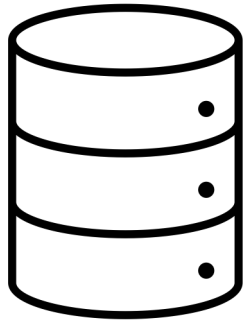
Autoregressive Language Models

Zhaolin Gao

CS 5788: Introduction to Generative Models

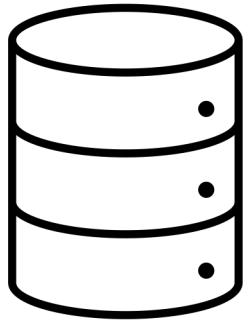

GPT-5
Gemini
deepseek
Grok
Claude

Pre-training



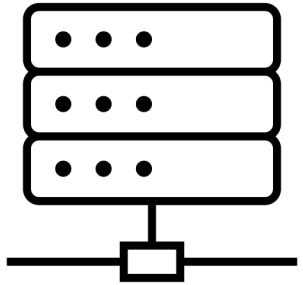
- Internet-scale data
- Learns grammar / syntax / world knowledge
- **Does not follow instructions well**

Pre-training



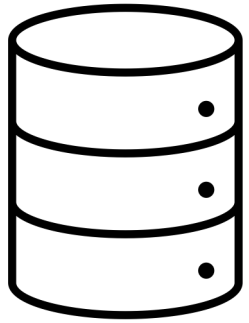
- Internet-scale data
- Learns grammar / syntax / world knowledge
- **Does not follow instructions well**

Mid-training



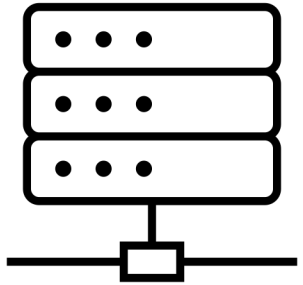
- Highest quality data (math / coding / reasoning)
- Preparing the model for post-training

Pre-training



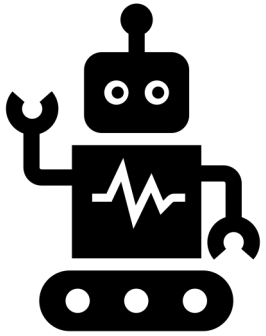
- Internet-scale data
- Learns grammar / syntax / world knowledge
- **Does not follow instructions well**

Mid-training



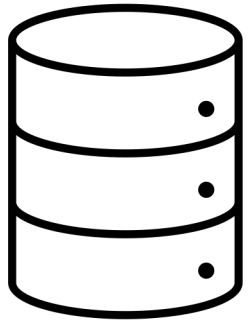
- Highest quality data (math / coding / reasoning)
- Preparing the model for post-training

Post-training



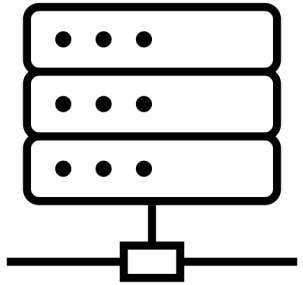
- Trajectories generated by the model itself
- Learns instruction following / reasoning / alignment

Pre-training



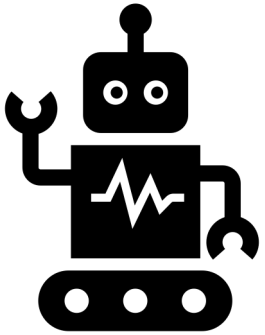
- Internet-scale data
- Learns grammar / syntax / world knowledge
- **Does not follow instructions well**

Mid-training



- Highest quality data (math / coding / reasoning)
- Preparing the model for post-training

Post-training



- Trajectories generated by the model itself
- Learns instruction following / reasoning / alignment

Today’s Class

- Autoregressive Language Model
- Architecture
- Tokenization
- Case study on GPT / Deepseek
- Class activity for project group [15 min]

Autoregressive Language Model

$\boldsymbol{\pi}$: Autoregressive Language Model

$\pi(y|x)$: The probability of generating y given x

$$\underbrace{the\ capital\ of\ France\ is\ Paris}_{\text{Response } y} \sim \pi(\cdot\,|\underbrace{what\ is\ the\ capital\ of\ France?}_{\text{Prompt } x})$$

Autoregressive Language Model

$$the \sim \pi(\cdot \,|what\ is\ the\ capital\ of\ France?\,)$$

$$capital \sim \pi(\cdot \,|what\ is\ the\ capital\ of\ France?\ the)$$

$$of \sim \pi(\cdot \,|what\ is\ the\ capital\ of\ France?\ the\ capital)$$

$$France \sim \pi(\cdot \,|what\ is\ the\ capital\ of\ France?\ the\ capital\ of)$$

$$is \sim \pi(\cdot \,|what\ is\ the\ capital\ of\ France?\ the\ capital\ of\ France)$$

$$Paris \sim \pi(\cdot \,|what\ is\ the\ capital\ of\ France?\ the\ capital\ of\ France\ is)$$

Autoregressive Language Model

$$\pi(y_1, y_2, y_3, y_4, y_5, y_6|x) = \pi(y_1|x)\pi(y_2|x, y_1) \ldots \pi(y_6|x, y_1, y_2, y_3, y_4, y_5)$$

$$\underbrace{\overbrace{the}^{y_1}\ \overbrace{capital\ of}^{y_2}\ \overbrace{France}^{y_3}\ \overbrace{is}^{y_4}\ \overbrace{Paris}^{y_5\ y_6}}_{\text{Response } y} \sim \pi(\cdot\,|\underbrace{what\ is\ the\ capital\ of\ France?}_{\text{Prompt } x})$$

History: Recurrent Neural Networks (RNNs)

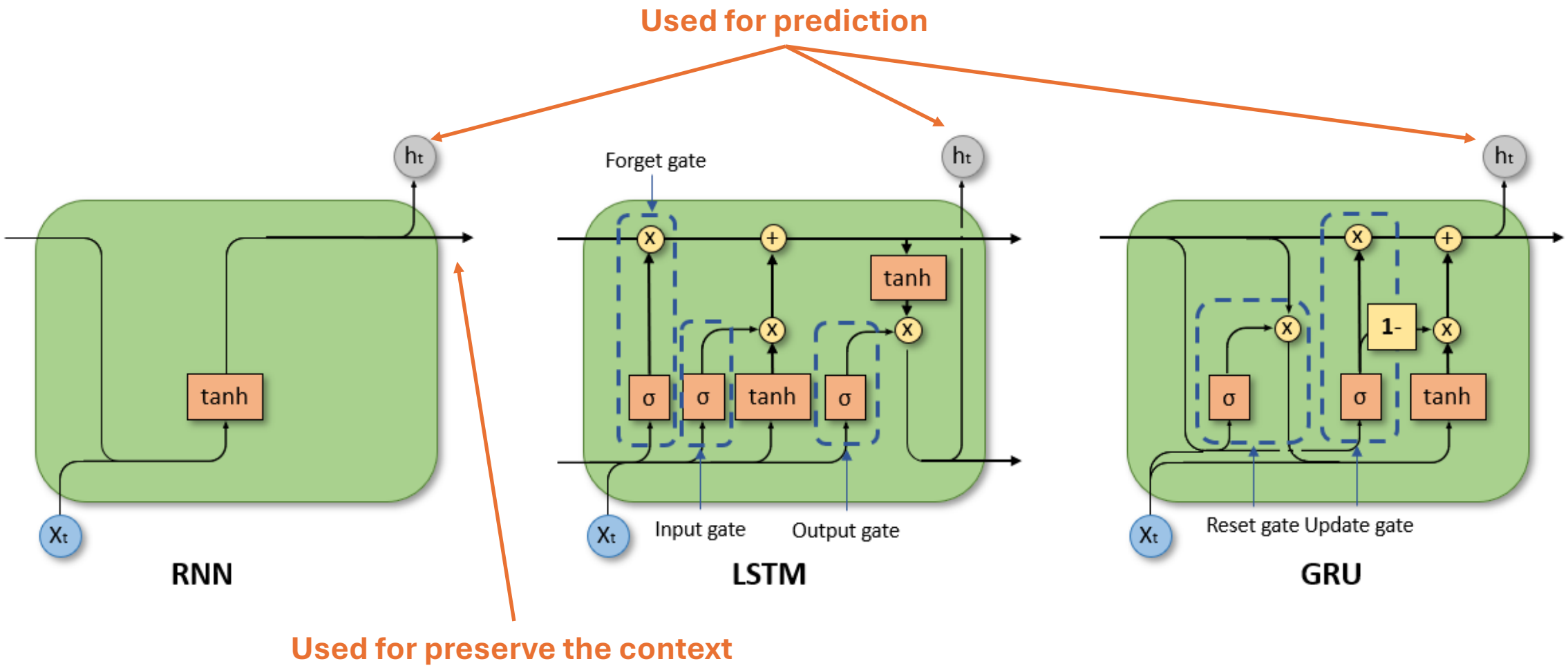


Image: Verma, Shourya, et al. "RouteNet-Fermi: Network Modeling With GNN (Analysis And Re-implementation)." *arXiv preprint arXiv:2412.05649* (2024).

History: Recurrent Neural Networks (RNNs)

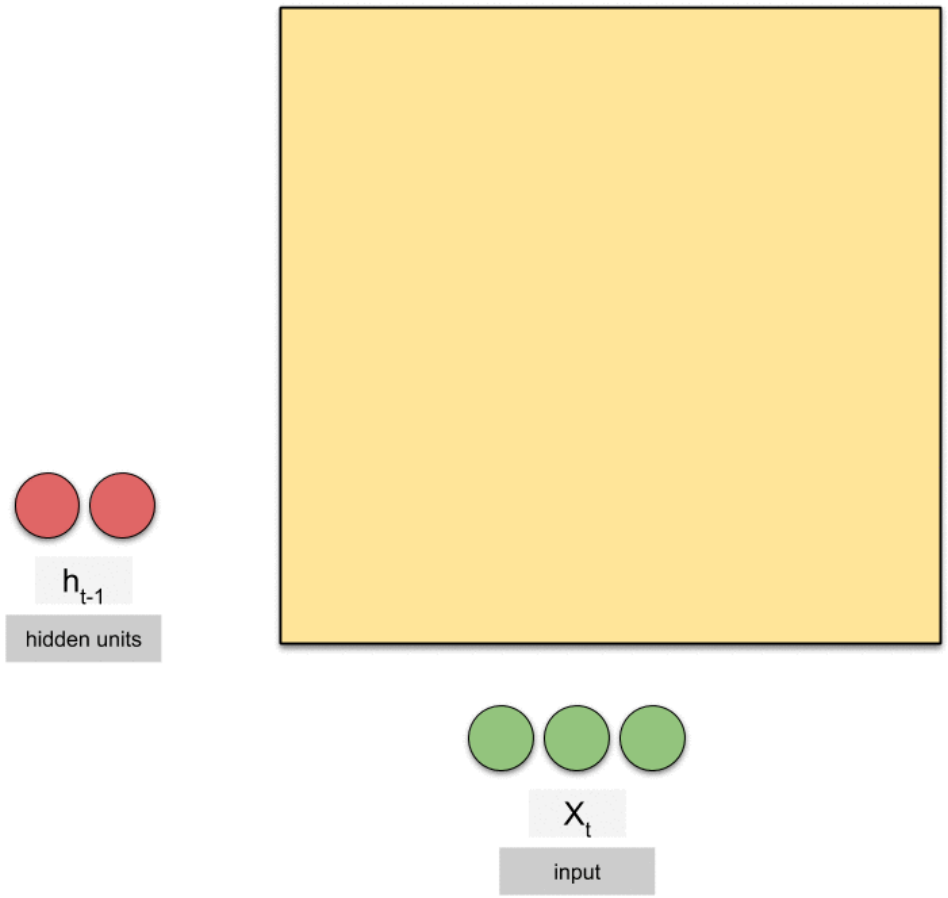


Image: Dhruv Matani. “Animated RNN, LSTM and GRU.” Medium, 2018.

History: RNNs for Translation

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT'14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous best result on this task. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

History: RNNs for Translation

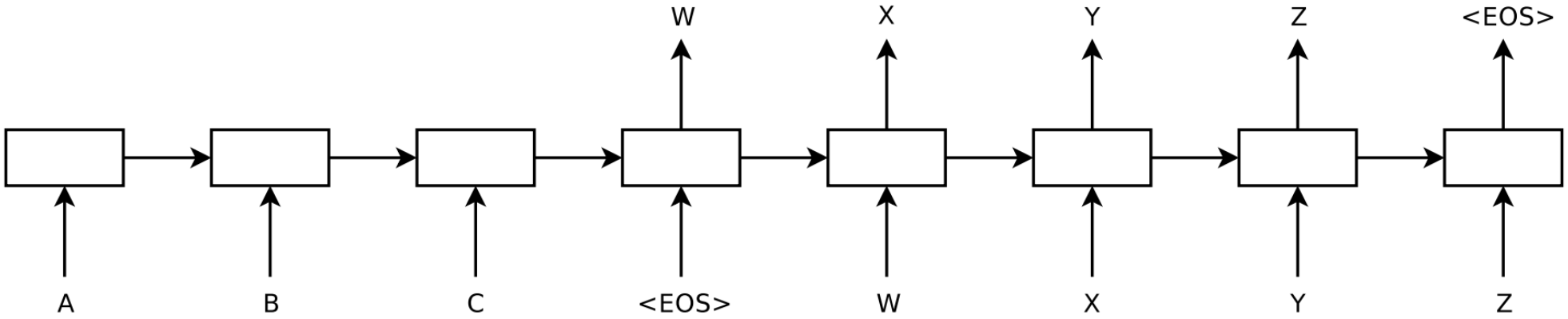
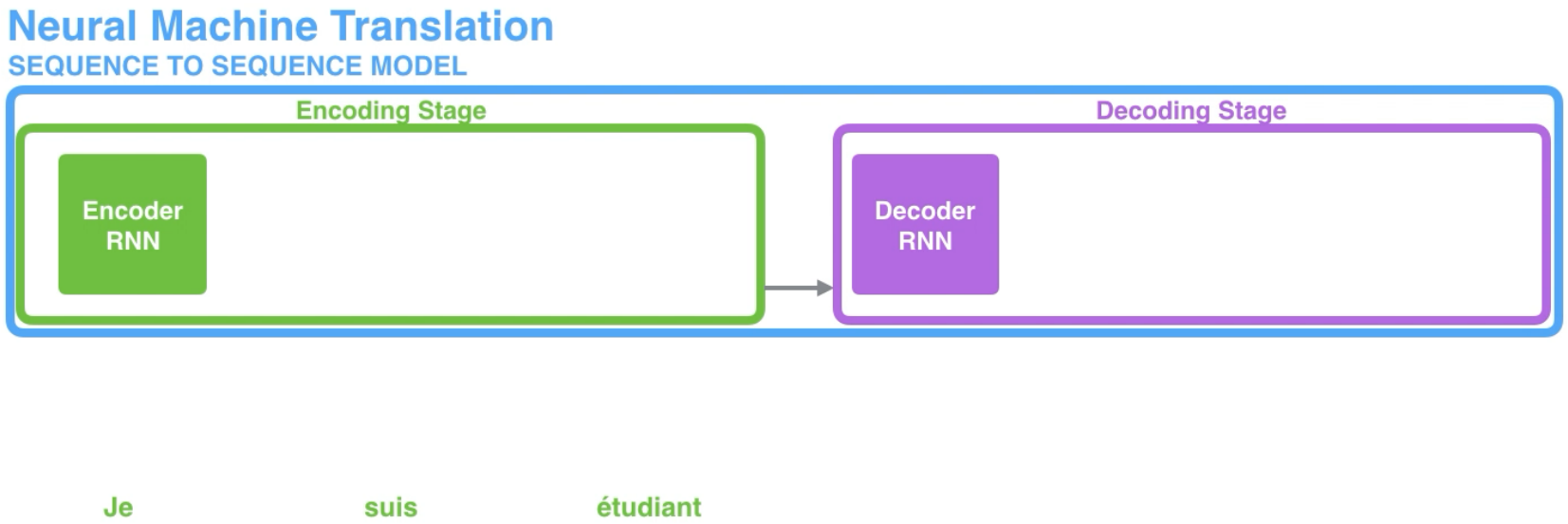


Image: Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems 27 (2014).

History: RNNs for Translation



- Context vector is a bottleneck for these types of models.
- It made it challenging for the models to deal with long sentences.

Image: Goplani, D. (2019). *Transformers*. Medium. https://medium.com/@deeksha.goplani/transformers-be818f3232a6

History: RNNs for Translation

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

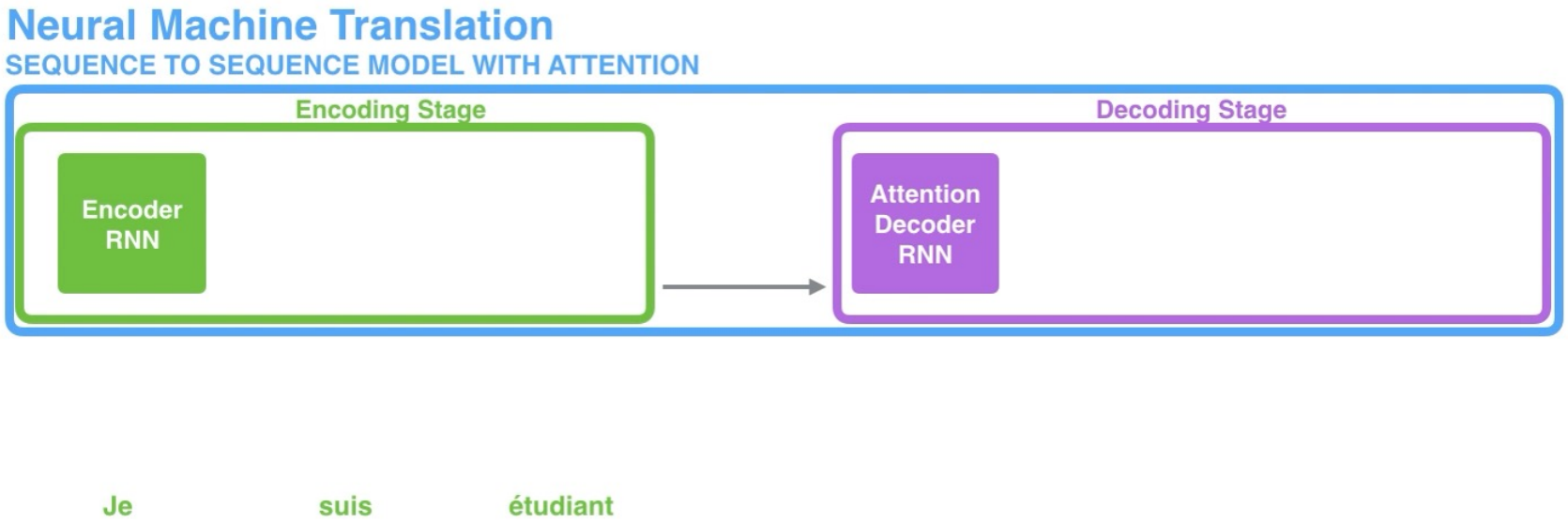
Dzmitry Bahdanau
Jacobs University Bremen, Germany

KyungHyun Cho **Yoshua Bengio***
Université de Montréal

Effective Approaches to Attention-based Neural Machine Translation

Minh-Thang Luong **Hieu Pham** **Christopher D. Manning**
Computer Science Department, Stanford University, Stanford, CA 94305
{lmthang,hyhieu,manning}@stanford.edu

History: RNNs for Translation



The encoder passes a lot more data to the decoder. Instead of passing the last hidden state of the encoding stage, the encoder passes *all* the hidden states to the decoder

Image: Goplani, D. (2019). *Transformers*. Medium. https://medium.com/@deeksha.goplani/transformers-be818f3232a6

History: RNNs for Translation

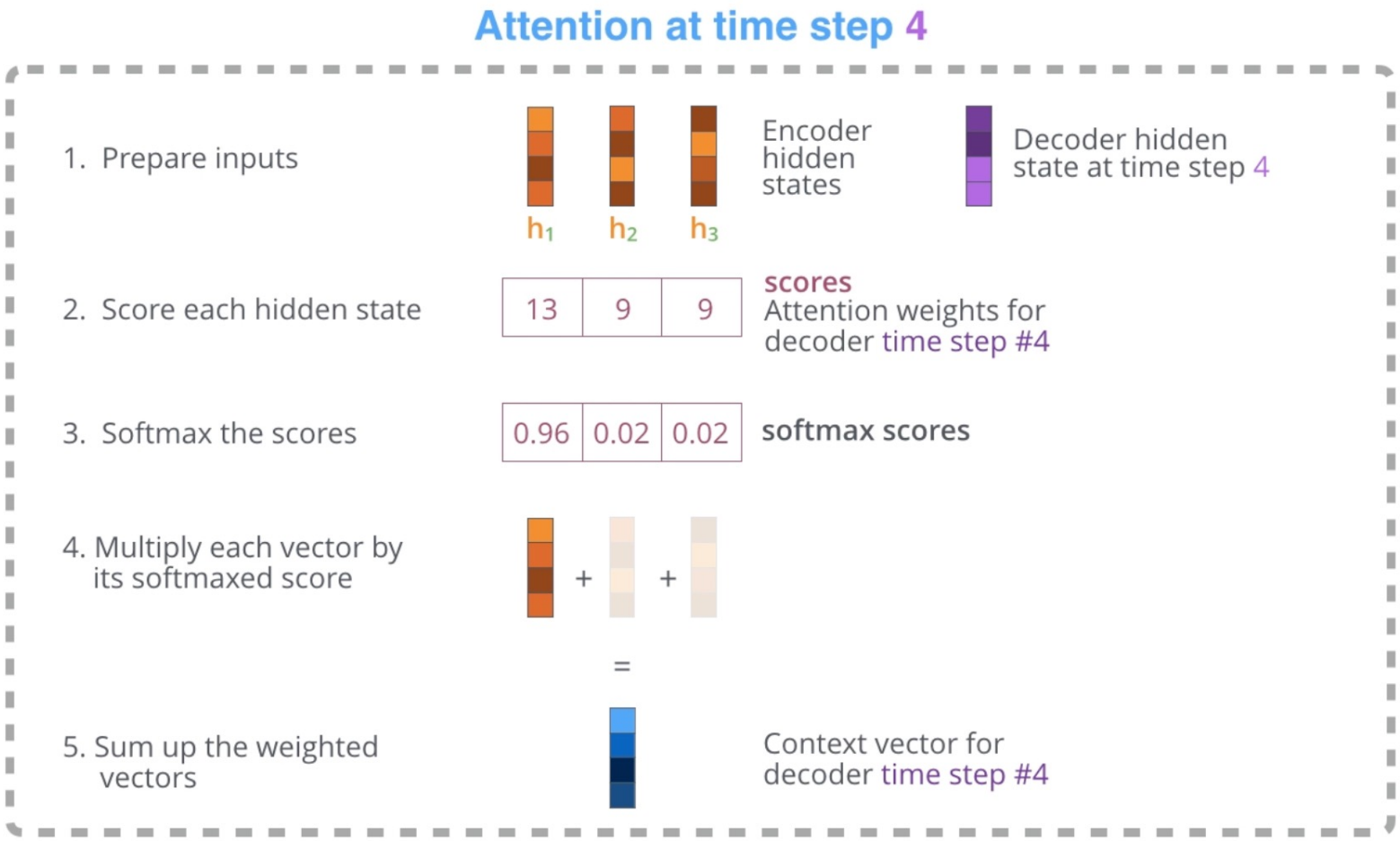


Image: Goplani, D. (2019). *Transformers*. Medium. https://medium.com/@deeksha.goplani/transformers-be818f3232a6

Transformer

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

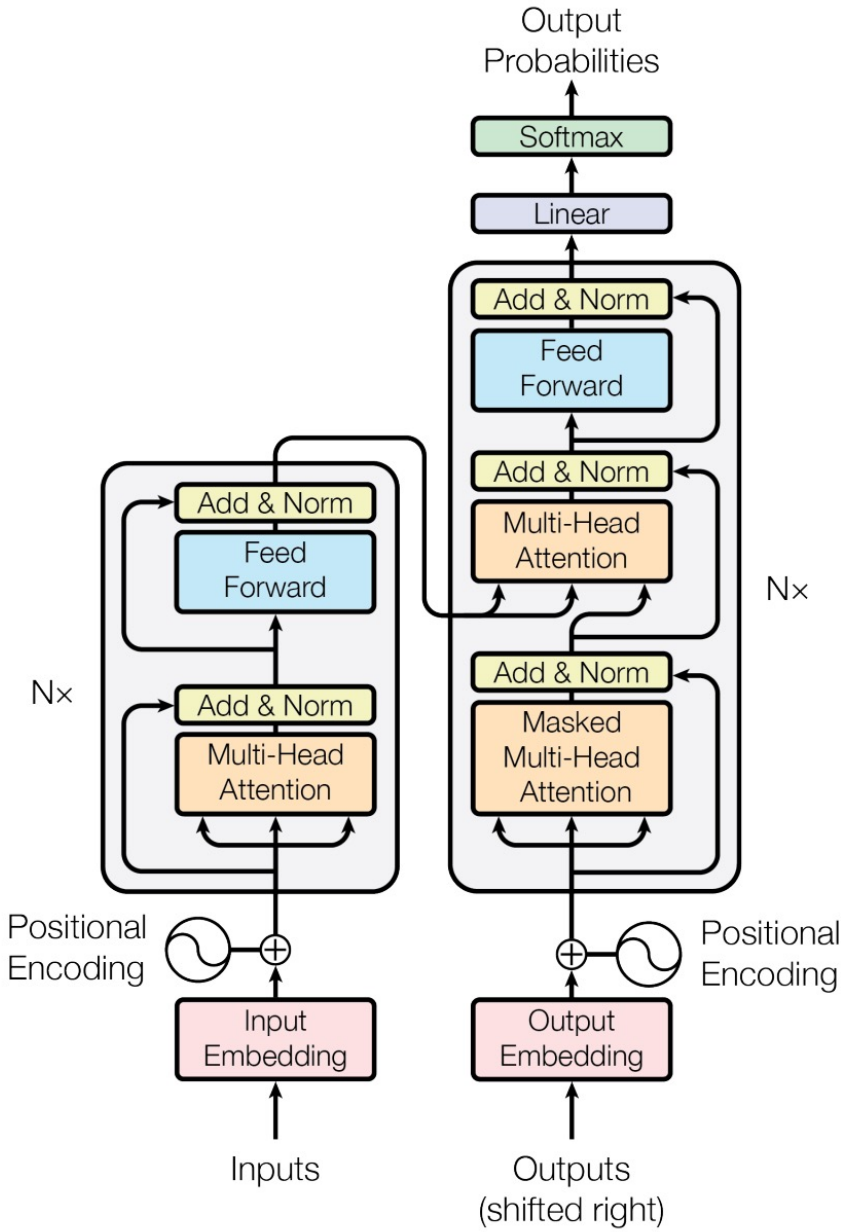
Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

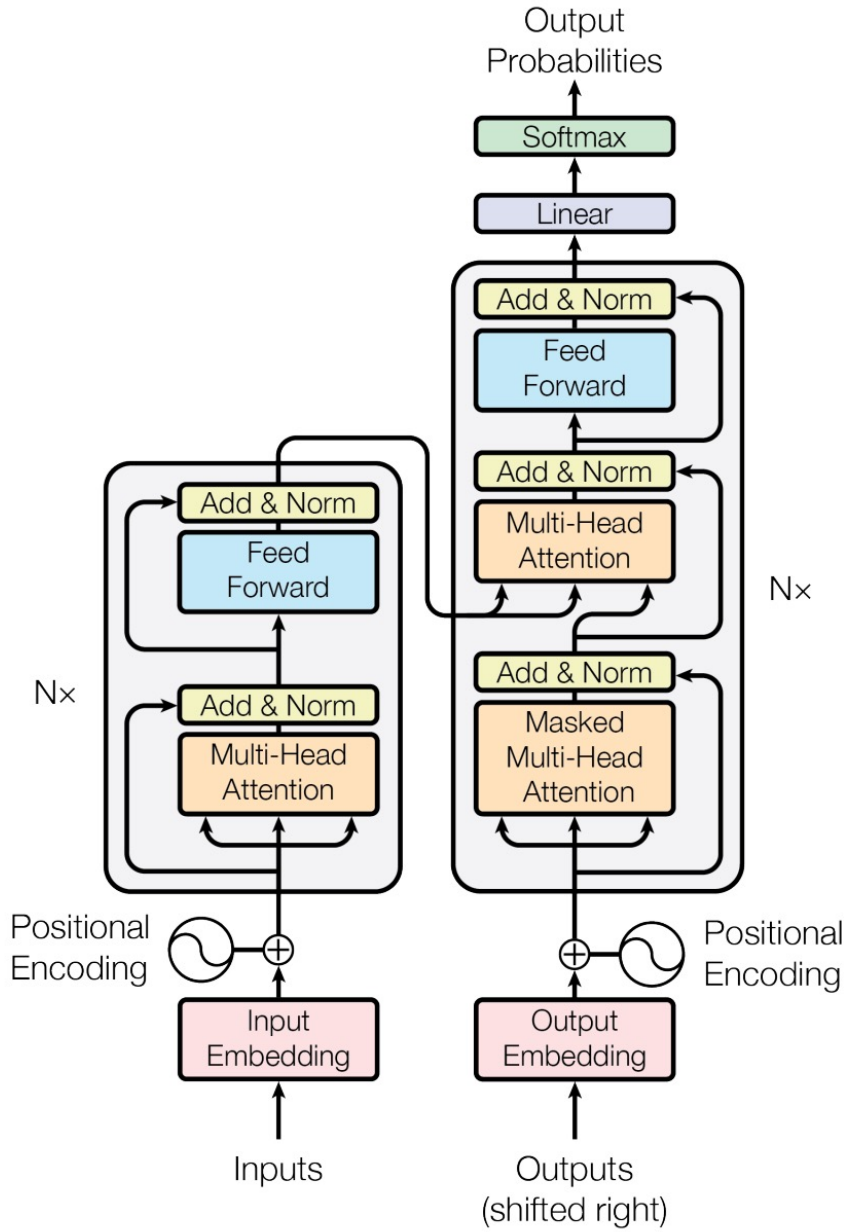
Illia Polosukhin* ‡ illia.polosukhin@gmail.com



Transformer

Non-Recurrent Encoder-Decoder architecture

- No hidden states
- Context information captured via attention and positional encodings
- Consists of stacks of layers with various sublayer



Transformer

Non-Recurrent Encoder-Decoder architecture

- No hidden states
- Context information captured via attention and positional encodings
- Consists of stacks of layers with various sublayer

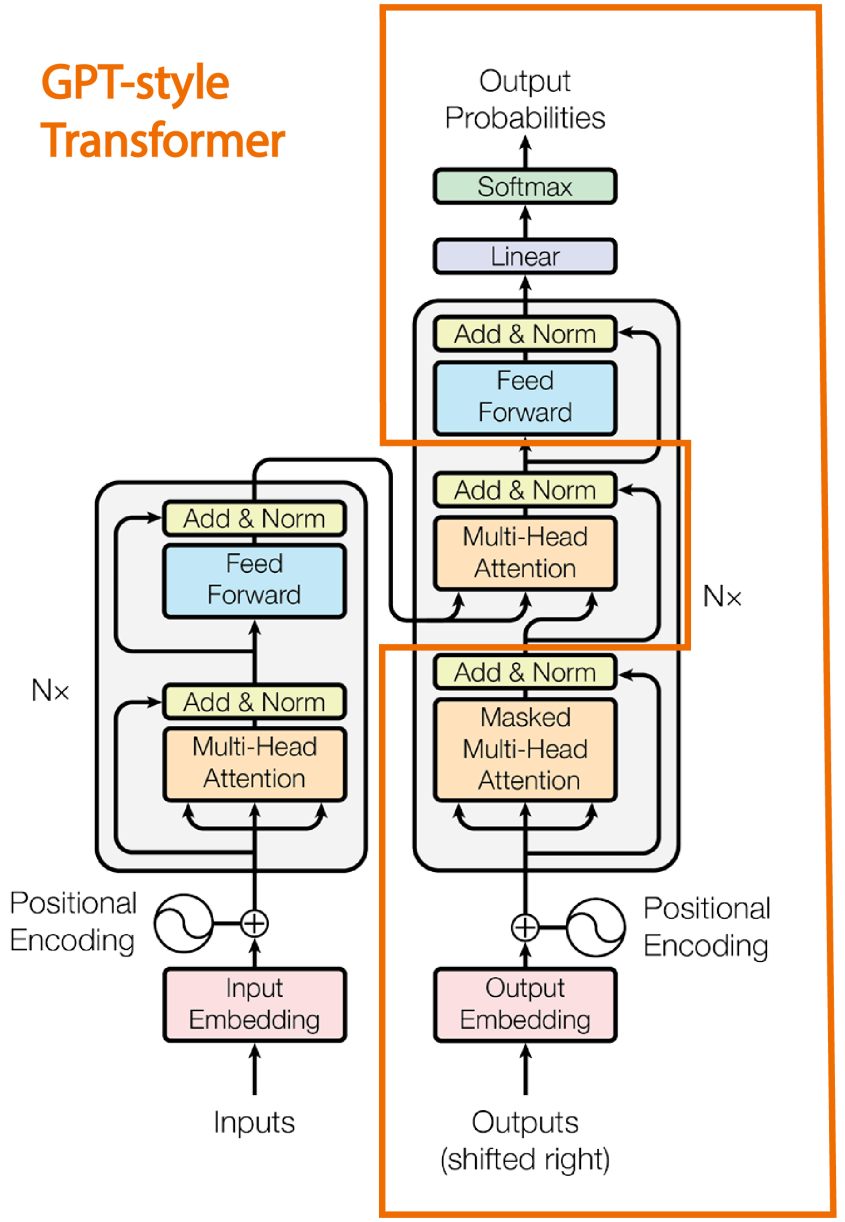
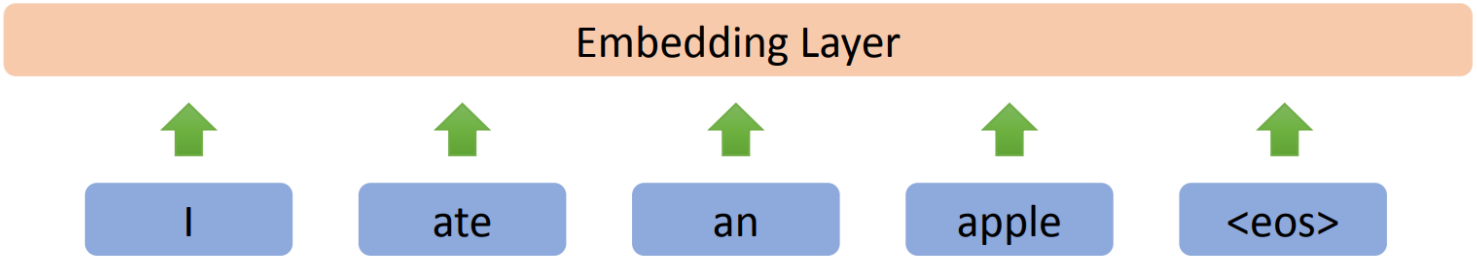


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Embedding



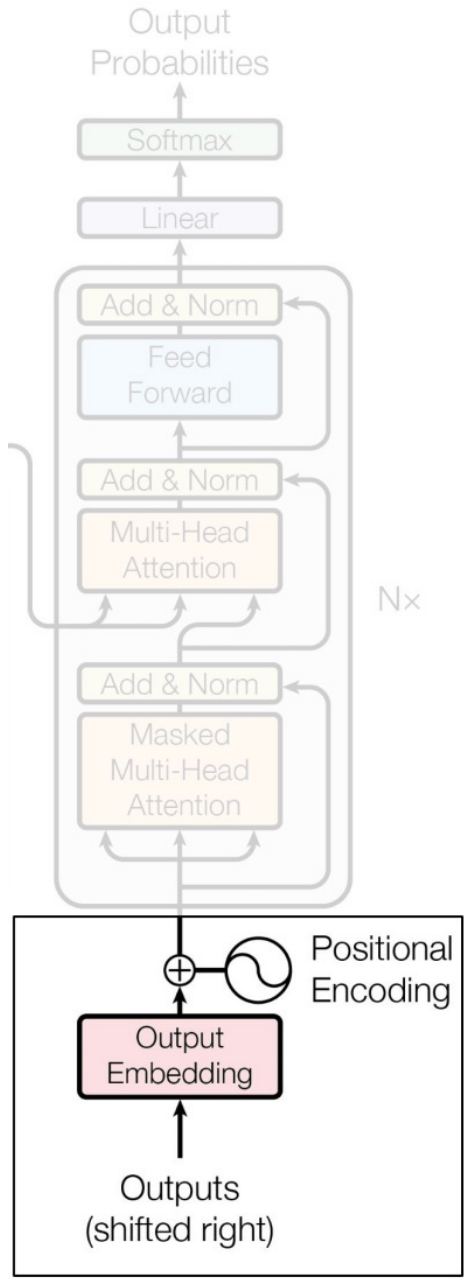
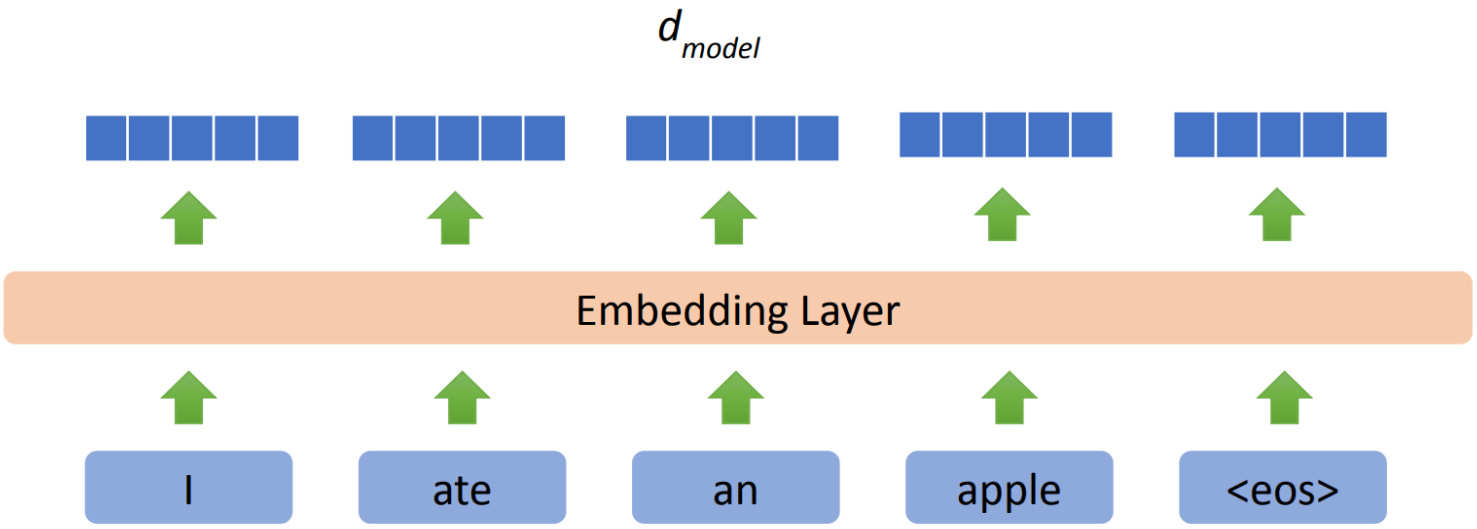


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Embedding



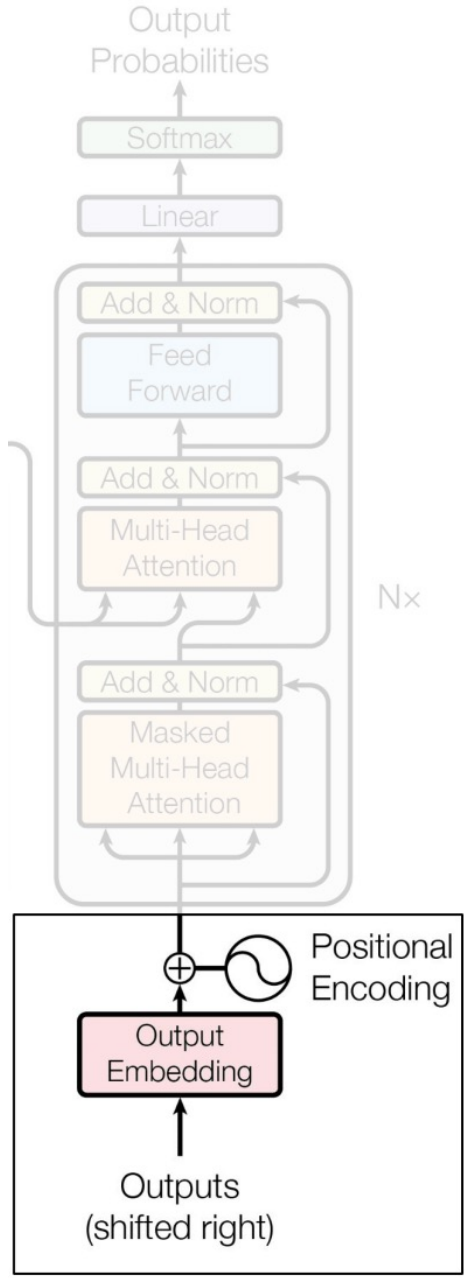
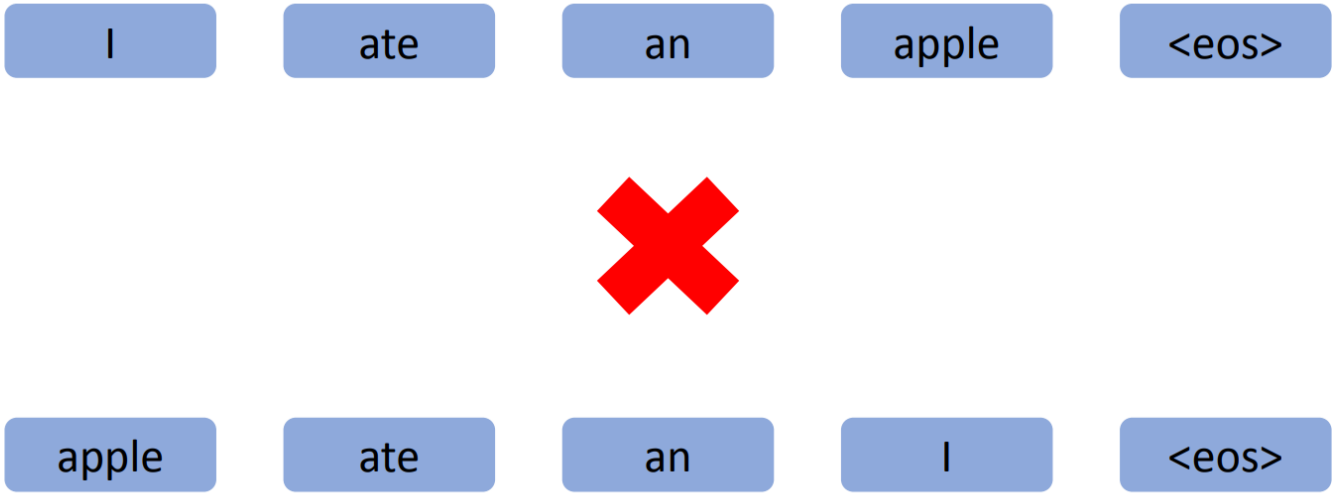


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Positional Encoding



We need to **encode** the position of each token

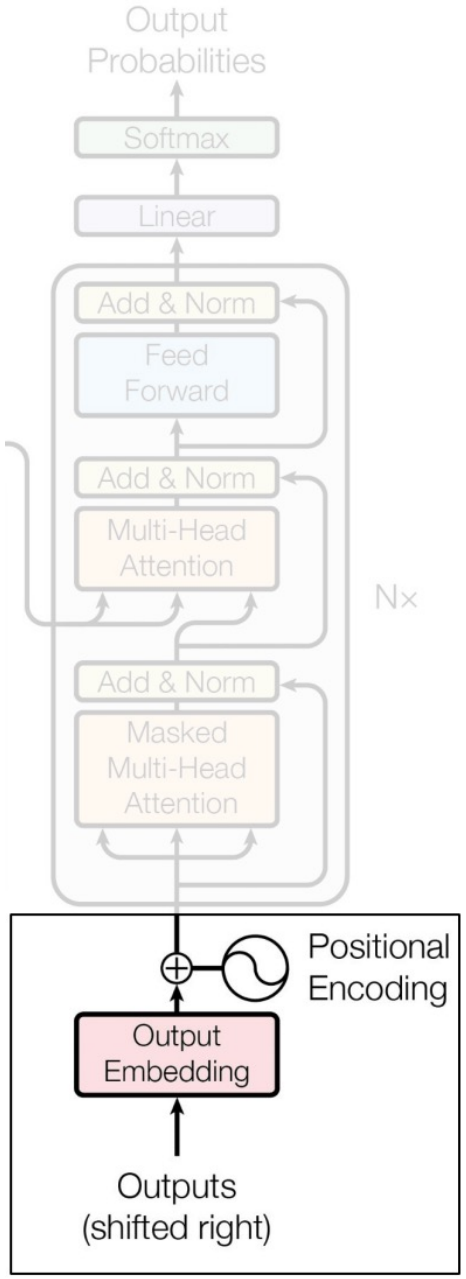
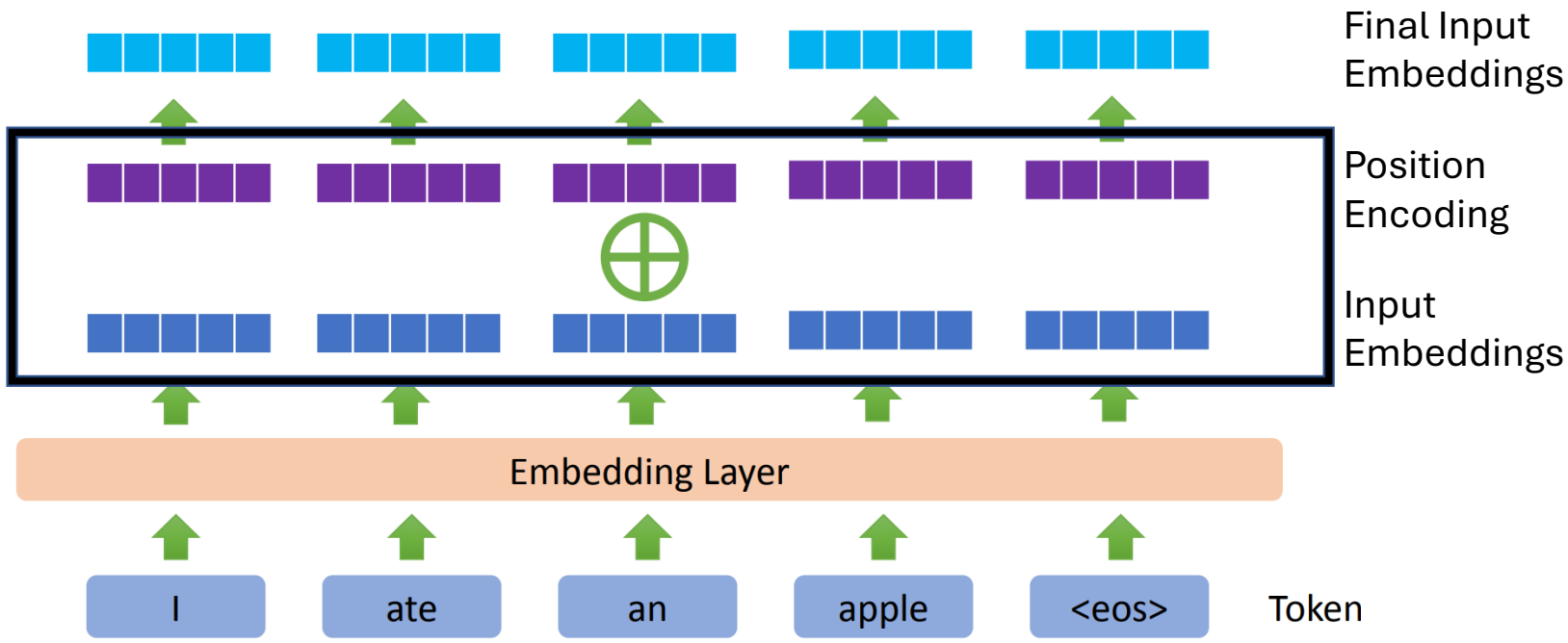


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Positional Encoding

Method 1: Position Encoding

- Add a unique fixed vector to each position



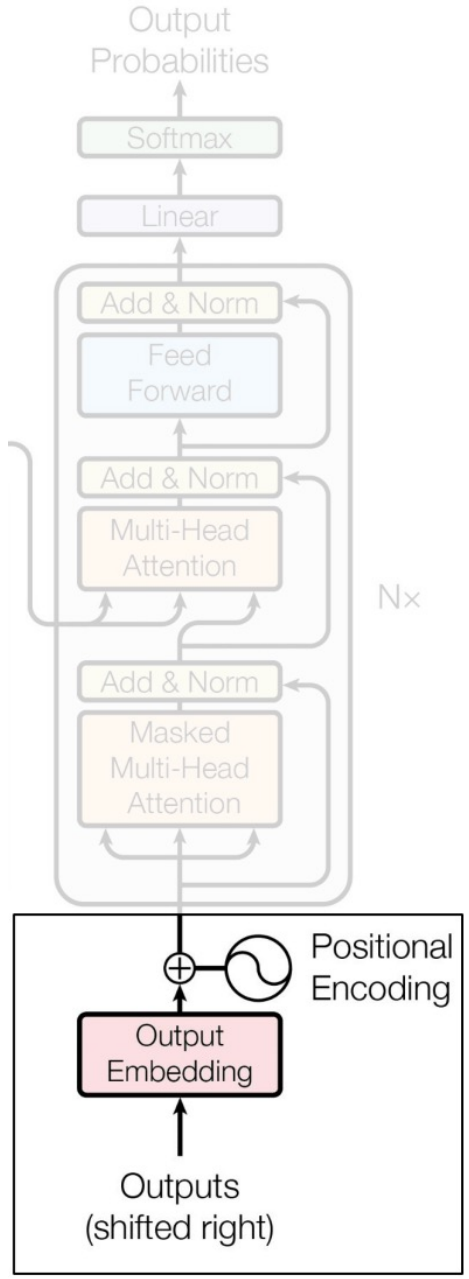


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Positional Encoding

Method 1: Position Encoding

- Add a unique fixed vector to each position

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

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

pos -> idx of the token in input sentence

i -> i^{th} dimension out of d

d model -> embedding dimension of each token

Different calculations for odd and even embedding indices

```
Positional Encoding:
          0     1      2      3      4
Dim 1 0.000 0.841  0.909  0.141 -0.757
Dim 2 1.000 0.540 -0.416 -0.990 -0.654
Dim 3 0.000 0.025  0.050  0.075  0.100
Dim 4 1.000 1.000  0.999  0.997  0.995
Dim 5 0.000 0.001  0.001  0.002  0.003
```

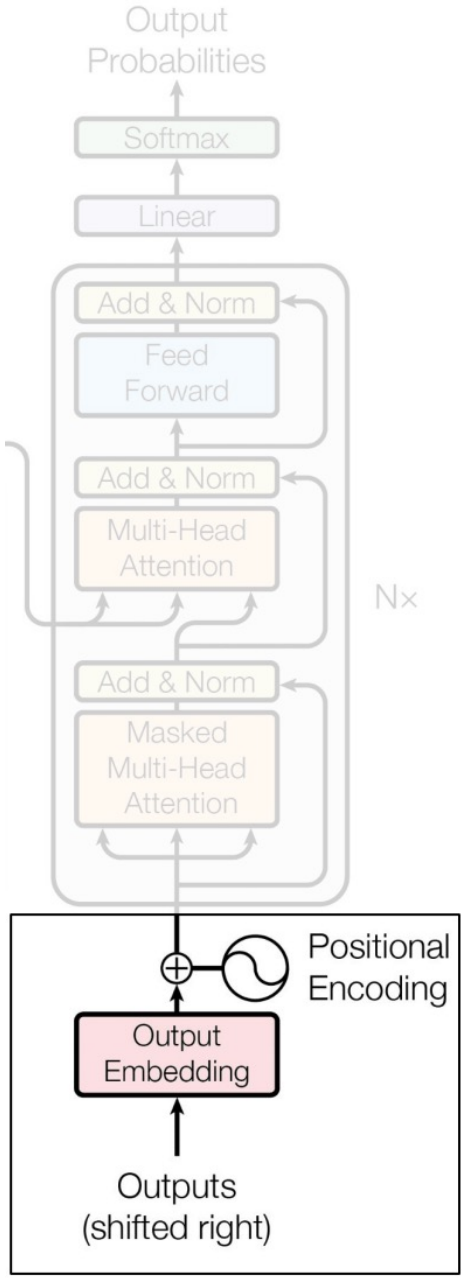
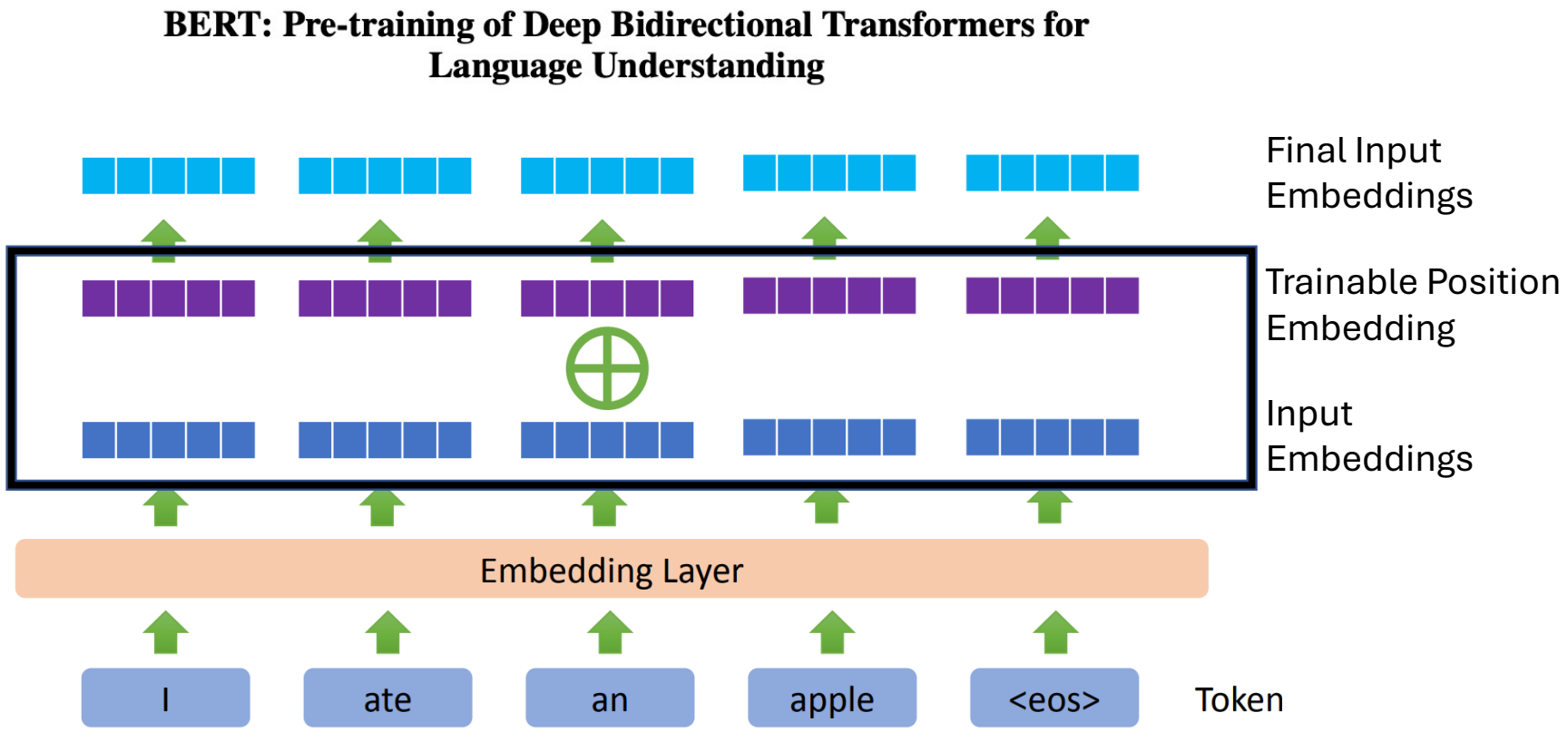


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Positional Encoding

Method 2: Position Embedding

- Add a trainable vector to each position



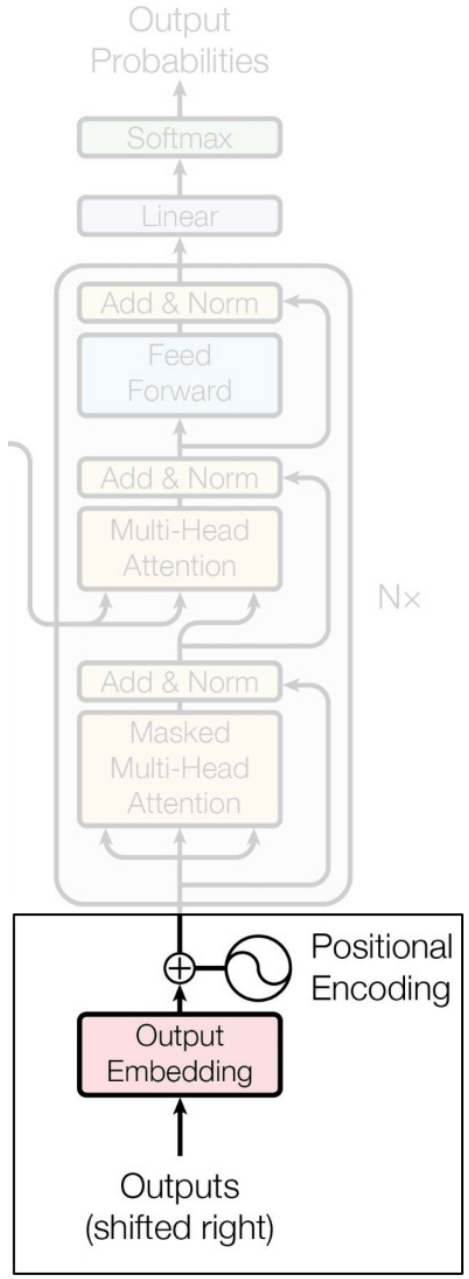
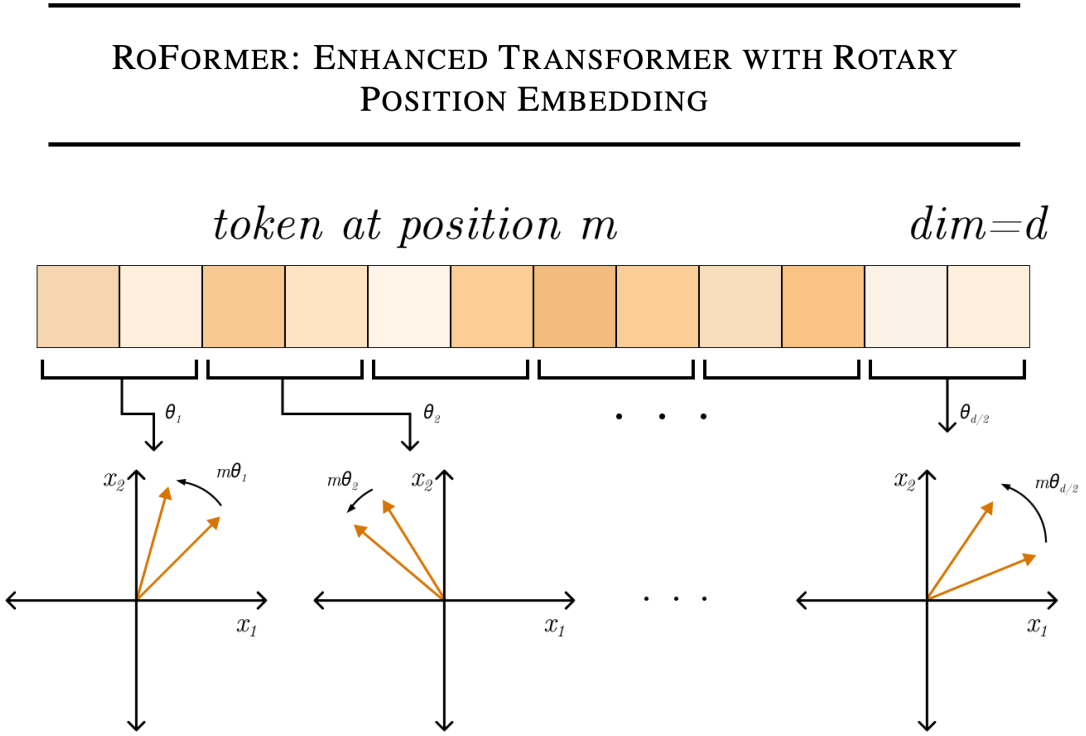


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Positional Encoding

Method 3: Rotary Position Embedding (RoPE)

- Rotate the input embedding based on its position



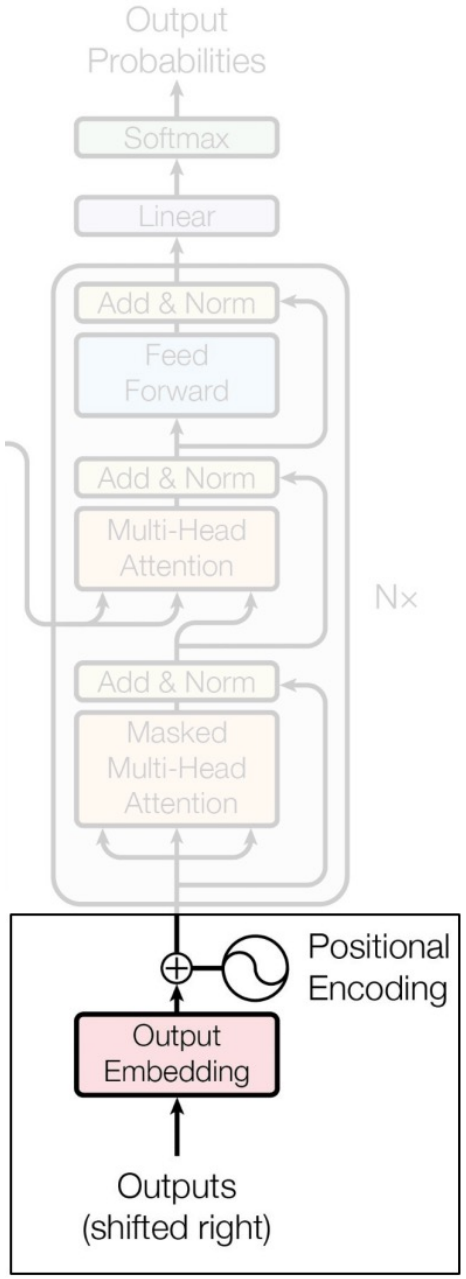


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Positional Encoding

Position Encoding / Embedding:

- These methods are **absolute**.
- It assigns a unique fixed vector to "position 1," "position 2," and so on.
- The model then has to learn that if word A is at position i and word B is at position i+5, they have a specific relationship.

Rotary Position Embedding (RoPE)

- RoPE naturally captures **relative** distance. As two tokens get further apart, their dot product decreases.
- Long-Context Extrapolation: since RoPE relies on rotation, we could interpolate these angles.
  - Allows a model trained on a 4k context window to be fine-tuned to handle 32k or even 128k tokens.

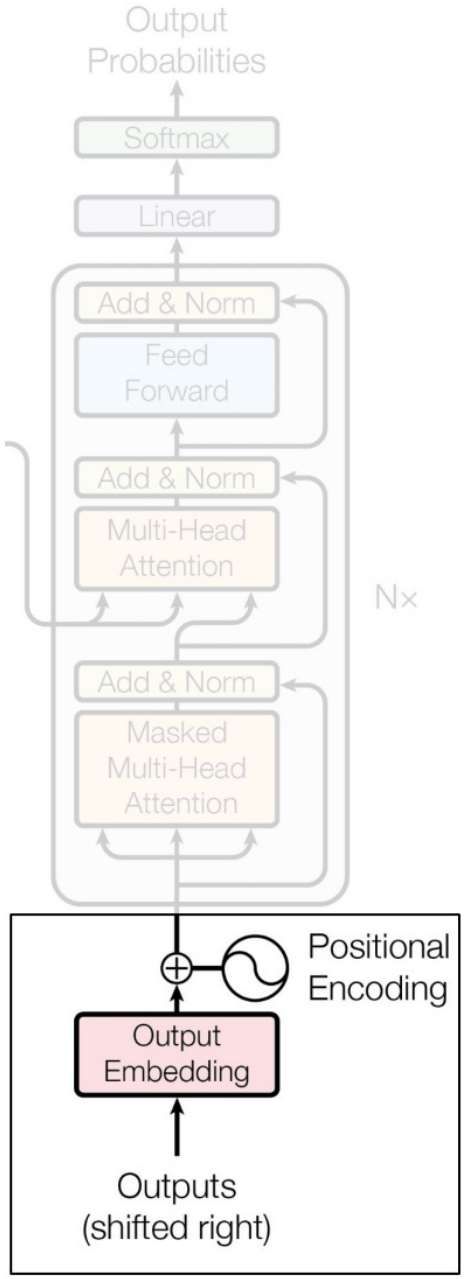


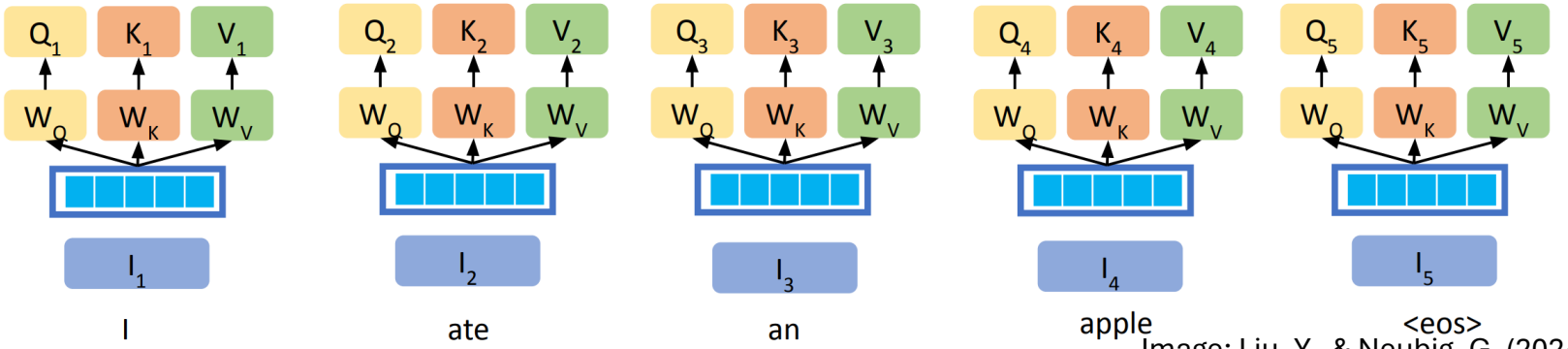
Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention

Q (query)
- What is the current token looking for?

K (key)
- What information does the current token have?

V (Value)
- The actual content of the current token.



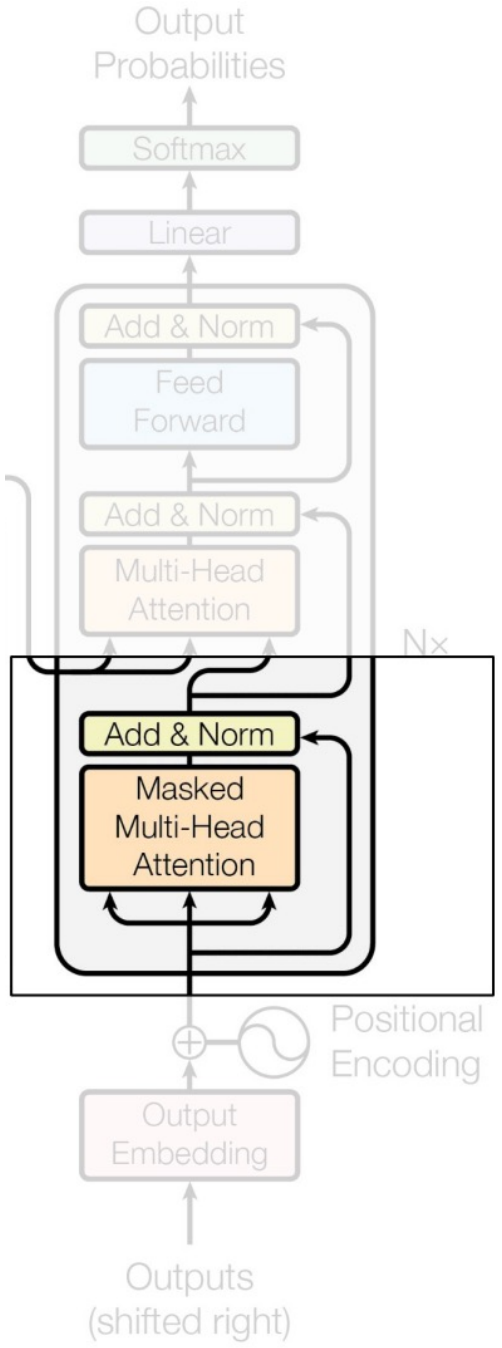
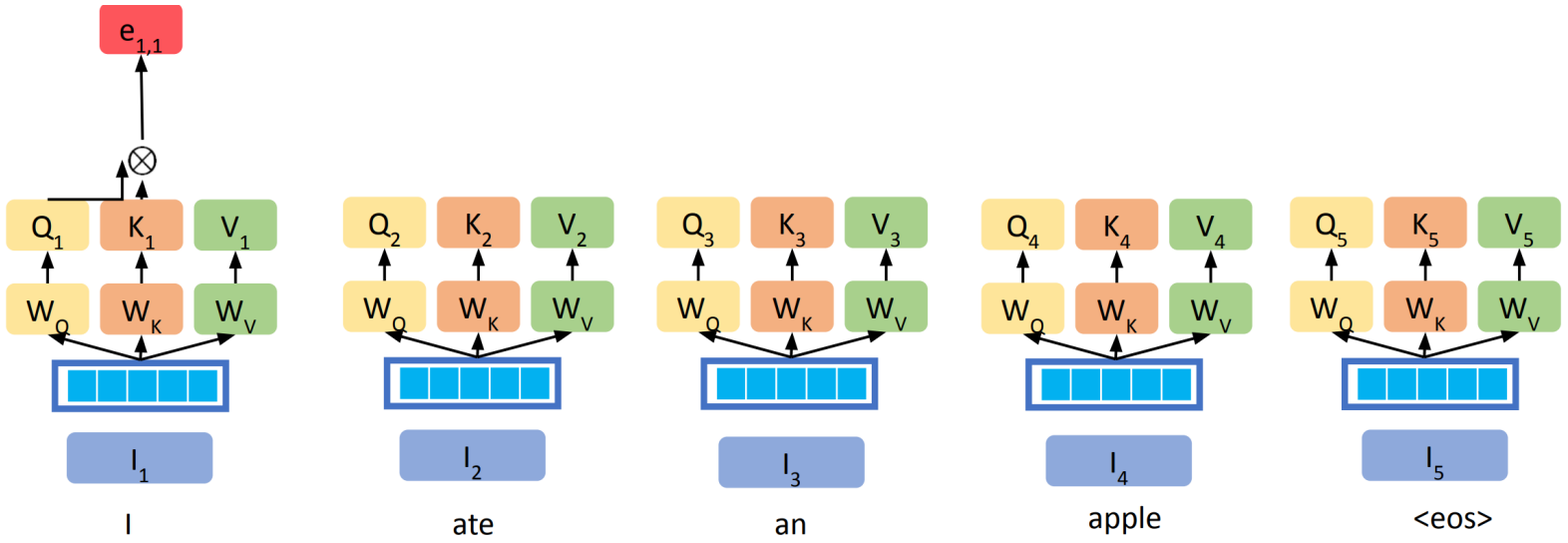


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention



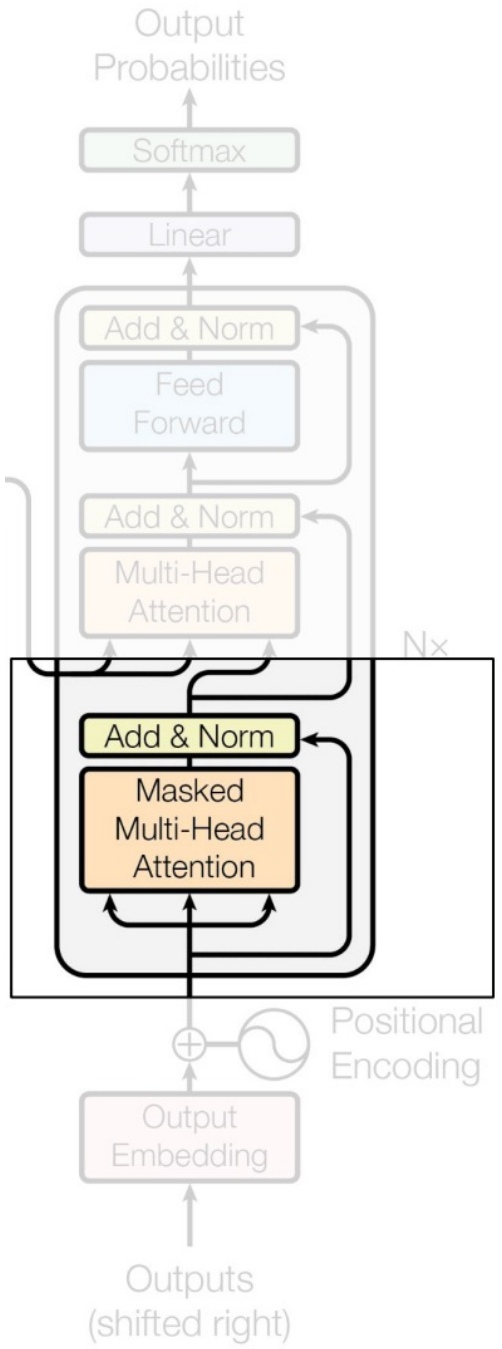
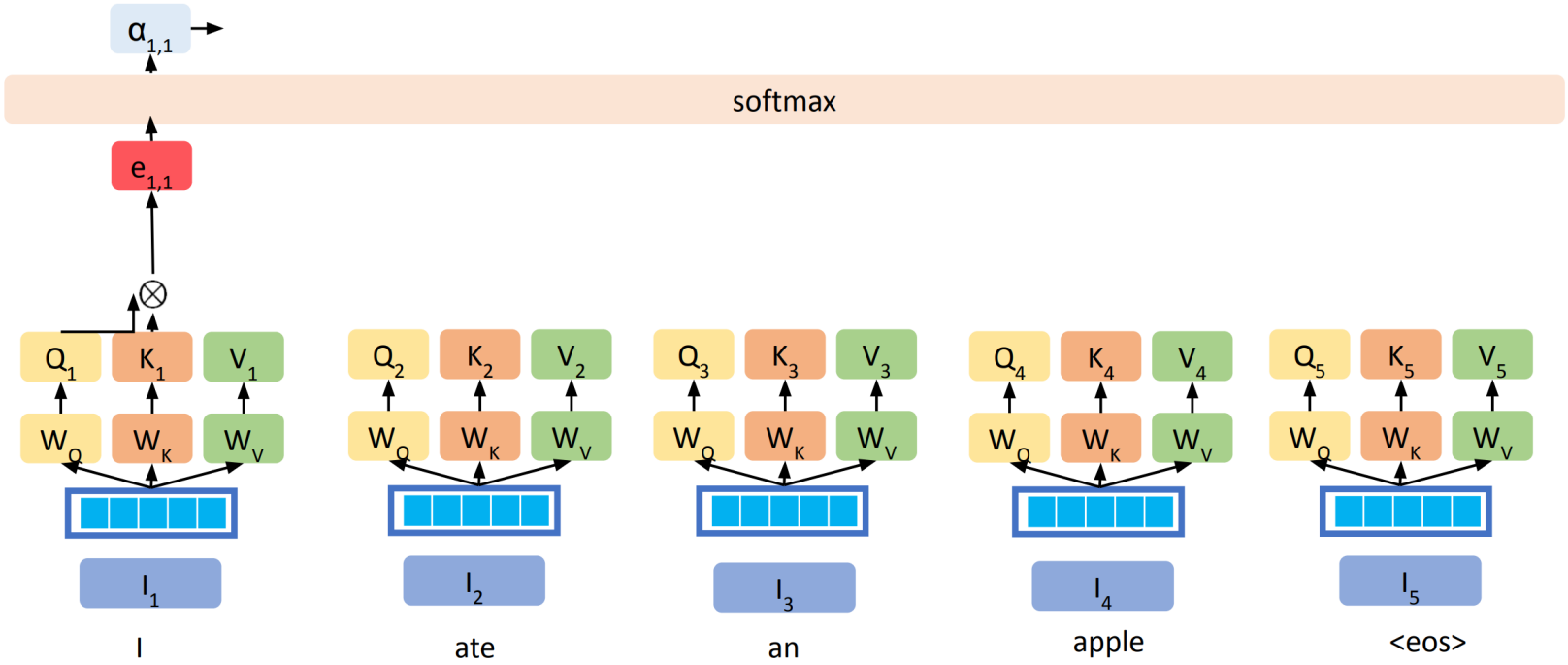


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention



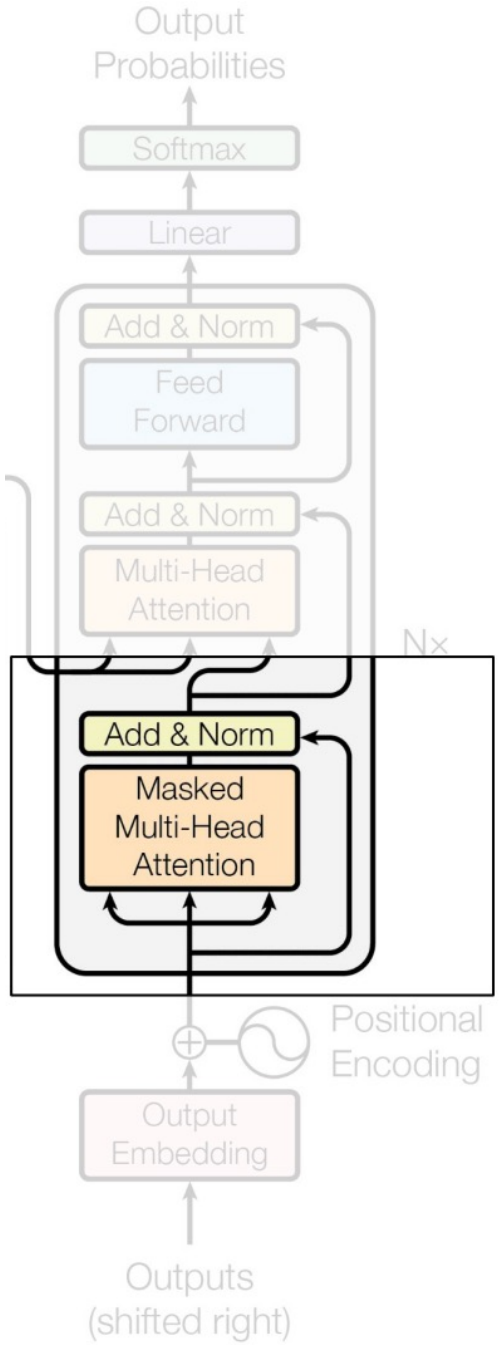
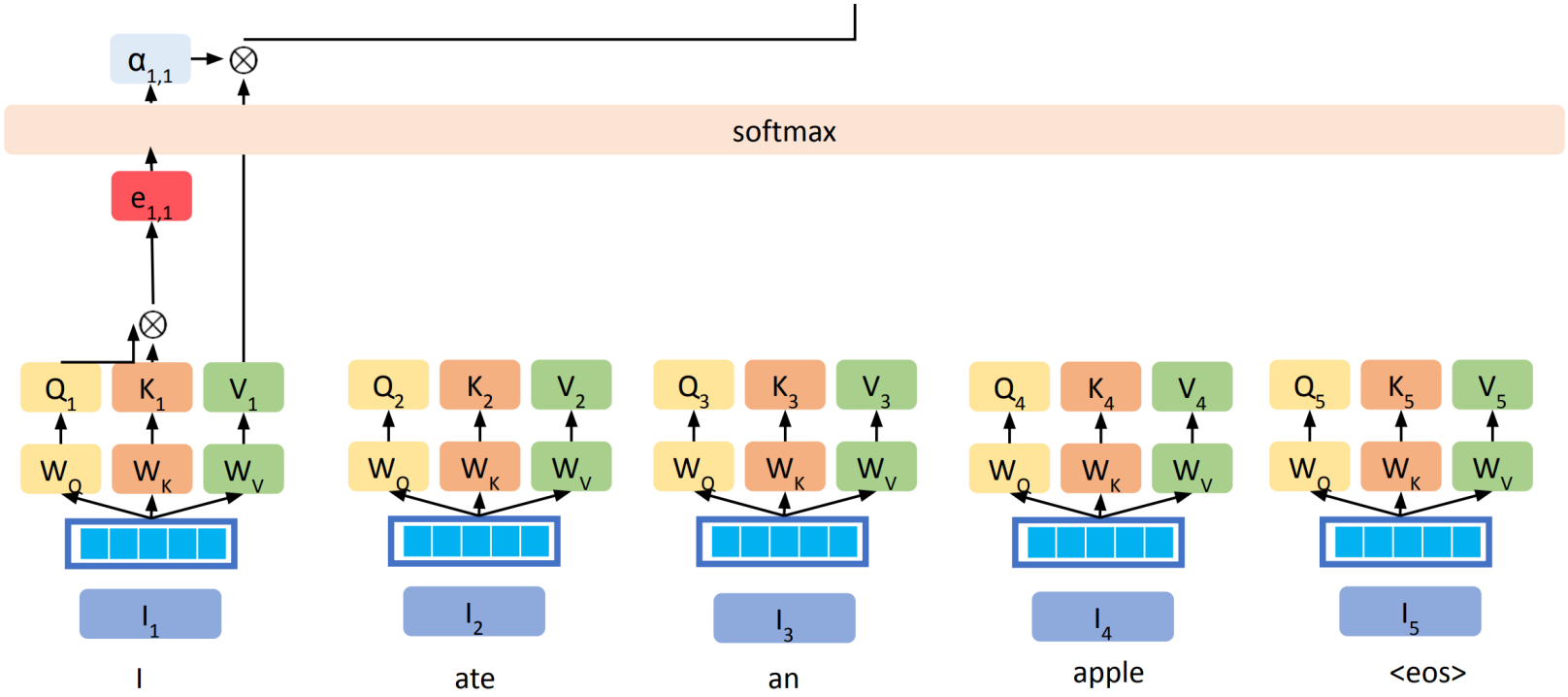


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention



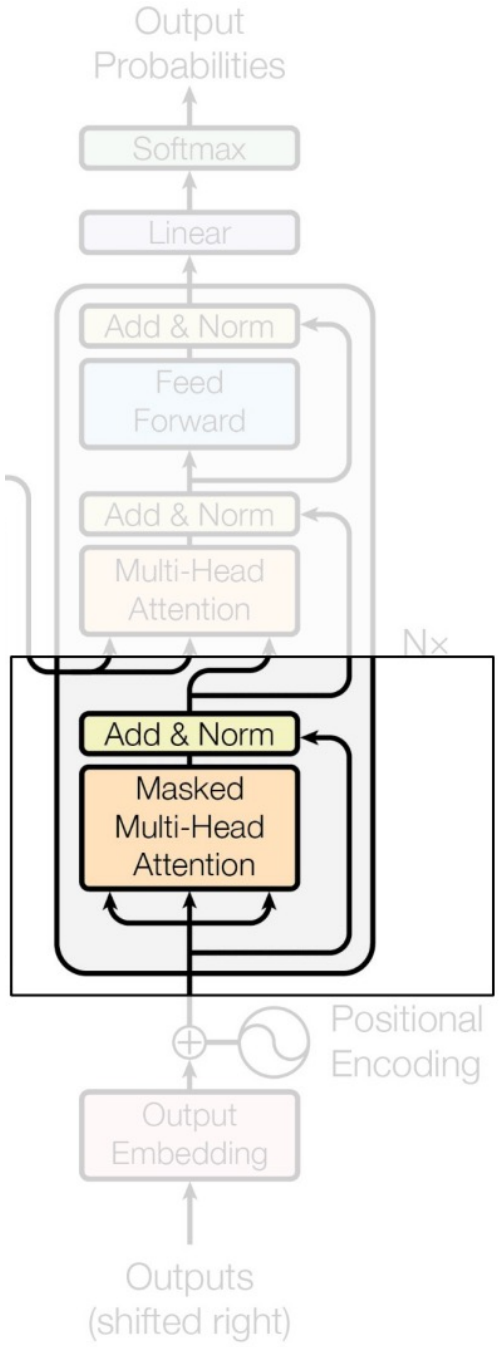
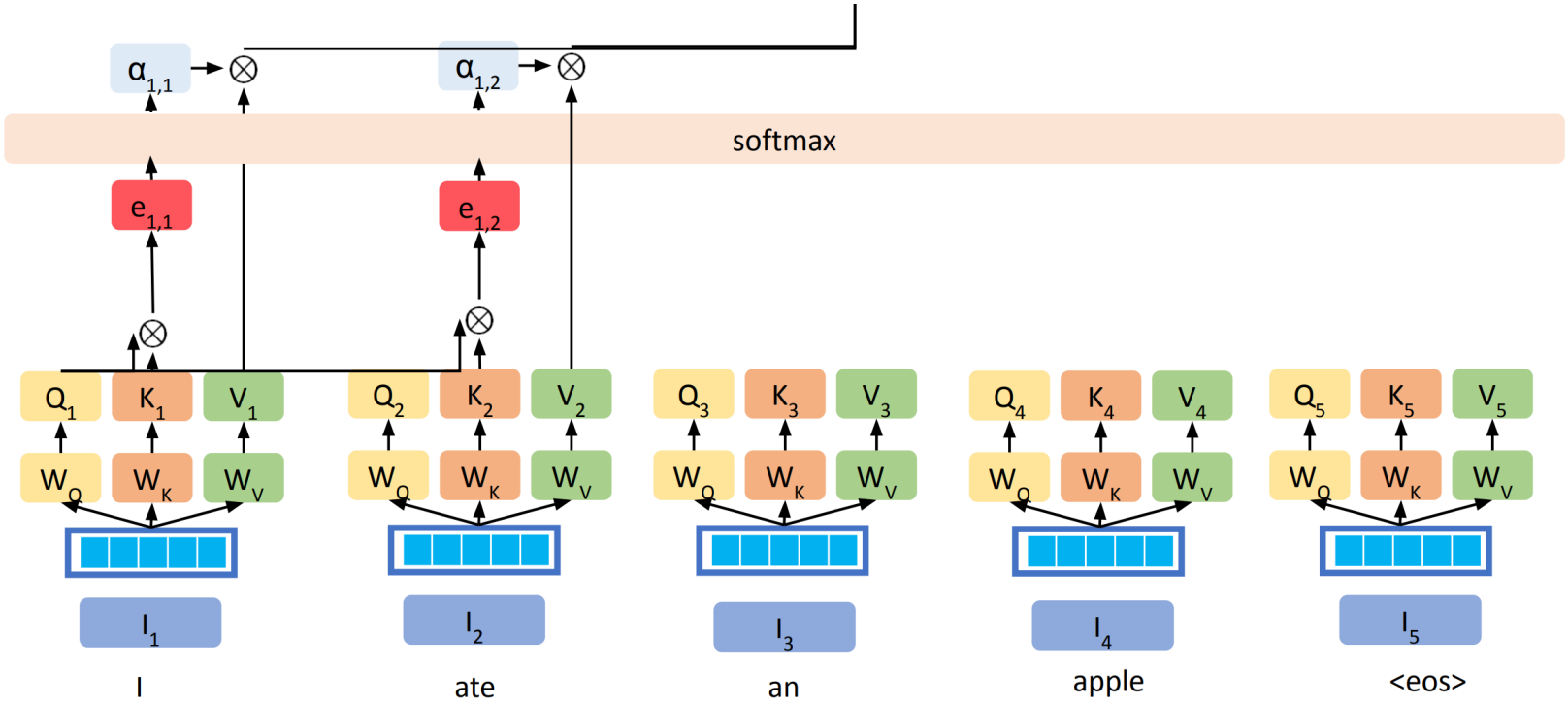


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention



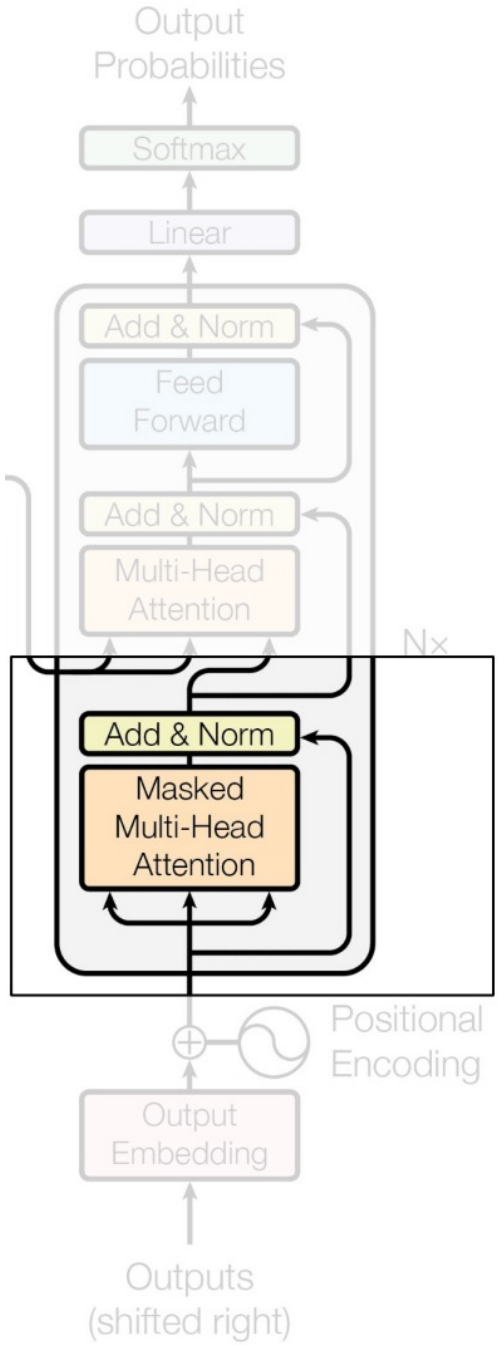
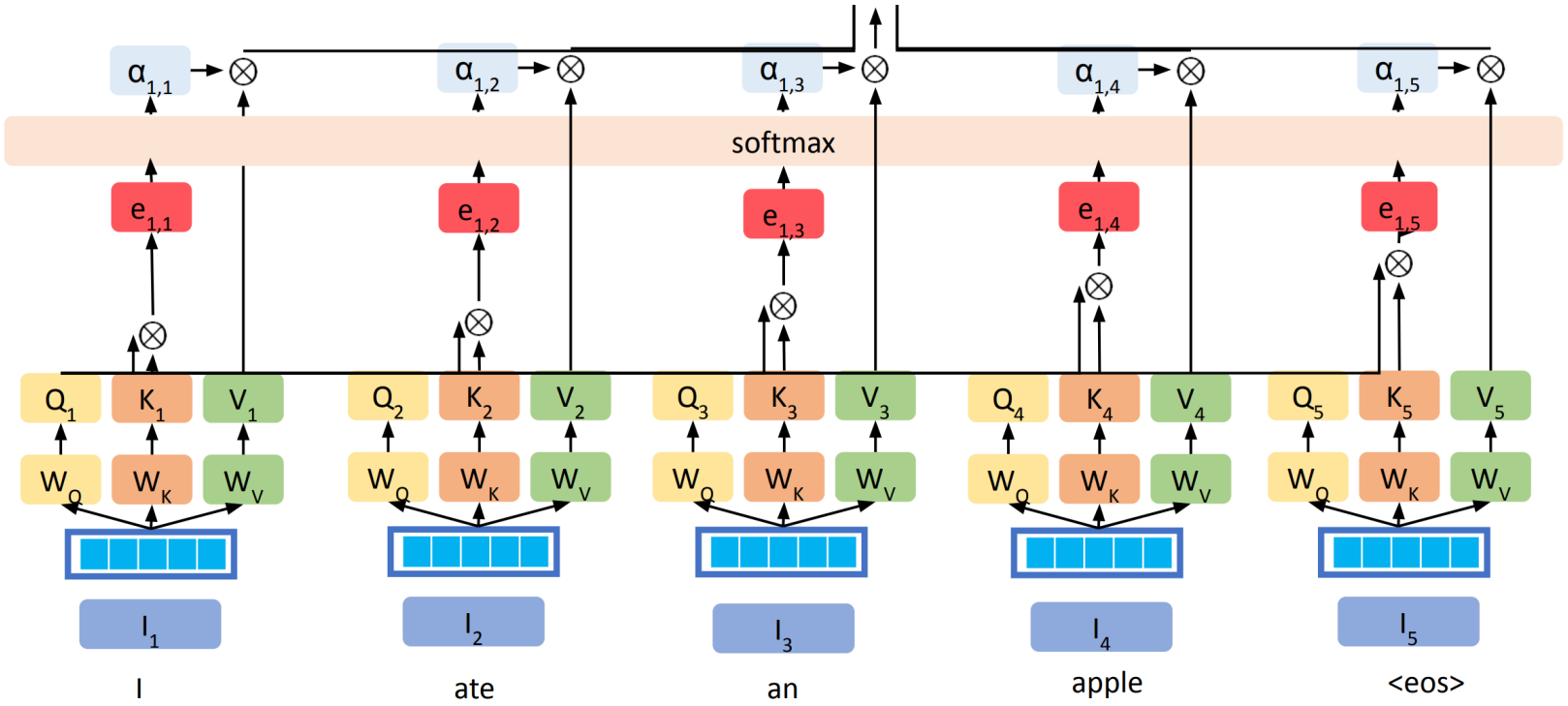


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention



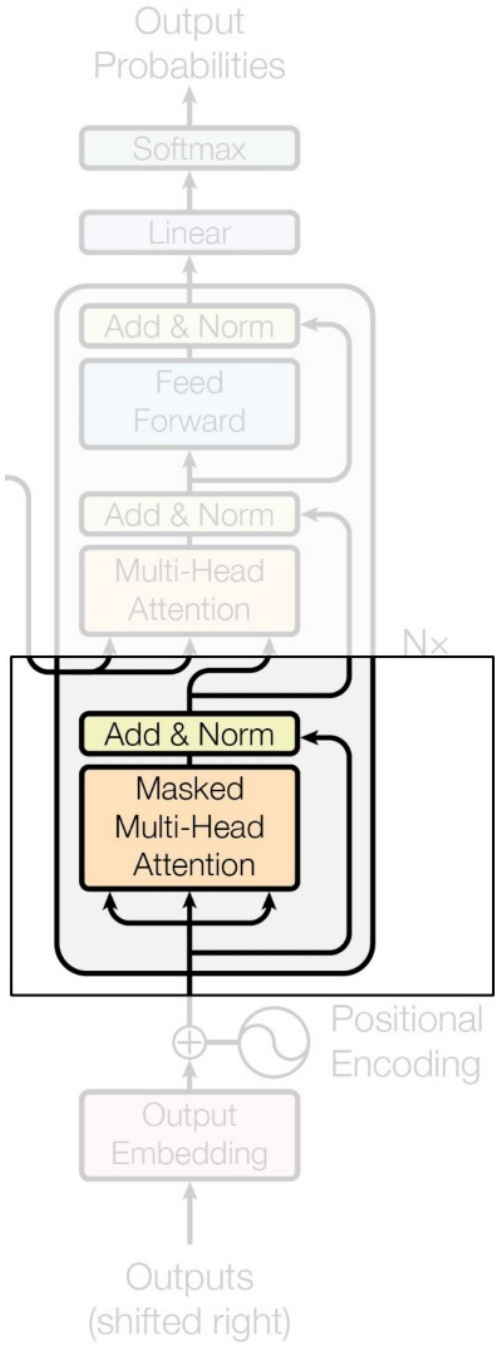
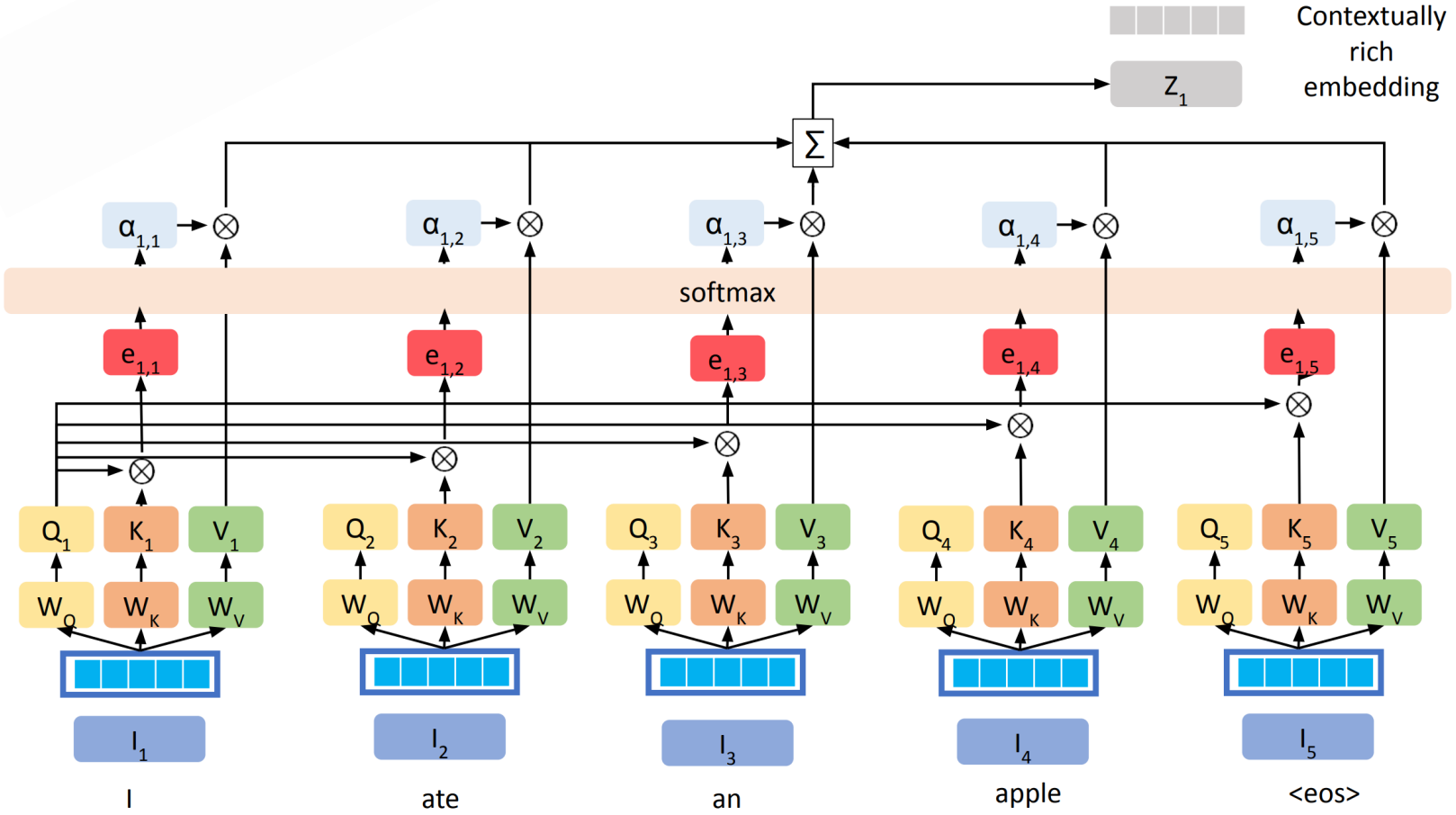


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention



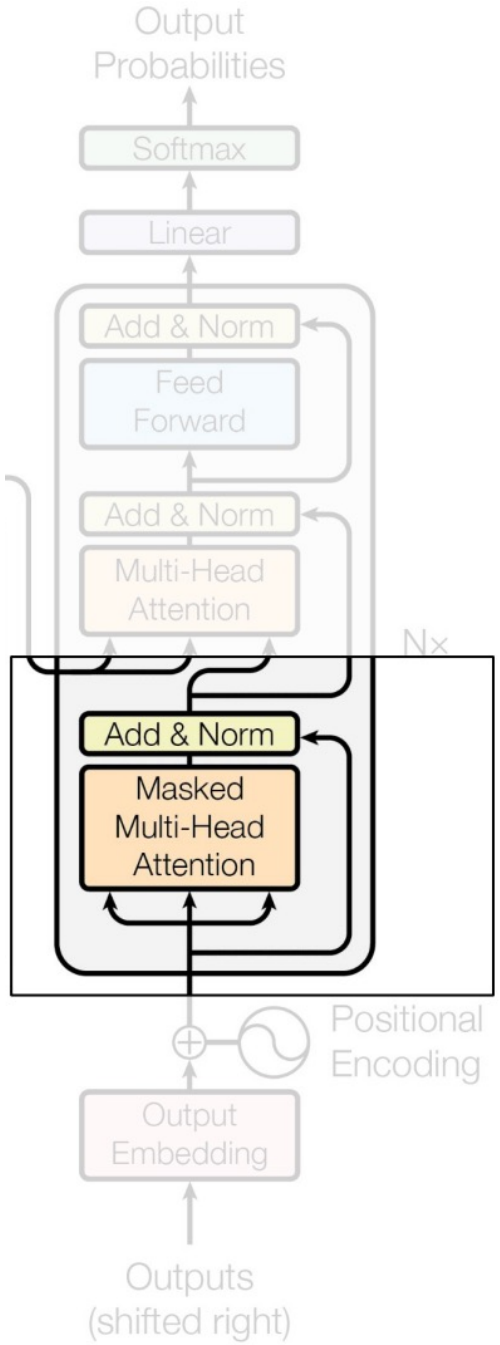


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention

Parallelized!

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

Q, K, V: matrices with size (T, d)

T: length of the sequence

Preserve the original variance

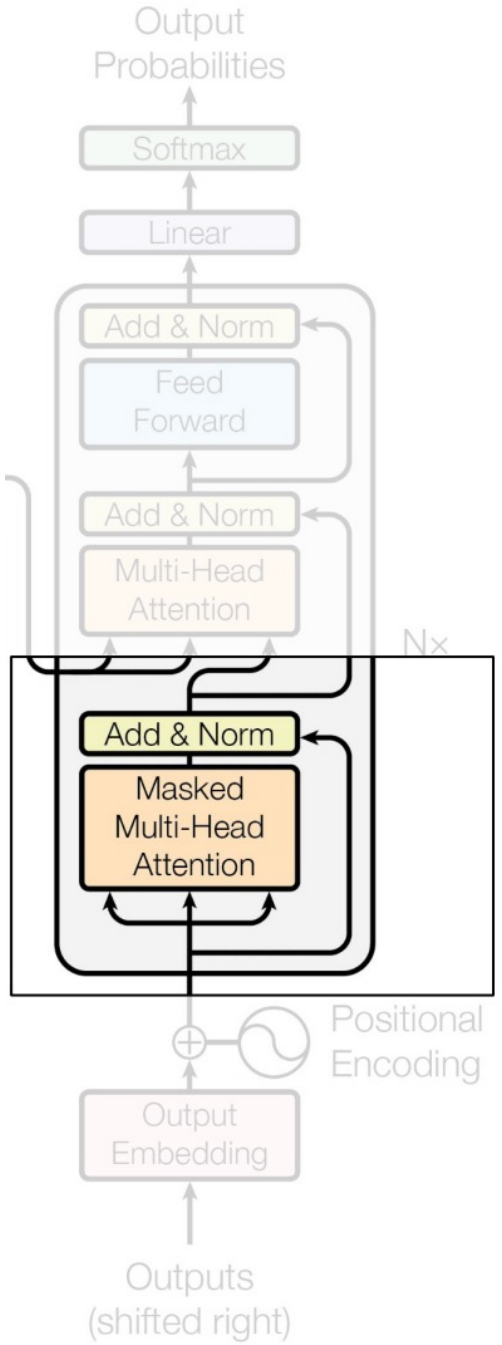
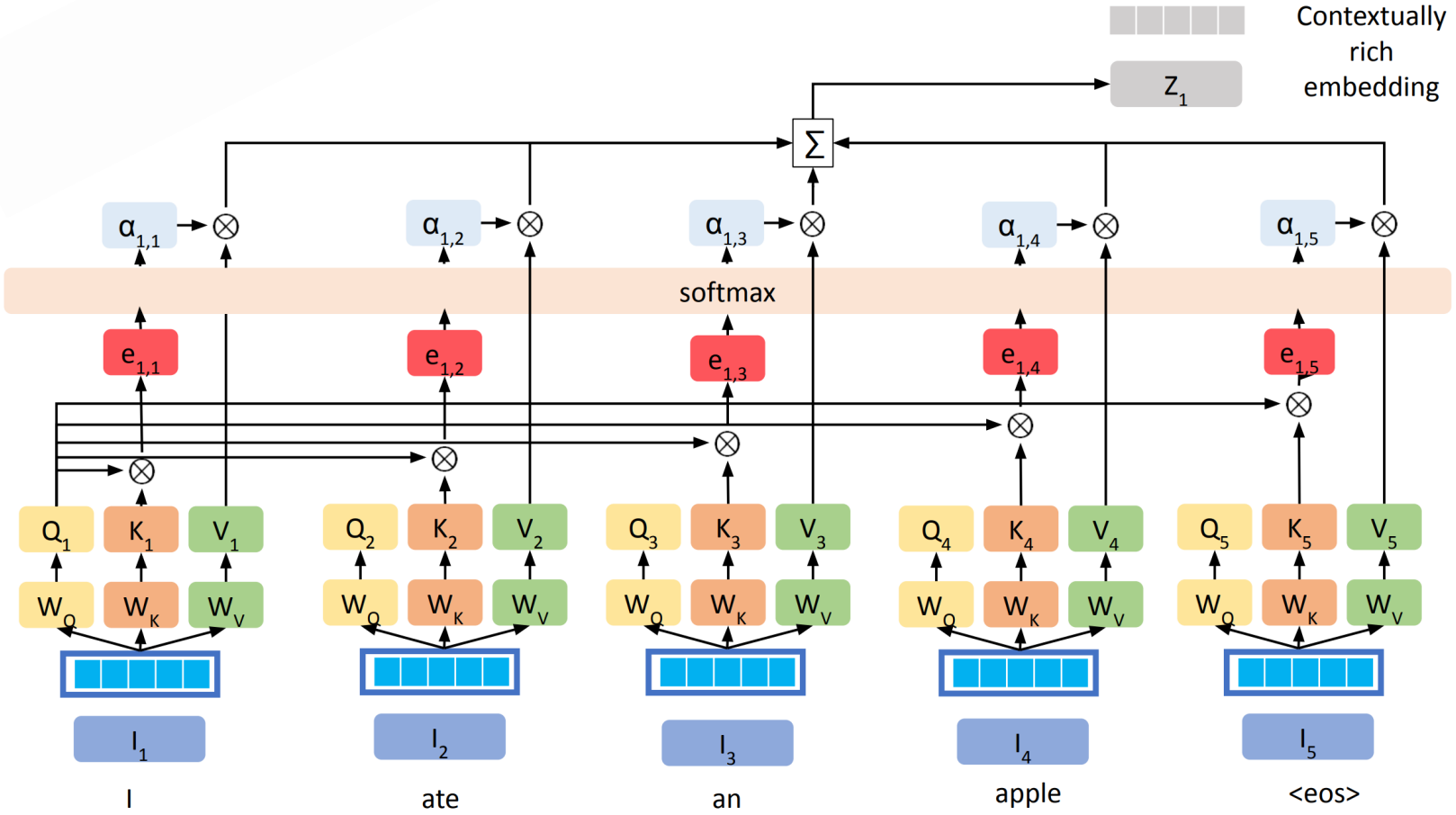


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention



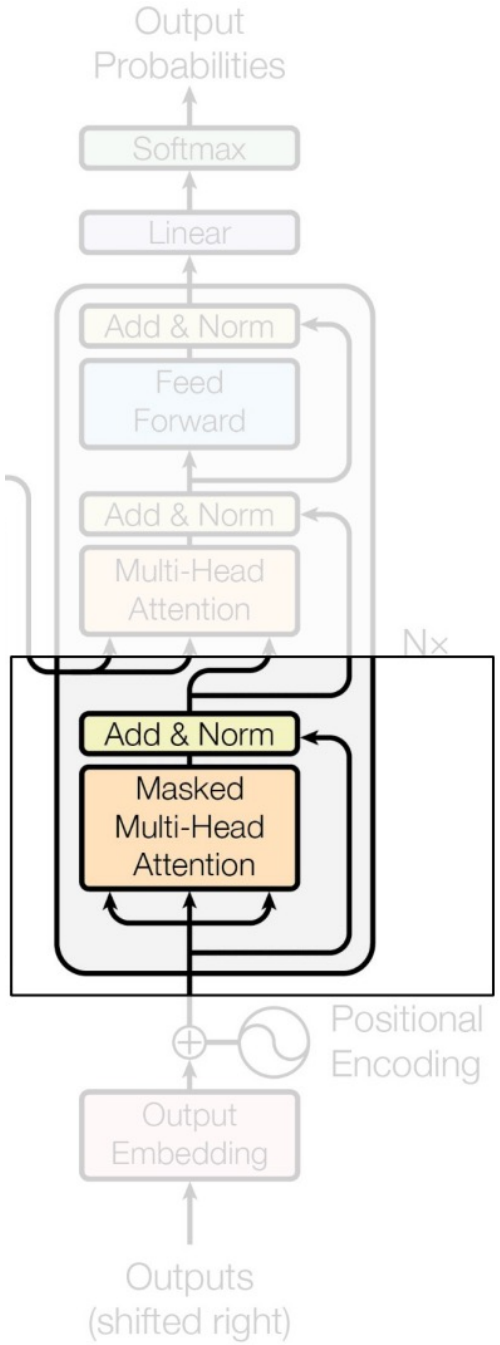
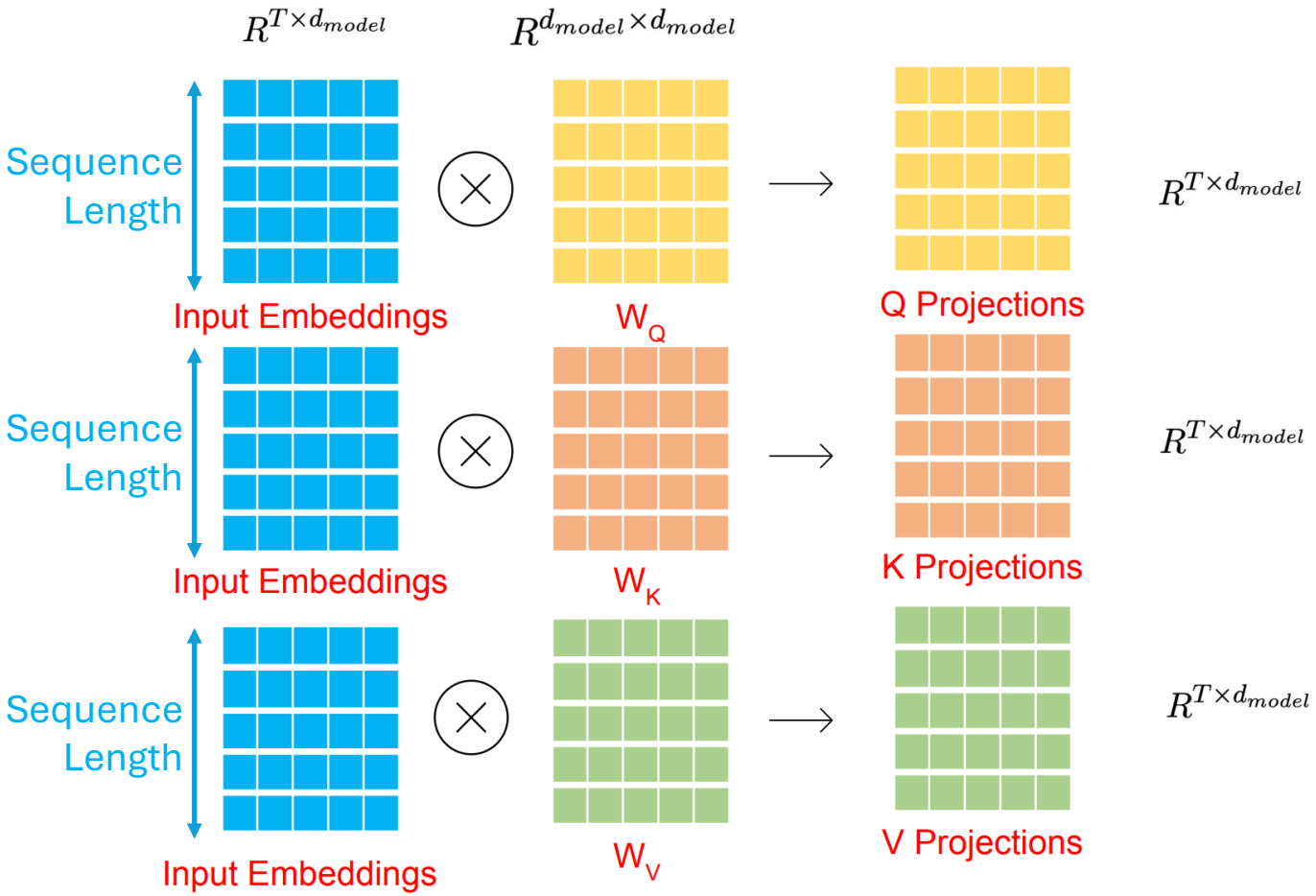


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention



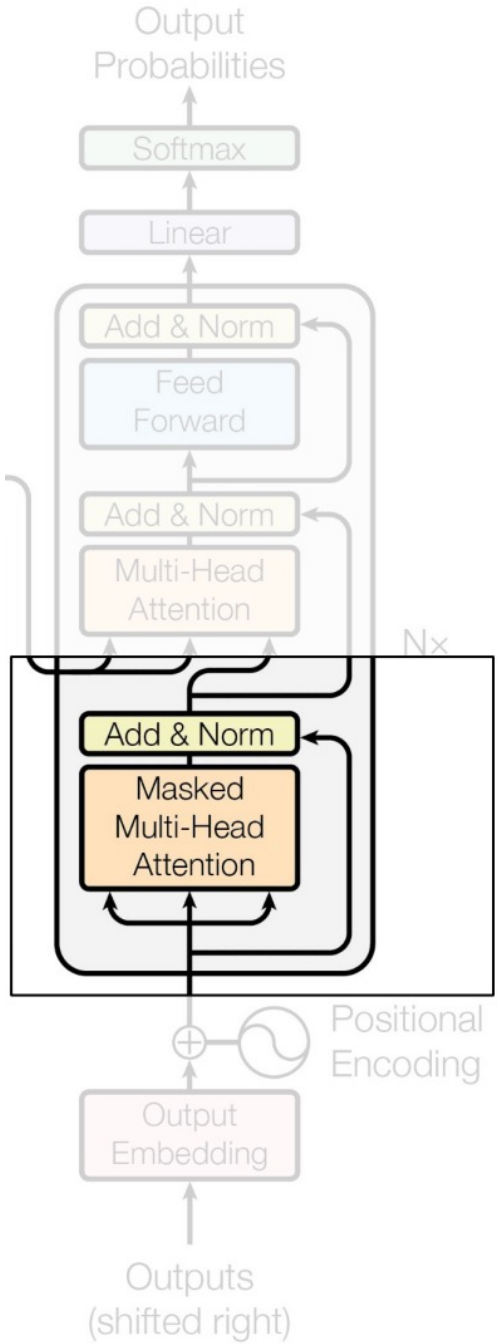
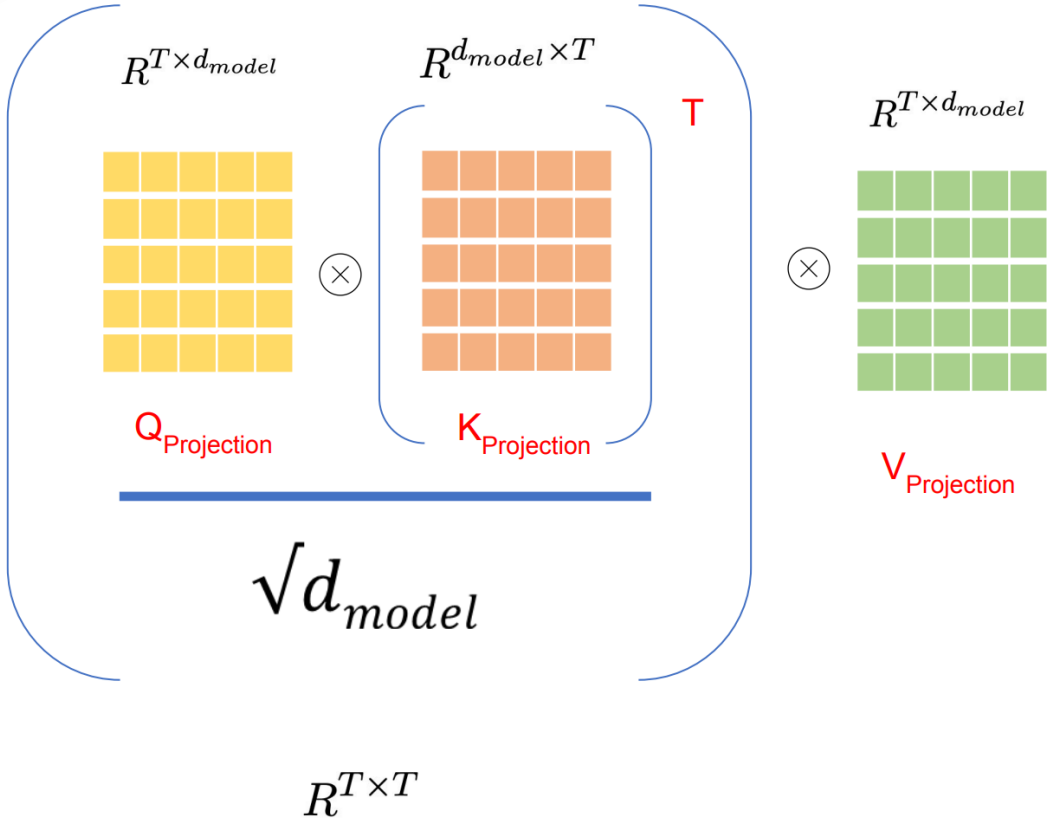


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention

softmax



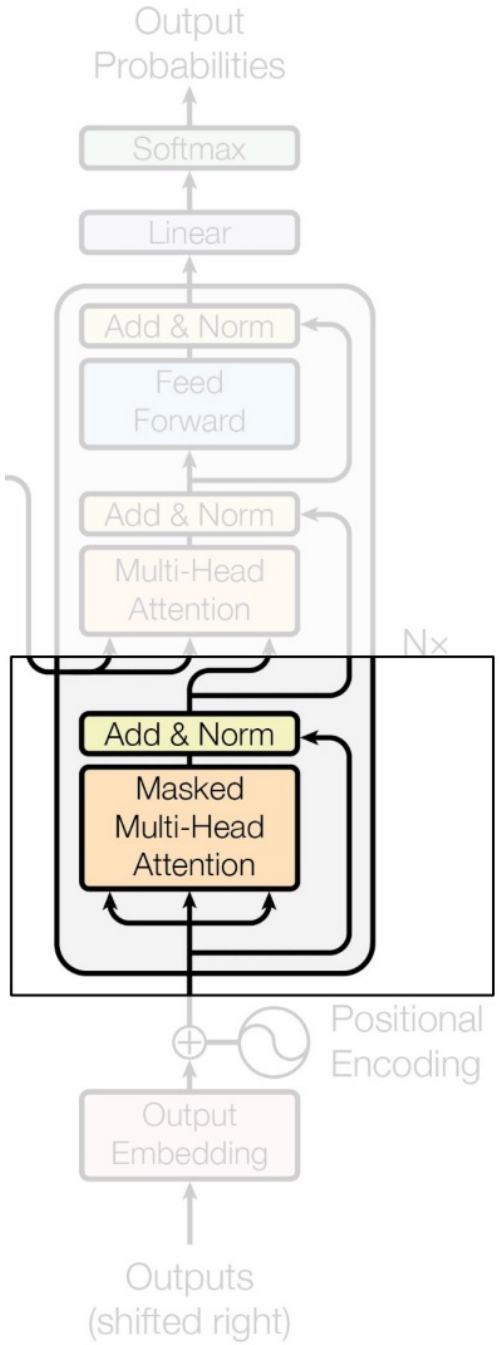
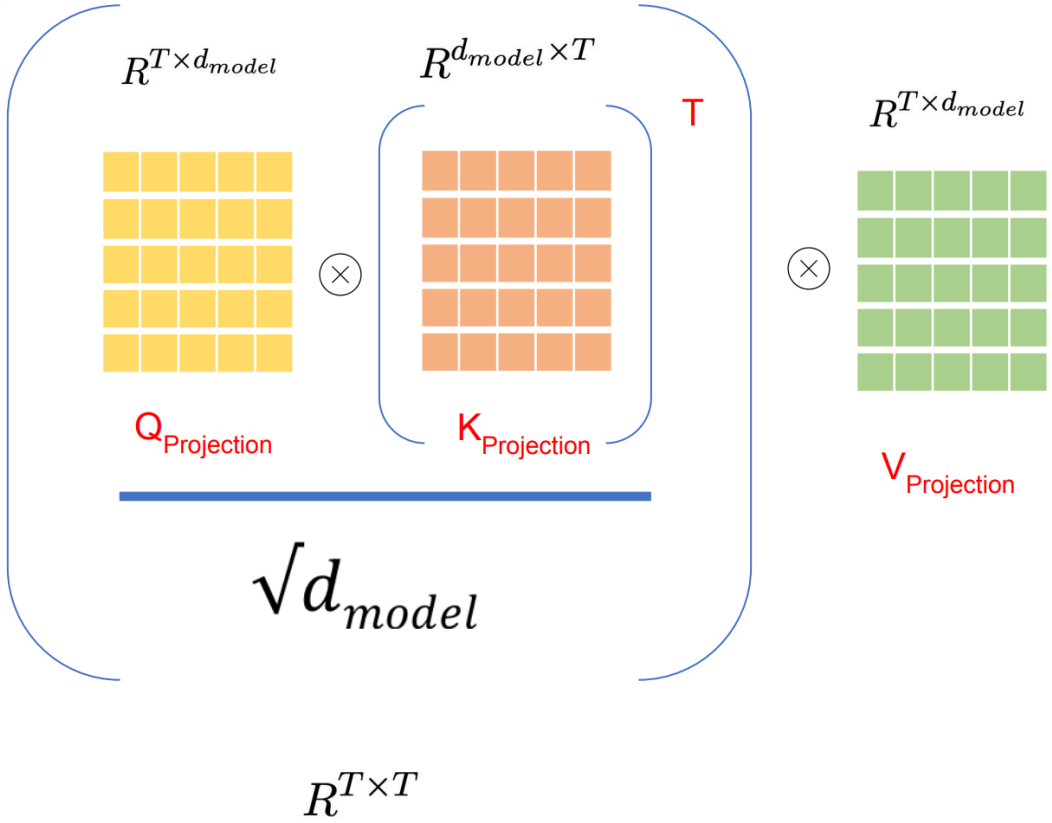


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Attention

Complexity: $O(T^2 d_{model})$

softmax



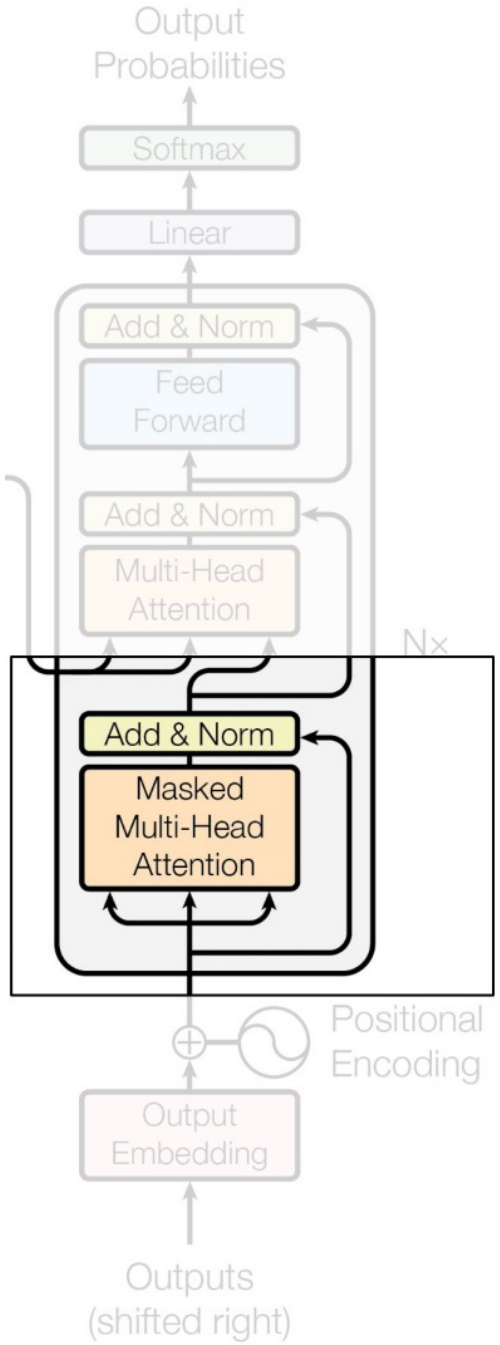
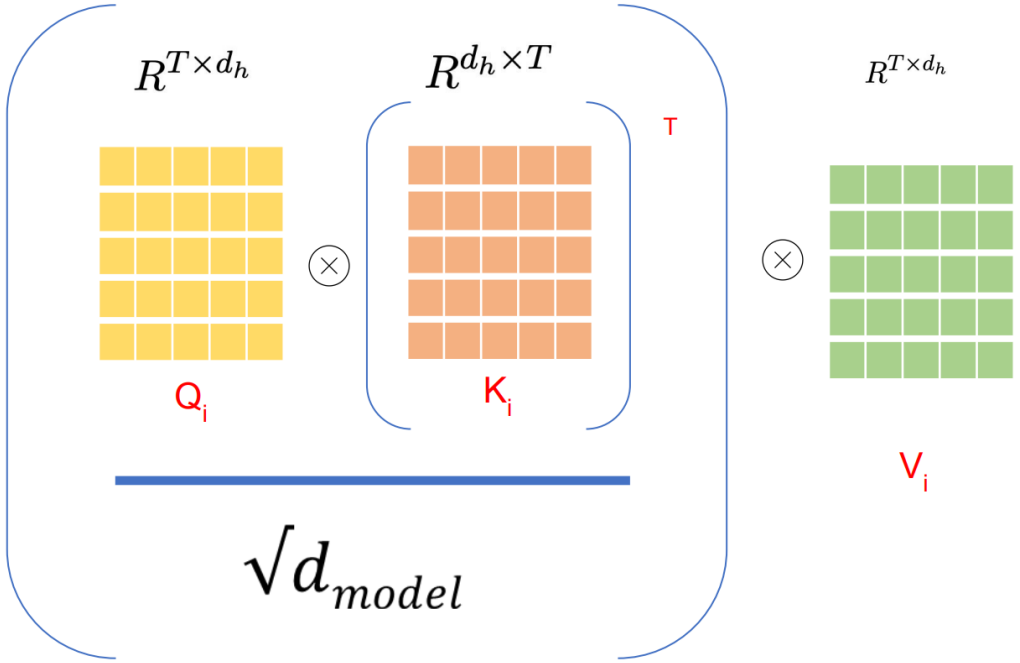


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Multi-Head Attention

$$d_h = \frac{d_{model}}{h}$$

softmax



for all i ∈ [1, h]

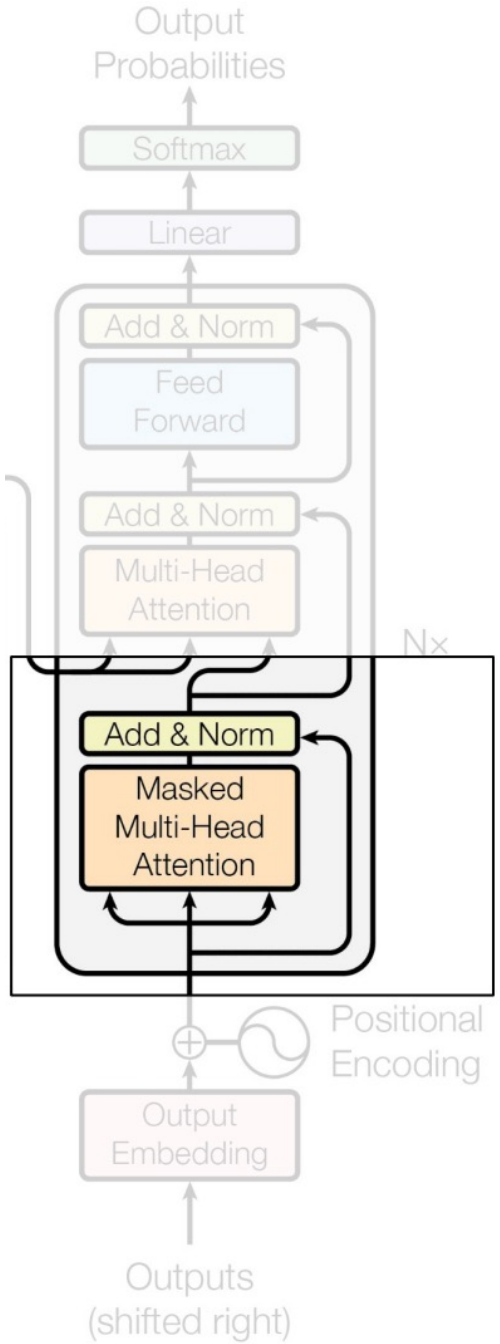
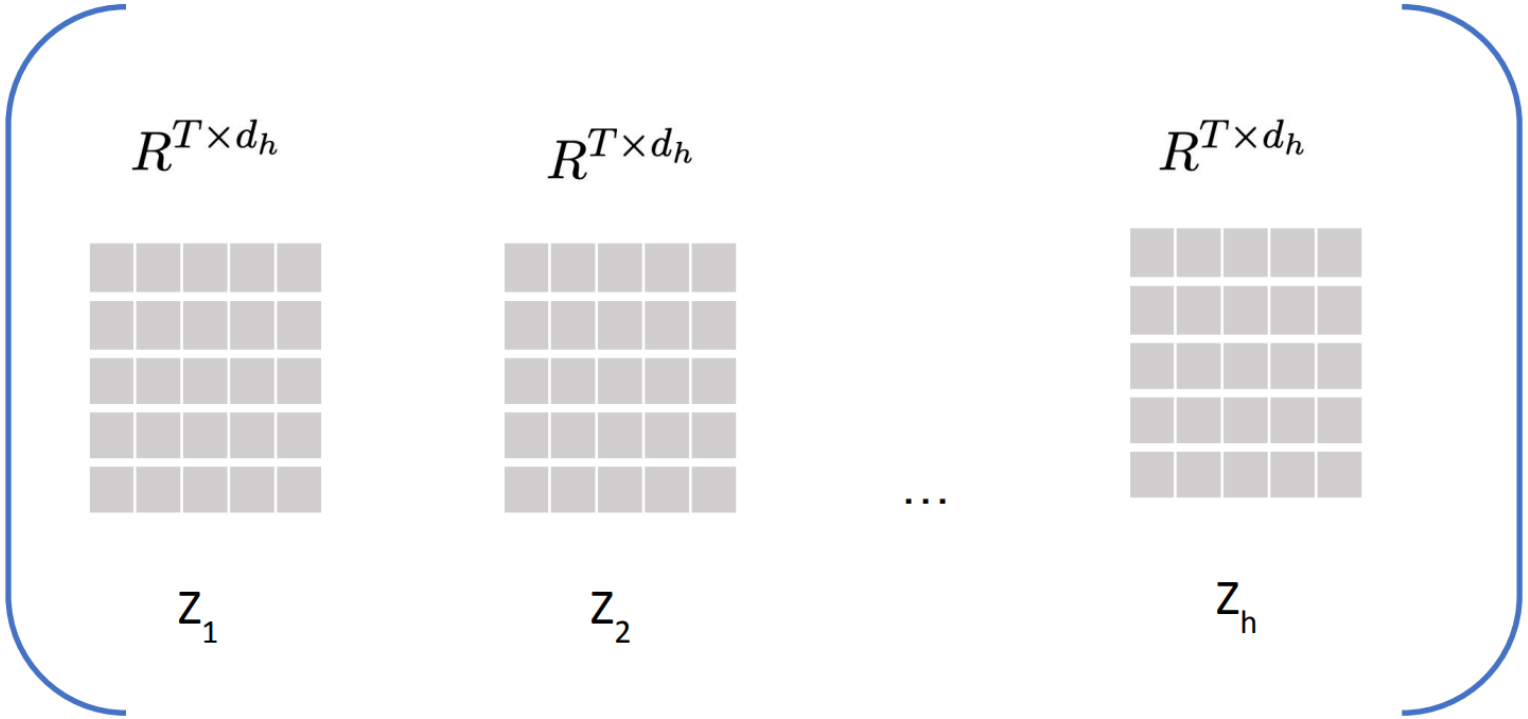


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Multi-Head Attention



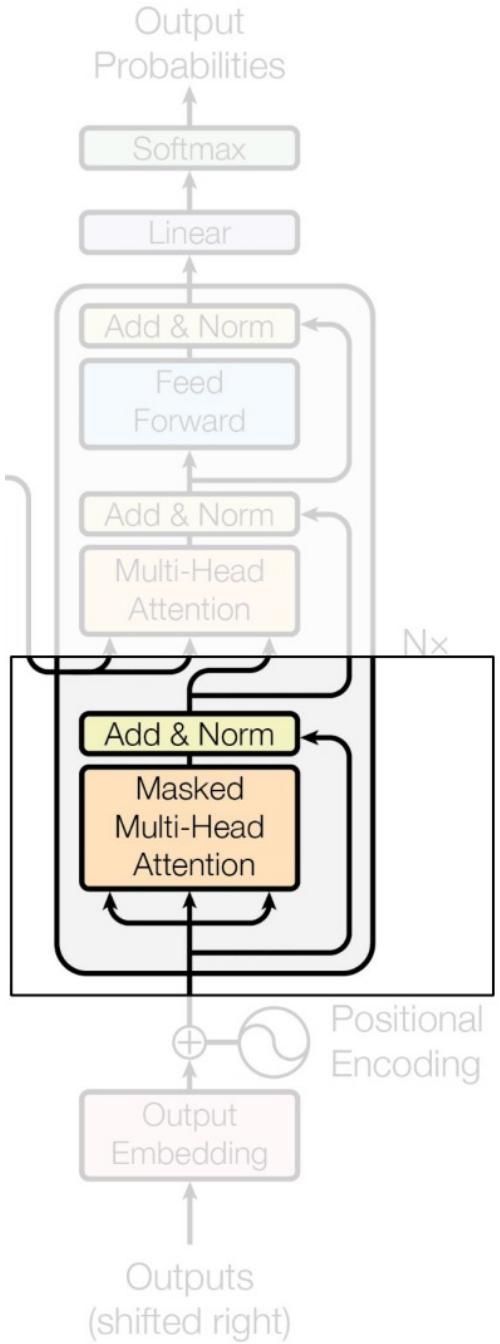


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Masked Attention

1	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
2	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
3	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
4	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
5	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
6	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>
7	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>

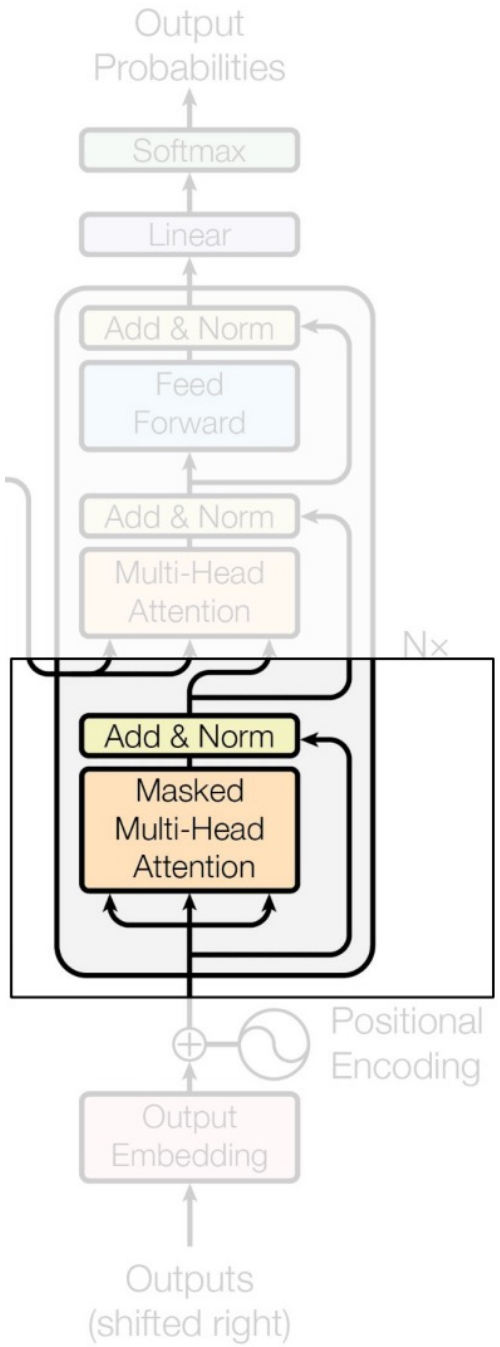


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Masked Attention

1	<sos>	- ∞	- ∞	- ∞	- ∞	- ∞	- ∞
2	<sos>	Ich	- ∞	- ∞	- ∞	- ∞	- ∞
3	<sos>	Ich	habe	- ∞	- ∞	- ∞	- ∞
4	<sos>	Ich	habe	einen	- ∞	- ∞	- ∞
5	<sos>	Ich	habe	einen	Apfel	- ∞	- ∞
6	<sos>	Ich	habe	einen	Apfel	gegessen	- ∞
7	<sos>	Ich	habe	einen	Apfel	gegessen	<eos>

Softmax - ∞ -> 0

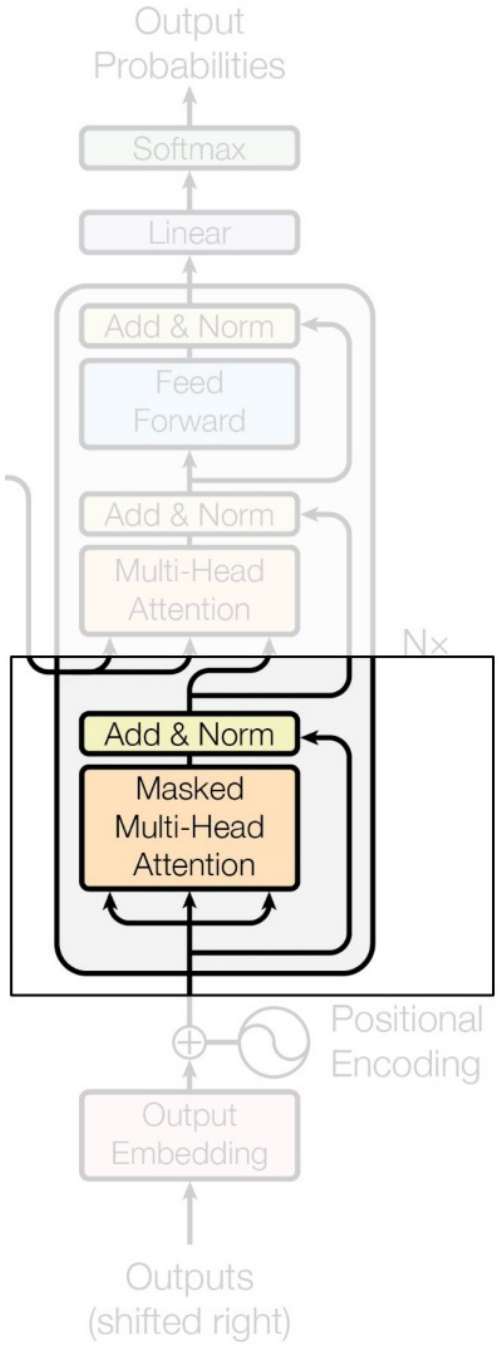
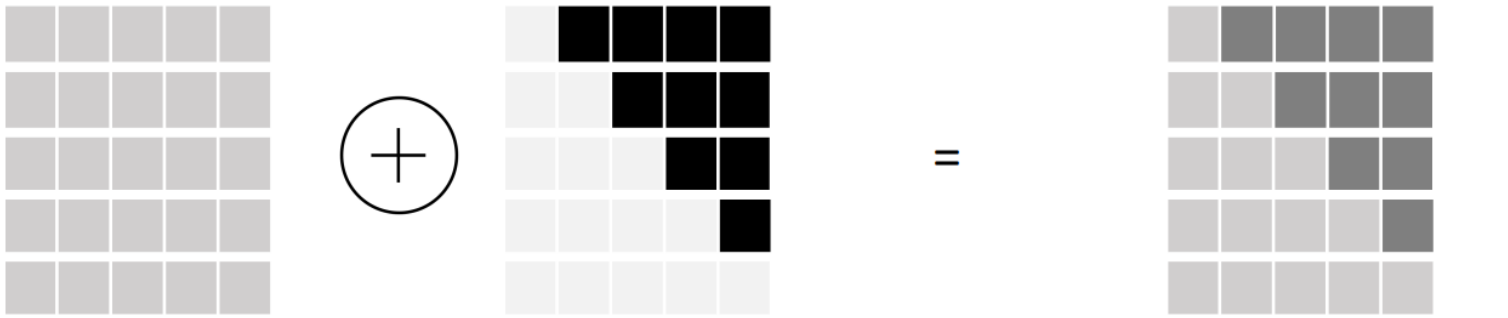


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Masked Attention

$R^{T \times T}$ $R^{T \times T}$



QK^T Attention Mask: M Masked Attention

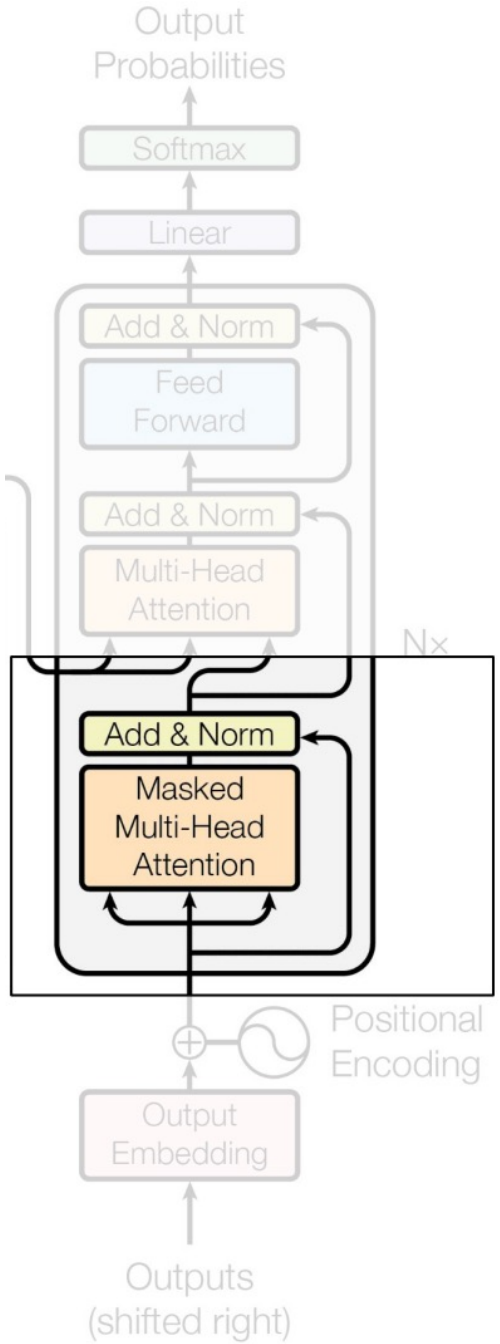
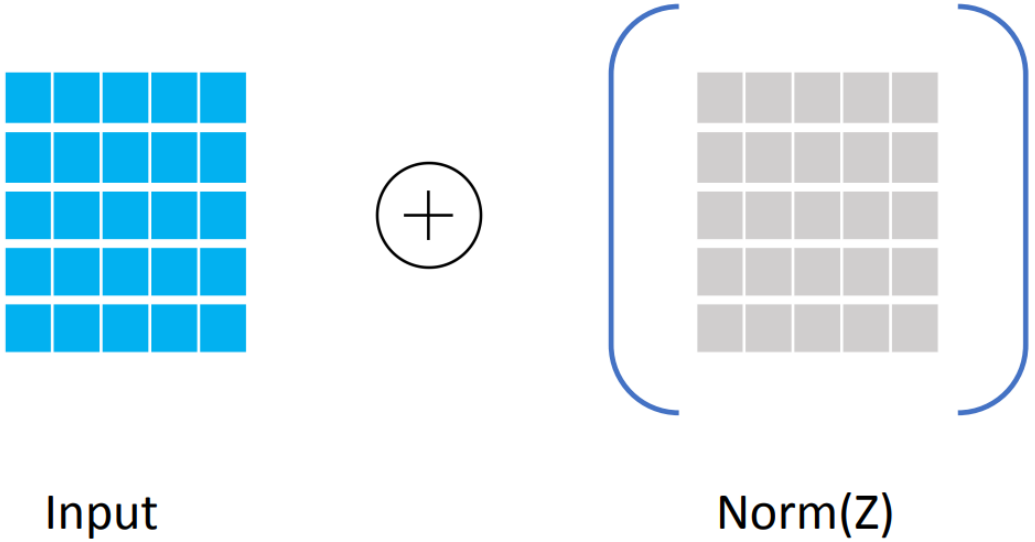


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Add & Norm



Add
- Avoid vanishing gradient
- Train deeper networks

Norm
- Normalize to be mean 0 std 1
- Stabilize training

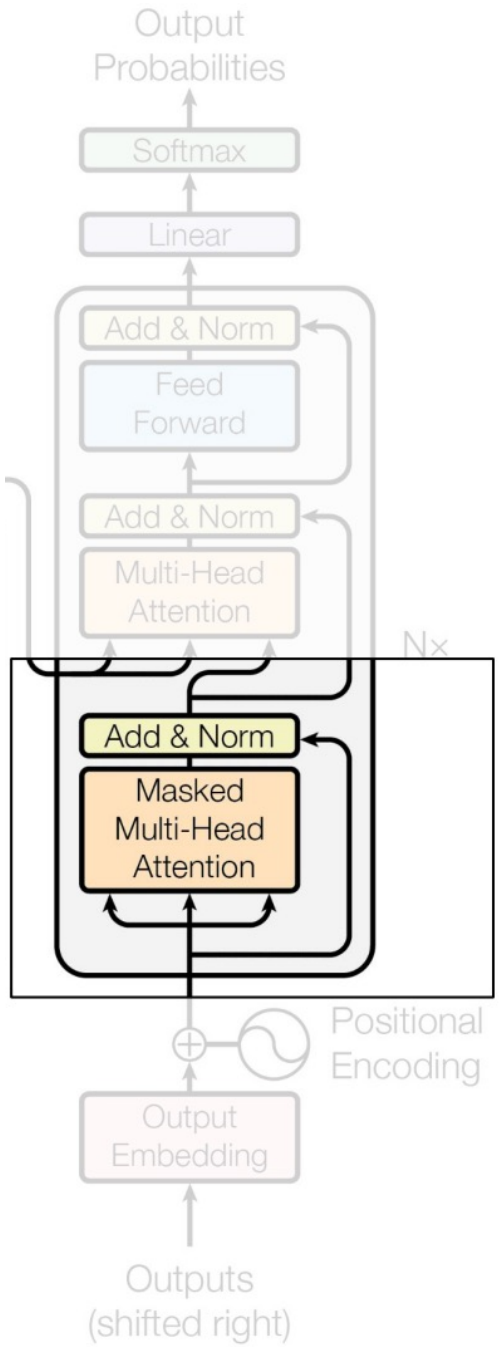
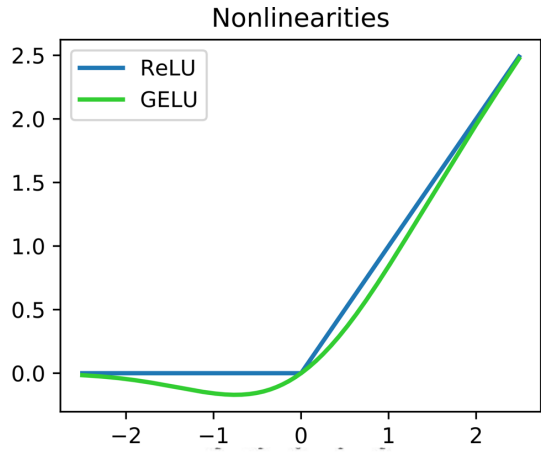
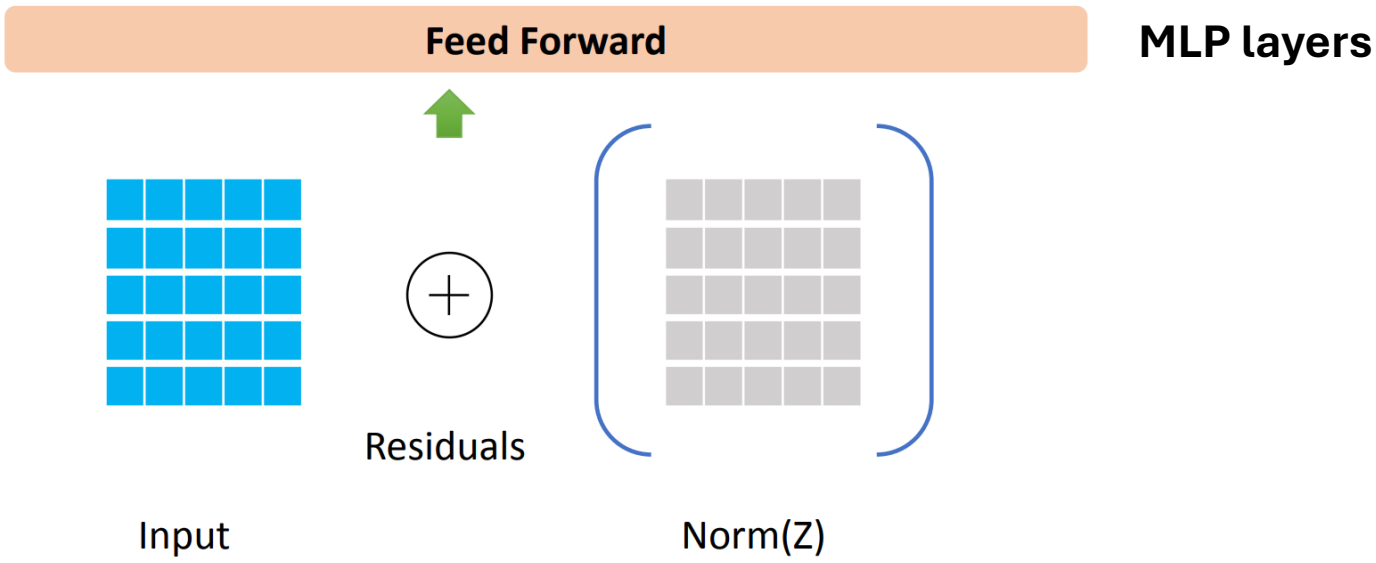


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Feed Forward





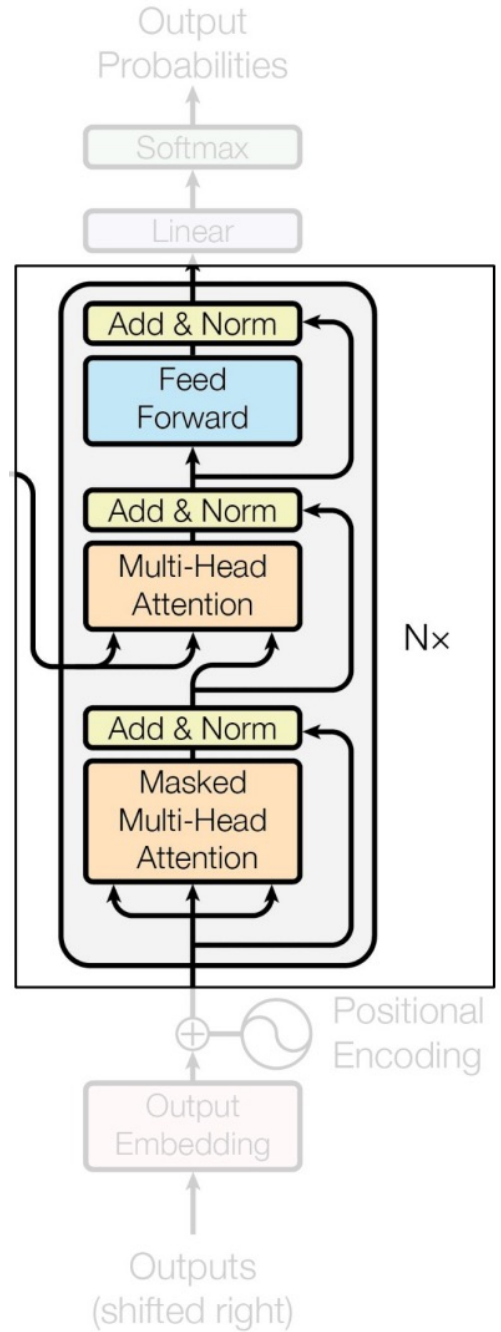
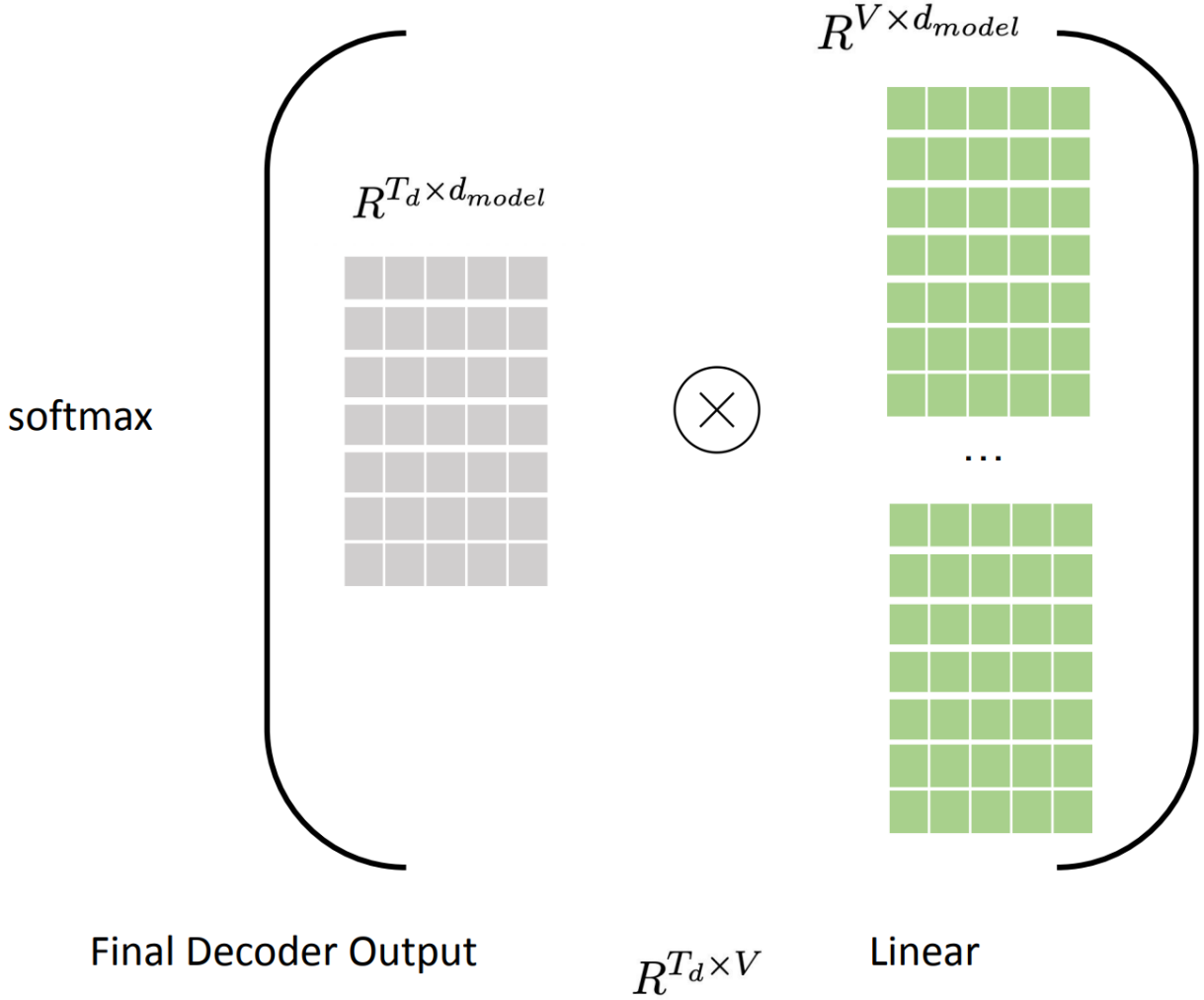


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer: Linear & Softmax



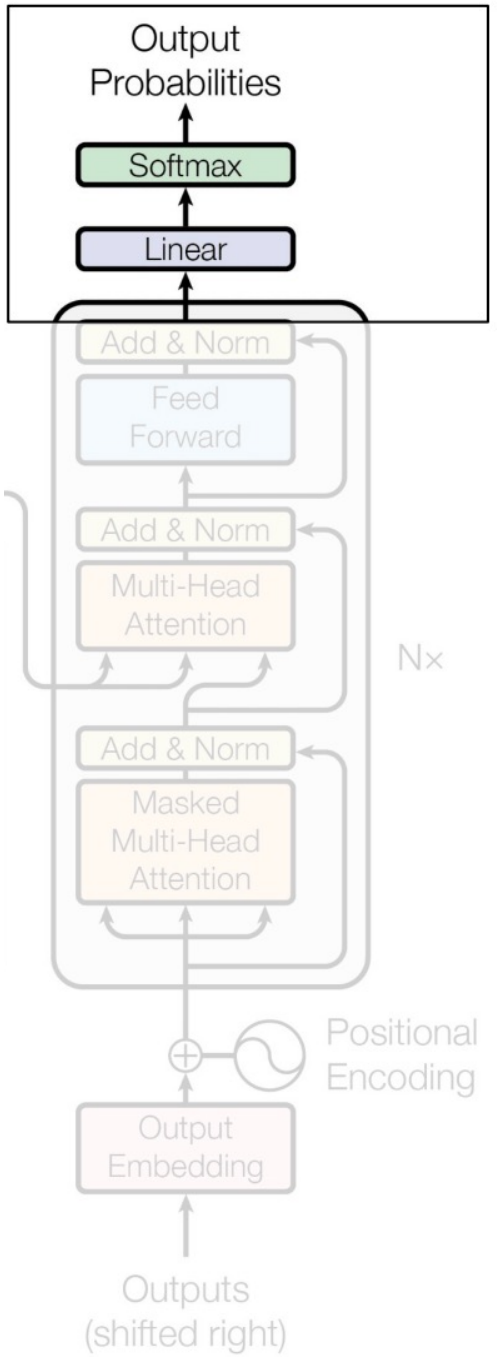
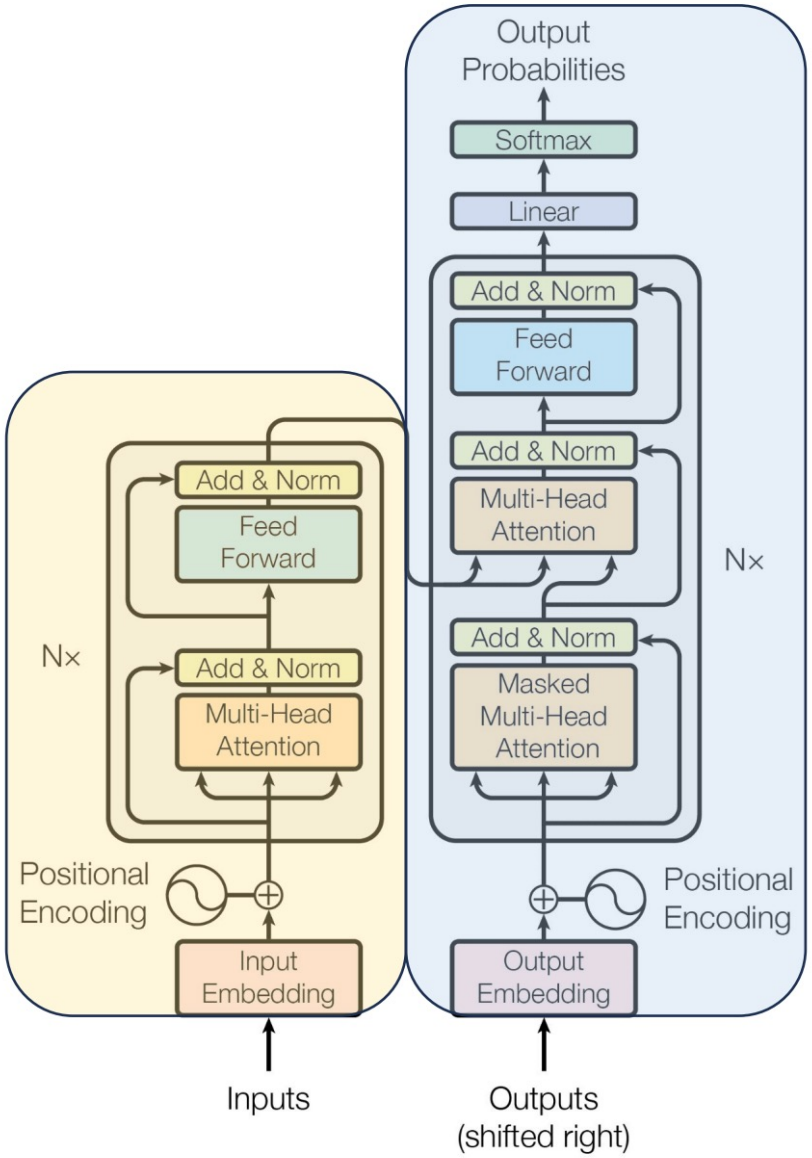


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer



Representation

Learning the representations of the input sequence

Generation

Model is auto-regressive with previous timesteps' outputs

Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Transformer

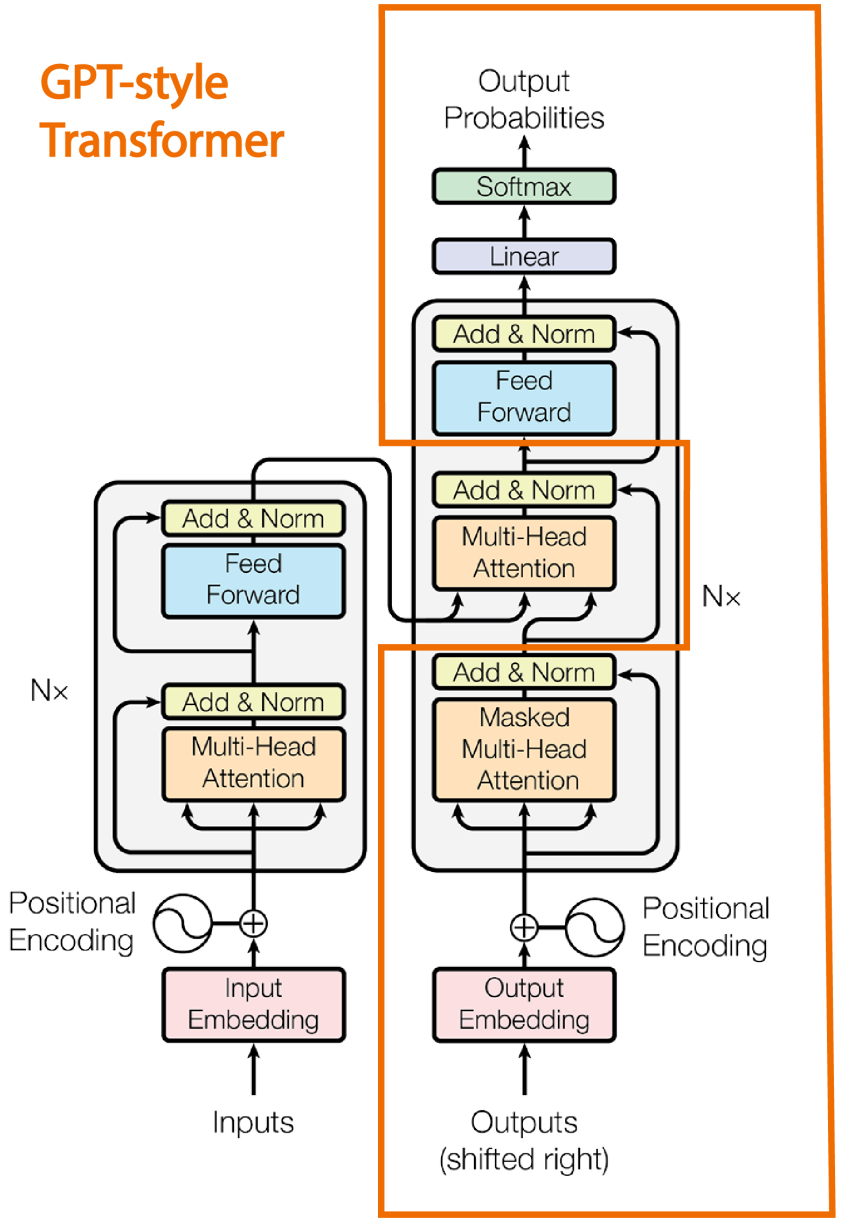
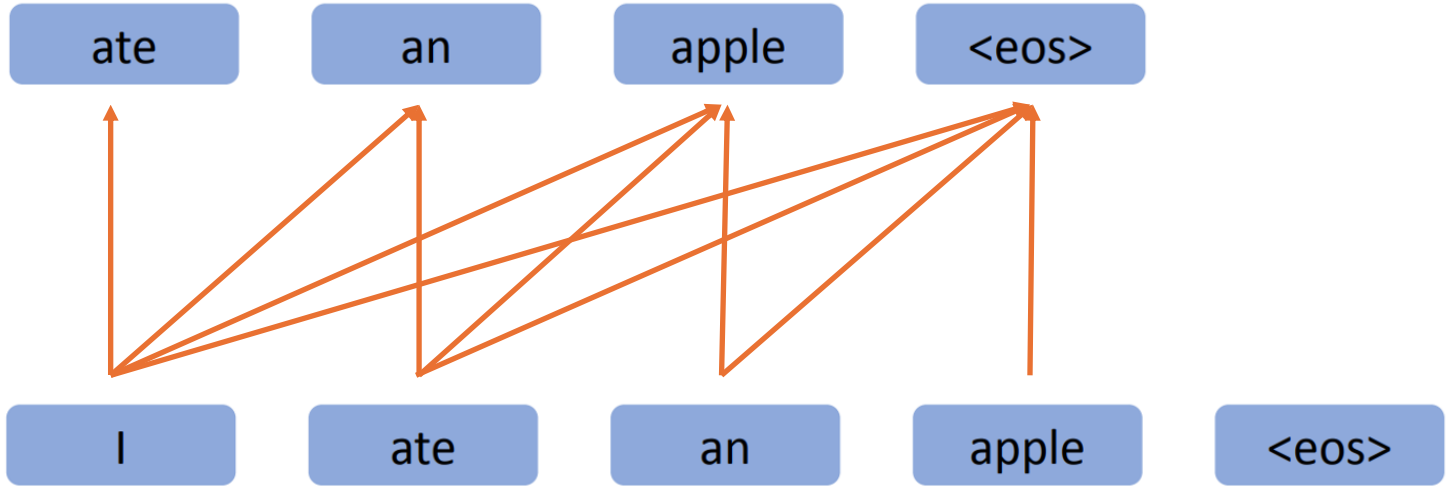


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Training: Teacher forcing



- Feeding the ground truth token from the previous time step as input for the current time step
  - rather than the model's own, potentially incorrect, prediction
- Stabilizes training by preventing early errors from compounding

Training: Teacher forcing

Given a sequence of tokens from dataset, $x_{1:T}$:

$$\max_{\pi} \pi(x_{2:T}|x_1)$$

$$\max_{\pi} \prod_{t=2}^{T} \pi(x_t|x_{1:t-1})$$

$$\max_{\pi} \log \prod_{t=2}^{T} \pi(x_t|x_{1:t-1})$$

$$\max_{\pi} \sum_{t=2}^{T} \log \pi(x_t|x_{1:t-1})$$

Tokenization

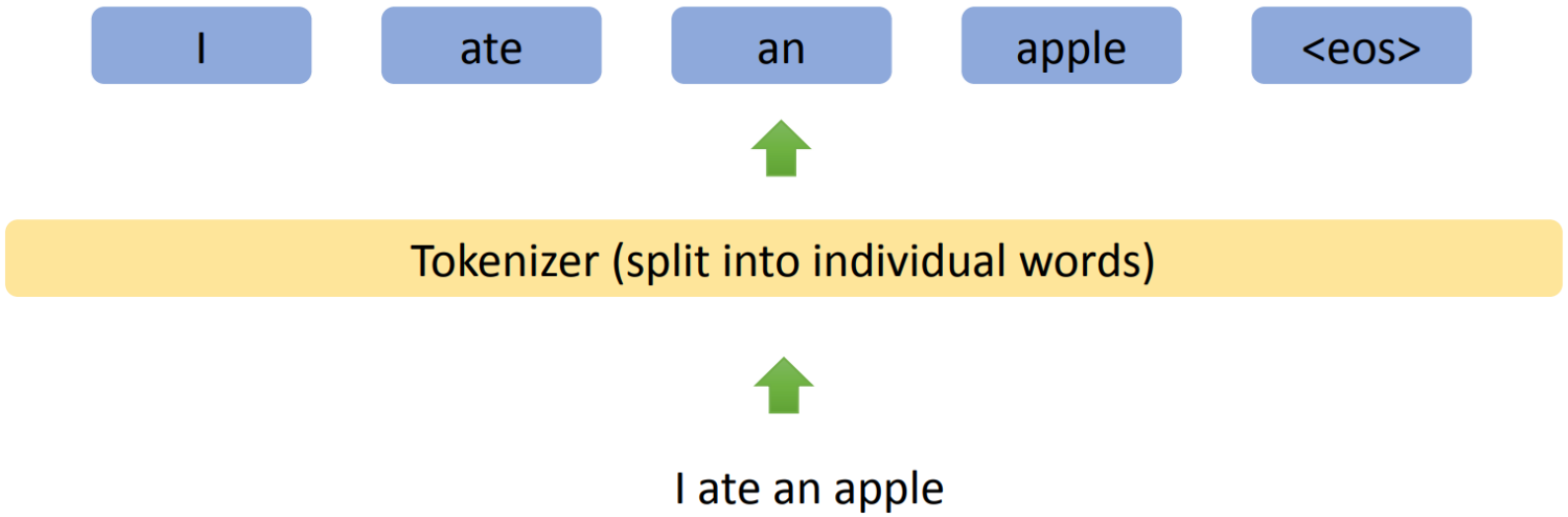
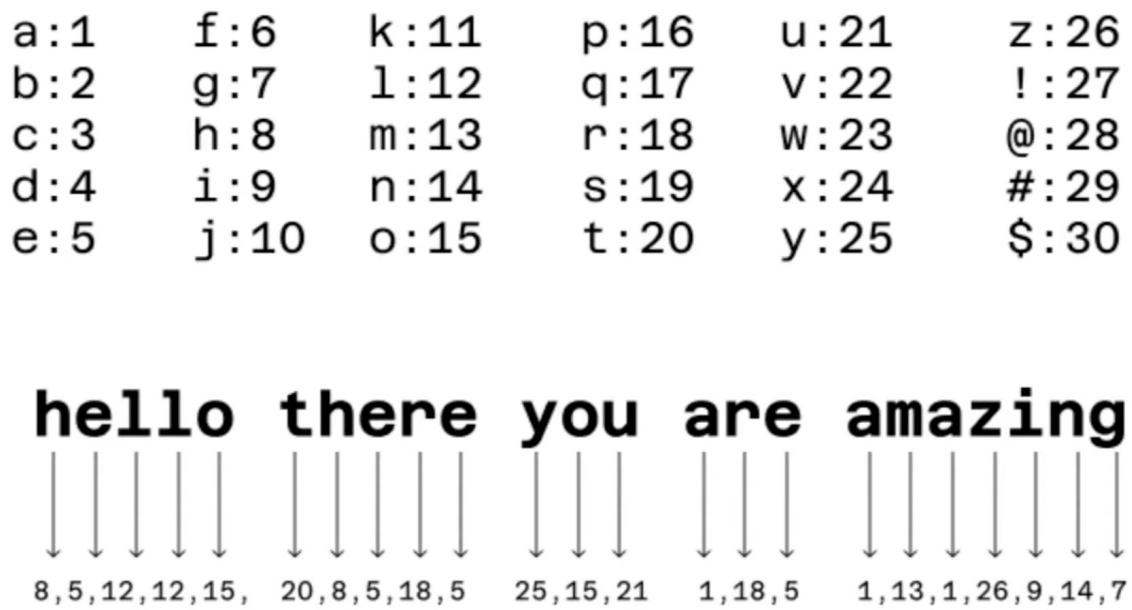


Image: Liu, Y., & Neubig, G. (2024). *Transformers* [Lecture slides]. Carnegie Mellon University.

Tokenization: Character Tokenization

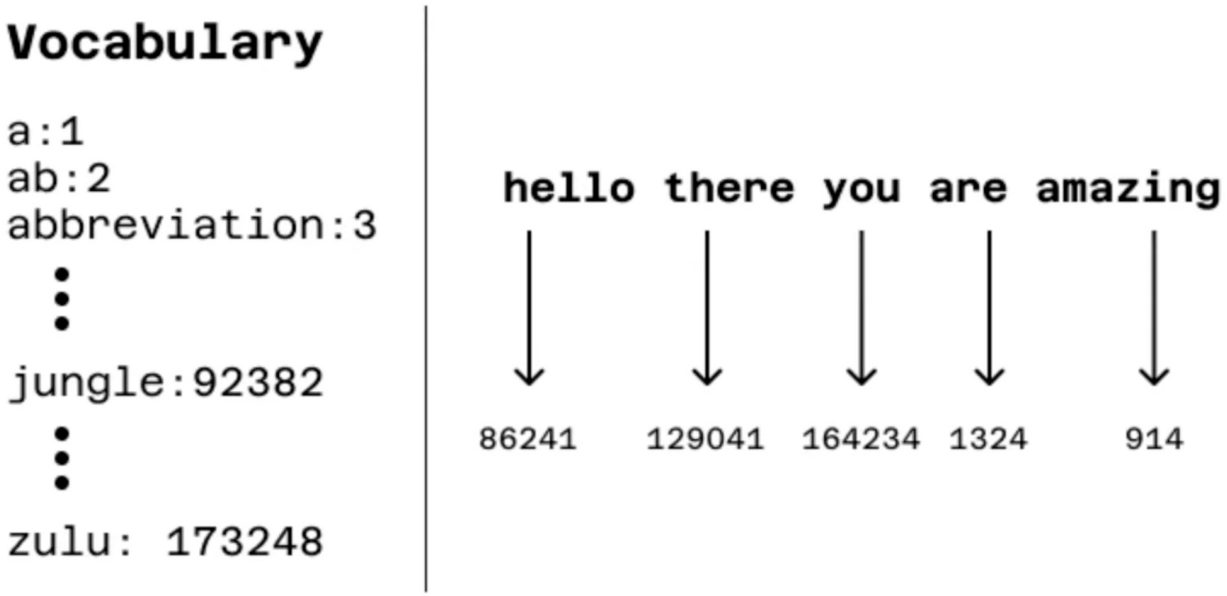


Small vocabulary

Long tokenized sequence

Image: Pulapakura, R. (2023). *Byte pair encoding – How does the BPE algorithm work?*

Tokenization: Word Tokenization



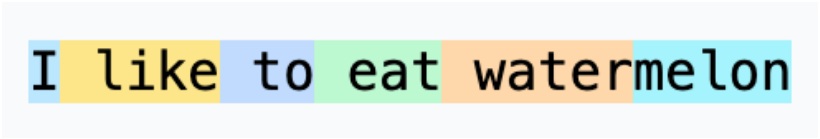
Large vocabulary

Short tokenized sequence

Image: Pulapakura, R. (2023). *Byte pair encoding – How does the BPE algorithm work?*

Tokenization: Sub-word Tokenization

Balance between word and character tokenization



Tokenization: Byte-Pair Encoding (BPE)

Used by GPT 1/2/3

- Start with vocabulary of all individual characters {a, b, c, …, z}
- Repeat:
  - Choose two tokens that are most frequently adjacent in the training corpus
  - Merge those two tokens to a new token and add to the vocabulary
- Until k merges have been done

Tokenization: Byte-Pair Encoding (BPE)

fred fed ted bread, and ted fed fred bread

vocabulary = { 'a', 'b', 'd', 'e', 'f', 'n', 'r', 't', ' ' }

1. Choose the two symbols that are most frequently adjacent in the training corpus

d_ : 7
ed: 6
re: 4
_f: 3

Image: Pulapakura, R. (2023). *Byte pair encoding – How does the BPE algorithm work?*

Tokenization: Byte-Pair Encoding (BPE)

fred fed ted bread, and ted fed fred bread

vocabulary = { 'a', 'b', 'd', 'e', 'f', 'n', 'r', 't', ' ', 'd ' }

1. Choose the two symbols that are most frequently adjacent in the training corpus
2. Add the new merged symbol to the vocabulary
3. Replace every adjacent 'd'+' ' with 'd ' in the corpus

d_ : 7
ed: 6
re: 4
_f: 3

Image: Pulapakura, R. (2023). *Byte pair encoding – How does the BPE algorithm work?*

Tokenization: Byte-Pair Encoding (BPE)

fred fed ted bread, and ted fed fred bread

vocabulary = { 'a', 'b', 'd', 'e', 'f', 'n', 'r', 't', ' ', **'d '** }

1. Choose the two symbols that are most frequently adjacent in the training corpus

ed : 6
re: 4
d f: 4
fr: 3

Image: Pulapakura, R. (2023). *Byte pair encoding – How does the BPE algorithm work?*

Tokenization: Byte-Pair Encoding (BPE)

fred fed ted bread, and ted fed fred bread

vocabulary = { 'a', 'b', 'd', 'e', 'f', 'n', 'r', 't', ' ', **'d '**, 'ed ' }

1. Choose the two symbols that are most frequently adjacent in the training corpus
2. Add the new merged symbol to the vocabulary
3. Replace every adjacent 'e'+'d ' with 'ed ' in the corpus

ed : 6
re: 4
d f: 4
fr: 3

Image: Pulapakura, R. (2023). *Byte pair encoding – How does the BPE algorithm work?*

GPT 1: Generative Pretrained Transformer

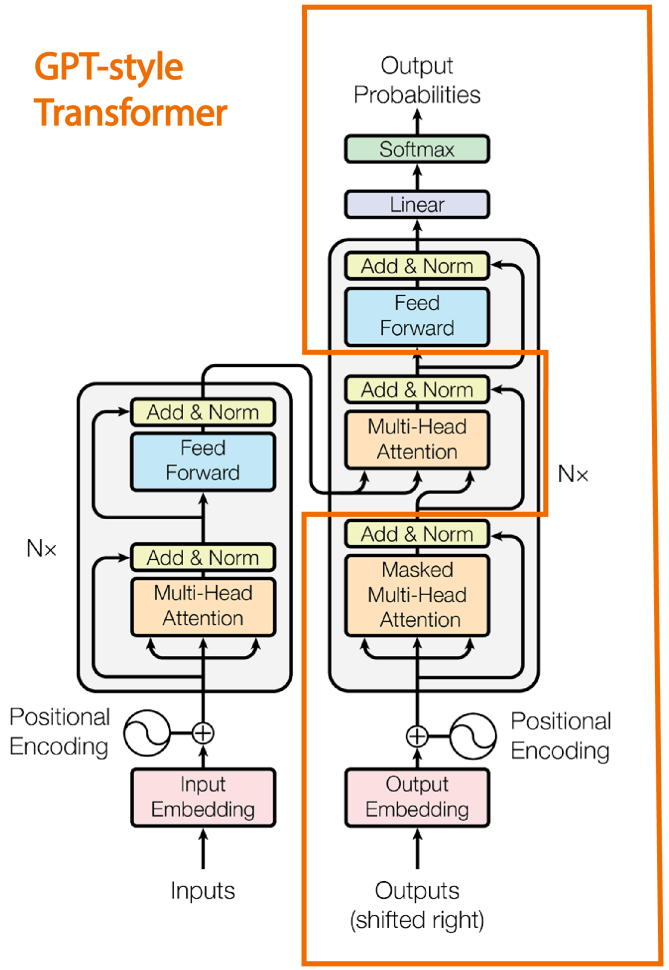
Decoder-only Transformer

GPT Pre-Training Corpus:

- BooksCorpus and English Wikipedia

GPT Pre-Training Tasks:

- Predict the next token, given the previous tokens
- 117M Params



GPT 1: Generative Pretrained Transformer

The man worked as a teacher for the college

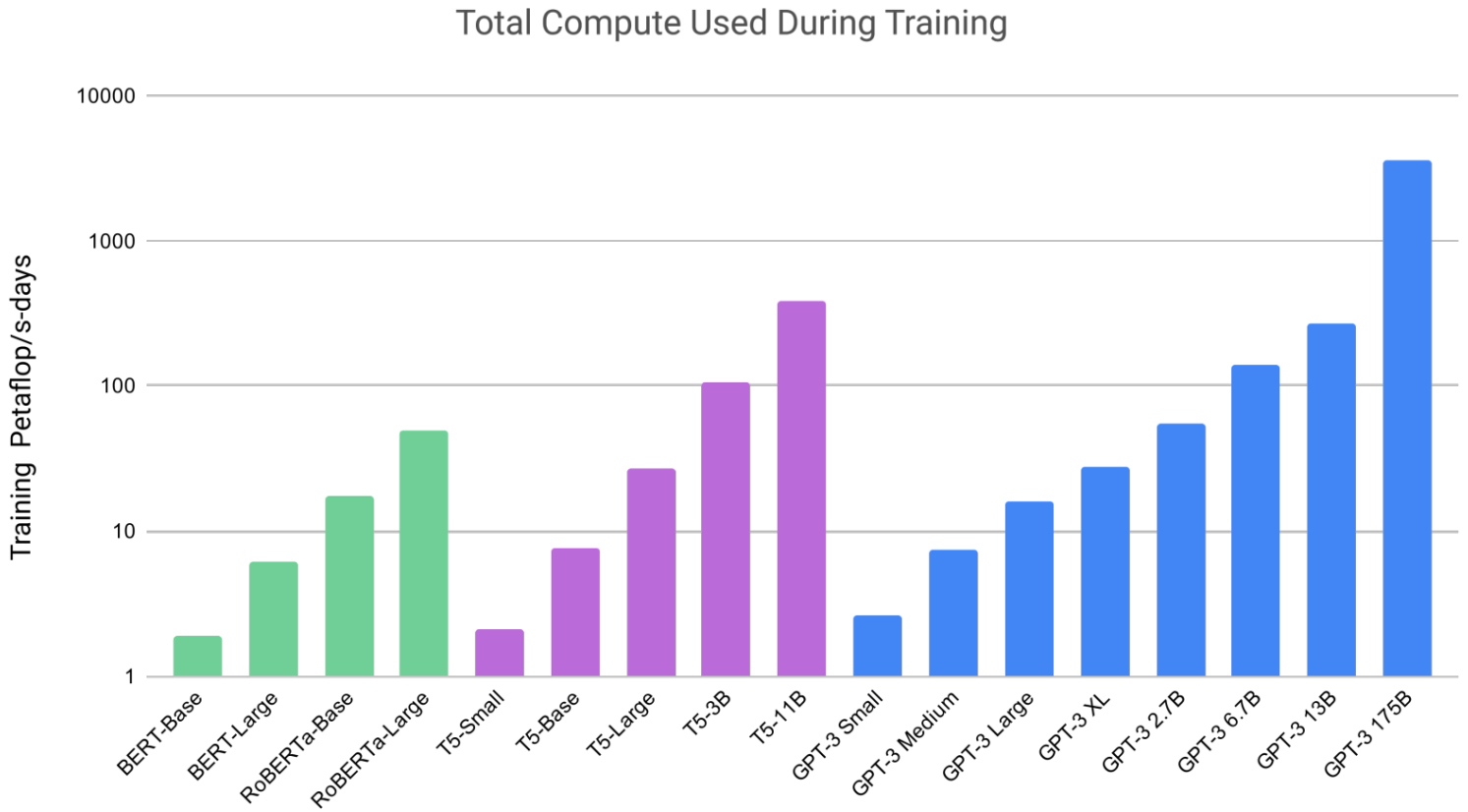
The city of Paris is known for its historic architecture and cultural heritage

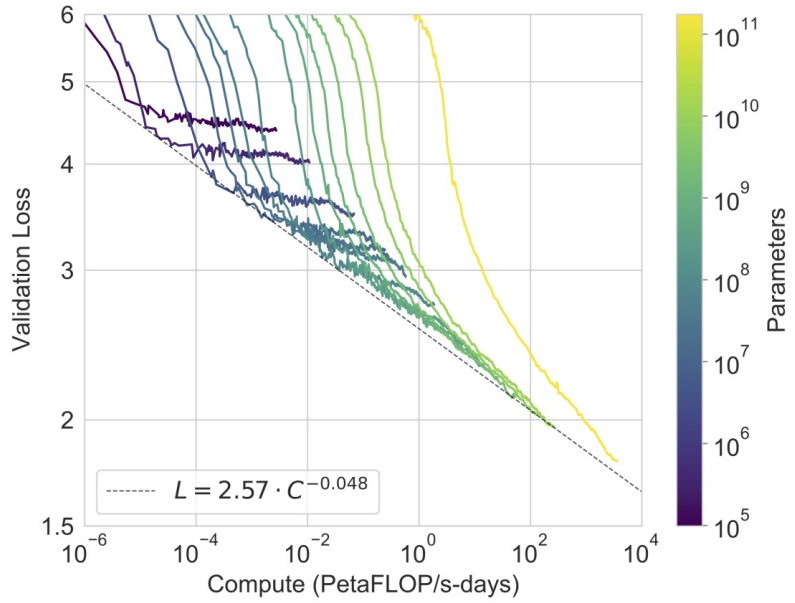
- Short sentences (due to small size of 117M)
- Only text continuations

GPT-3: Scaling Up

GPT Pre-Training Corpus:

- Common Crawl and many more
  - Trillions of tokens





GPT-3: Few-shot Prompting

Arithmetic reasoning

Q: What is 27 + 35?
A: 62

Q: What is 123 + 456?
A: 579

Few-shot translation

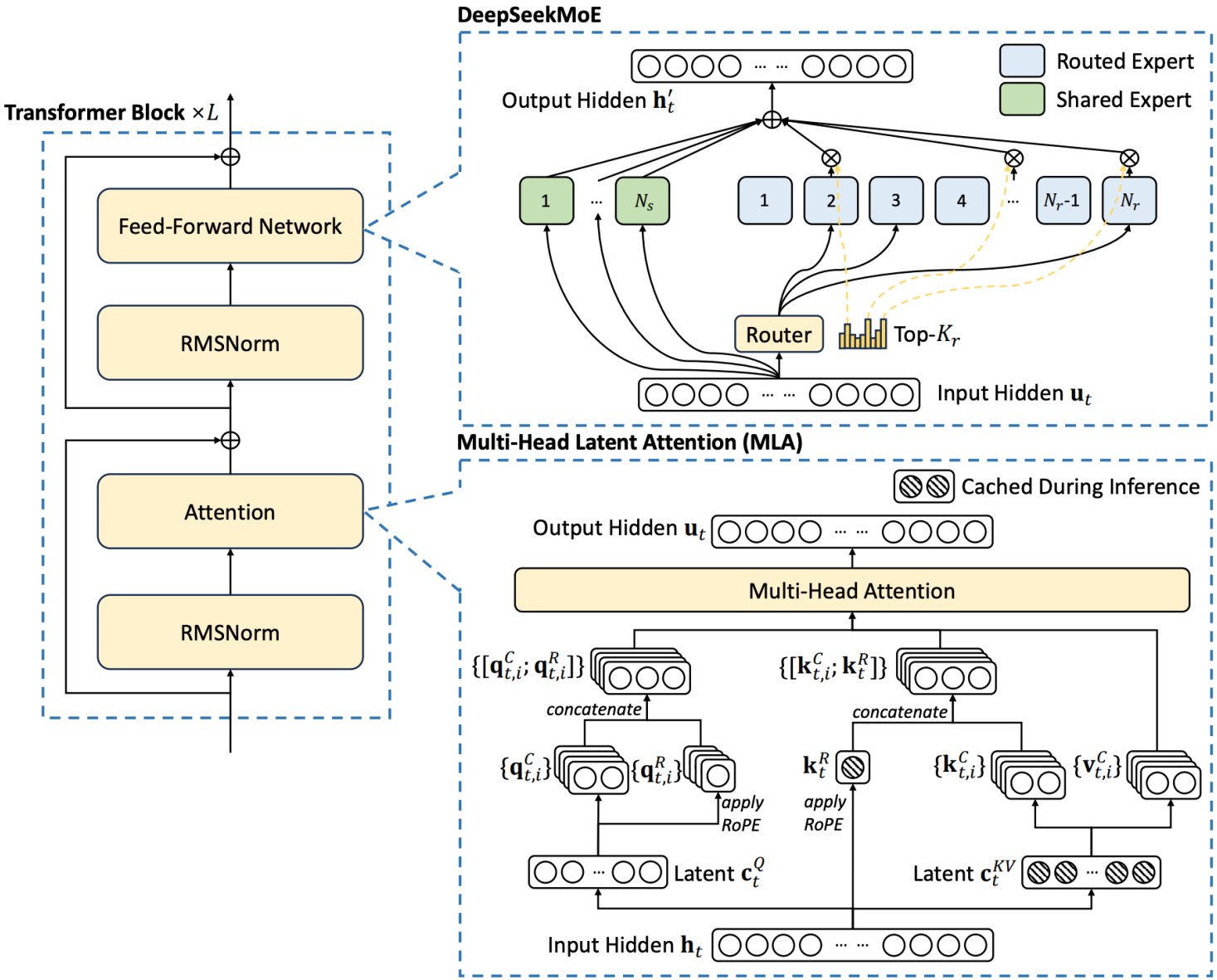
English: sea otter
French: loutre de mer

English: peppermint
French: menthe poivrée

English: cheese
French: fromage

- Multi-paragraph
- Few-shot learning

Case study: Deepseek-V3



Mixture-of-experts

Covered later in class

Multi-token prediction

Given a sequence of tokens from dataset, $x_{1:T}$:

Next-token Prediction

$$\max_{\pi} \sum_{t=2}^{T} \log \pi(x_t | x_{1:t-1})$$

Multi-token Prediction

$$\max_{\pi} \sum_{t=2}^{T} \sum_{i=0}^{k-1} \log \pi(x_{t+i} | x_{1:t-1})$$

Directly predicting the next k tokens

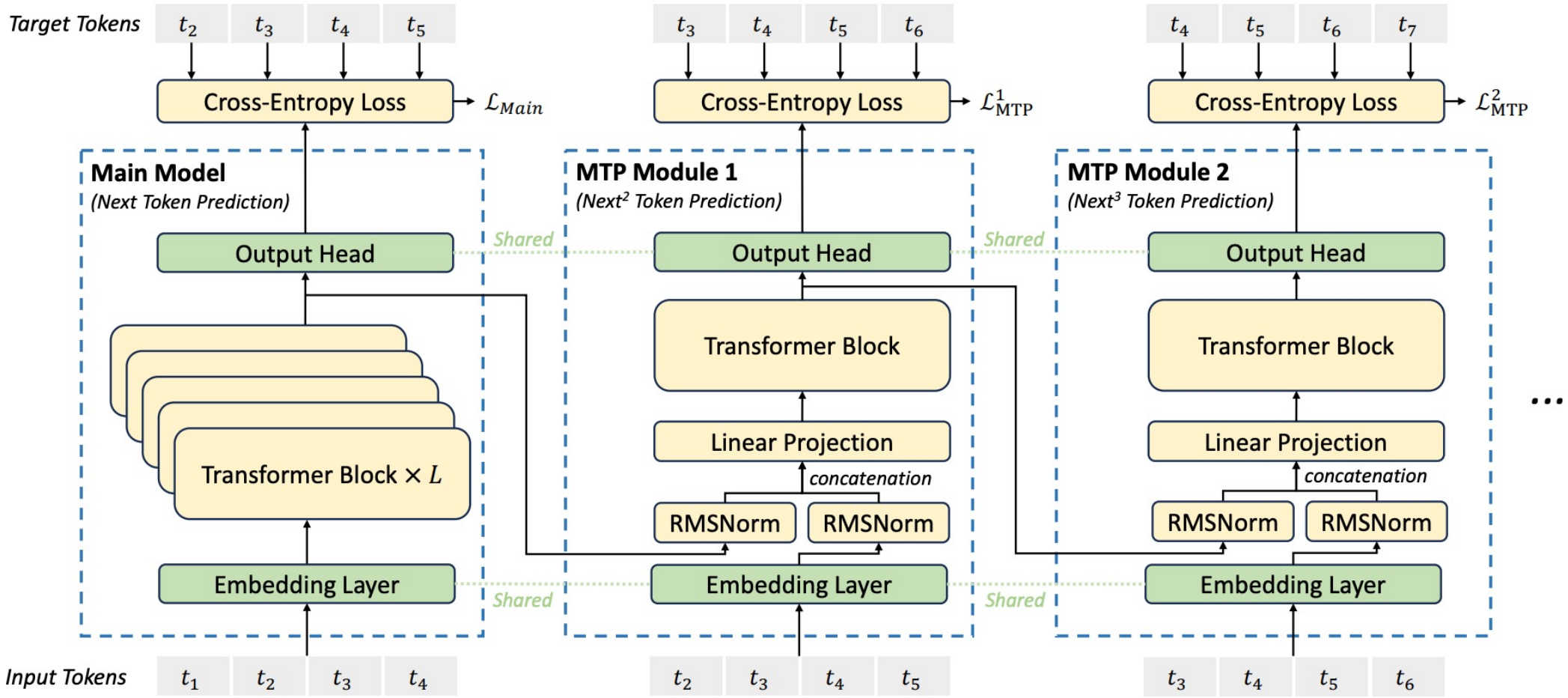
Multi-token prediction

$$the\ capital \sim \pi(\cdot\,|what\ is\ the\ capital\ of\ France?)$$

$$of France \sim \pi(\cdot\,|what\ is\ the\ capital\ of\ France?\ the\ capital)$$

$$is\ Paris \sim \pi(\cdot\,|what\ is\ the\ capital\ of\ France?\ the\ capital\ of\ France)$$

Case study: Deepseek-V3



Multi-token Prediction

Case study: Deepseek-V3

671B Parameters

Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	$5.328M	$0.238M	$0.01M	$5.576M

Project Group

Image generation

(e.g. diffusion)

Language generation

(e.g. LLM)

Other modality and multimodal