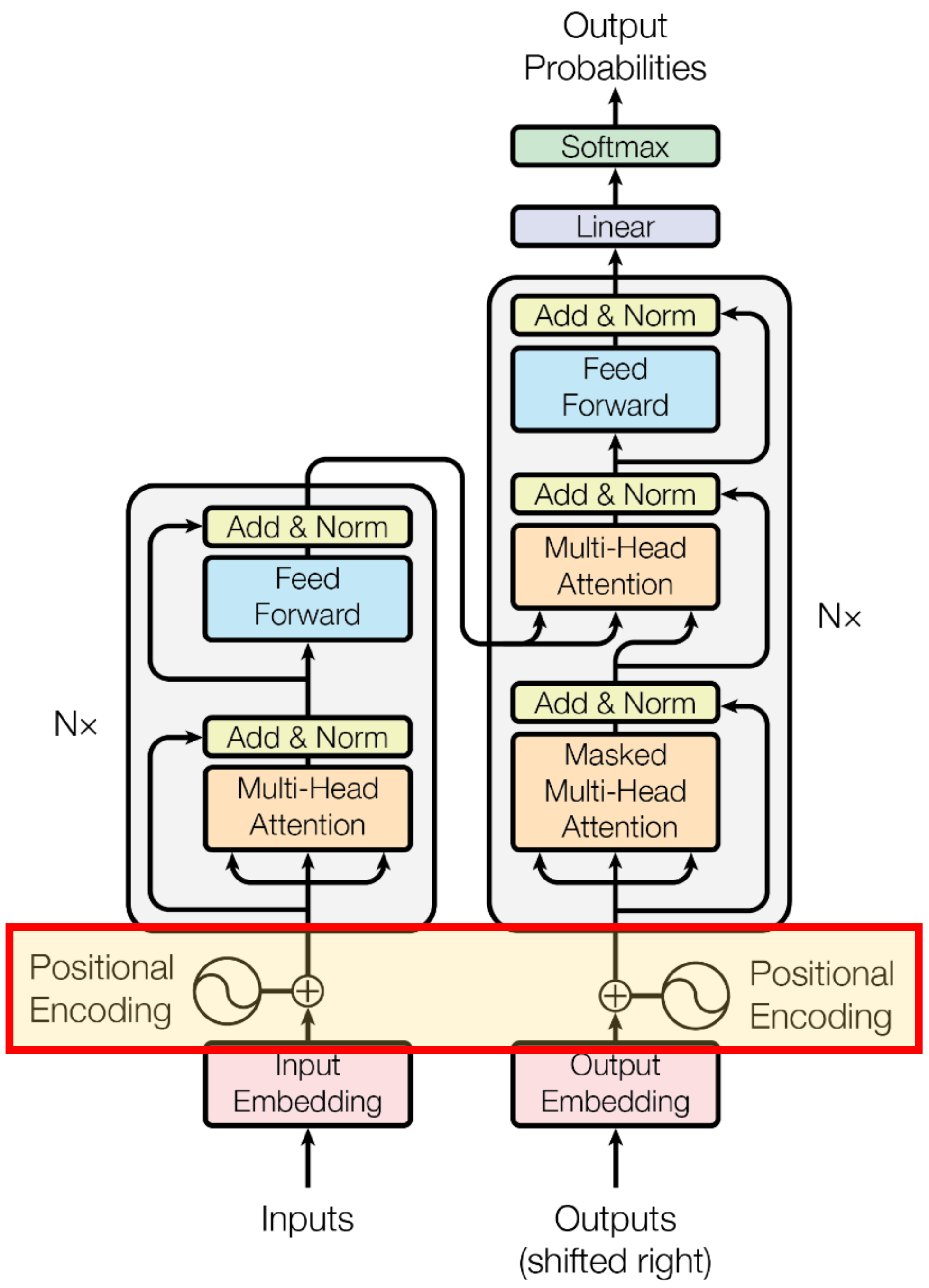
- Help avoiding vanishing gradients
- Layer(x) = LayerNorm(x + Sublayer(x))
- Note: All sub-layers have the same output size: d_{model}
- LayerNorm: mean and variance of each element-vector is normalized independently



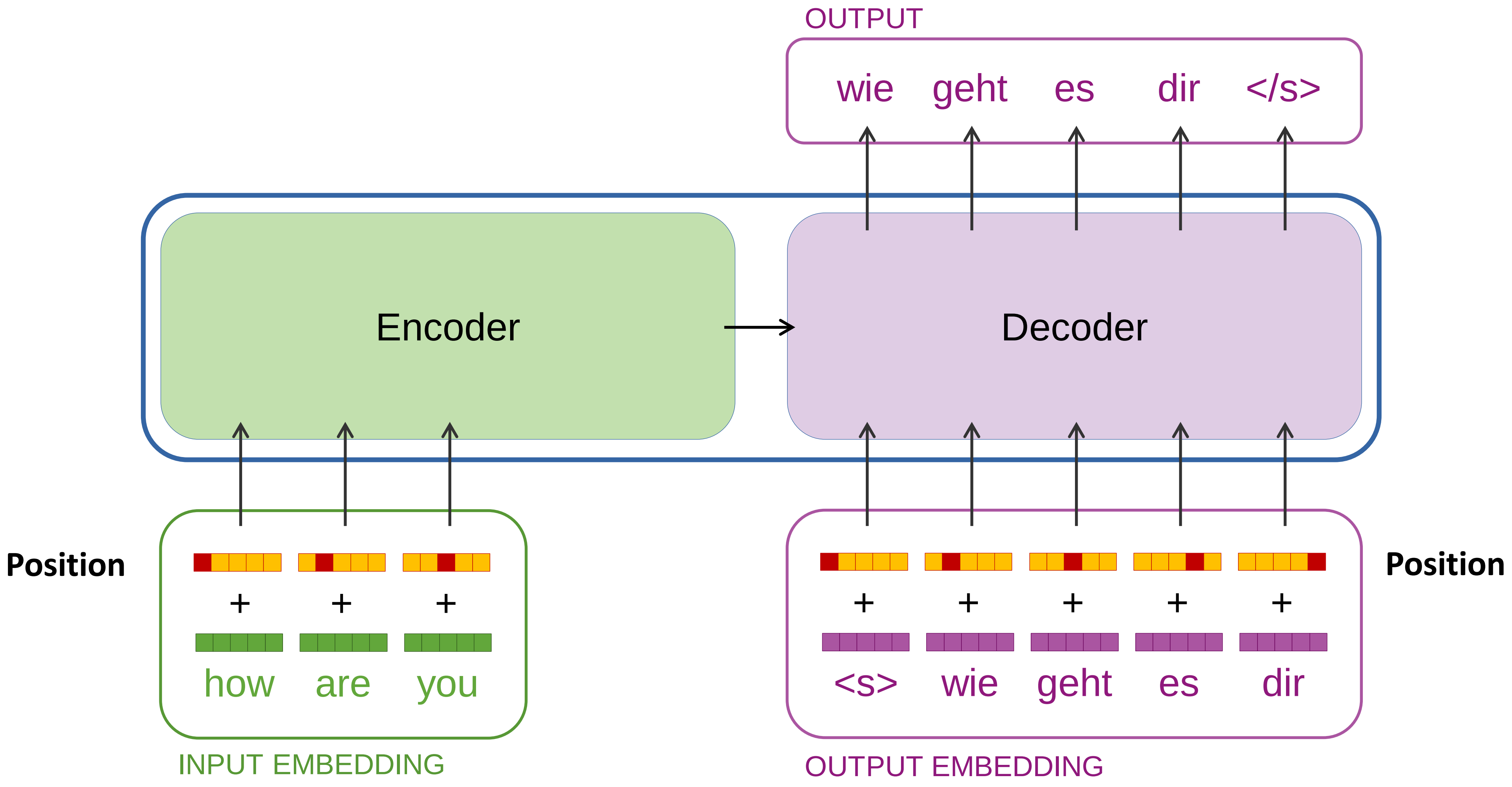
[Image Source]

Key Components

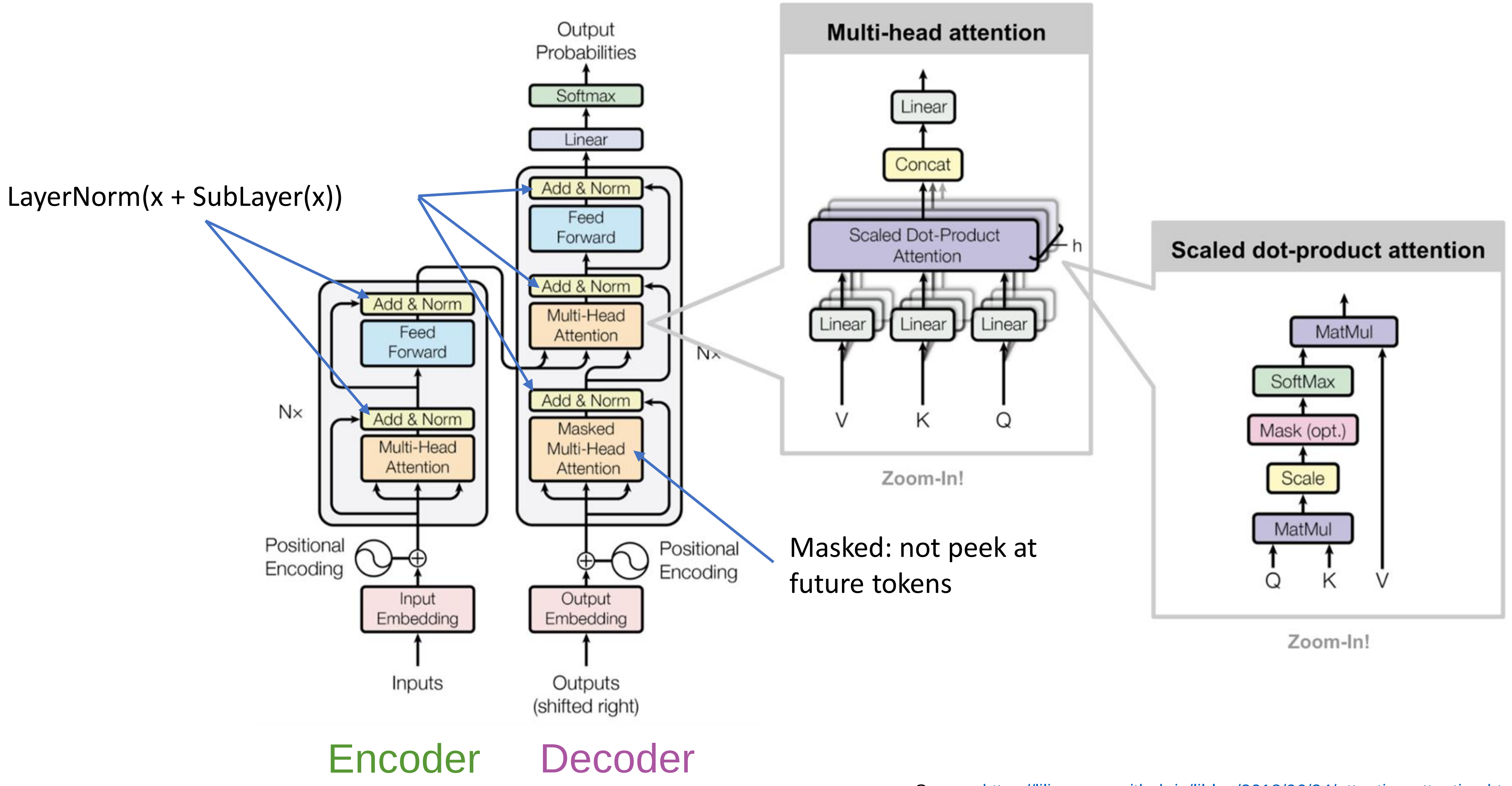
- Encoder & Decoder
- Input Embedding
- Next Token Classifier
- Autoregressive Decoder
- Encoder-Decoder Attention
- Self-Attention
- Feed-Forward Layer
- Residual & Normalization
- **Positional Encoding**



Positional Encoding



The Full Transformer

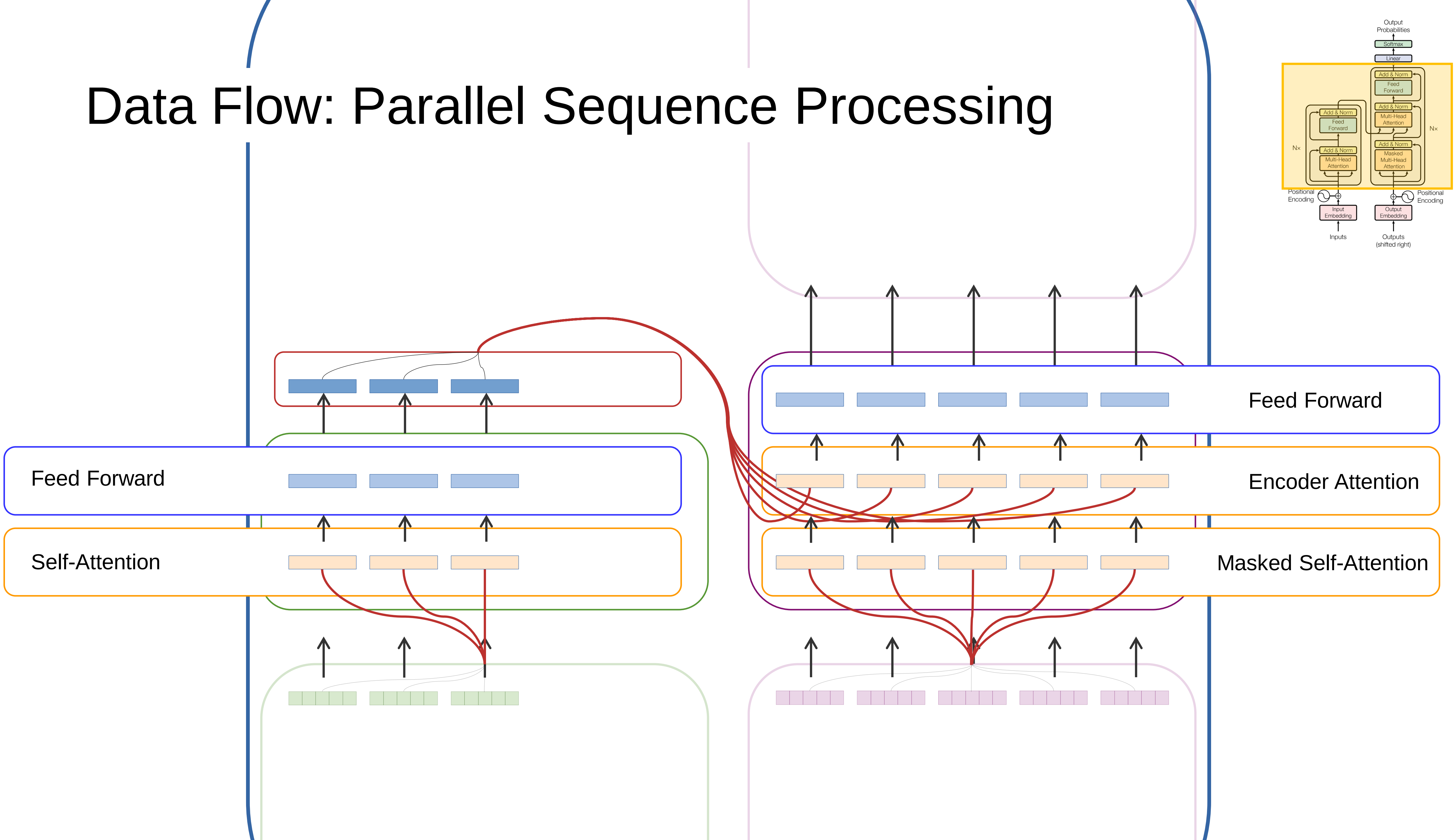


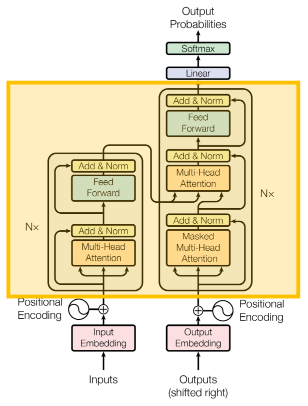
LayerNorm(x + SubLayer(x))

Masked: not peek at future tokens

Source: https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

Data Flow: Parallel Sequence Processing





Selected Resources: Go And Transform

Fun:

- https://talktotransformer.com/
- https://transformer.huggingface.co/ (Write With Transformer)

Blogs:

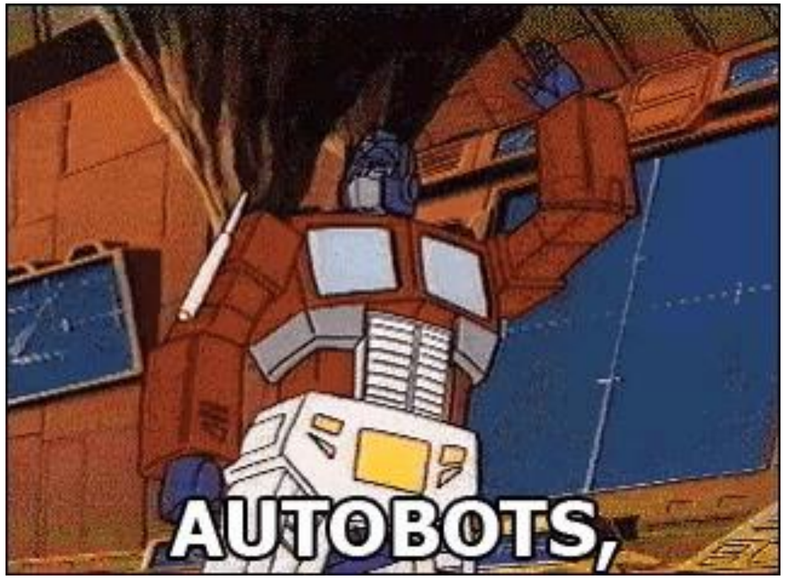
- http://jalammar.github.io/
- http://nlp.seas.harvard.edu/2018/04/03/attention.html (The Annotated Transformer)
- https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

PyTorch:

- https://huggingface.co/pytorch-transformers/
- https://github.com/pytorch/fairseq
- https://docs.fast.ai/text.models.html

Tensorflow:

- https://www.tensorflow.org/beta/tutorials/text/transformer
- https://github.com/google-research/bert



Questions?

Thanks for your attention! ;-)

Bonus Slides

GPT-2 Sample Text

"The Transformer" is a story about a young inventor's attempt to win a competition to see who can build and develop a robot that can drive a car. His robot will be more powerful than other designs.

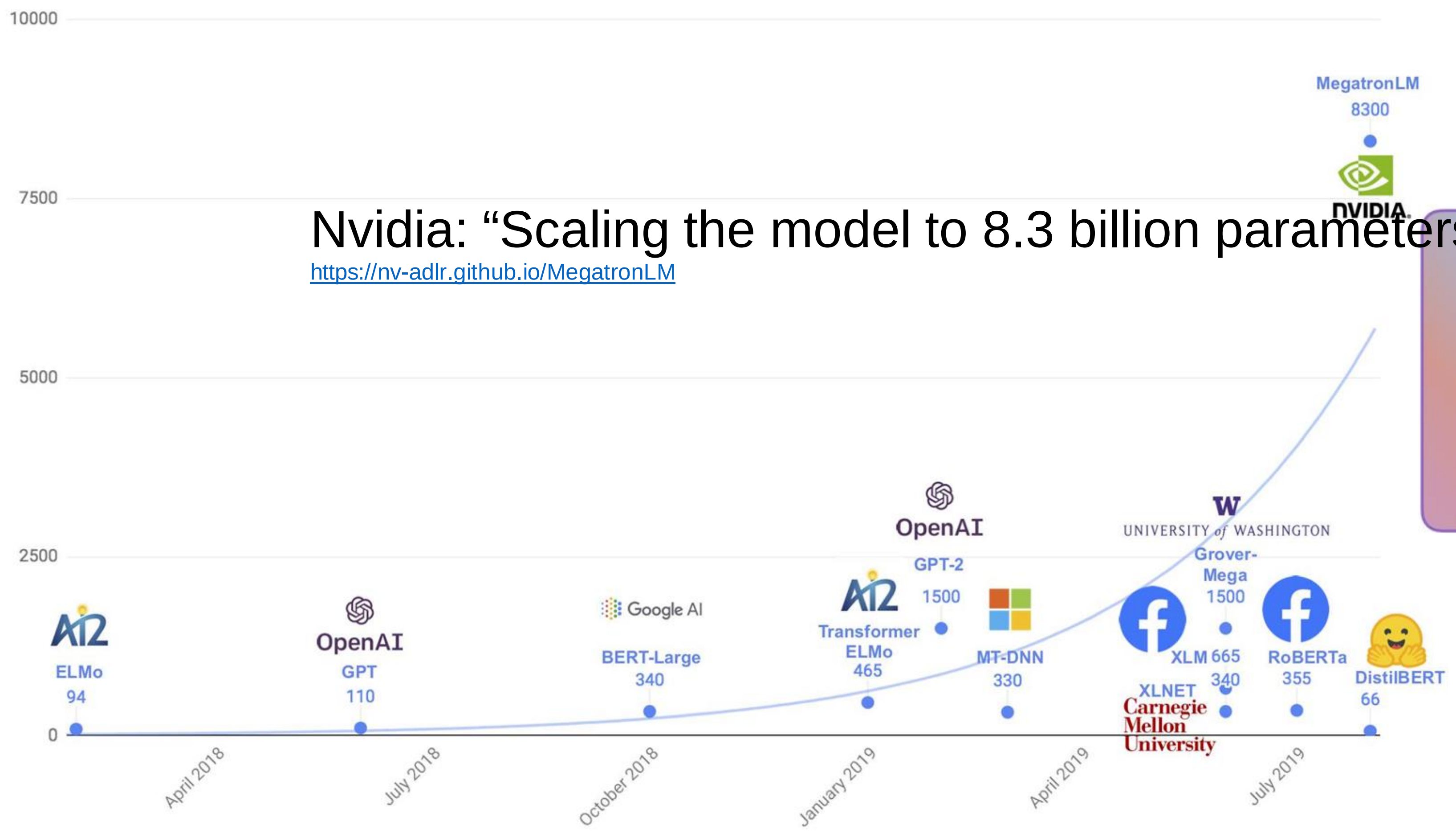
The robot is named Mimi. It uses a three-cylinder engine which will allow it to achieve speeds on the open road that would otherwise be impossible. The Transformer will also have the ability to drive and accelerate on its own. Mimi will only drive after being charged with an electric current. Each charge will generate enough electricity to accelerate the robot up to a top speed of 20 mph. The Transformer could be driven from the garage and charged in a matter of seconds when the battery is depleted. The Transformer utilizes a special battery pack designed specifically for Mimi and can run for several months without charging.

Each Transformer is designed to look similar to a toy and includes an eye-catching LED headlights. The Transformer's wheels are also painted a bright orange color and the Transformer can also travel backward in time if it gets lost.

Generated by https://talktotransformer.com/

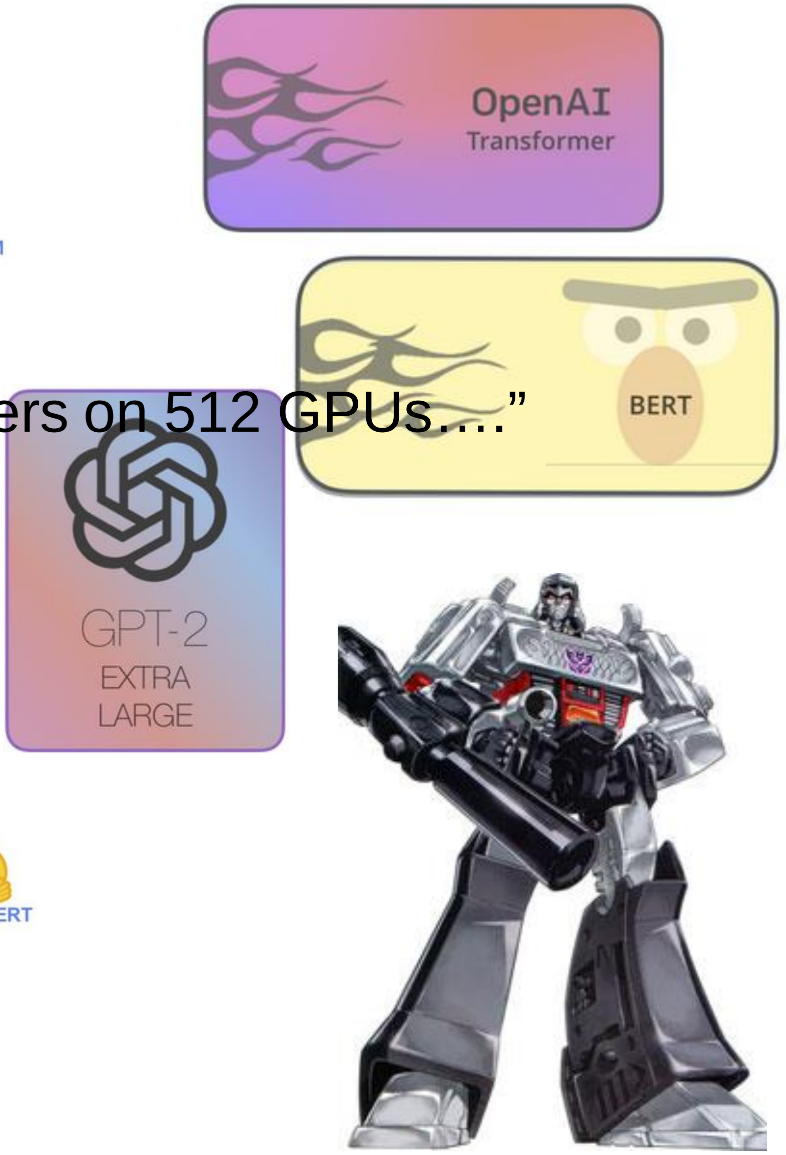
The Offspring

Model Parameters (Mio)



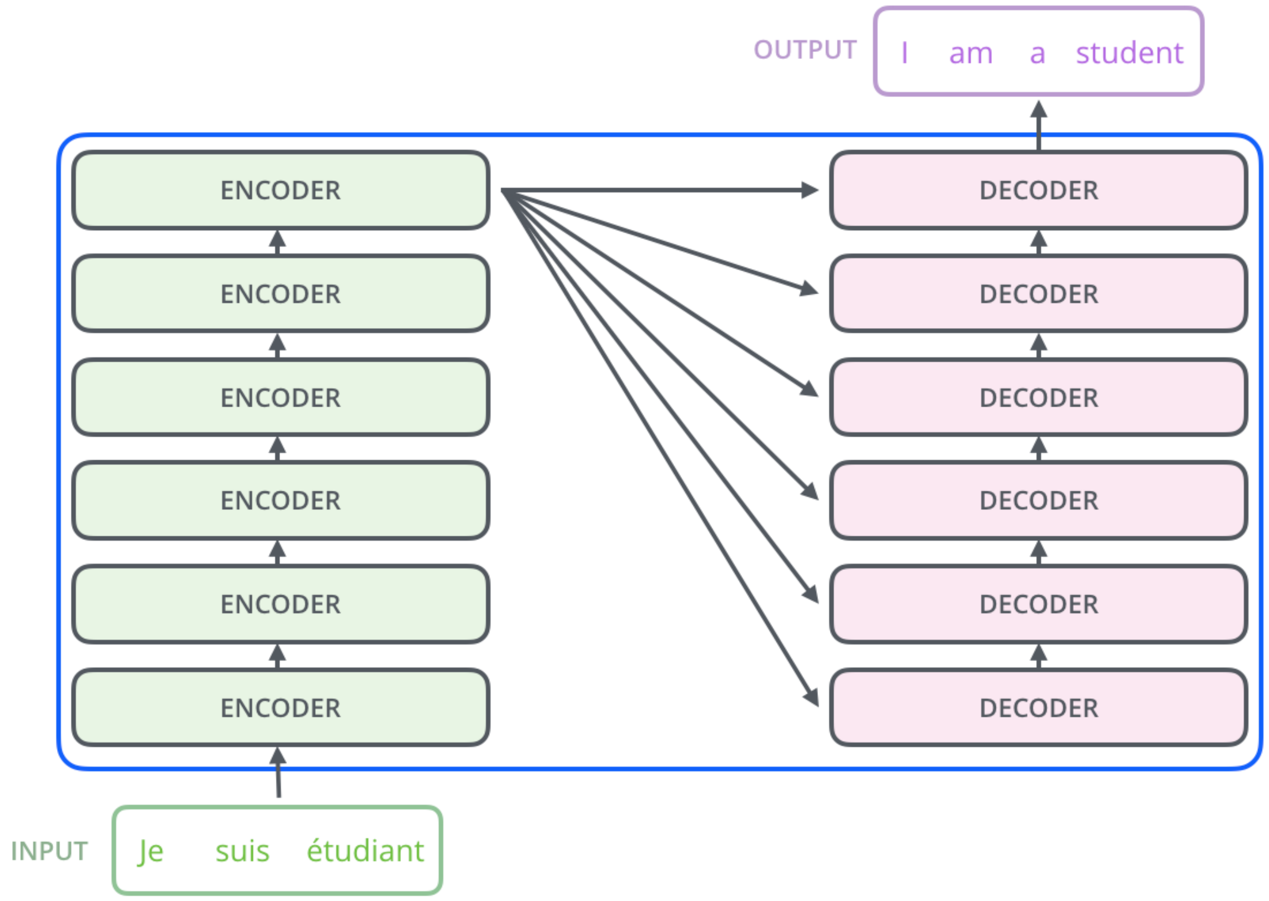
Nvidia: “Scaling the model to 8.3 billion parameters on 512 GPUs….”

https://nv-adlr.github.io/MegatronLM

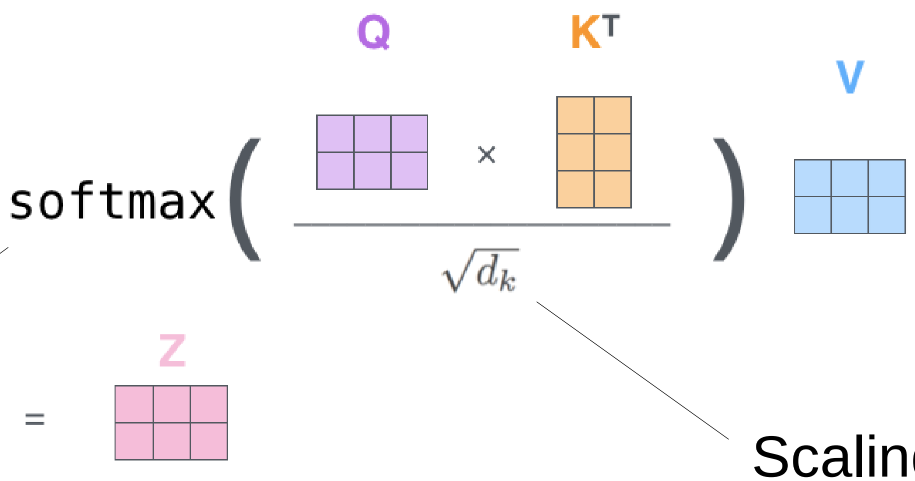


Sources:
https://medium.com/huggingface/distilbert-8cf3380435b5
https://jalammar.github.io/illustrated-bert/

Data Flow: Multilayer Encoder & Decoder



Scaled Dot Product Attention

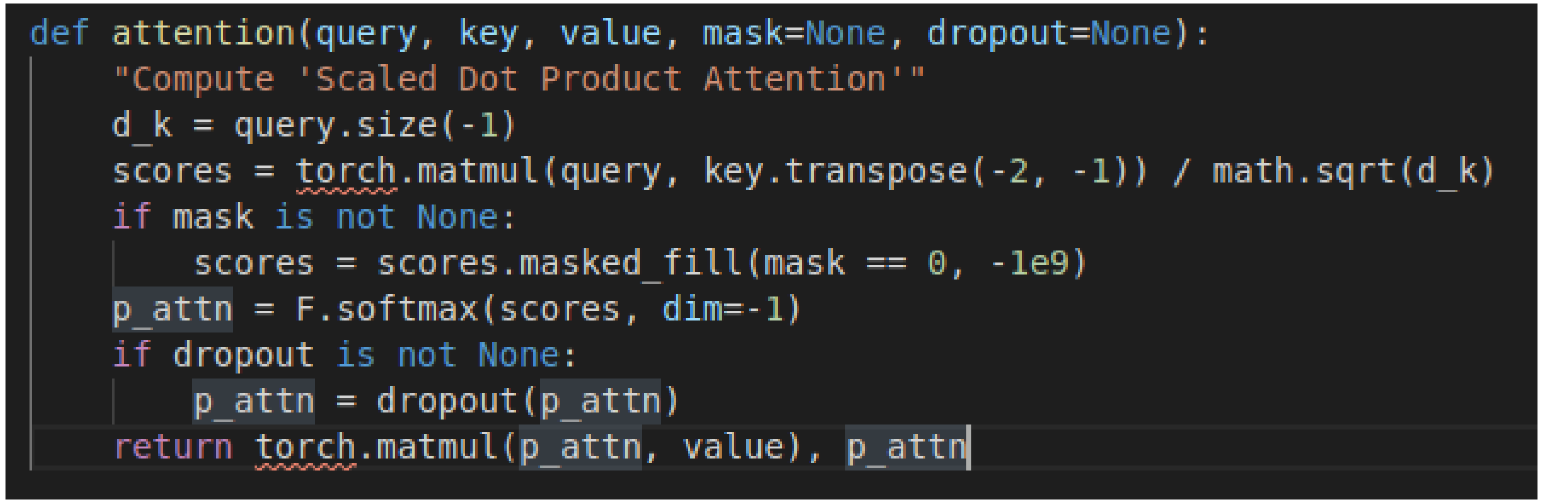


Q:	Query
K:	Key
V:	Value
Z:	Output

row-wise

$$\frac{\exp(y_i)}{\sum_j \exp\left(y_j\right)}$$

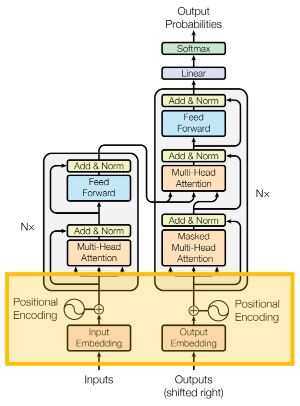
Scaling factor, lowers softmax ‘temperature’

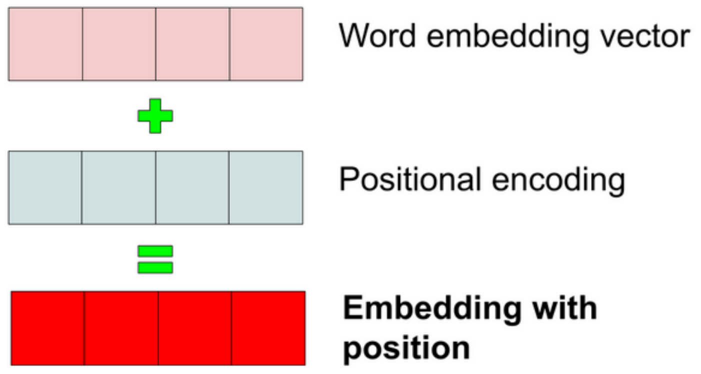


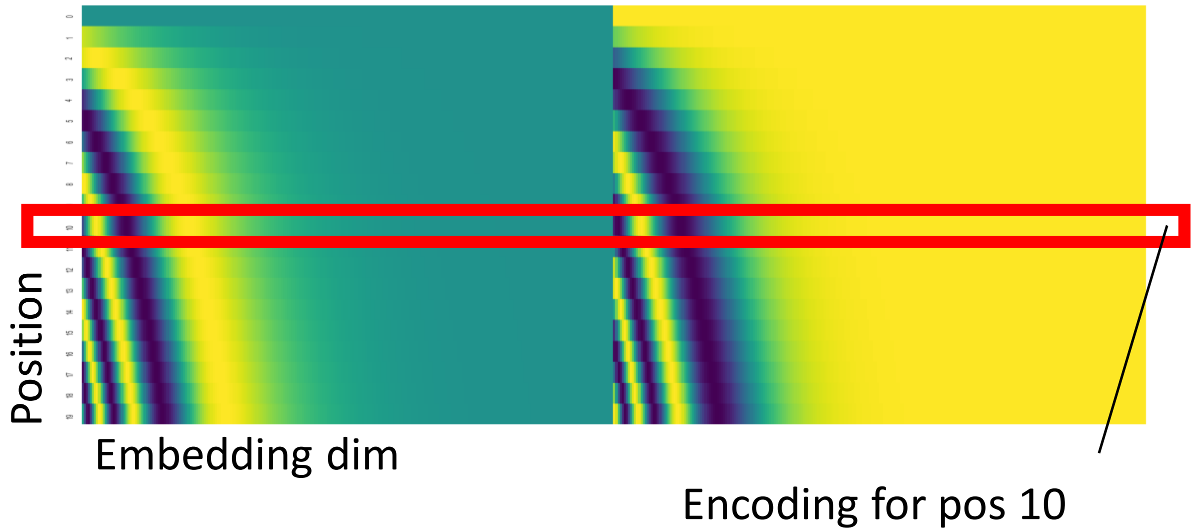
```python
def attention(query, key, value, mask=None, dropout=None):
    "Compute 'Scaled Dot Product Attention'"
    d_k = query.size(-1)
    scores = torch.matmul(query, key.transpose(-2, -1)) / math.sqrt(d_k)
    if mask is not None:
        scores = scores.masked_fill(mask == 0, -1e9)
    p_attn = F.softmax(scores, dim=-1)
    if dropout is not None:
        p_attn = dropout(p_attn)
    return torch.matmul(p_attn, value), p_attn
```

Positional Encoding

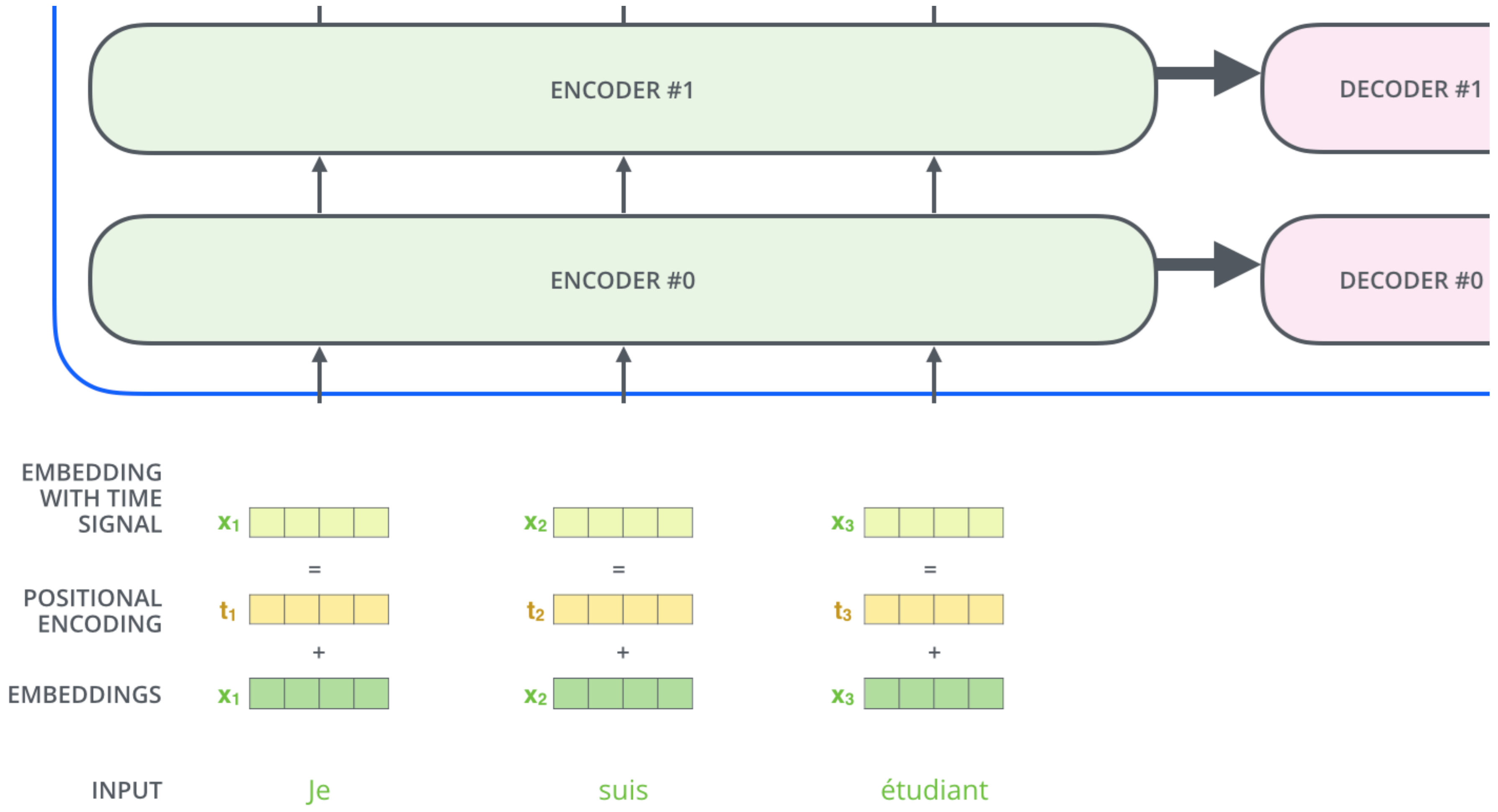
- Problem: Model is **invariant to input order**: En-/Decoder attention is omnidirectional
- Solution: Add vector to each input element to inject position information
- Encoding can be learned or hard-coded via fixed formulas
- Fixed encoding may help to attend by relative positions







Positional Encoding



Source: http://jalammar.github.io/illustrated-transformer/

Decoding Strategies

- **Greedy Search**
  pick most likely next token
- **Beam Search**
  keep a fixed number (beam-width) of candidates per step
- **Top-k Sampling**
  sample from distribution of top-k probabilities per step
- **Nucleus Sampling (top-p)**
  sample from top-slice of probability mass per step