UMass · CS685 | Advanced Natural Language Processing (2020)

CS685 (2020)· 课程资料包 @ShowMeAI

视频
中英双语字幕

课件
一键打包下载

笔记
官方笔记翻译

代码
作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1BL411t7RV

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/umass-cs685

NLP 语言模型 问答系统 文本生成 BERT 语义解析 GPT-3 知识推理 模型蒸馏 迁移学习 transformer 注意力机制

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？** 回复 [添砖加瓦]

Transformers and sequence-to-sequence learning

CS 685, Fall 2020

Mohit Iyyer
College of Information and Computer Sciences
University of Massachusetts Amherst

some slides from Emma Strubell

sequence-to-sequence learning

Used when inputs and outputs are both sequences of words (e.g., machine translation, summarization)

- we'll use French (*f*) to English (*e*) as a running example
- **goal**: given French sentence *f* with tokens f_1, f_2, … f_n produce English translation *e* with tokens e_1, e_2, … e_m
- **real goal**: compute $\arg\max_{e} p(e \mid f)$

This is an instance of *conditional language modeling*

$$
\begin{aligned}
p(e \mid f) &= p(e_1, e_2, \ldots, e_m \mid f) \\
&= p(e_1 \mid f) \cdot p(e_2 \mid e_1, f) \cdot p(e_3 \mid e_2, e_1, f) \cdot \ldots \\
&= \prod_{i=1}^{m} p(e_i \mid e_1, \ldots, e_{i-1}, f)
\end{aligned}
$$

Just like we've seen before, except we additionally condition our prediction of the next word on some other input (here, the French sentence)

seq2seq models

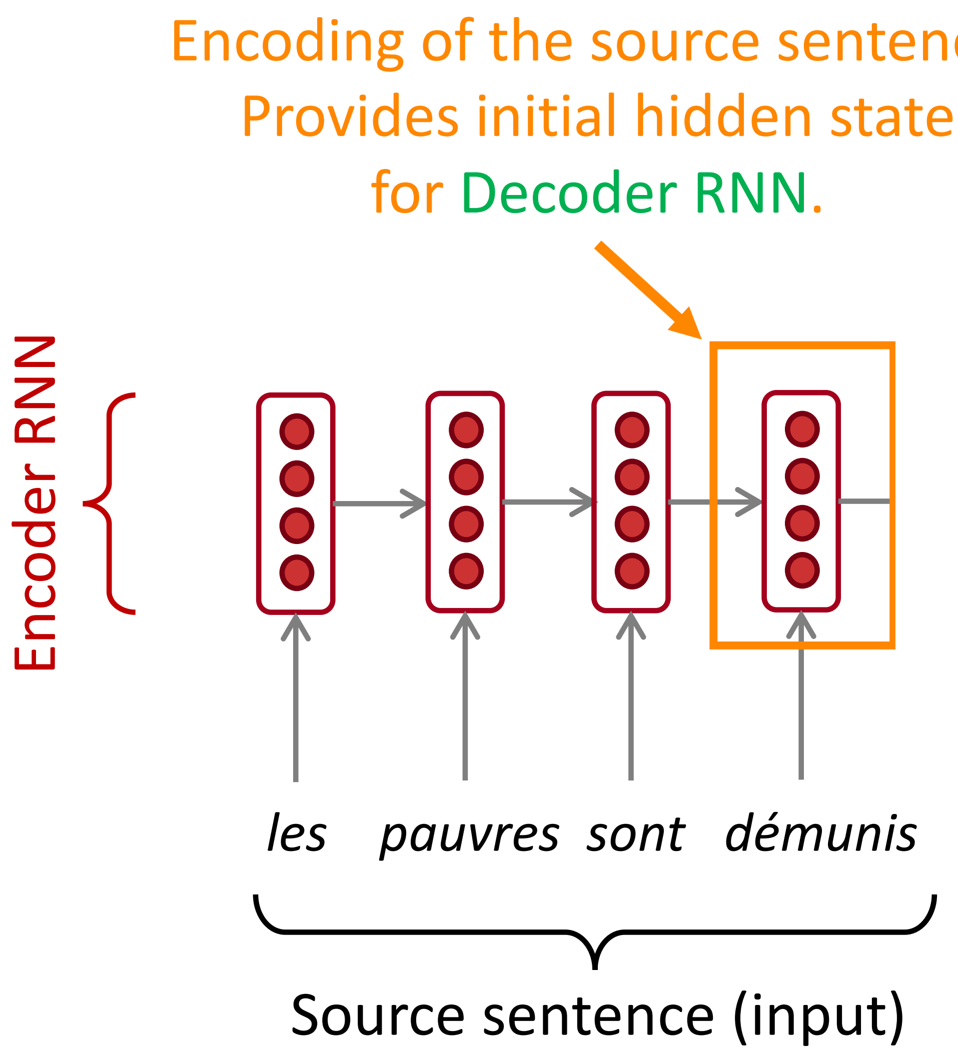
- use two different neural networks to model

$$\prod_{i=1}^{L} p(e_i \,|\, e_1, \ldots, e_{i-1}, f)$$

- first we have the *encoder*, which encodes the French sentence *f*
- then, we have the *decoder,* which produces the English sentence *e*

Neural Machine Translation (NMT)

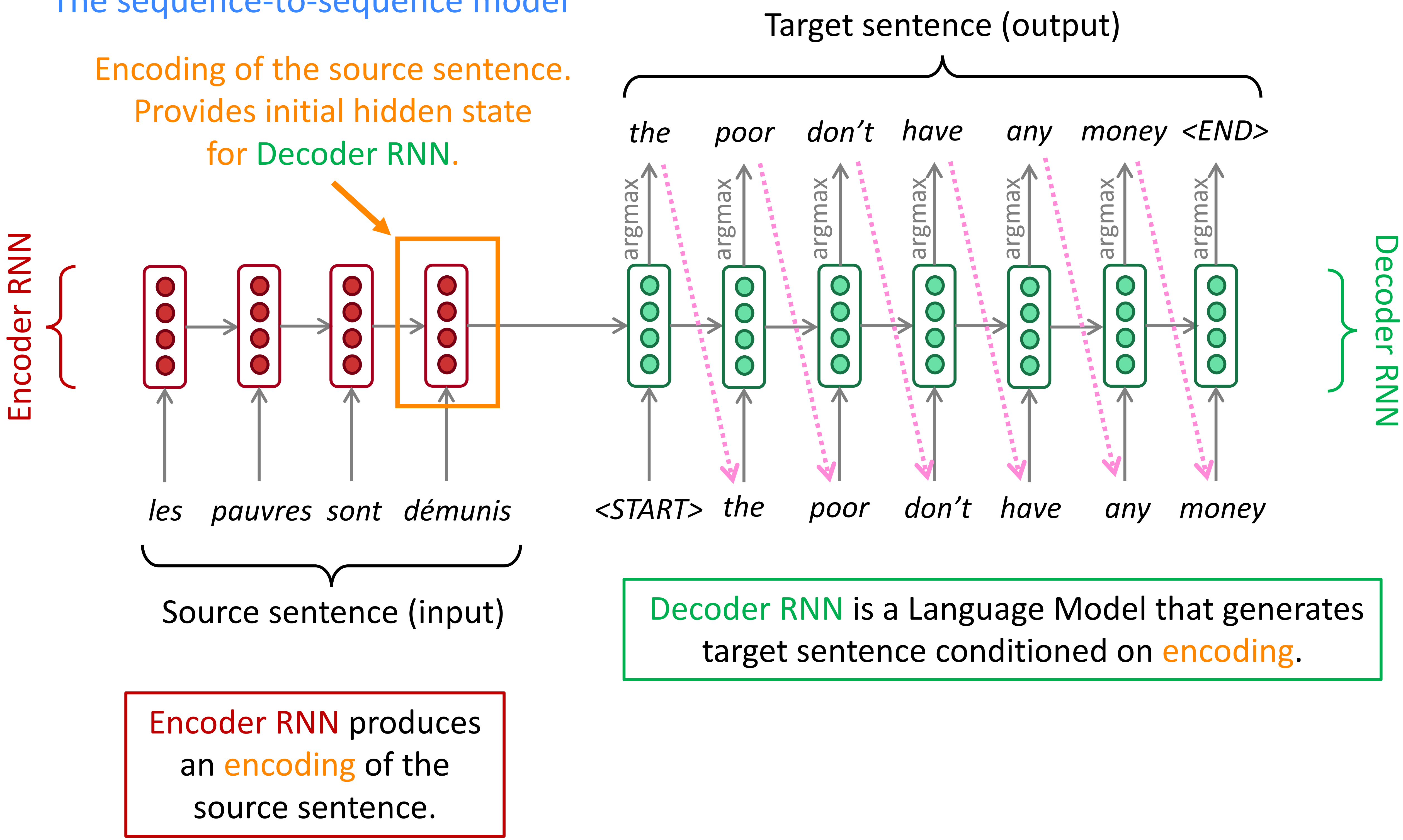
The sequence-to-sequence model



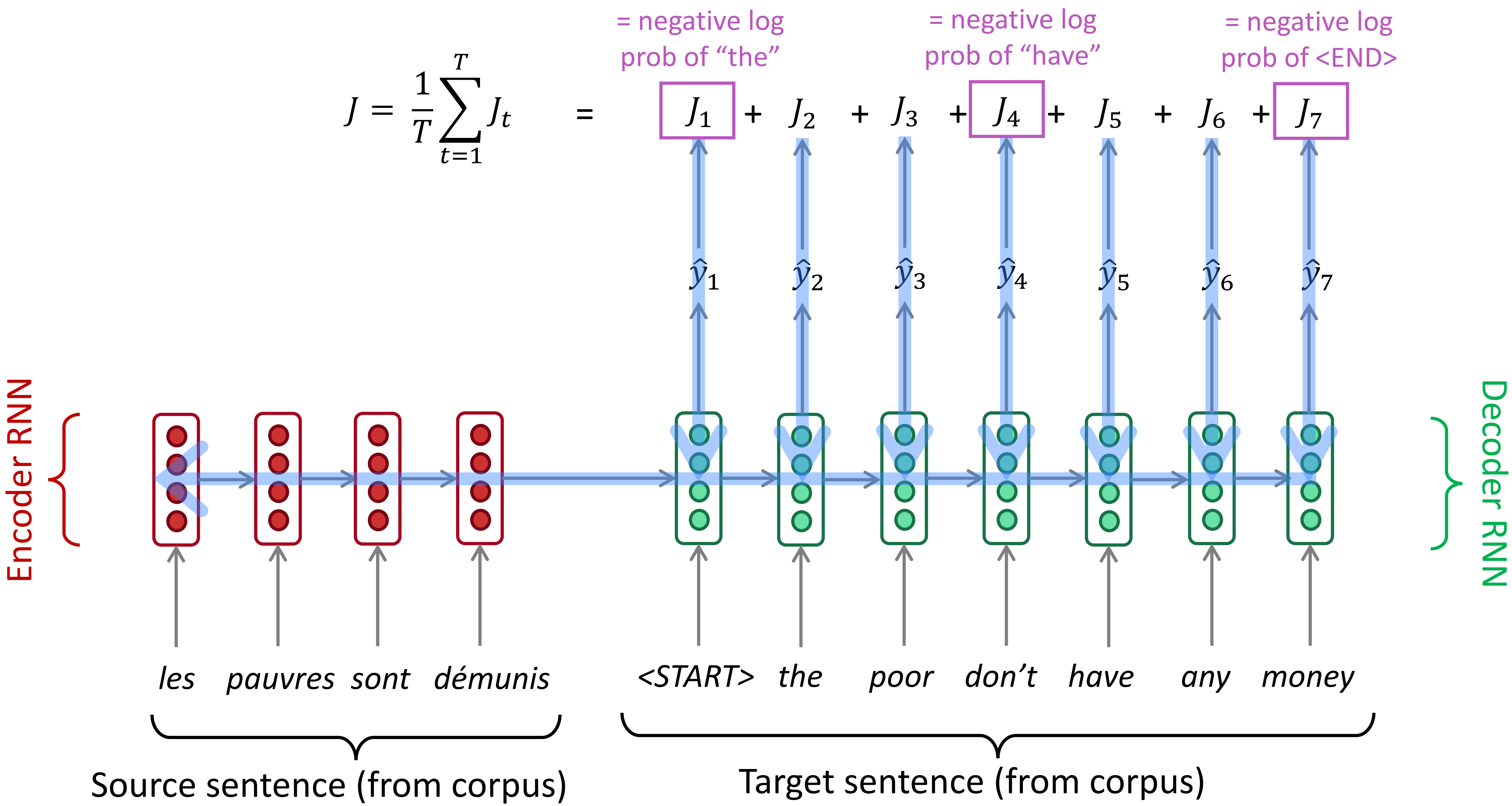
Encoder RNN produces an encoding of the source sentence.

Neural Machine Translation (NMT)

The sequence-to-sequence model



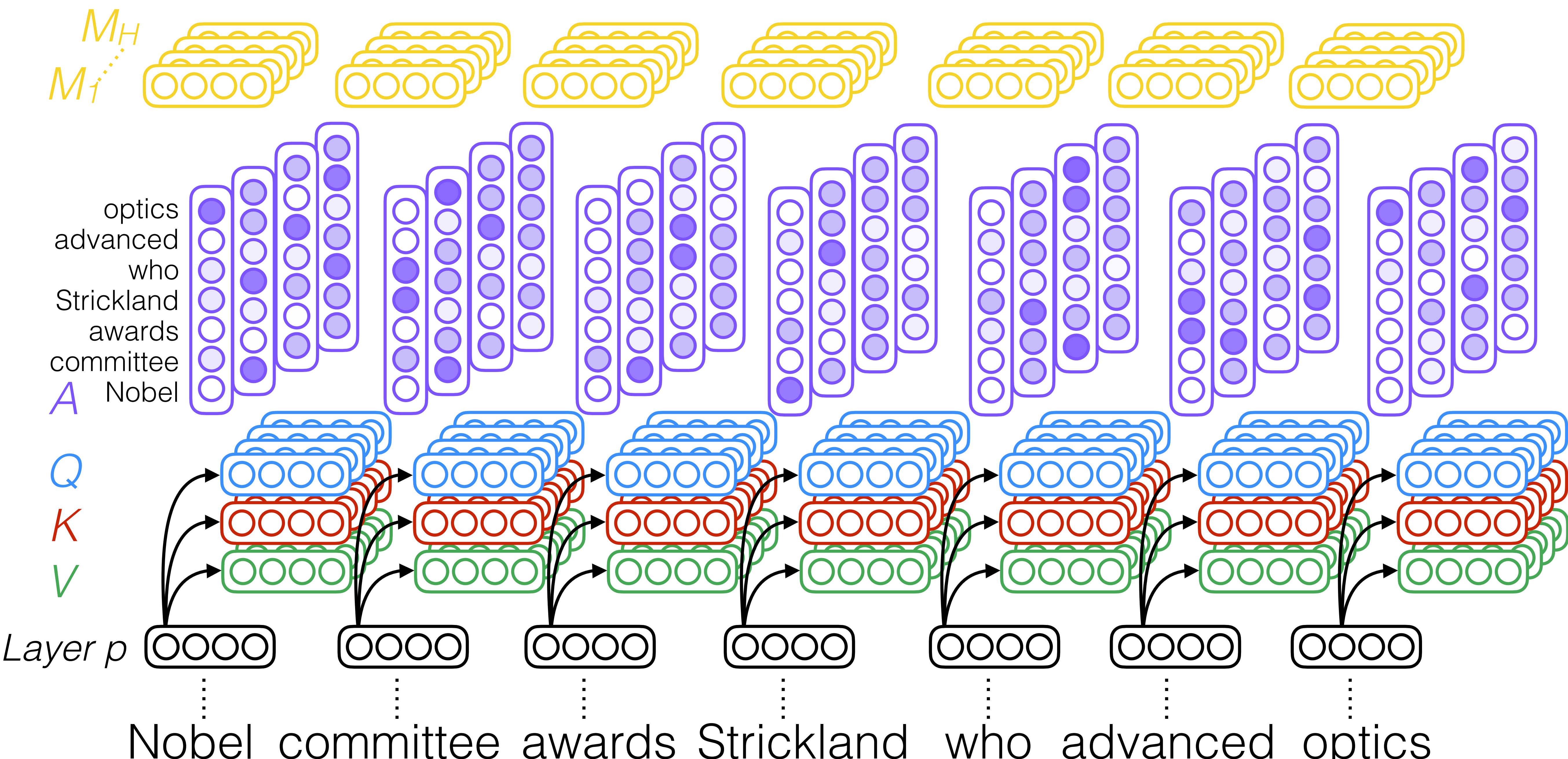
Training a Neural Machine Translation system



We’ll talk much more about machine translation / other seq2seq problems later… but for now, let’s go back to the Transformer

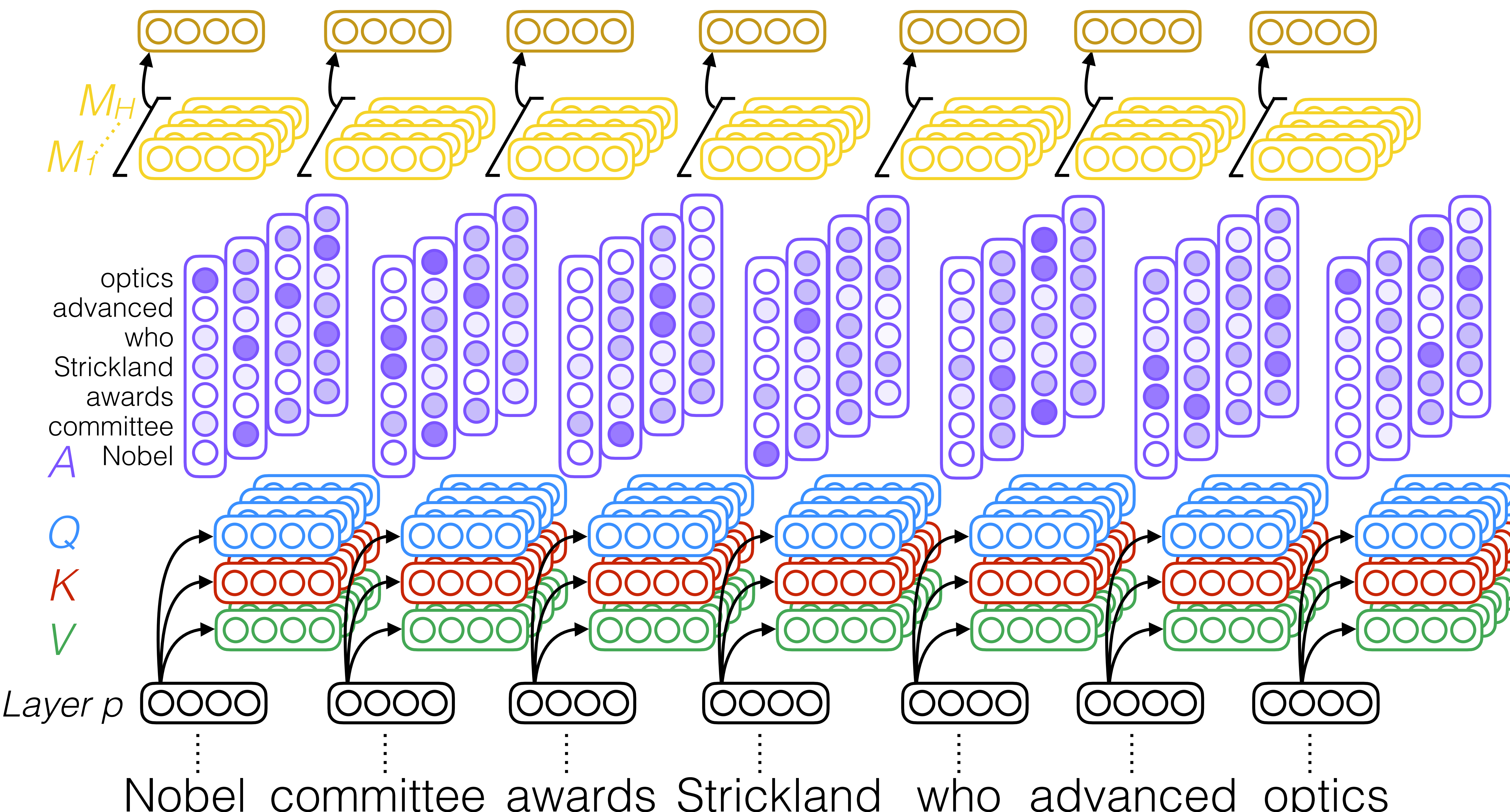
By Emma Strubell

Multi-head self-attention



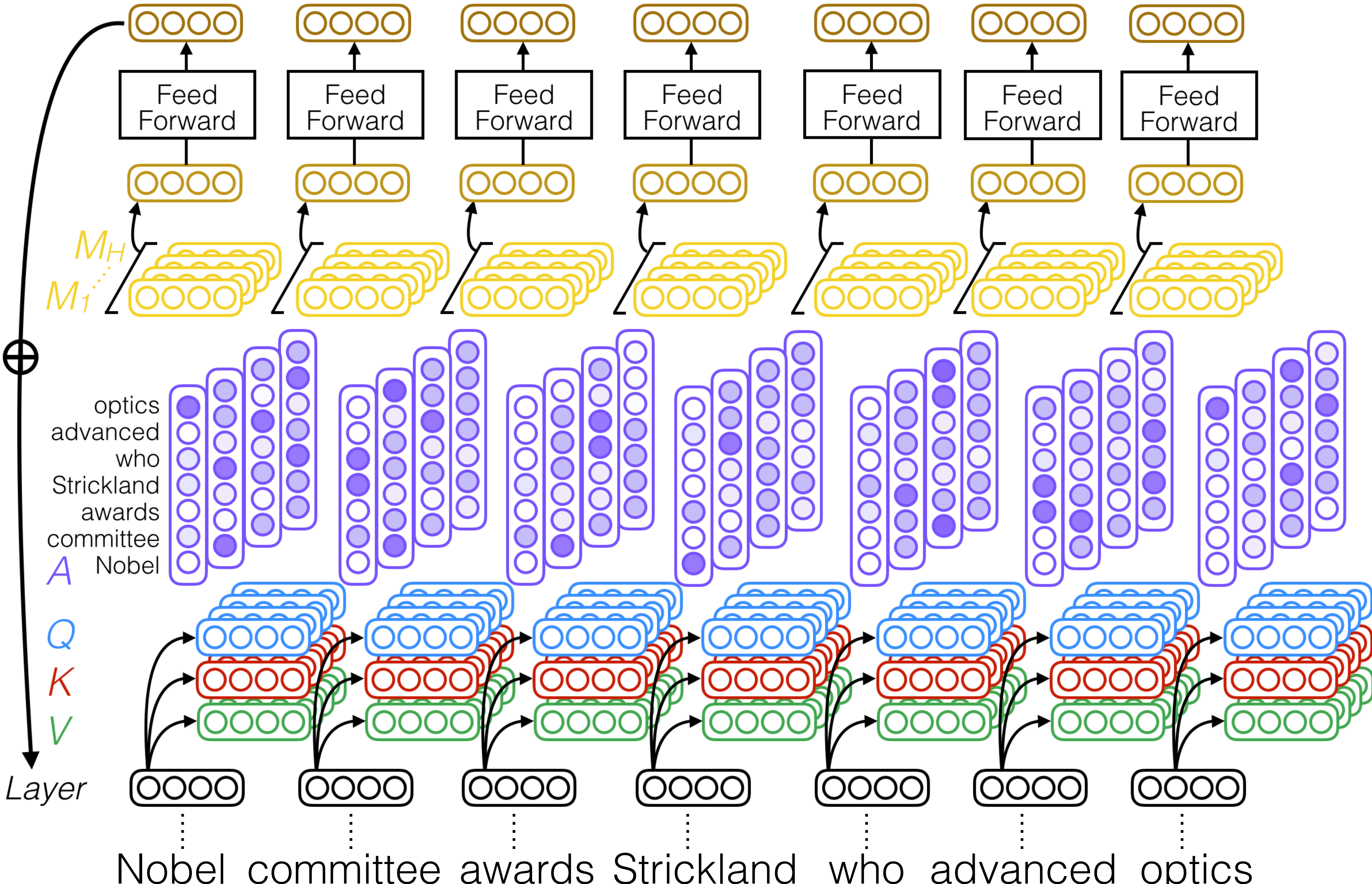
By Emma Strubell

Multi-head self-attention



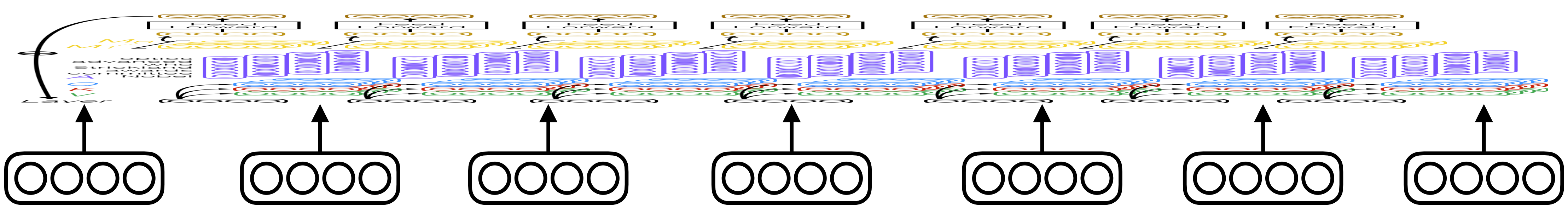
By Emma Strubell

Multi-head self-attention



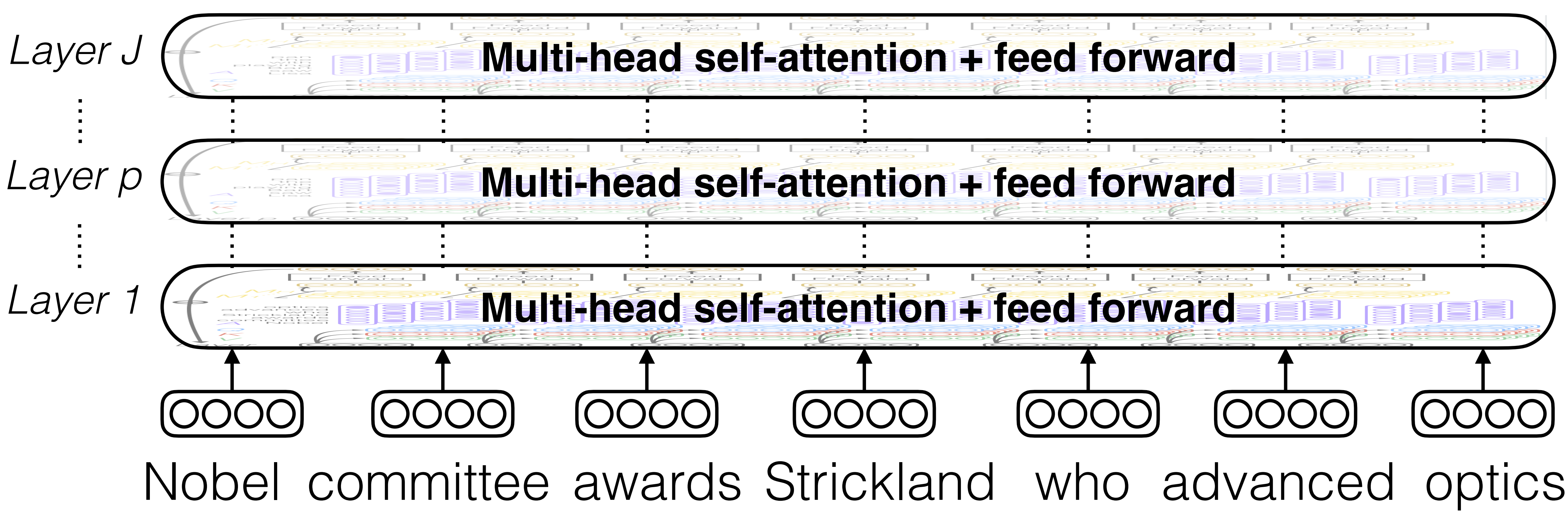
By Emma Strubell

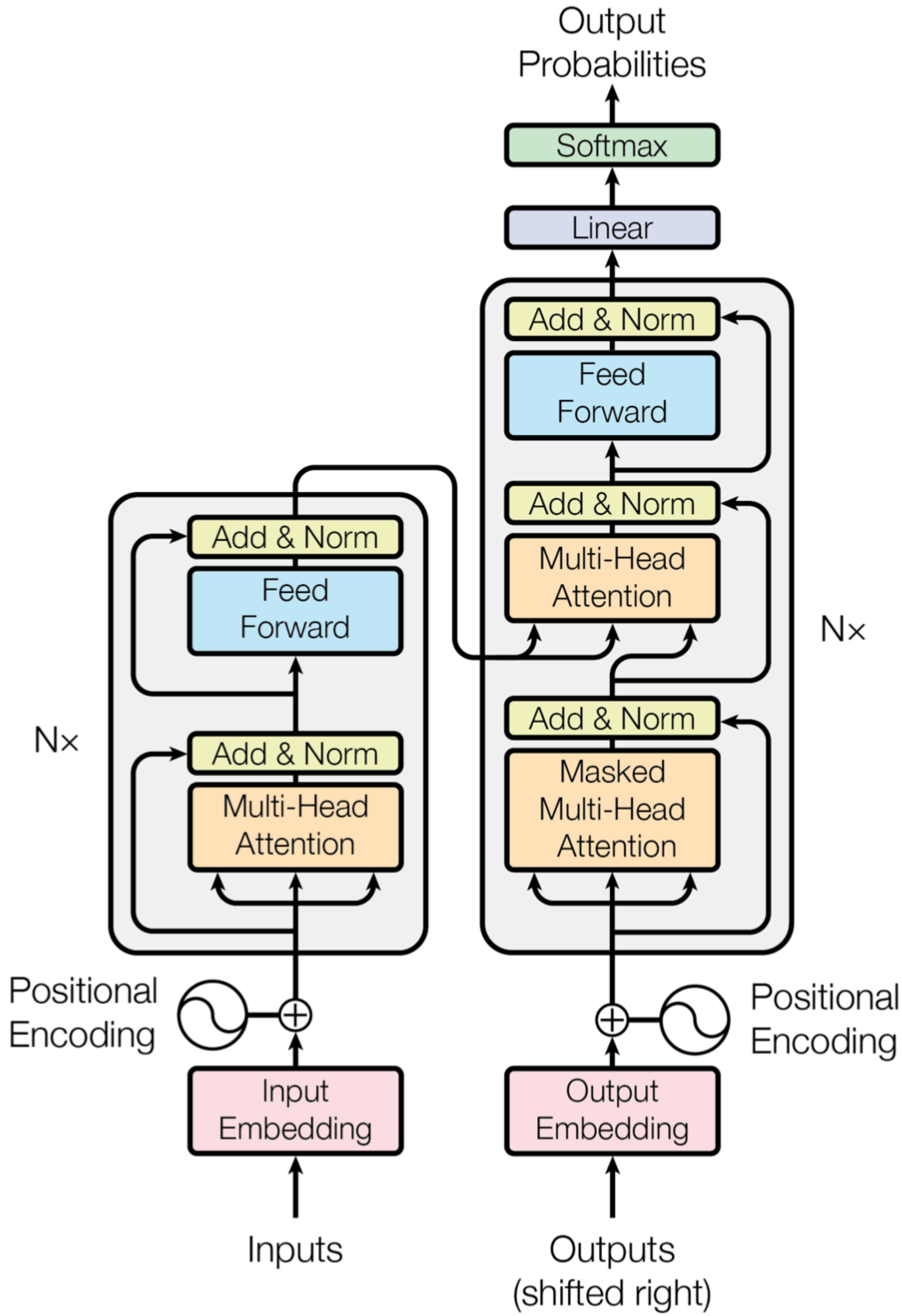
Multi-head self-attention

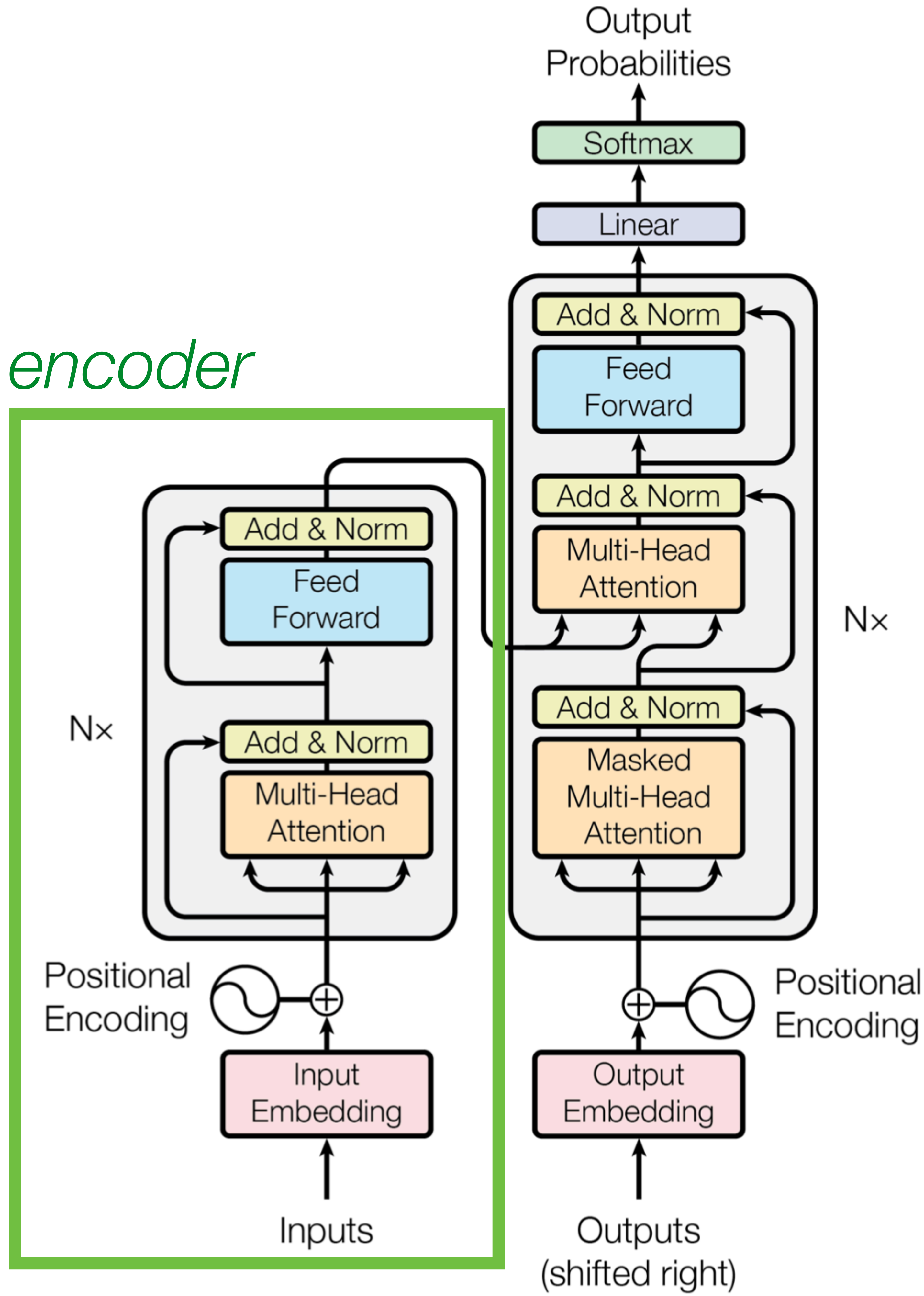


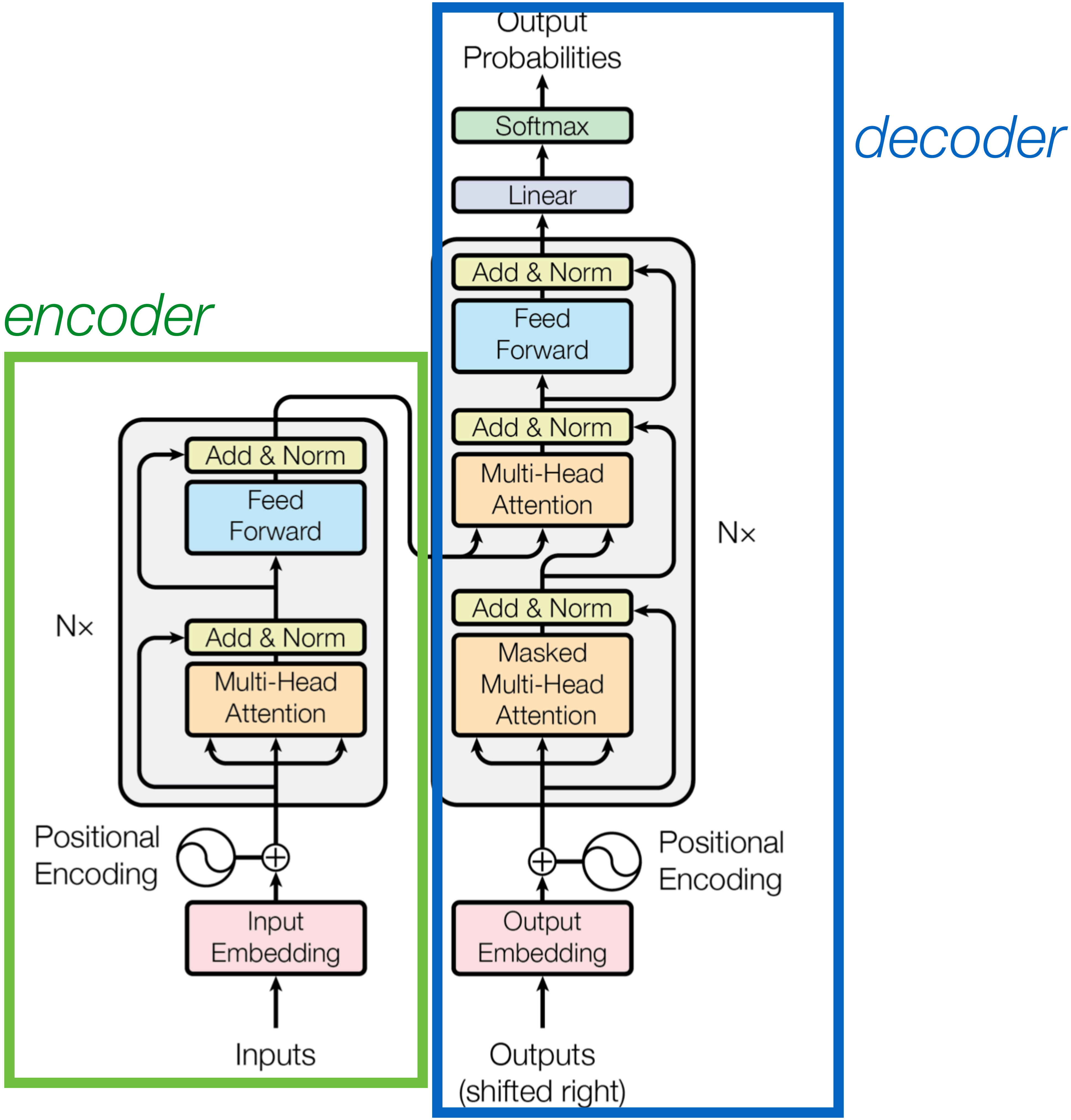
By Emma Strubell

Multi-head self-attention

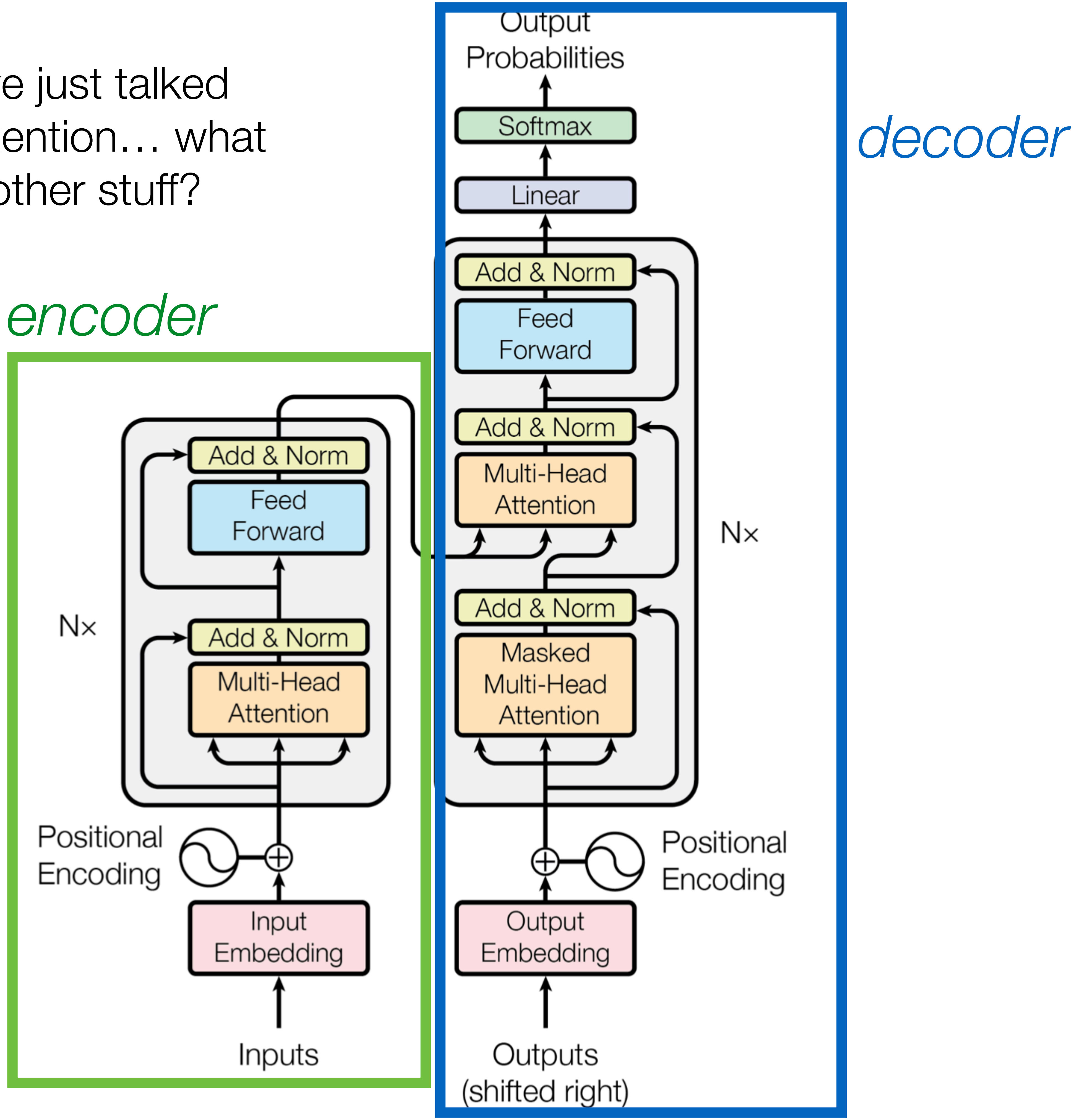


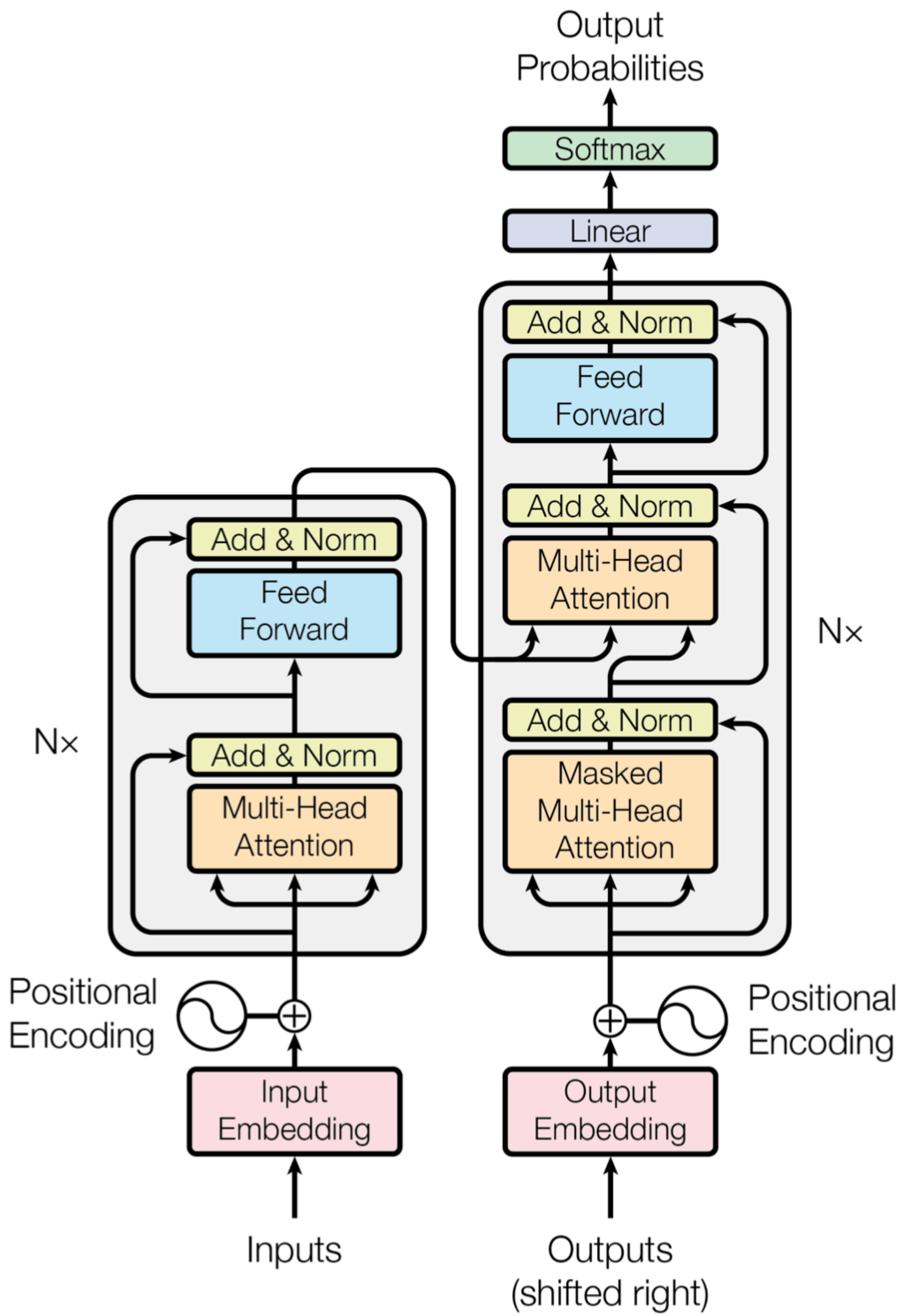

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)


encoder
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Input Embedding
Inputs
Positional Encoding
Output Embedding
Outputs (shifted right)

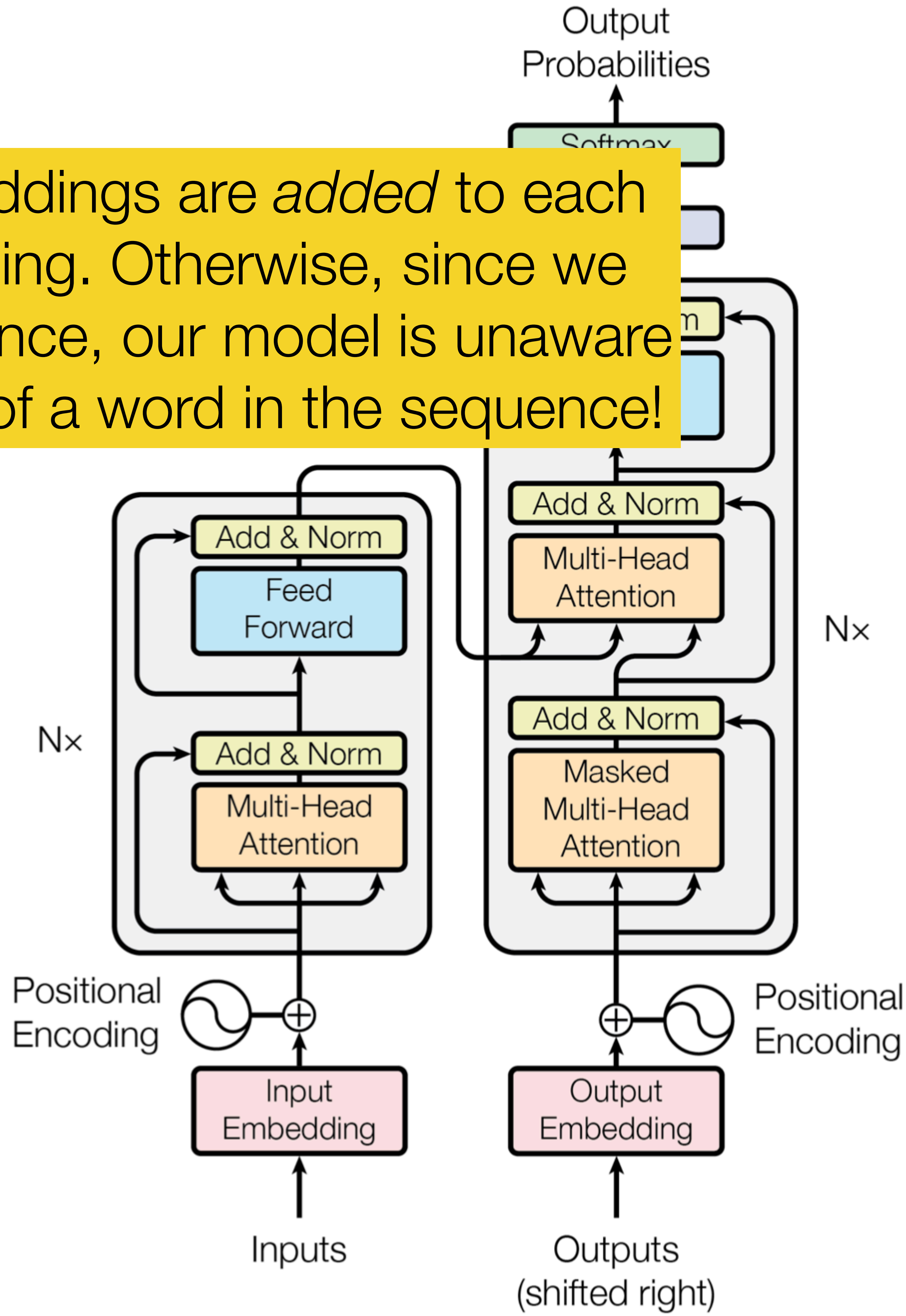

encoder
decoder
Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

So far we've just talked about self-attention… what is all this other stuff?

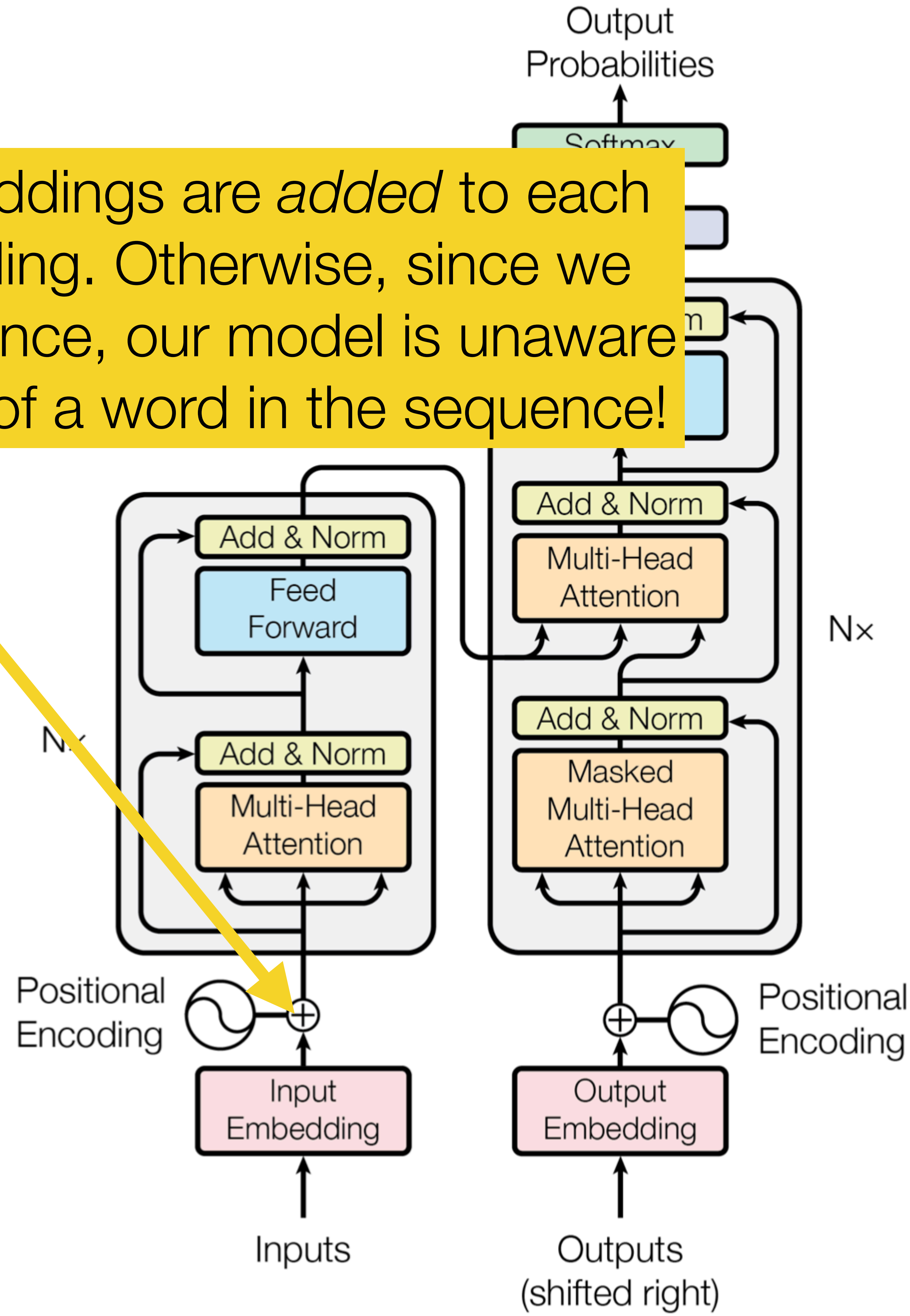


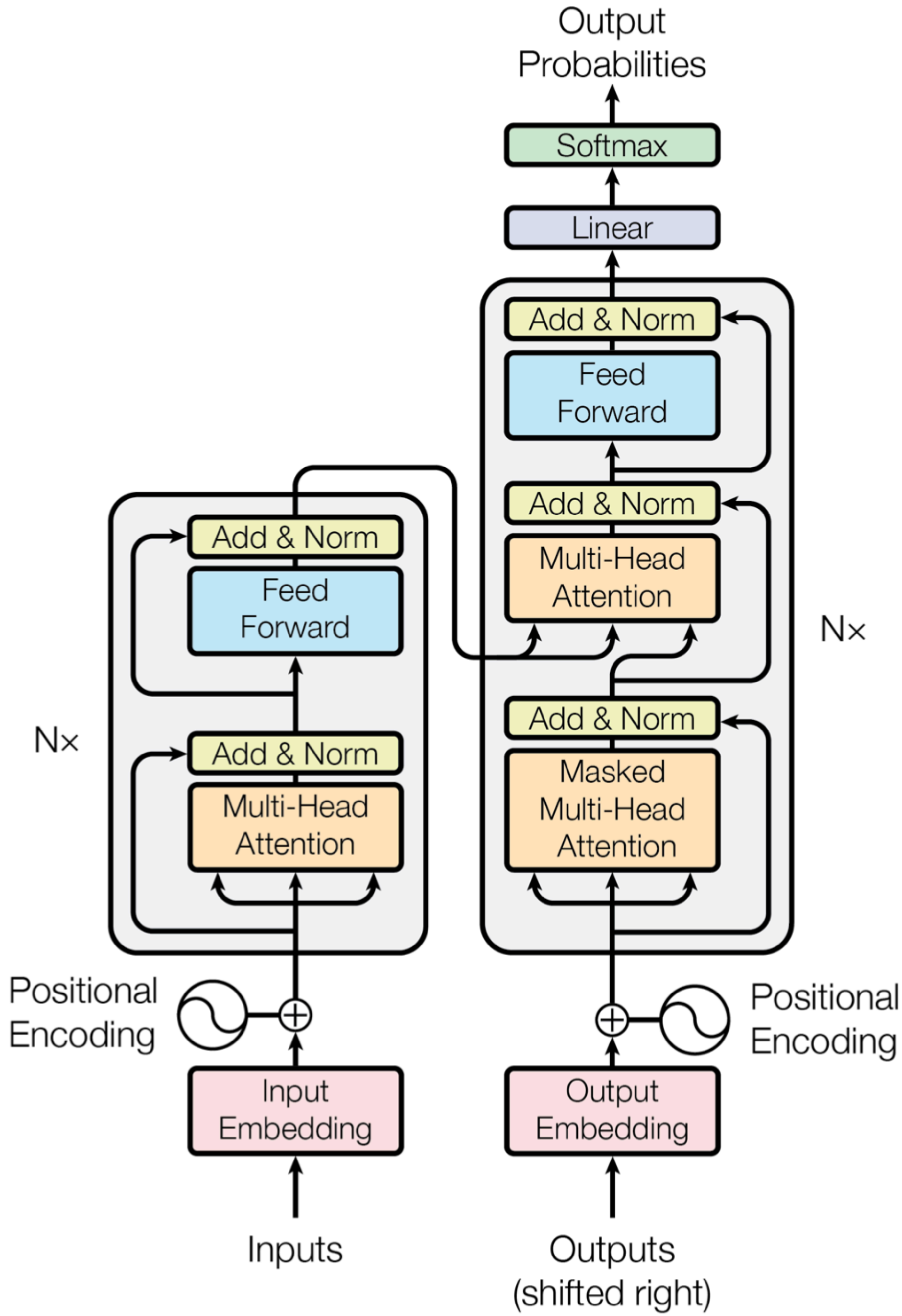

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

Position embeddings are *added* to each word embedding. Otherwise, since we have no recurrence, our model is unaware of the position of a word in the sequence!



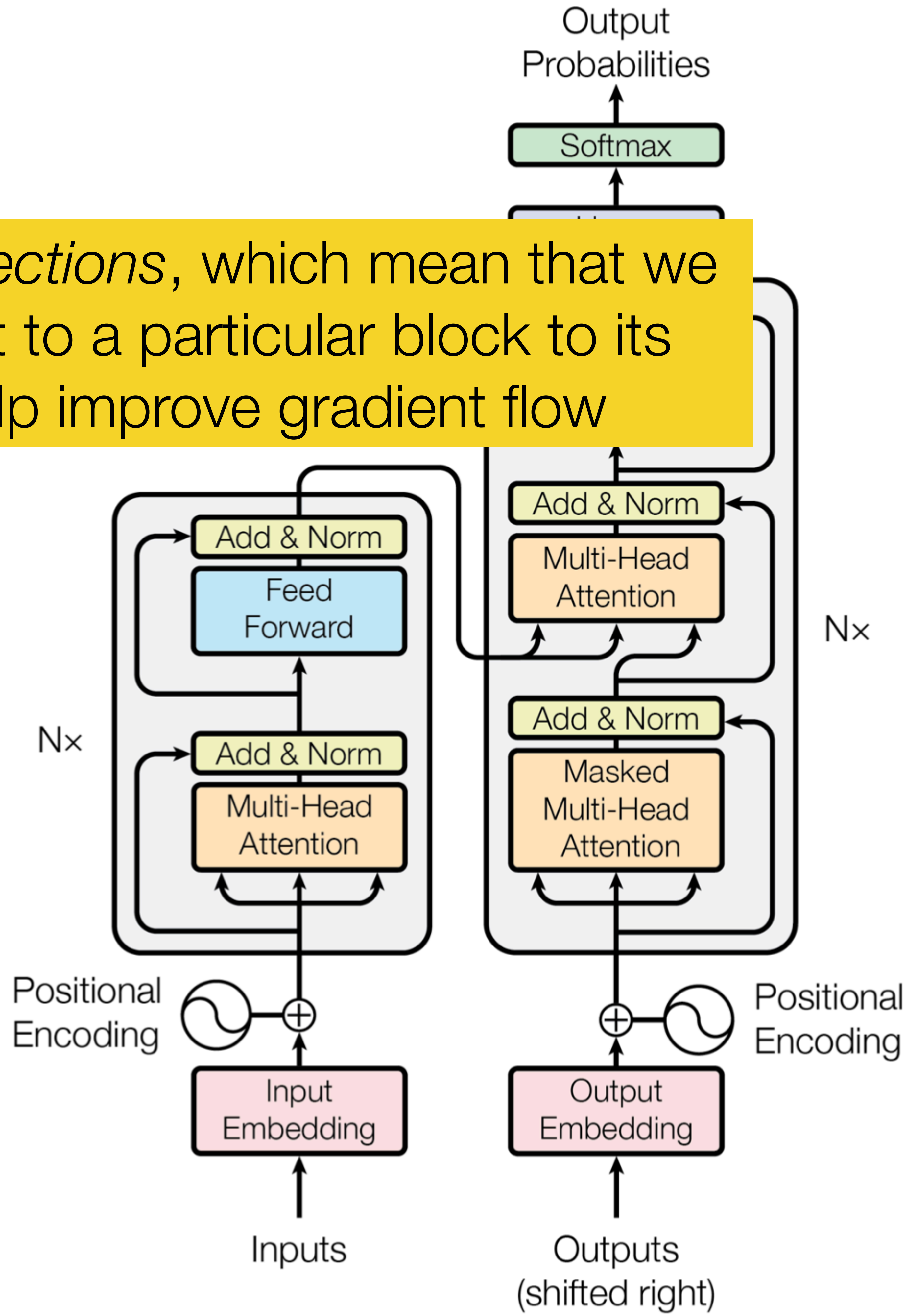
Position embeddings are *added* to each word embedding. Otherwise, since we have no recurrence, our model is unaware of the position of a word in the sequence!



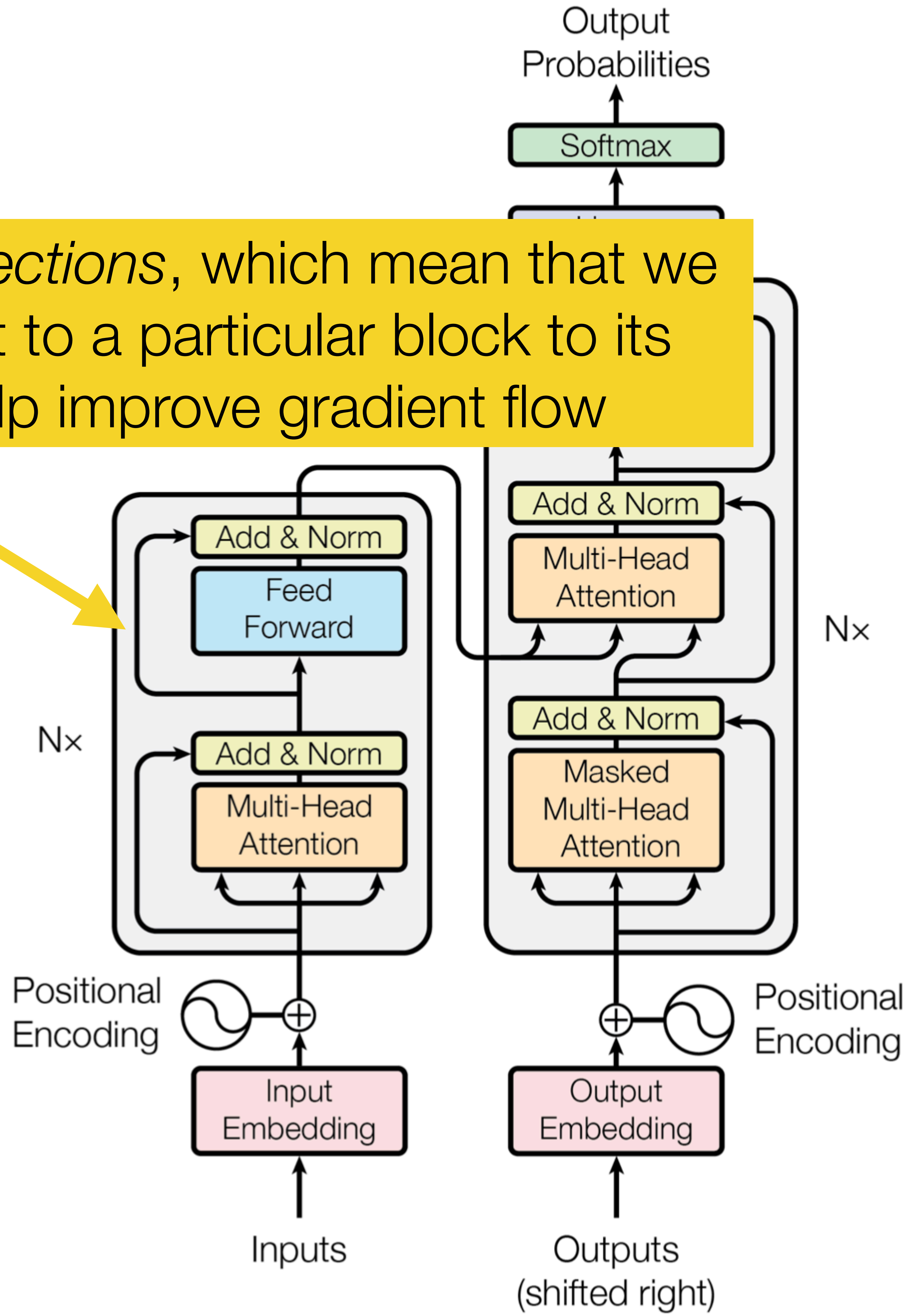


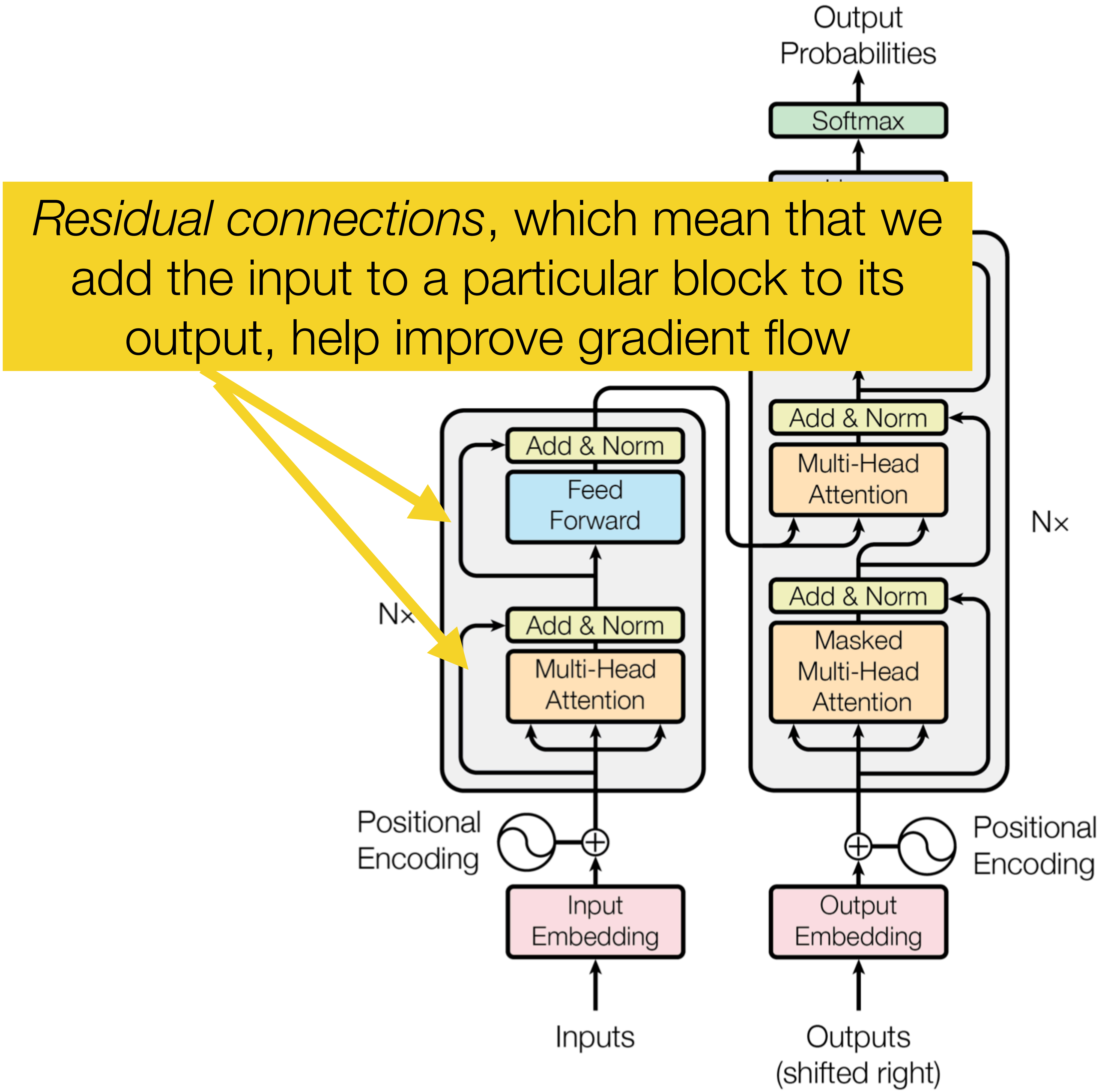
Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

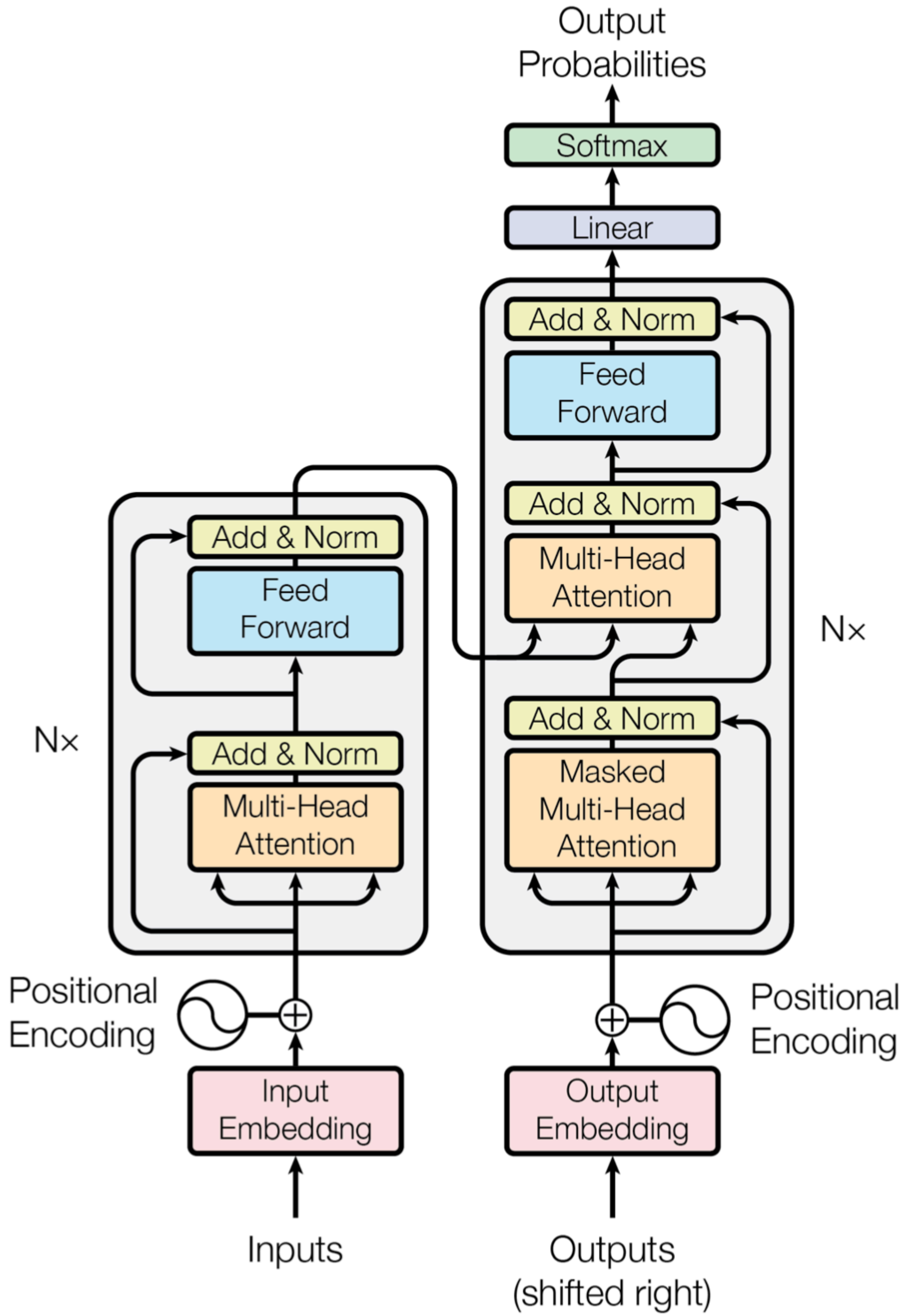
Residual connections, which mean that we add the input to a particular block to its output, help improve gradient flow



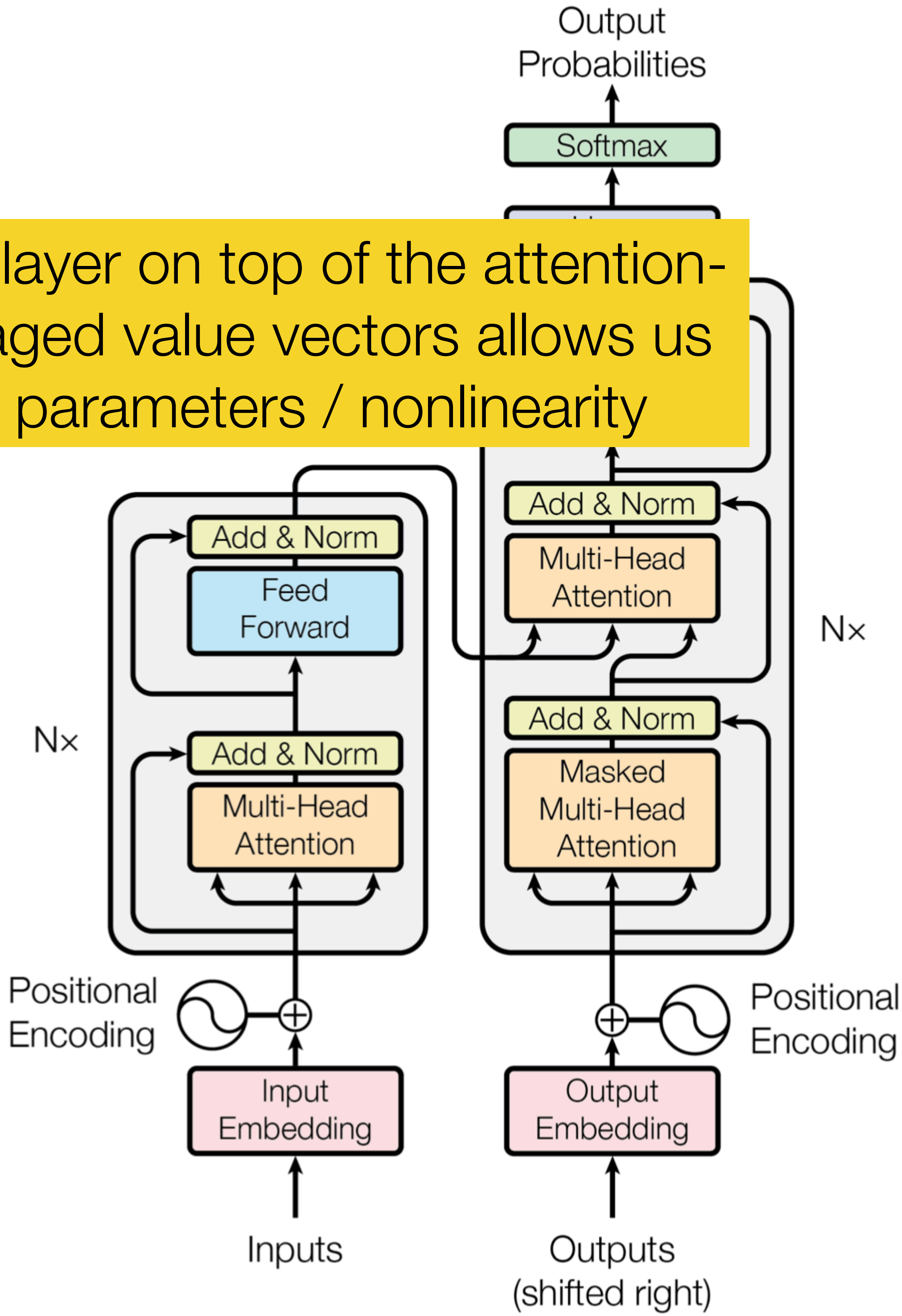
Residual connections, which mean that we add the input to a particular block to its output, help improve gradient flow



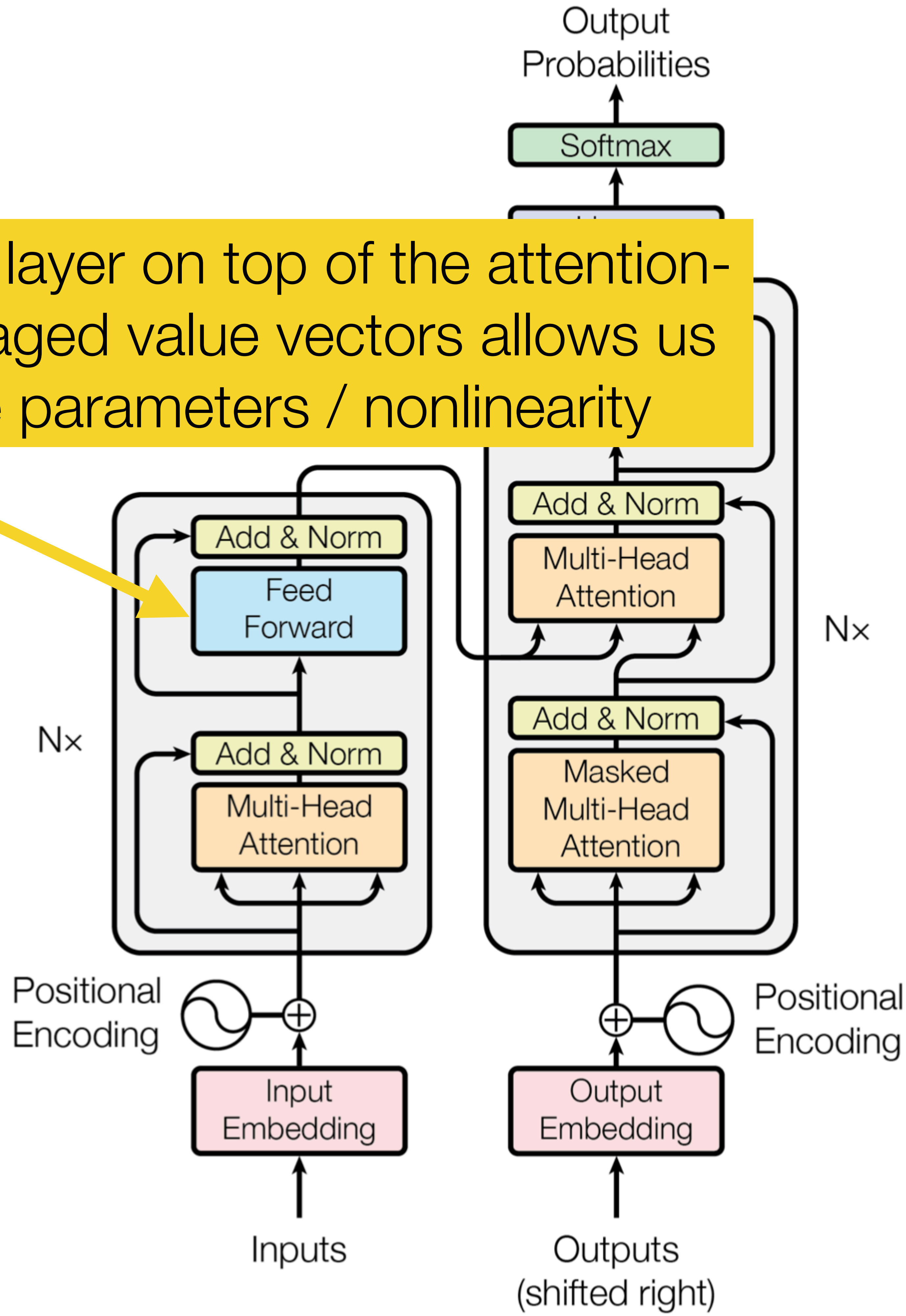

Residual connections, which mean that we add the input to a particular block to its output, help improve gradient flow
Output Probabilities
Softmax
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

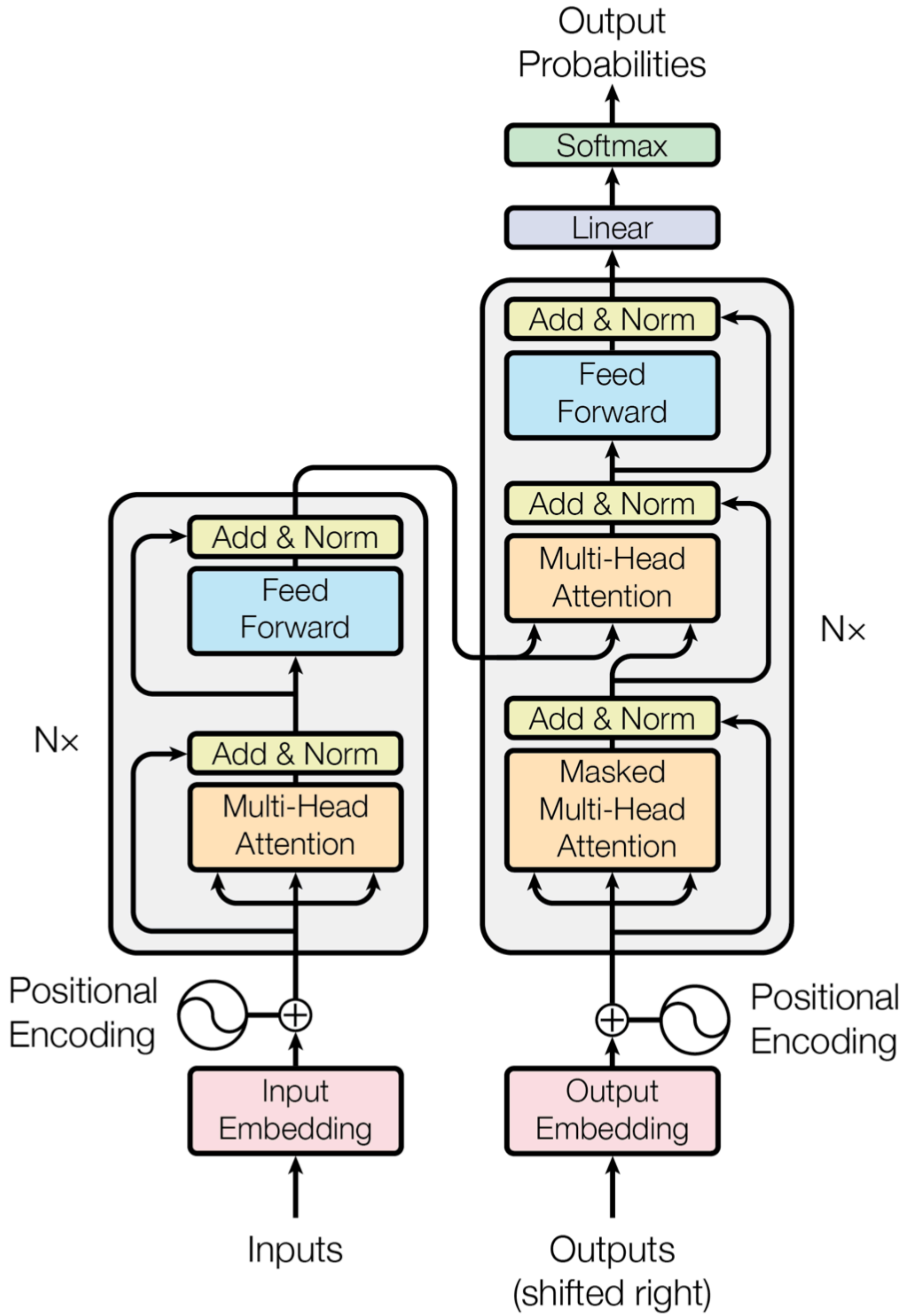

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Output
Embedding
Positional
Encoding
Outputs
(shifted right)

A feed-forward layer on top of the attention-weighted averaged value vectors allows us to add more parameters / nonlinearity

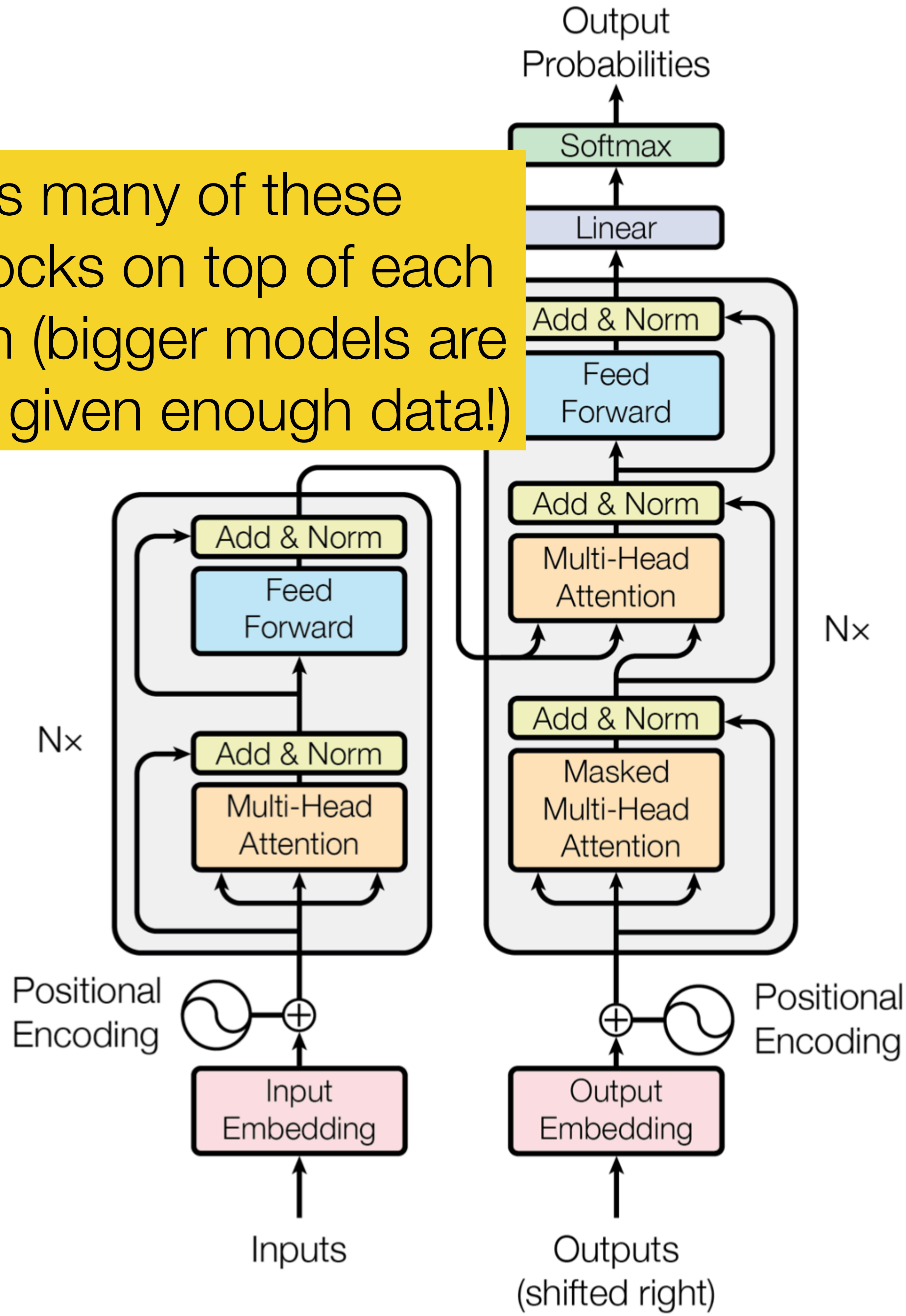


A feed-forward layer on top of the attention-weighted averaged value vectors allows us to add more parameters / nonlinearity

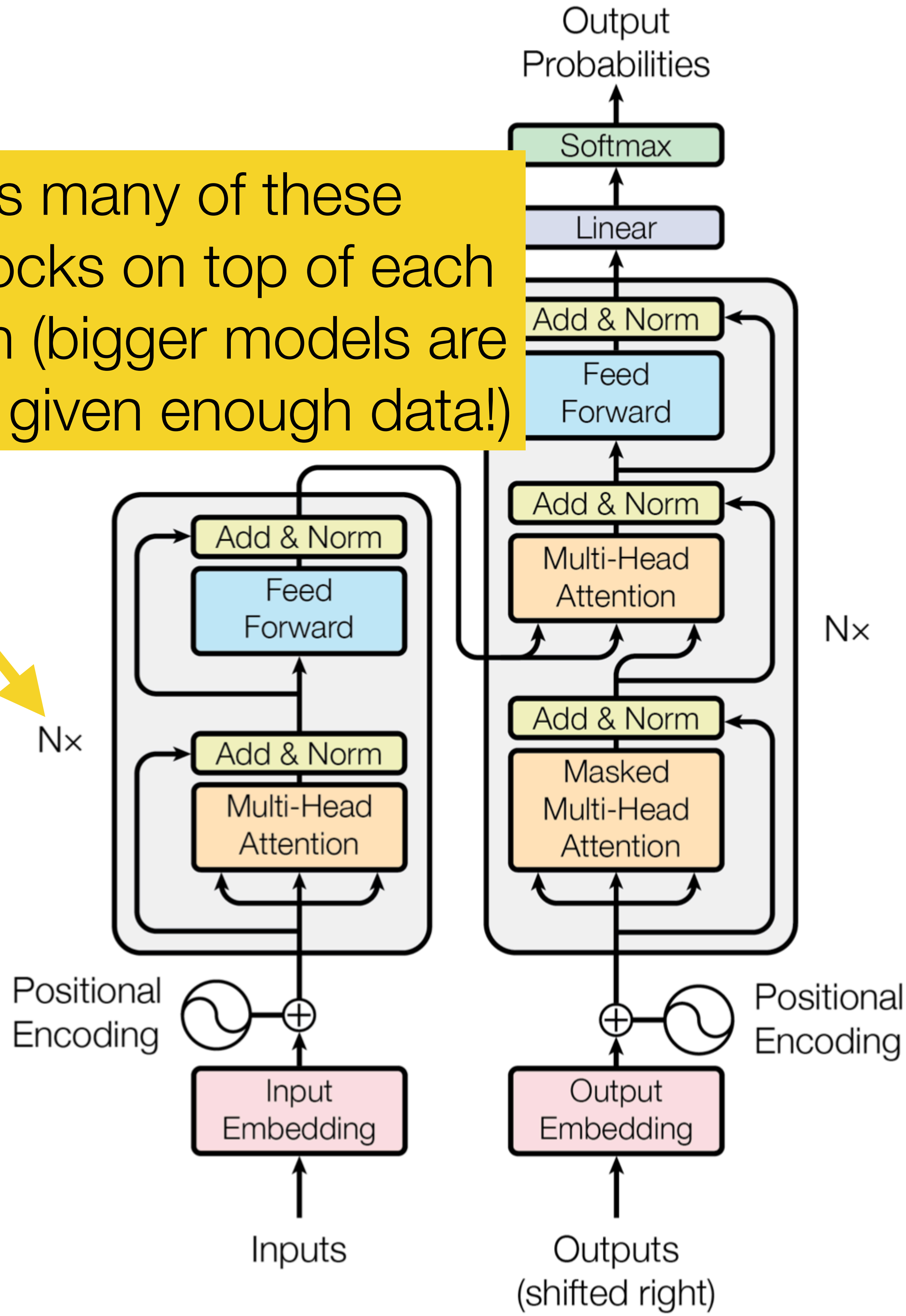


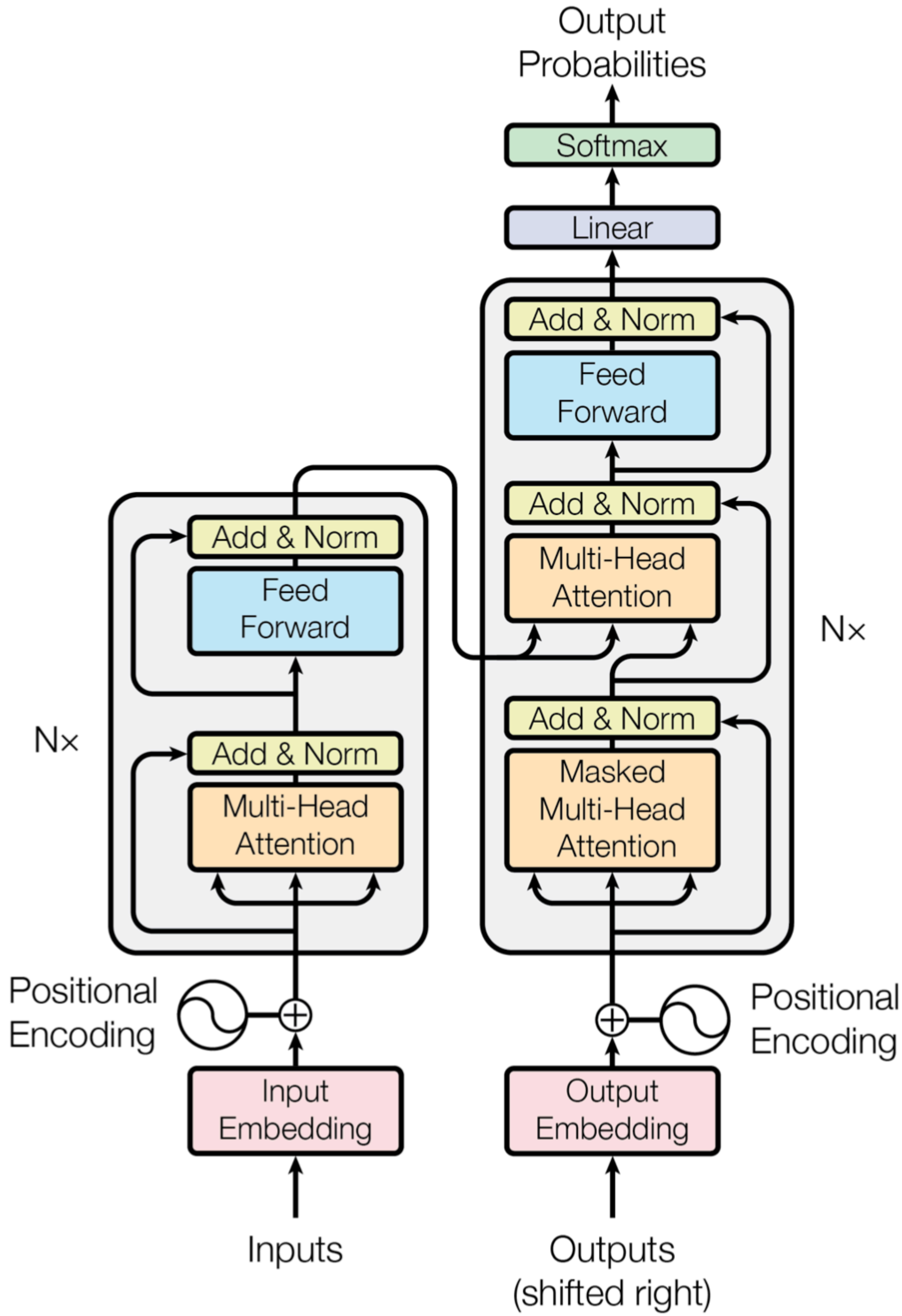

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

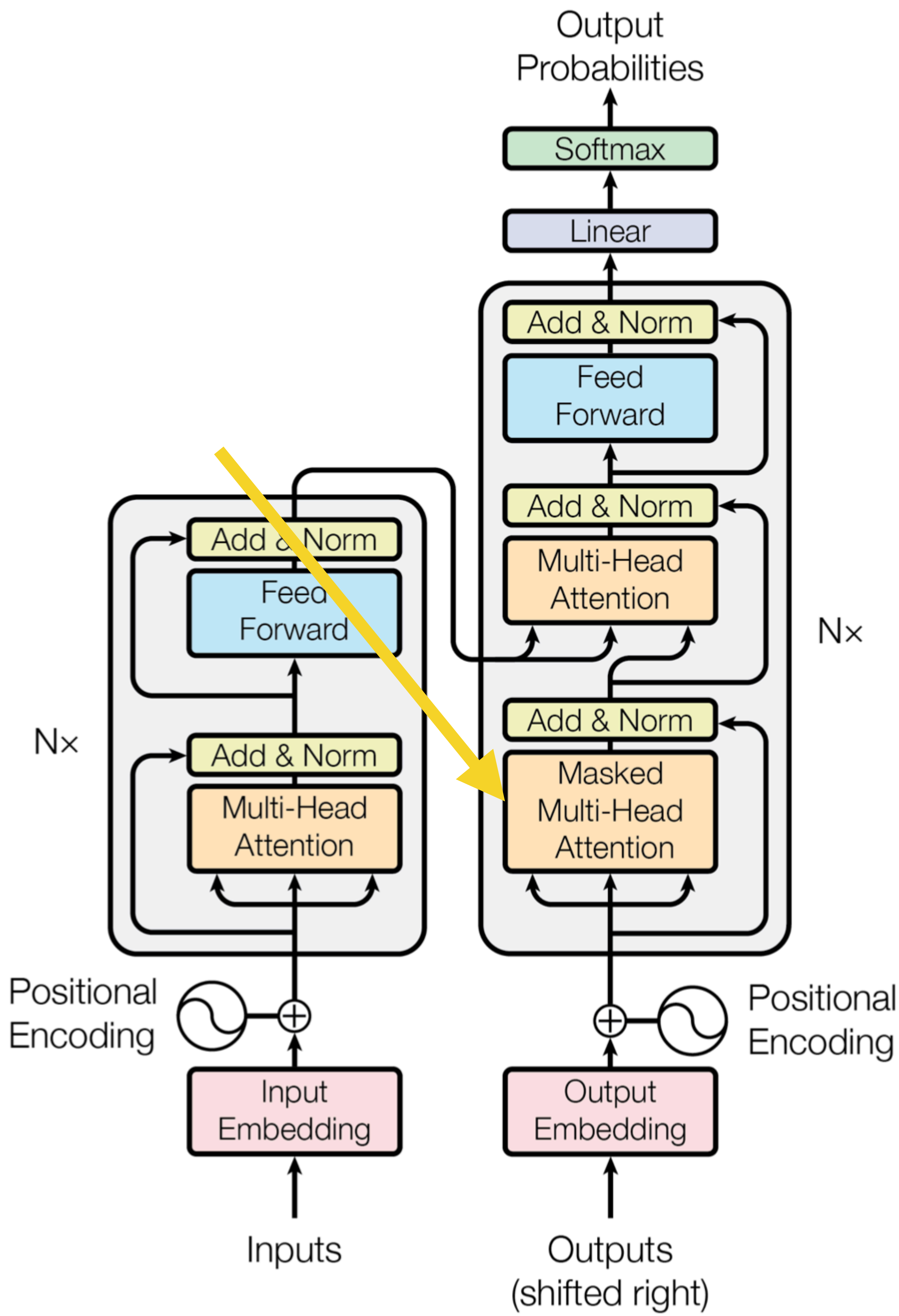
We stack as many of these *Transformer* blocks on top of each other as we can (bigger models are generally better given enough data!)



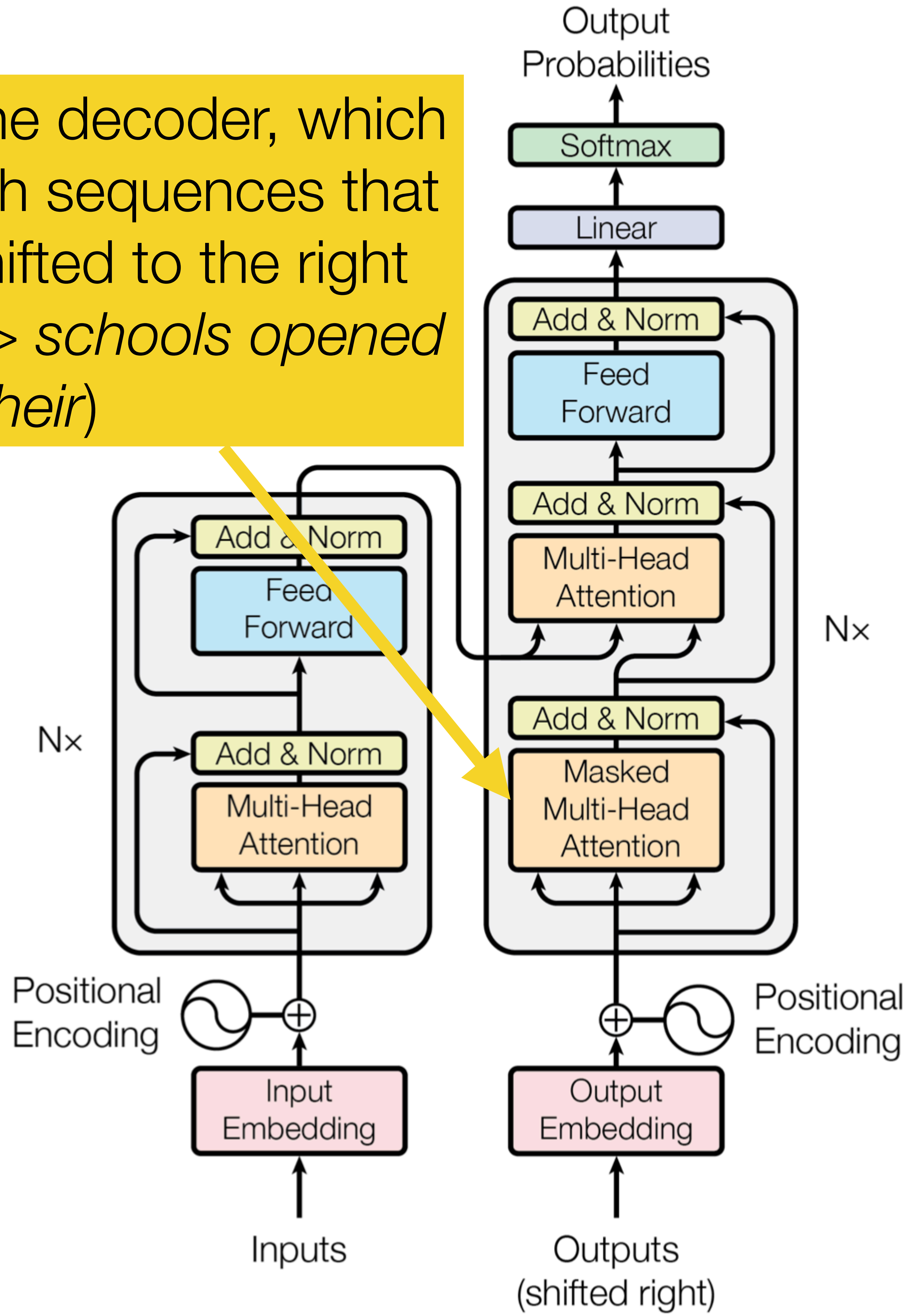
We stack as many of these *Transformer* blocks on top of each other as we can (bigger models are generally better given enough data!)

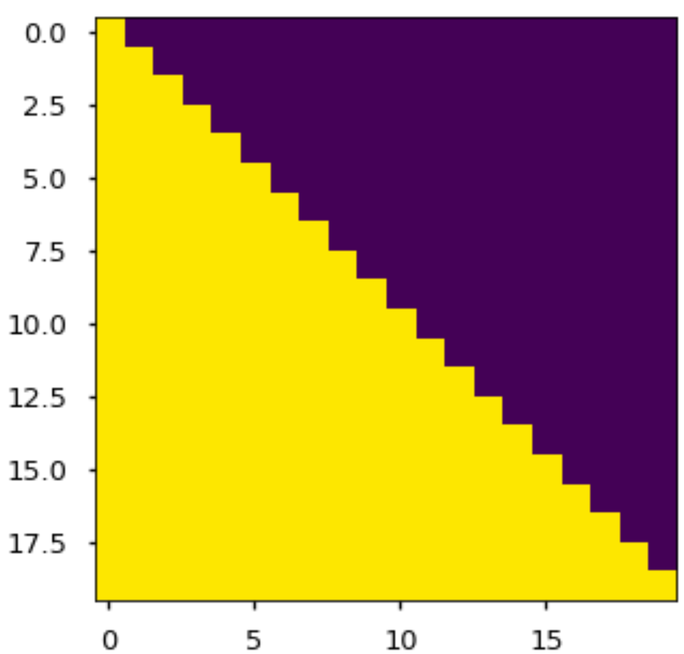


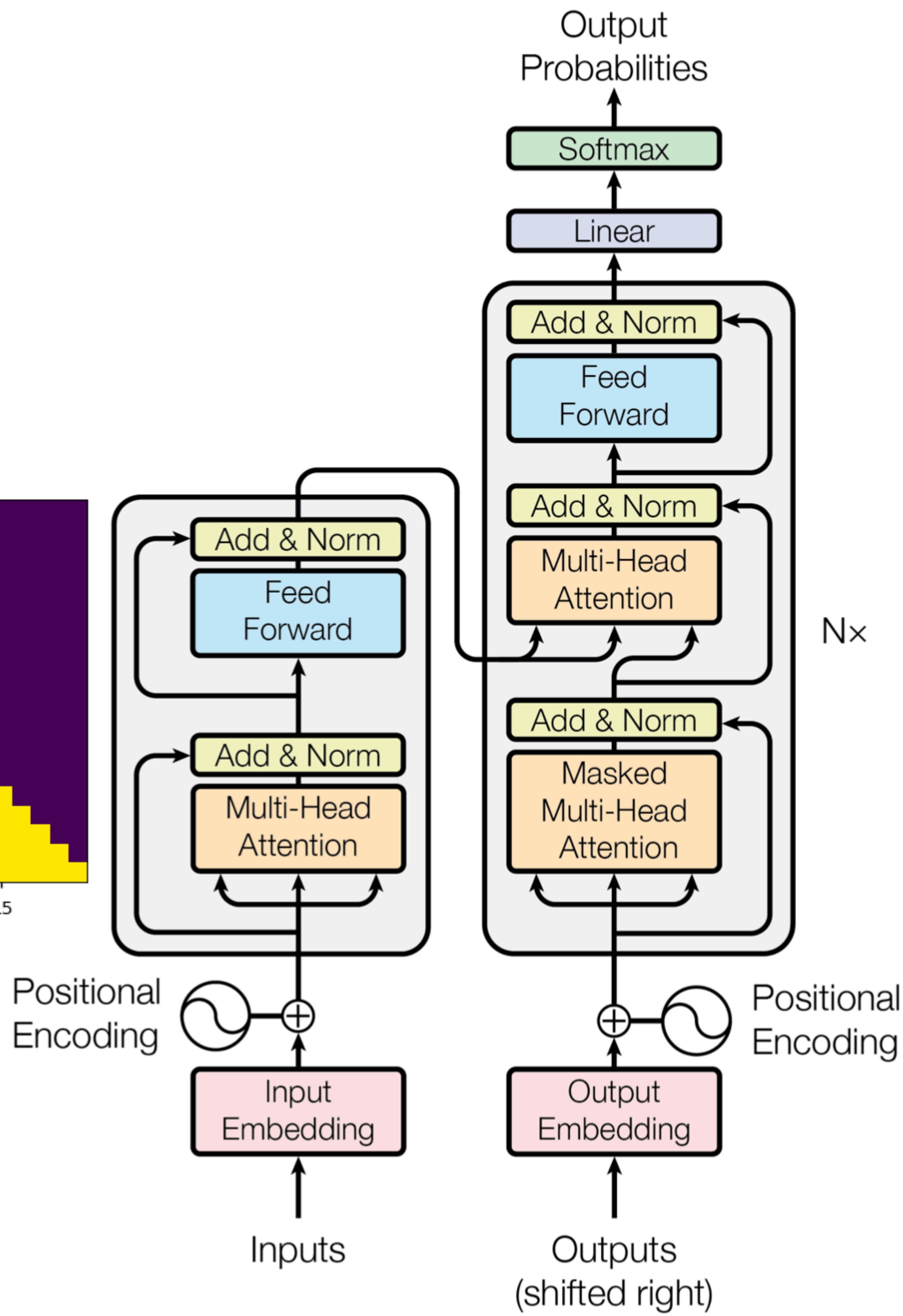

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

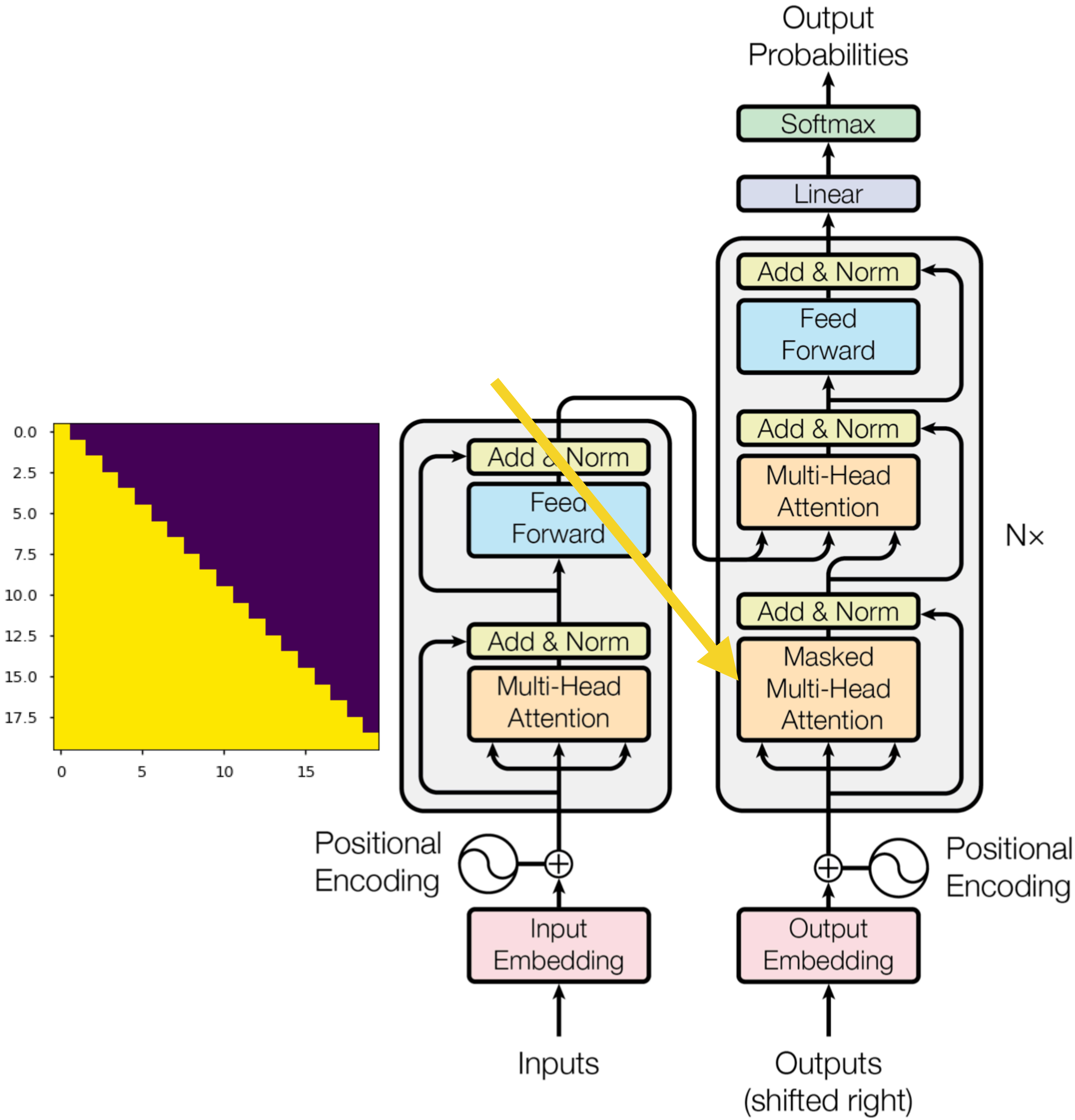

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

Moving onto the decoder, which takes in English sequences that have been shifted to the right (e.g., *<START> schools opened their*)

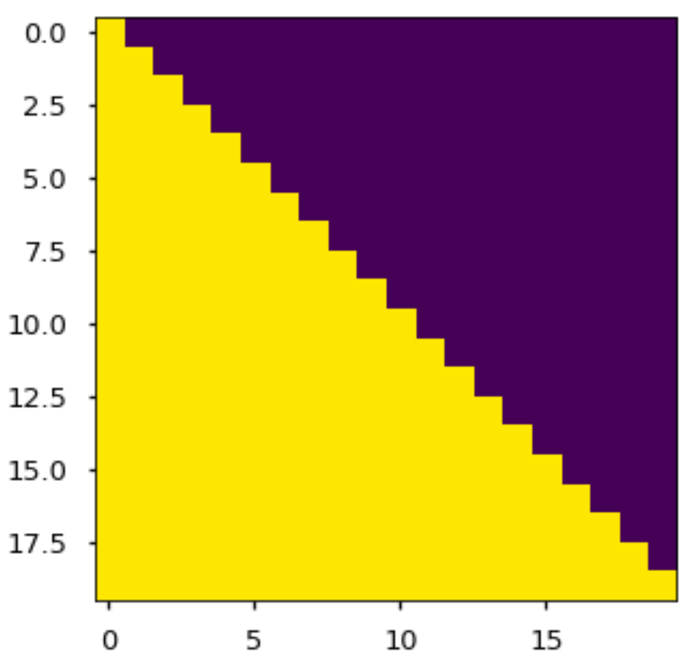


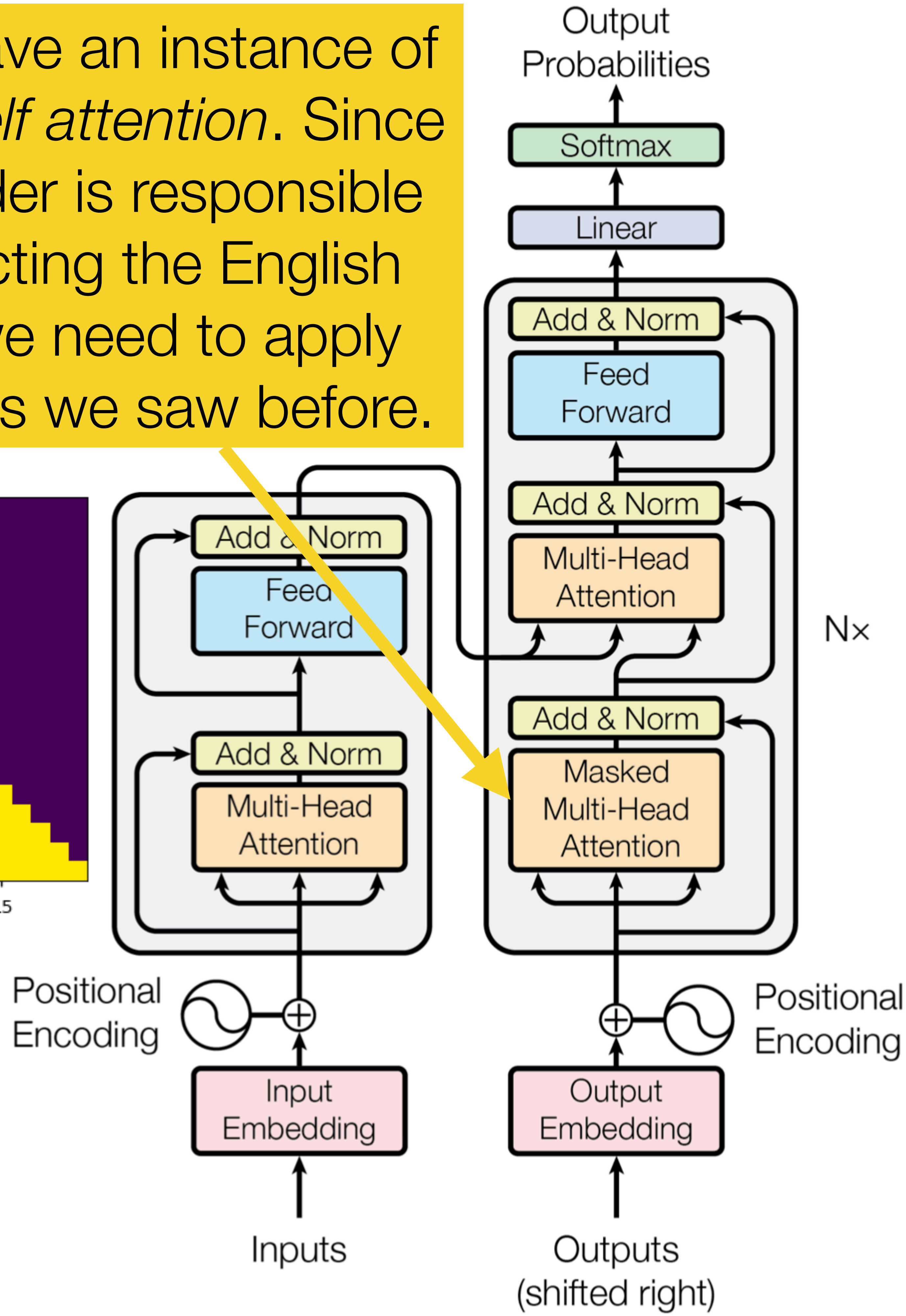

0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

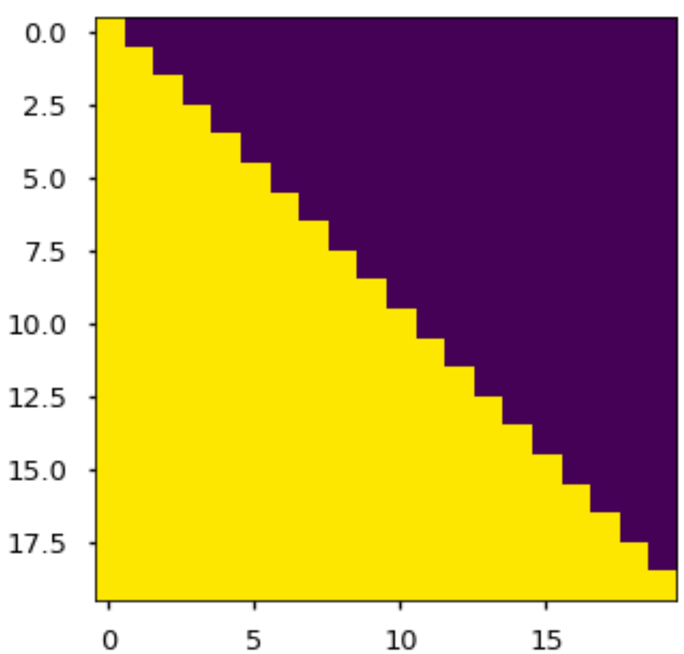

0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
5
10
15
Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

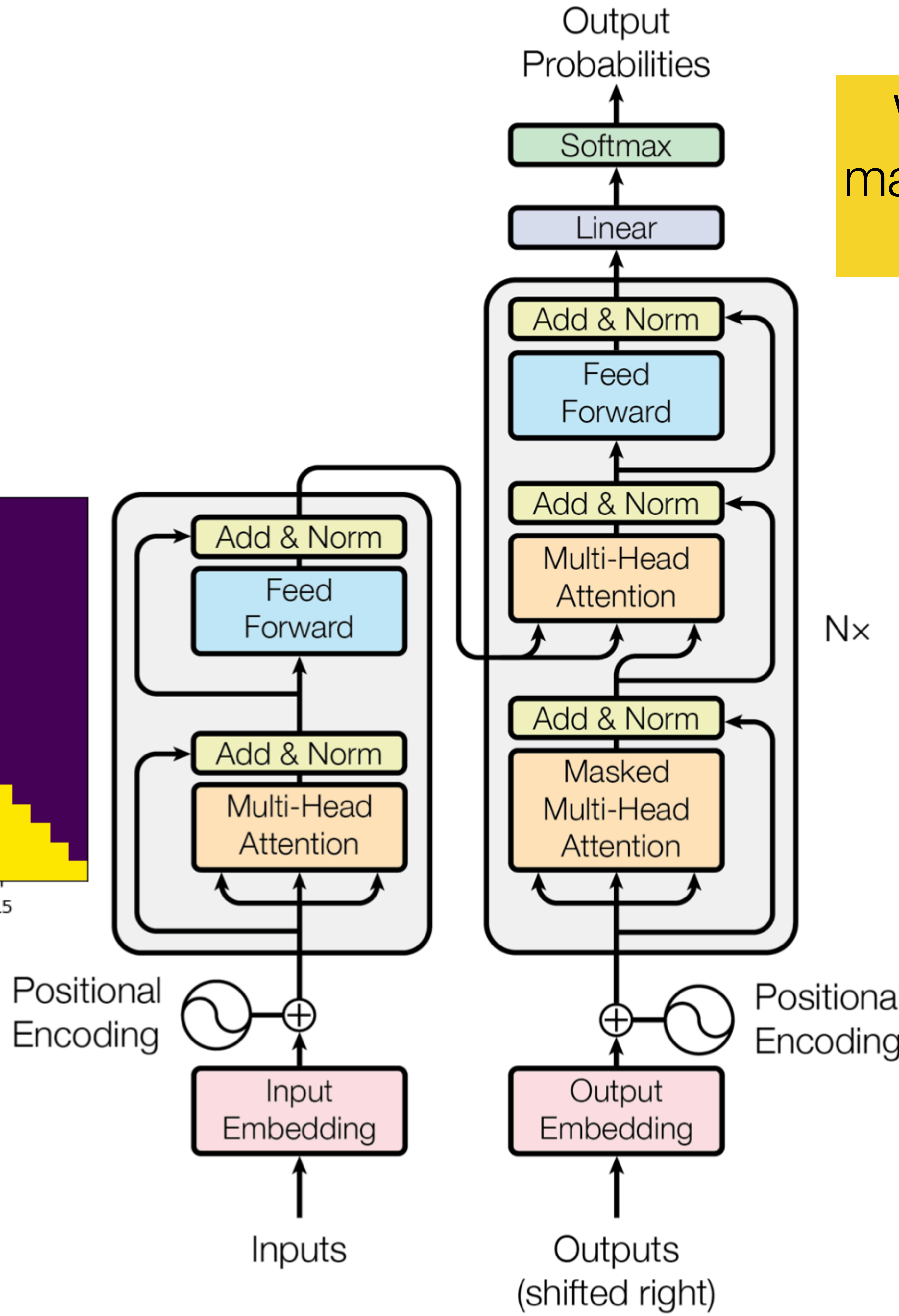
We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.





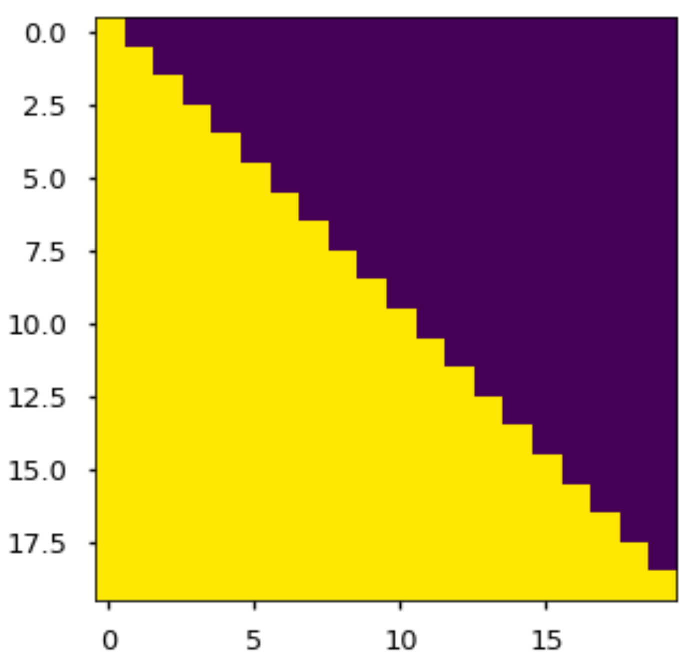
Why don’t we do masked self-attention in the encoder?

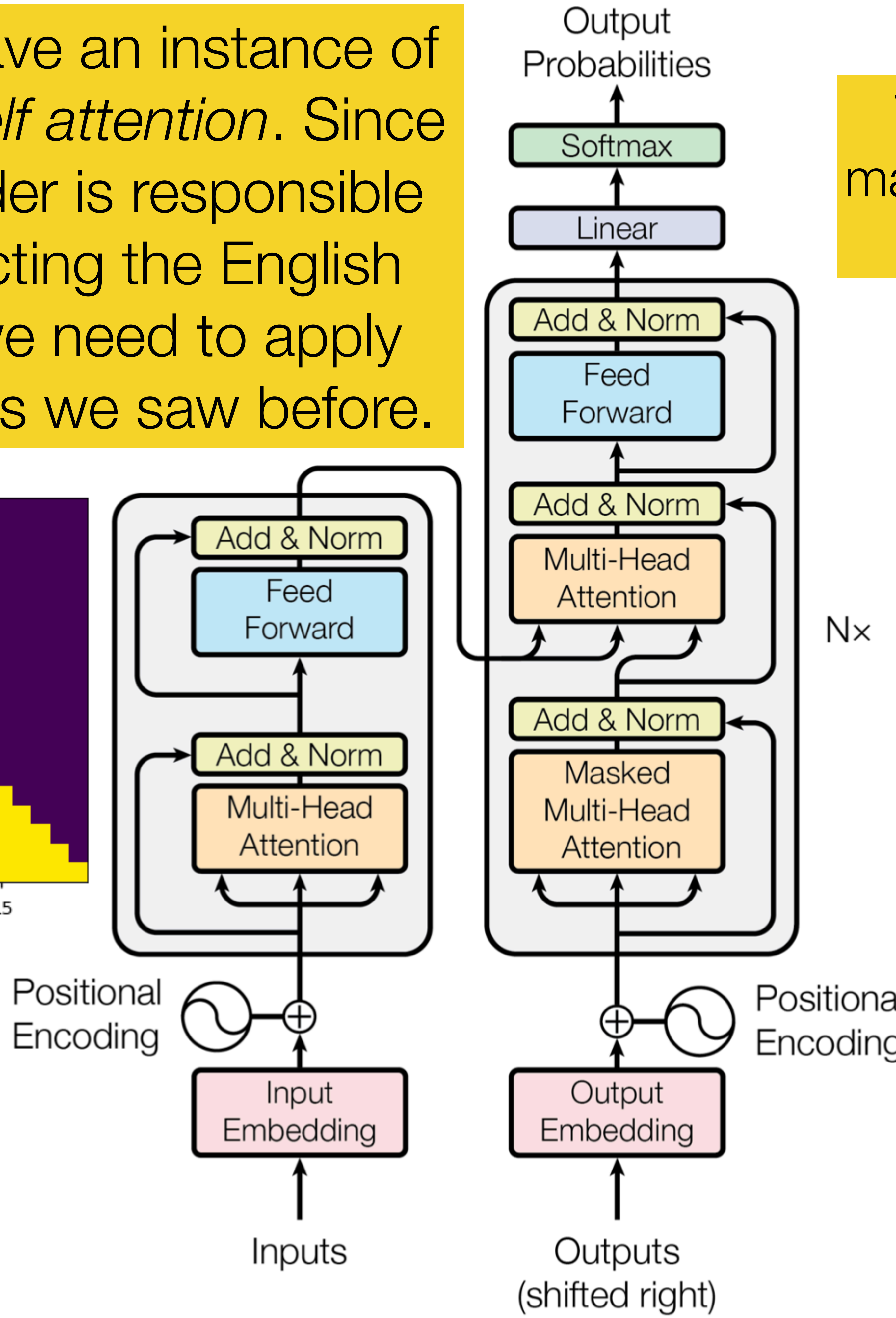




We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.

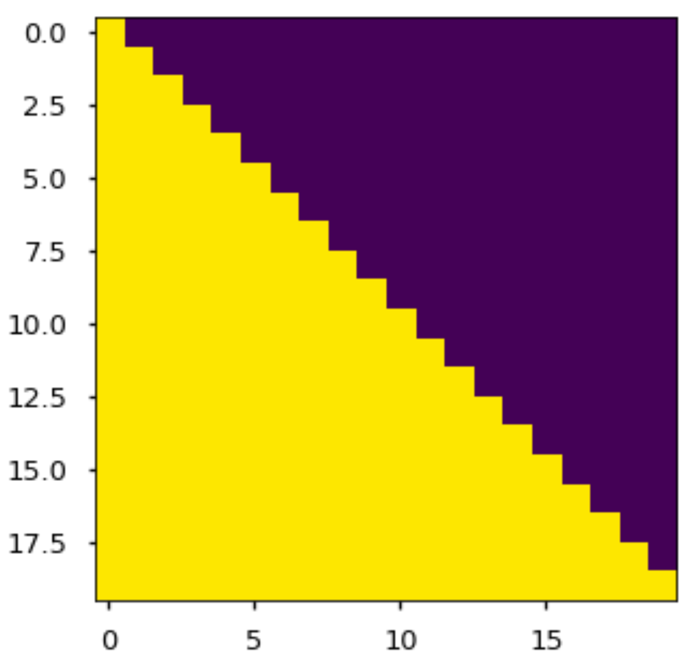
Why don't we do masked self-attention in the encoder?

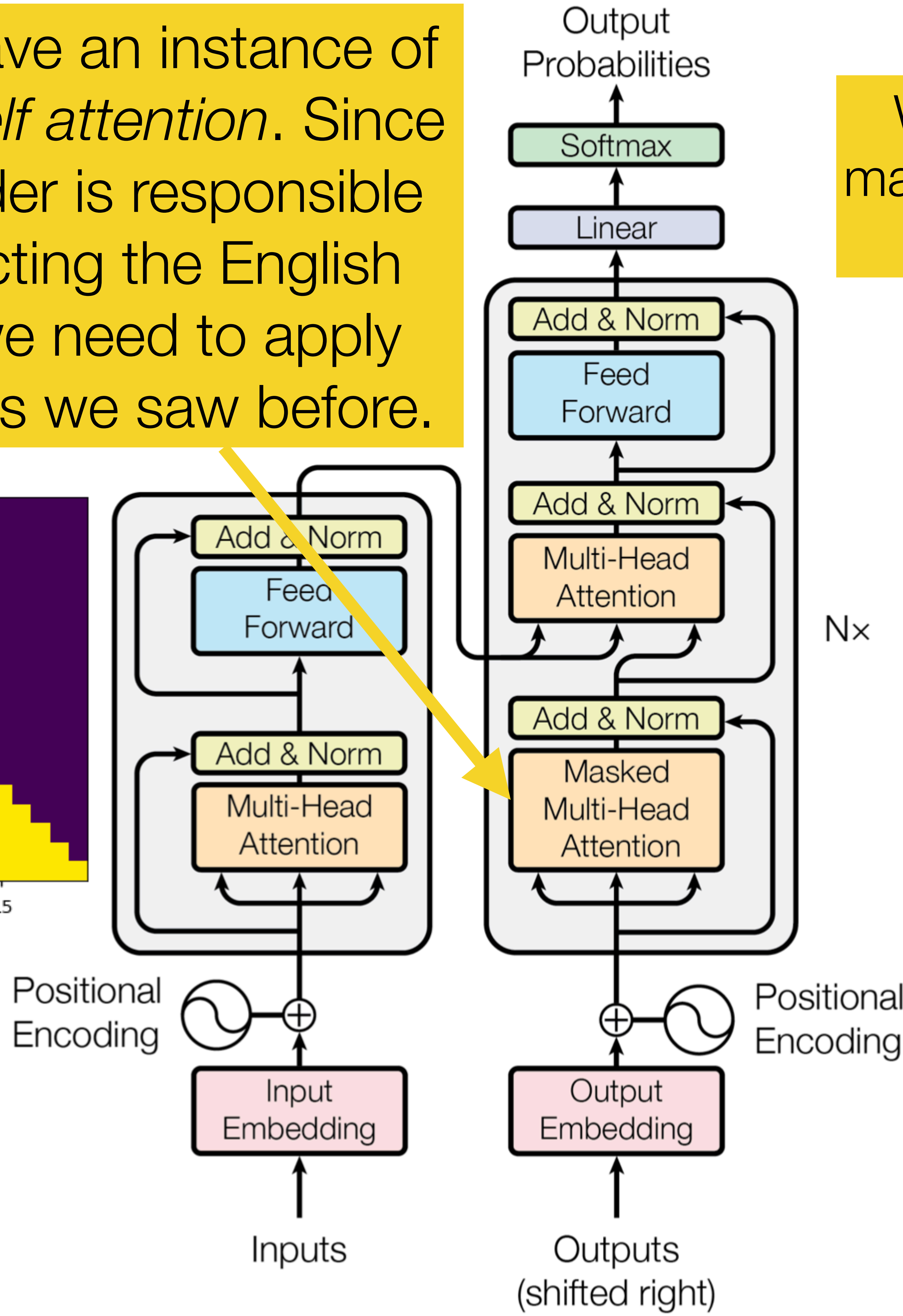


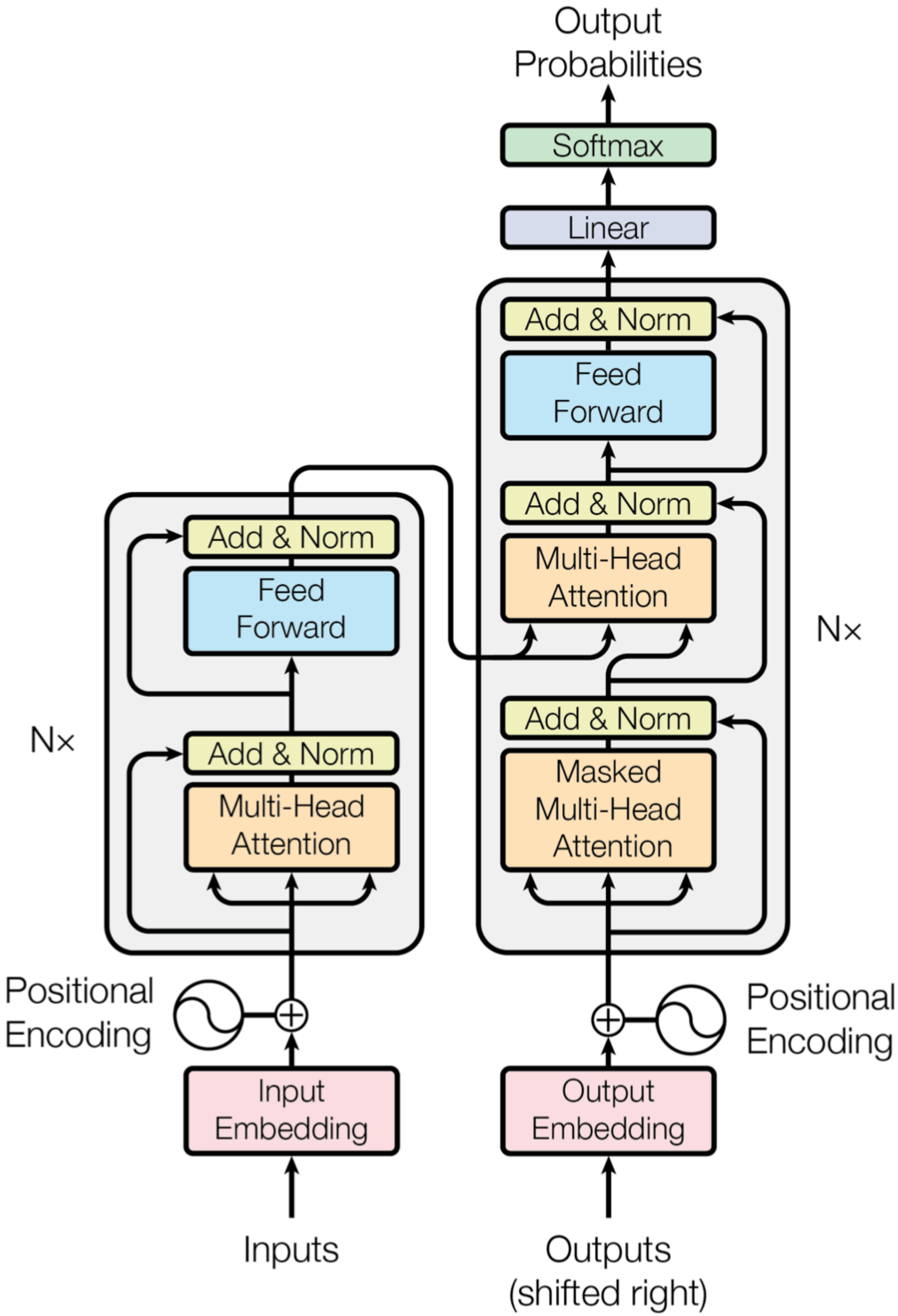


We first have an instance of *masked self attention*. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.

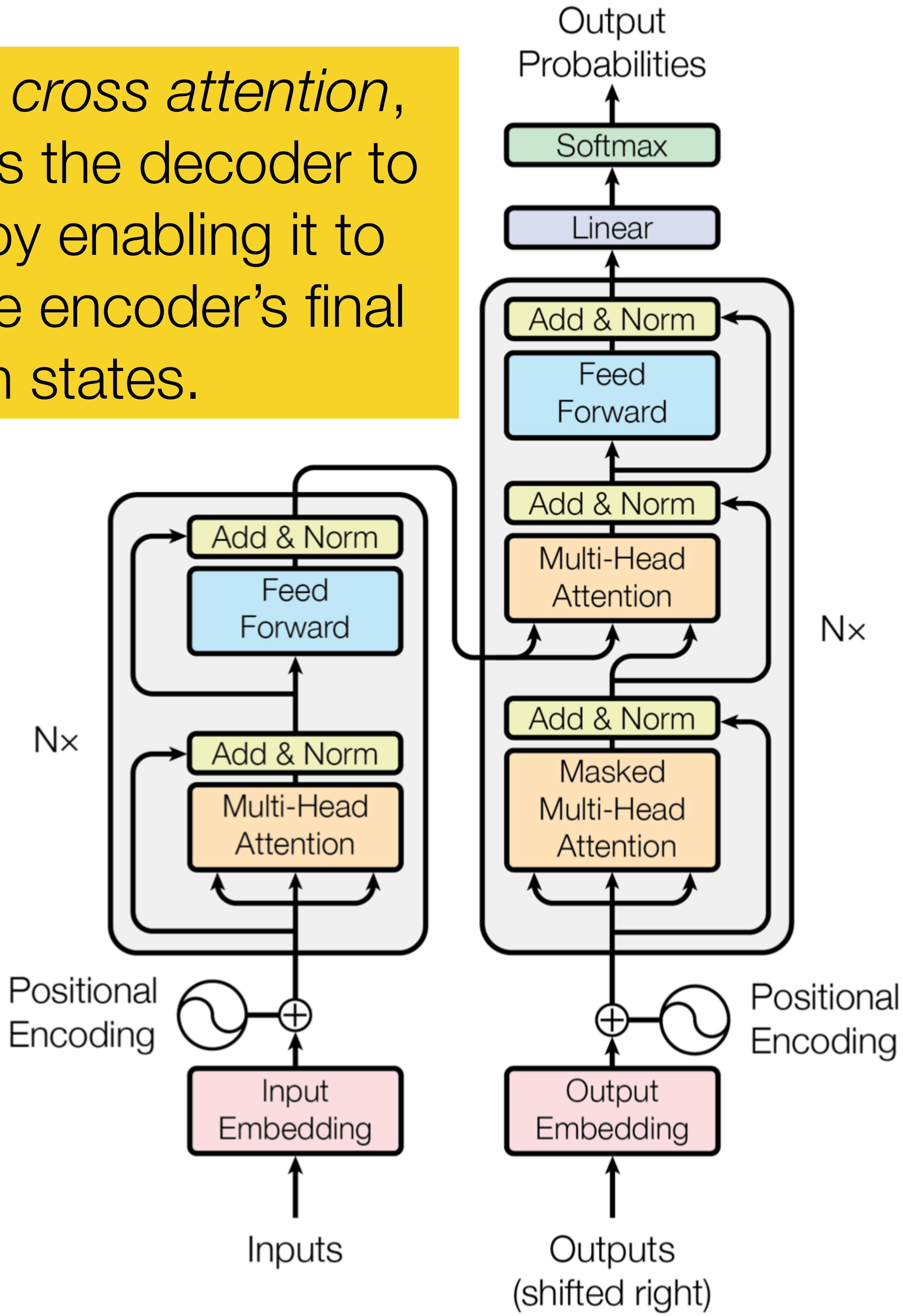
Why don't we do masked self-attention in the encoder?



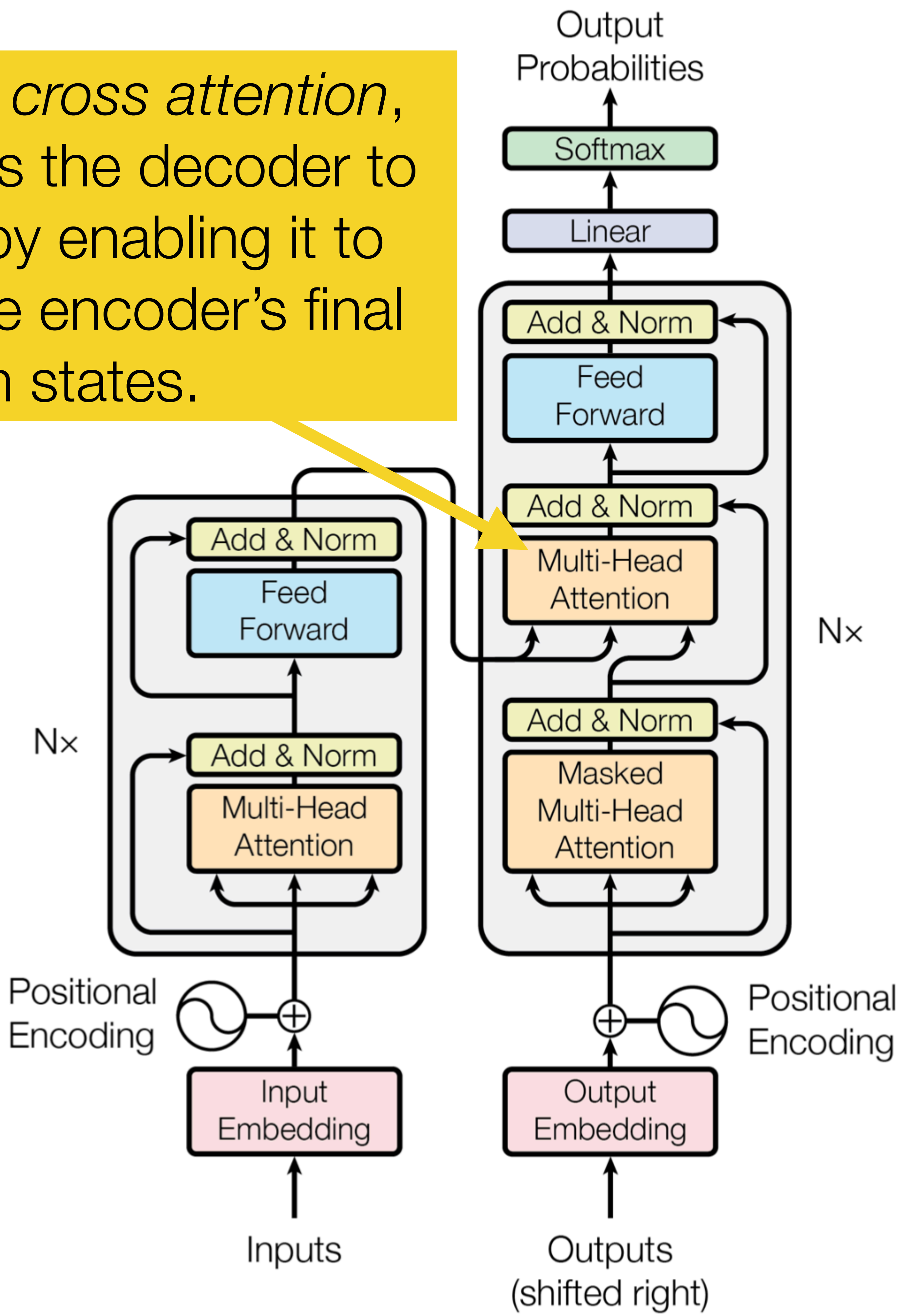


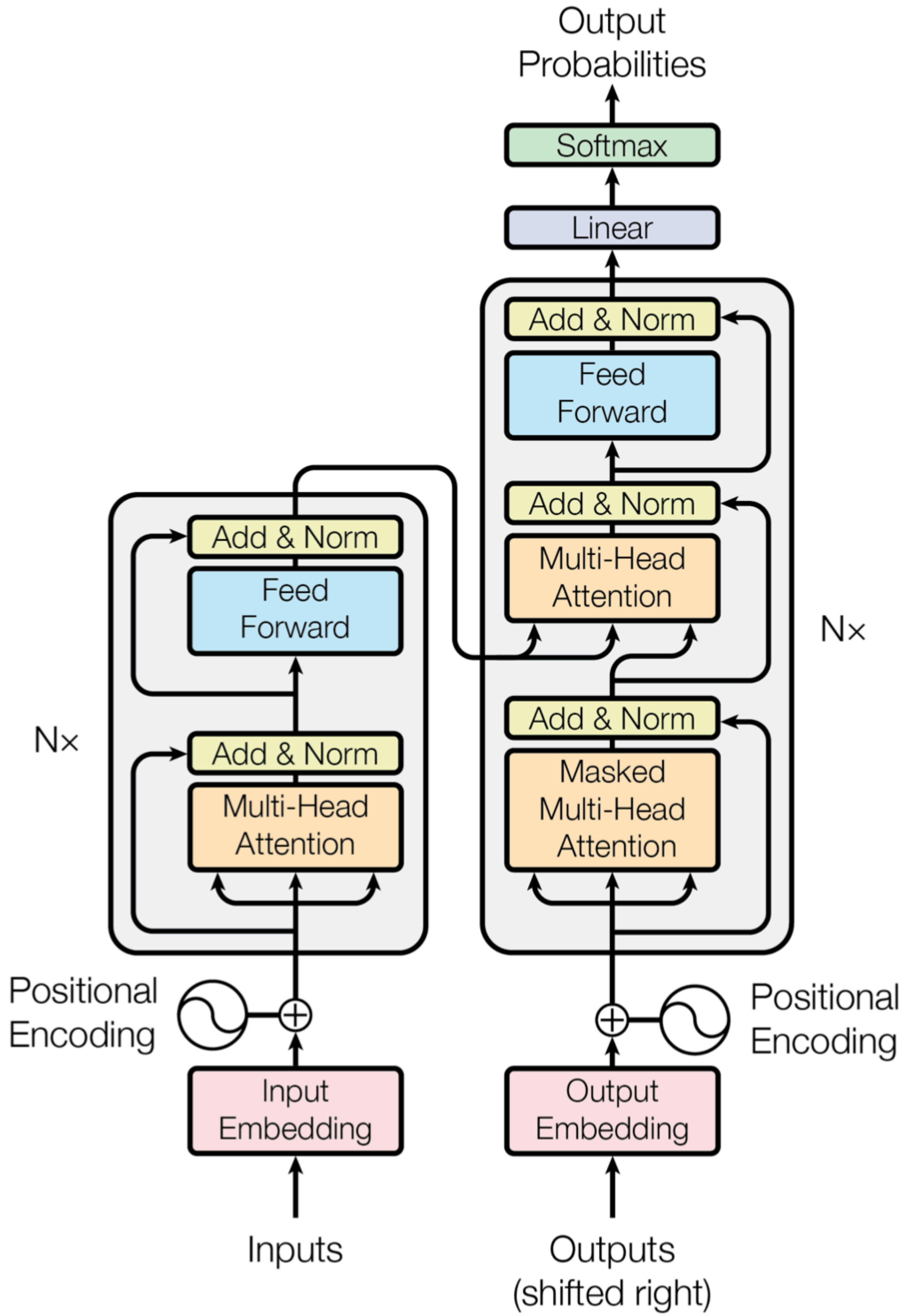

Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
Add & Norm
Feed
Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encoding
Positional
Encoding
Input
Embedding
Output
Embedding
Inputs
Outputs
(shifted right)

Now, we have *cross attention*, which connects the decoder to the encoder by enabling it to attend over the encoder's final hidden states.

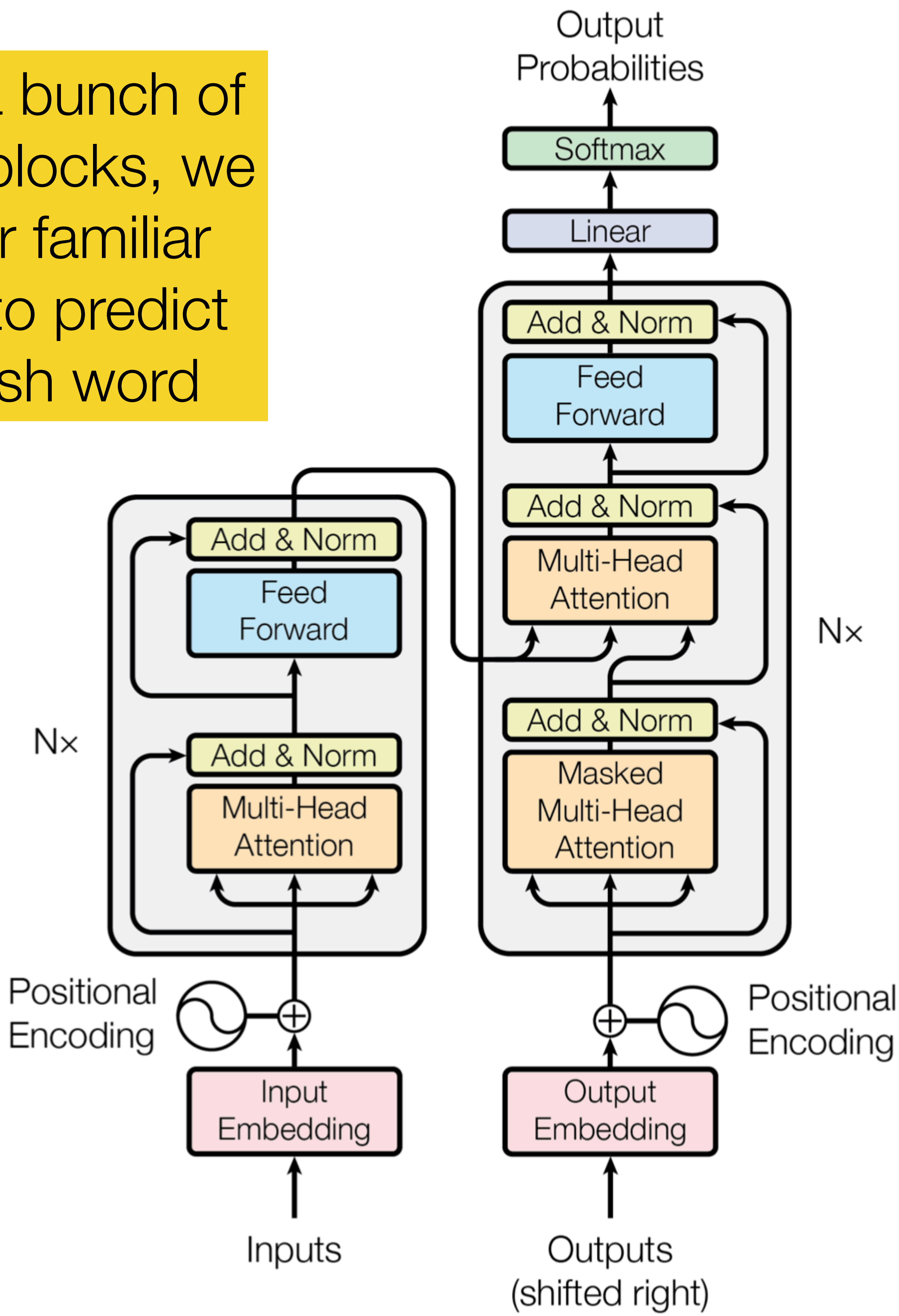


Now, we have *cross attention*, which connects the decoder to the encoder by enabling it to attend over the encoder's final hidden states.

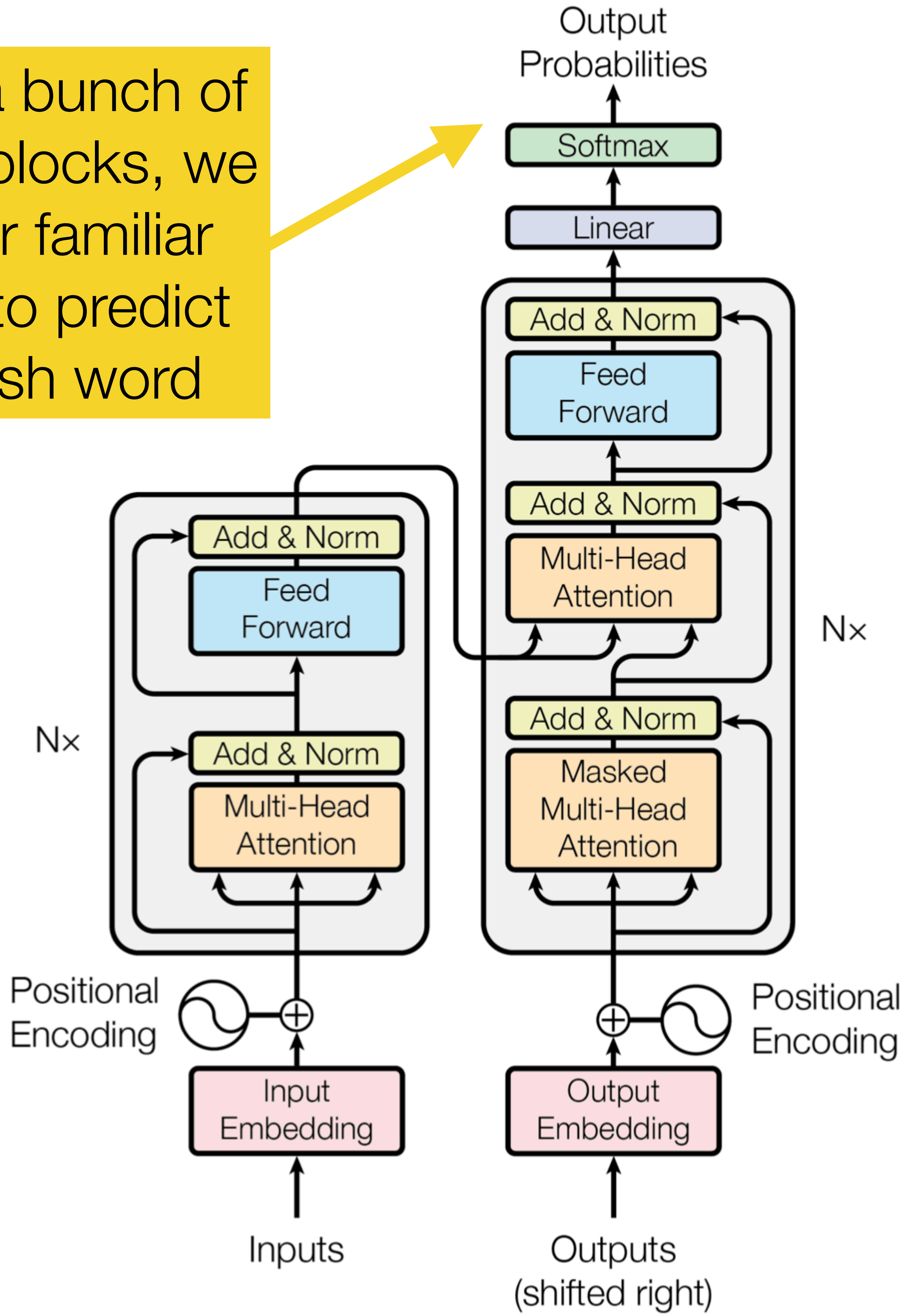



Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Inputs
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

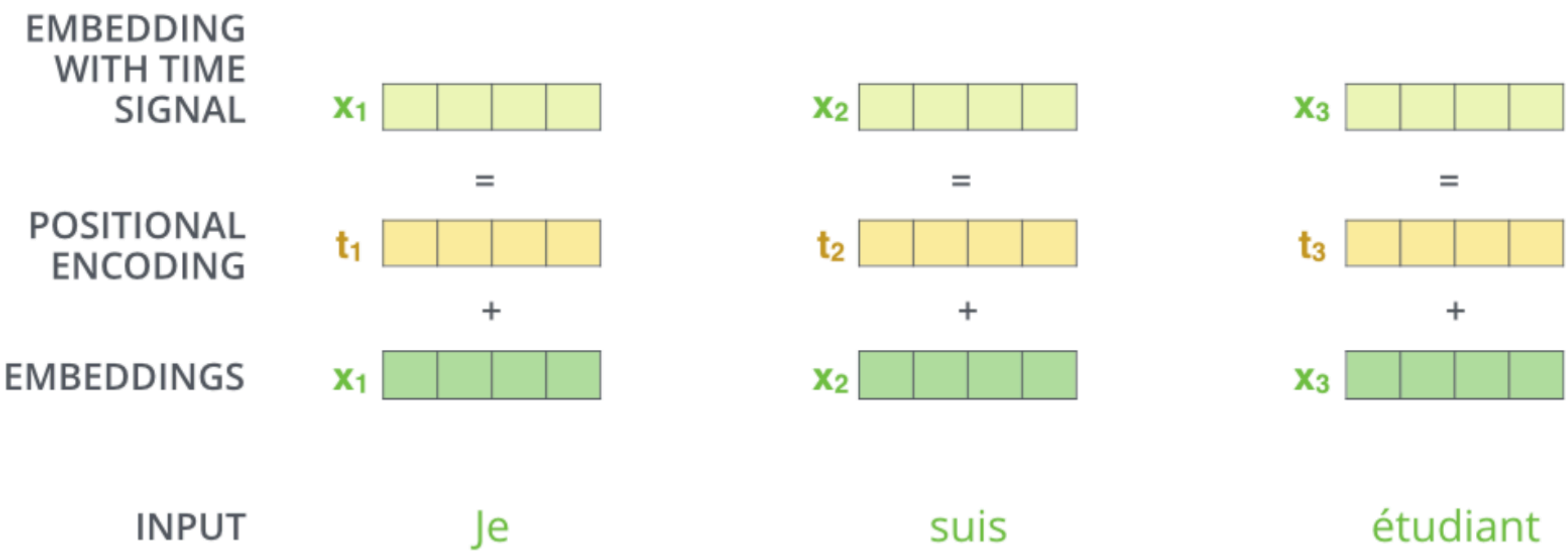
After stacking a bunch of these decoder blocks, we finally have our familiar Softmax layer to predict the next English word



After stacking a bunch of these decoder blocks, we finally have our familiar Softmax layer to predict the next English word



Positional encoding



Creating positional encodings?

- We could just concatenate a fixed value to each time step (e.g., 1, 2, 3, … 1000) that corresponds to its position, but then what happens if we get a sequence with 5000 words at test time?

- We want something that can generalize to arbitrary sequence lengths. We also may want to make attending to *relative positions* (e.g., tokens in a local window to the current token) easier.

Intuitive example

$0:$	0	0	0	0	$8:$	1	0	0	0
$1:$	0	0	0	1	$9:$	1	0	0	1
$2:$	0	0	1	0	$10:$	1	0	1	0
$3:$	0	0	1	1	$11:$	1	0	1	1
$4:$	0	1	0	0	$12:$	1	1	0	0
$5:$	0	1	0	1	$13:$	1	1	0	1
$6:$	0	1	1	0	$14:$	1	1	1	0
$7:$	0	1	1	1	$15:$	1	1	1	1

https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

Transformer positional encoding

$$PE_{(pos,2i)} = \sin(\frac{pos}{10000^{2i/d_{model}}})$$

$$PE_{(pos,2i+1)} = \cos(\frac{pos}{10000^{2i/d_{model}}})$$

Positional encoding is a 512d vector
i = a particular dimension of this vector
pos = dimension of the word
d_model = 512

Why this function???

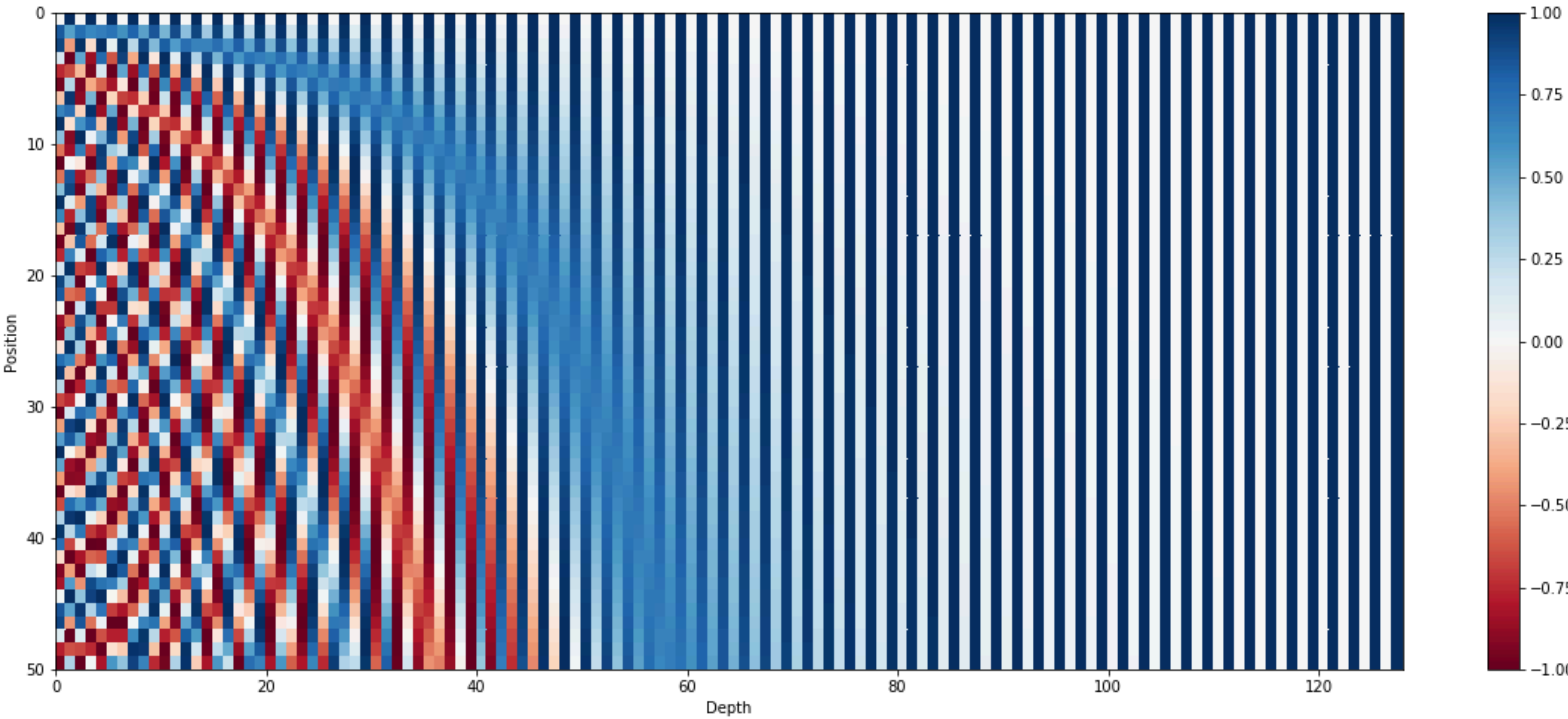
"We chose this function because we hypothesized it would allow the model to easily learn to attend by relative positions, since for any fixed offset k, PE_{pos+k} can be represented as a linear function of PE_{pos}."

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

What does this look like?

(each row is the pos. emb. of a 50-word sentence)



https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

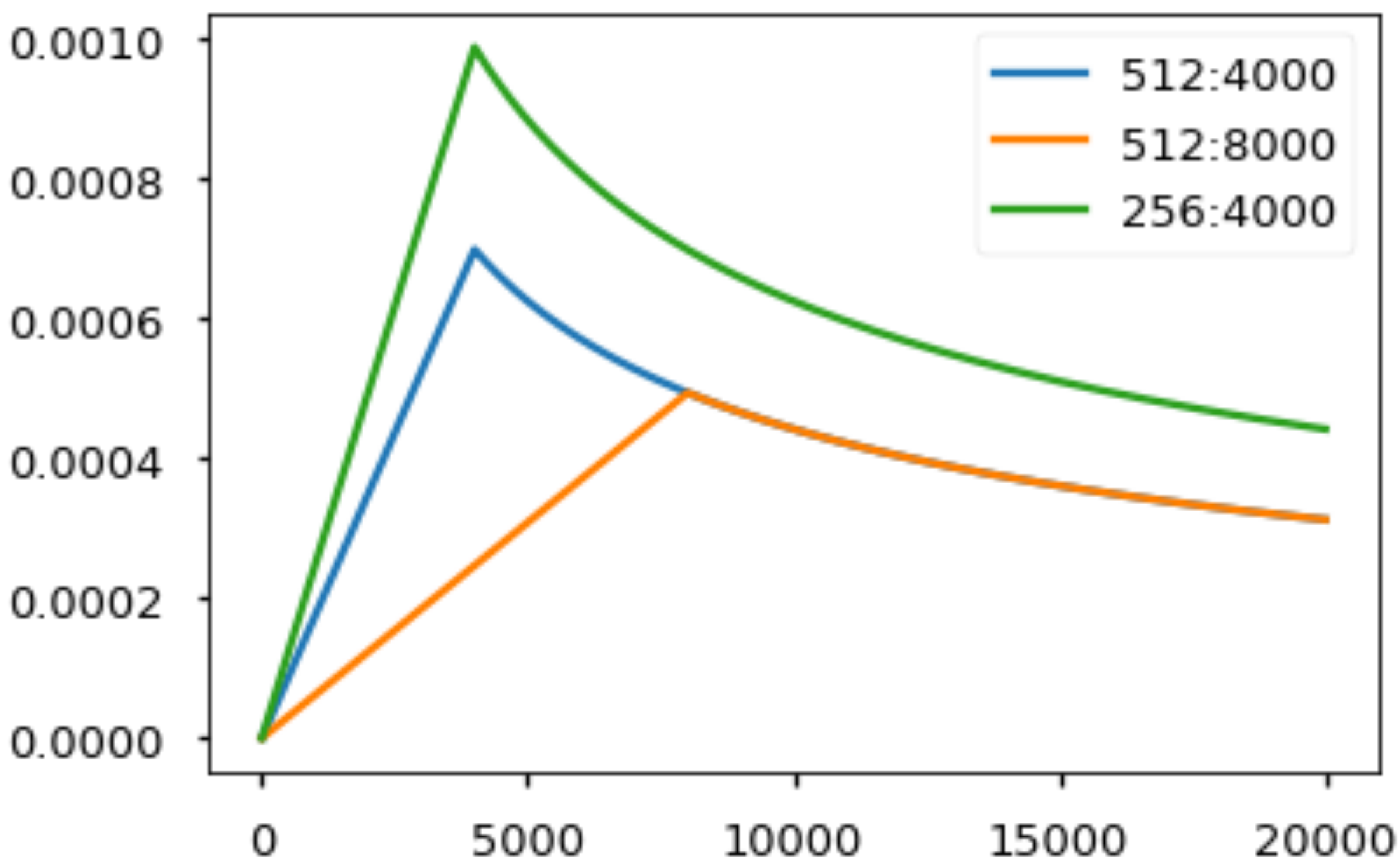
Despite the intuitive flaws, many models these days use *learned positional embeddings* (i.e., they cannot generalize to longer sequences, but this isn’t a big deal for their use cases)

Hacks to make Transformers work

Optimizer

We used the Adam optimizer (cite) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. We varied the learning rate over the course of training, according to the formula: $lrate = d_{\text{model}}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5})$ This corresponds to increasing the learning rate linearly for the first $warmup_steps$ training steps, and decreasing it thereafter proportionally to the inverse square root of the step number. We used $warmup_steps = 4000$.

> *Note: This part is very important. Need to train with this setup of the model.*



Label Smoothing

During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ (cite). This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

> *We implement label smoothing using the KL div loss. Instead of using a one-hot target distribution, we create a distribution that has `confidence` of the correct word and the rest of the `smoothing` mass distributed throughout the vocabulary.*

Label Smoothing

During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ (cite). This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

> *We implement label smoothing using the KL div loss. Instead of using a one-hot target distribution, we create a distribution that has `confidence` of the correct word and the rest of the `smoothing` mass distributed throughout the vocabulary.*

I went to class and took ___

cats TV notes took sofa

Label Smoothing

During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ (cite). This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

> *We implement label smoothing using the KL div loss. Instead of using a one-hot target distribution, we create a distribution that has `confidence` of the correct word and the rest of the `smoothing` mass distributed throughout the vocabulary.*

I went to class and took ___

cats	TV	notes	took	sofa
0	0	1	0	0

Label Smoothing

During training, we employed label smoothing of value $\epsilon_{ls} = 0.1$ (cite). This hurts perplexity, as the model learns to be more unsure, but improves accuracy and BLEU score.

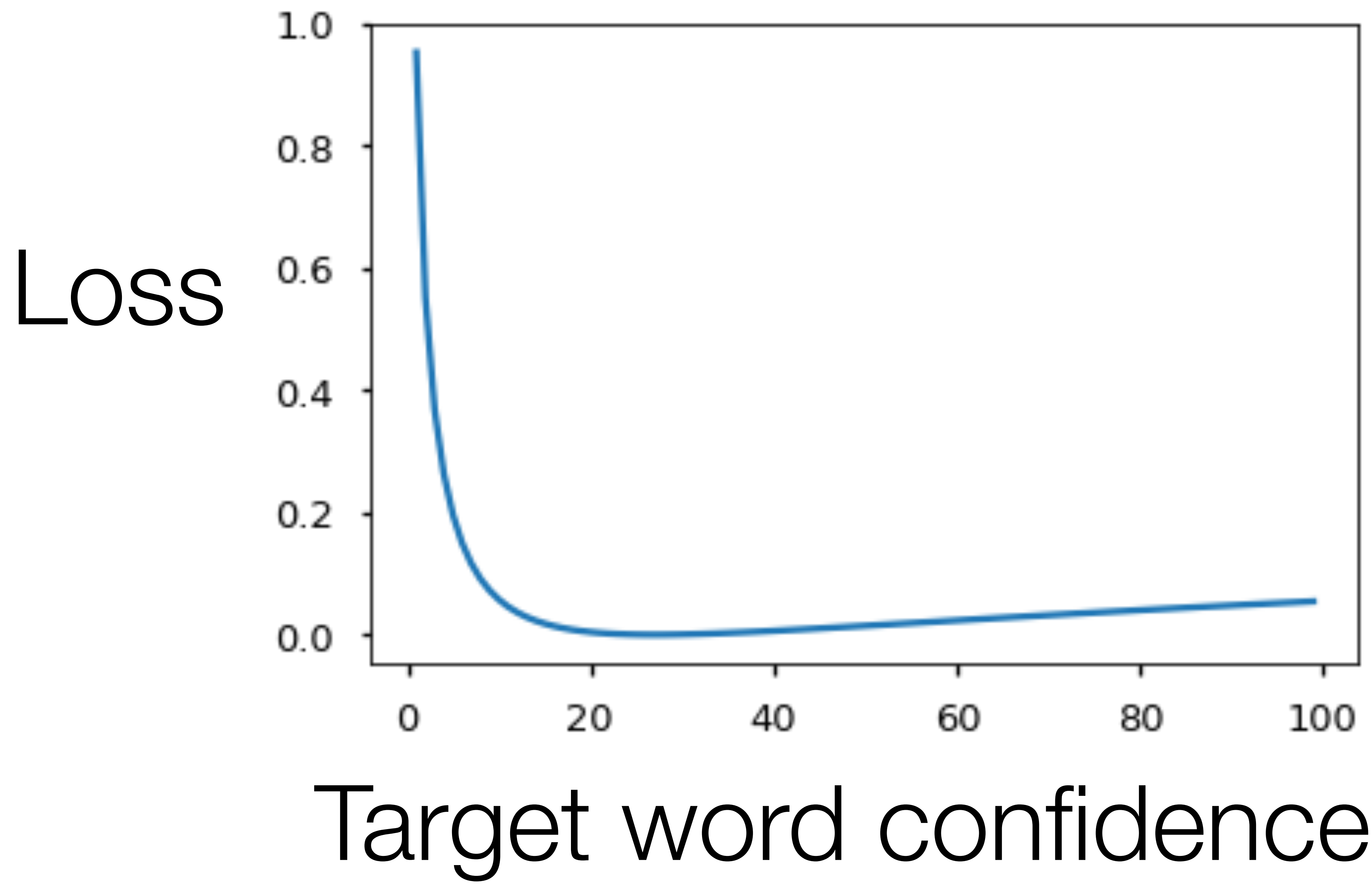
> *We implement label smoothing using the KL div loss. Instead of using a one-hot target distribution, we create a distribution that has `confidence` of the correct word and the rest of the `smoothing` mass distributed throughout the vocabulary.*

I went to class and took ___

cats	TV	notes	took	sofa
0	0	1	0	0
0.025	0.025	0.9	0.025	0.025

with label smoothing

Get penalized for overconfidence!



Byte pair encoding (BPE)

- Deal with rare words / large vocabulary by instead using *subword* tokenization

system	sentence
source	health research institutes
reference	Gesundheitsforschungsinstitute
WDict	Forschungsinstitute
C2-50k	Fo|rs|ch|un|gs|in|st|it|ut|io|ne|n
BPE-60k	Gesundheits|forsch|ungsinstitu|ten
BPE-J90k	Gesundheits|forsch|ungsin|stitute
source	asinine situation
reference	dumme Situation
WDict	asinine situation → UNK → asinine
C2-50k	as|in|in|e situation → As|in|en|si|tu|at|io|n
BPE-60k	as|in|ine situation → A|in|line-|Situation
BPE-J90K	as|in|ine situation → As|in|in-|Situation

Sennrich et al., ACL 2016

UMass · CS685 | Advanced Natural Language Processing (2020)

CS685 (2020)· 课程资料包 @ShowMeAI

视频	课件	笔记	代码
中英双语字幕	一键打包下载	官方笔记翻译	作业项目解析

视频 · B 站 [扫码或点击链接]
https://www.bilibili.com/video/BV1BL411t7RV

课件 & 代码 · 博客 [扫码或点击链接]
http://blog.showmeai.tech/umass-cs685

NLP 语言模型 问答系统 文本生成 BERT 语义解析 GPT-3 知识推理 模型蒸馏 迁移学习 transformer 注意力机制

Awesome AI Courses Notes Cheatsheets 是 ShowMeAI 资料库的分支系列，覆盖最具知名度的 **TOP50+** 门 AI 课程，旨在为读者和学习者提供一整套高品质中文学习笔记和速查表。

点击课程名称，跳转至课程**资料包**页面，**一键下载**课程全部资料！

机器学习	深度学习	自然语言处理	计算机视觉
Stanford · CS229	Stanford · CS230	Stanford · CS224n	Stanford · CS231n
# Awesome AI Courses Notes Cheatsheets· 持续更新中			
知识图谱	图机器学习	深度强化学习	自动驾驶
Stanford · CS520	Stanford · CS224W	UCBerkeley · CS285	MIT · 6.S094

微信公众号

资料下载方式 2：扫码点击底部菜单栏

称为 **AI 内容创作者？**回复 [添砖加瓦]