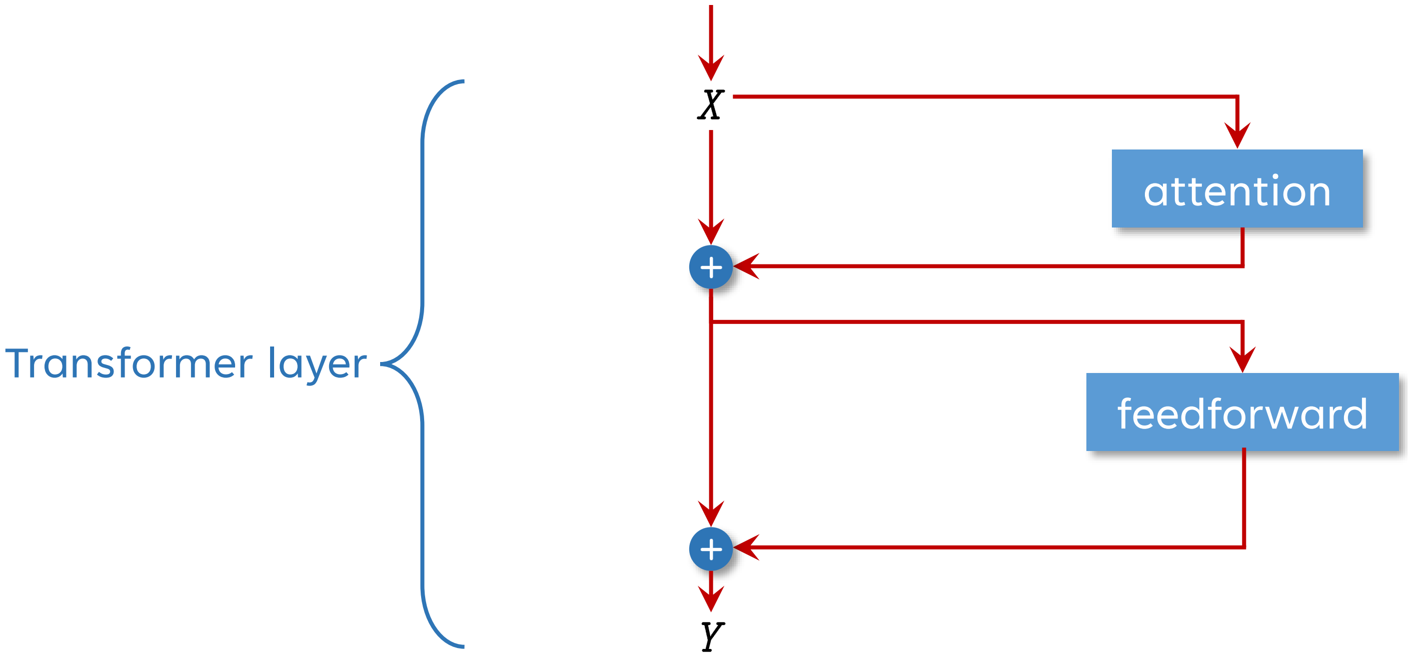
CS 577: NATURAL LANGUAGE PROCESSING

Abulhair Saparov

Lecture 7: Transformers II

PREVIOUS LECTURE: TRANSFORMERS



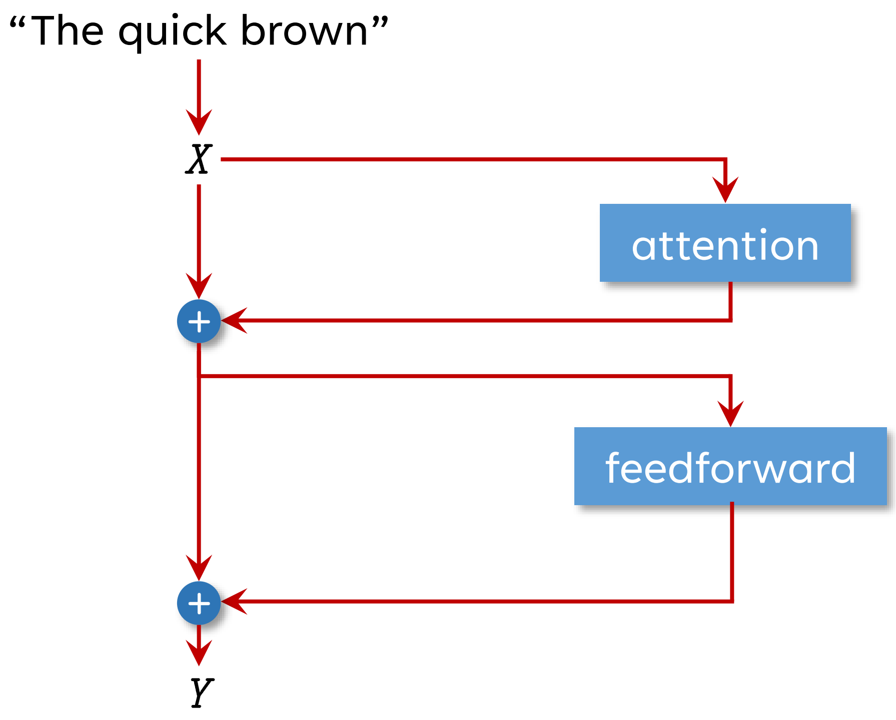
LAYER NORMALIZATION

- Suppose we are training a transformer on a classification task, so we have a softmax operation at the end of the network.
- Also suppose the input embeddings have large magnitude,
- It's very likely that the magnitude of the embeddings stays large throughout the transformer layers, up to the last softmax operation.
- Recall that the derivative of the softmax is close to zero if the input is a large positive or negative value.
  - Hint: The logistic function (i.e., sigmoid) is equivalent to softmax in two dimensions.
- Thus, in this example, the gradient would be very close to zero, and training would be extremely slow.

LAYER NORMALIZATION

- Thus, transformers with many layers can also sometimes suffer from vanishing gradients.
- To avoid this, transformers use layer normalization.

LAYER NORMALIZATION

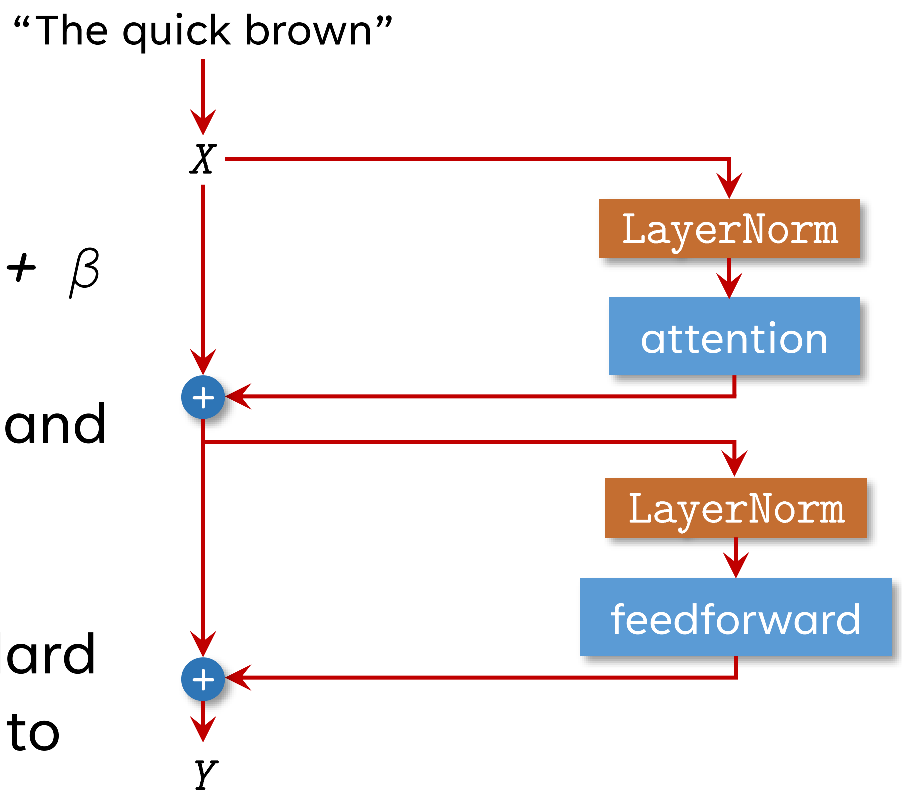


LAYER NORMALIZATION

$$\text{LayerNorm}(x_i) = \frac{x_i - avg(x_i)}{\sqrt{var(x_i) + \varepsilon}} \circ \gamma + \beta$$

Where ε is a small fixed constant, γ and β are vectors of learnable weights.

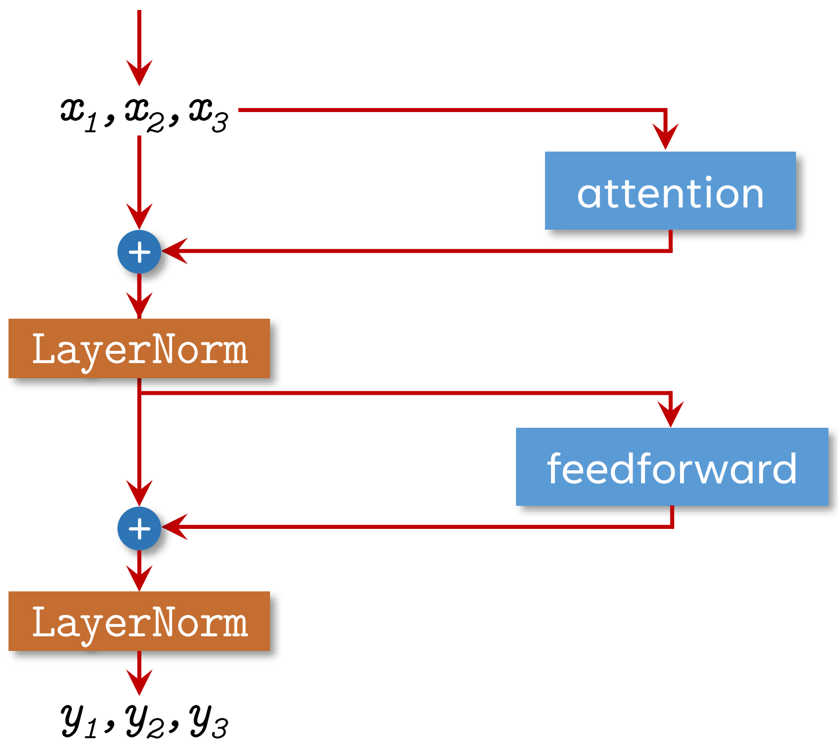
Since we scale the input by its standard deviation, layer normalization helps to prevent the activations from attaining very large positive or negative values.



POST-LAYER NORMALIZATION

The original transformer paper used post-layer normalization.

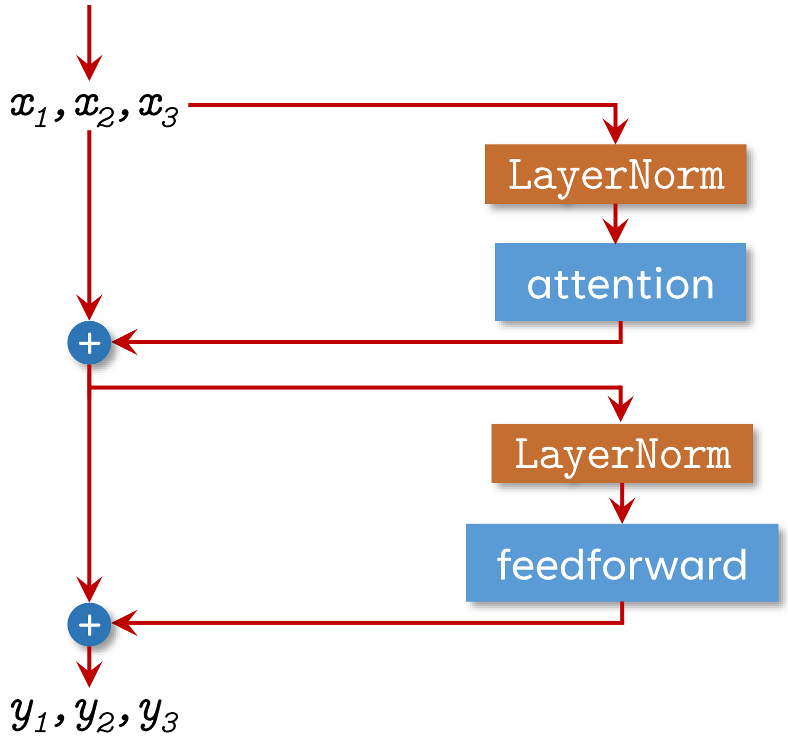
Layer normalization was applied on the residual stream (i.e., *after* residual connection).



[Vaswani et al., 2017]

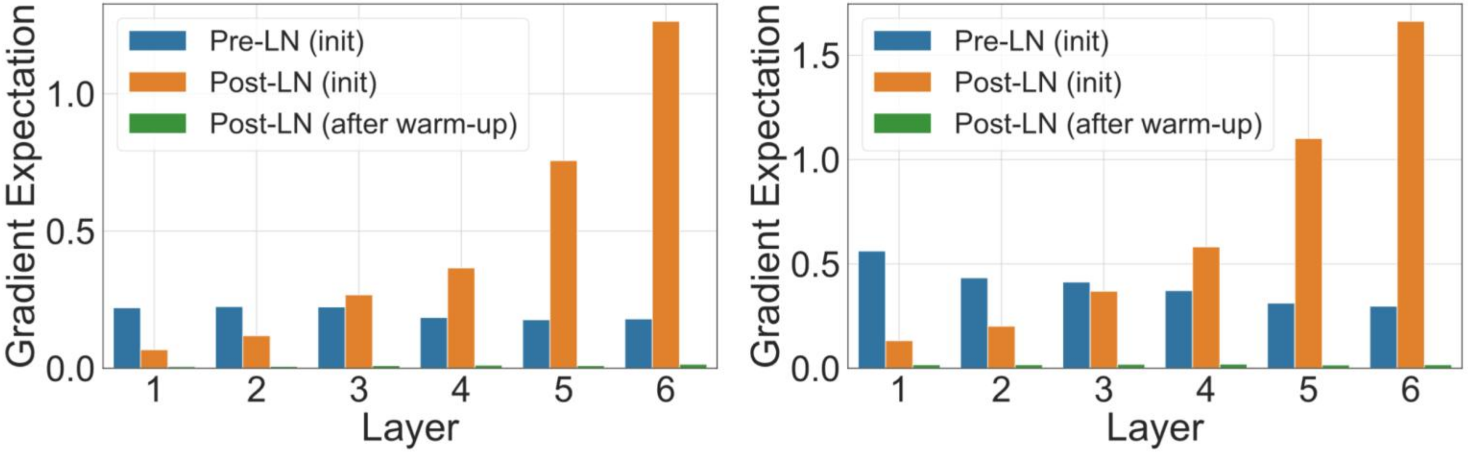
PRE-LAYER NORMALIZATION

Xiong et al., 2020, proposed moving the layer normalization *before* the attention and FF blocks.



PRE-LAYER NORMALIZATION

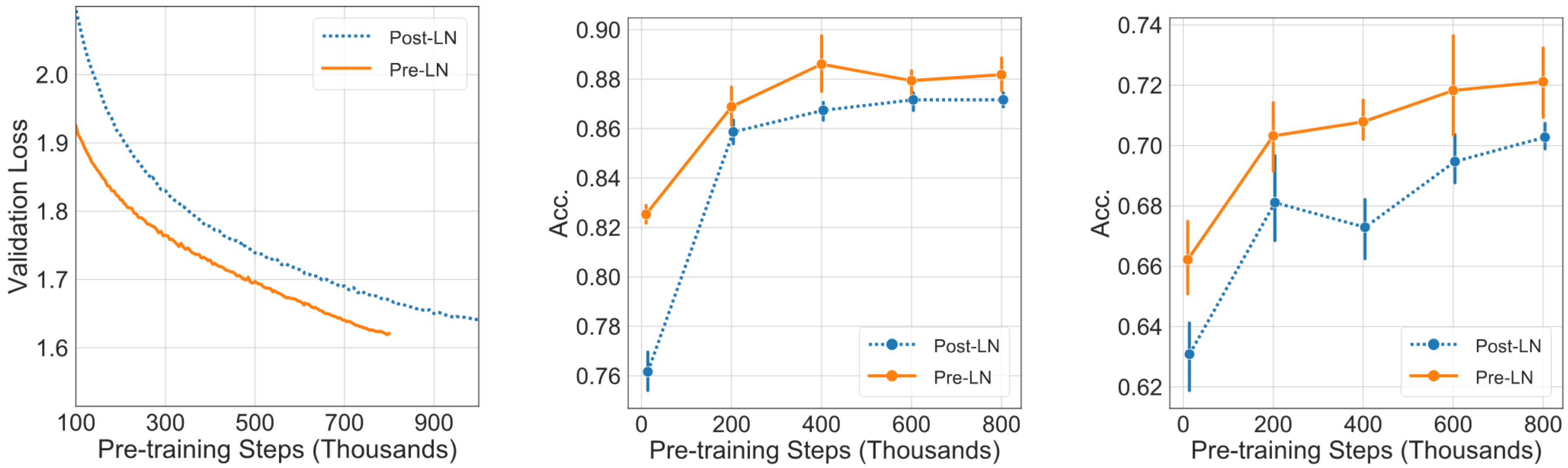
- Xiong et al., 2020, showed that the magnitudes of the gradients are more uniform across layers when using pre-layer normalization.
- Hypothesis: Learning occurs at a more uniform rate across layers when using pre-layer norm.



(a) W^1 in the FFN sub-layers (b) W^2 in the FFN sub-layers

PRE-LAYER NORMALIZATION

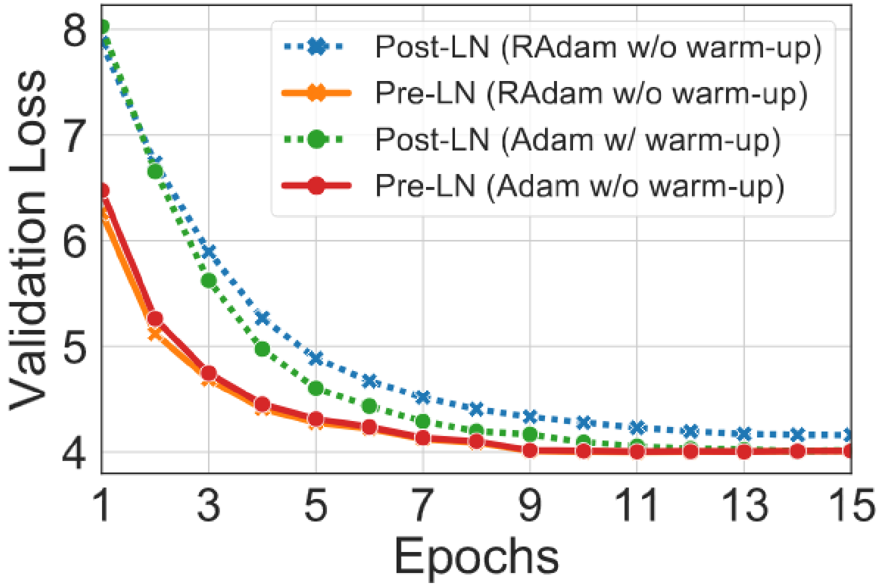
- Measure empirical performance on masked language modeling, semantic similarity (Microsoft Research Paragraph Corpus), and textual entailment (Recognizing Textual Entailment dataset).



(a) Validation Loss on BERT (b) Accuracy on MRPC (c) Accuracy on RTE

PRE-LAYER NORMALIZATION

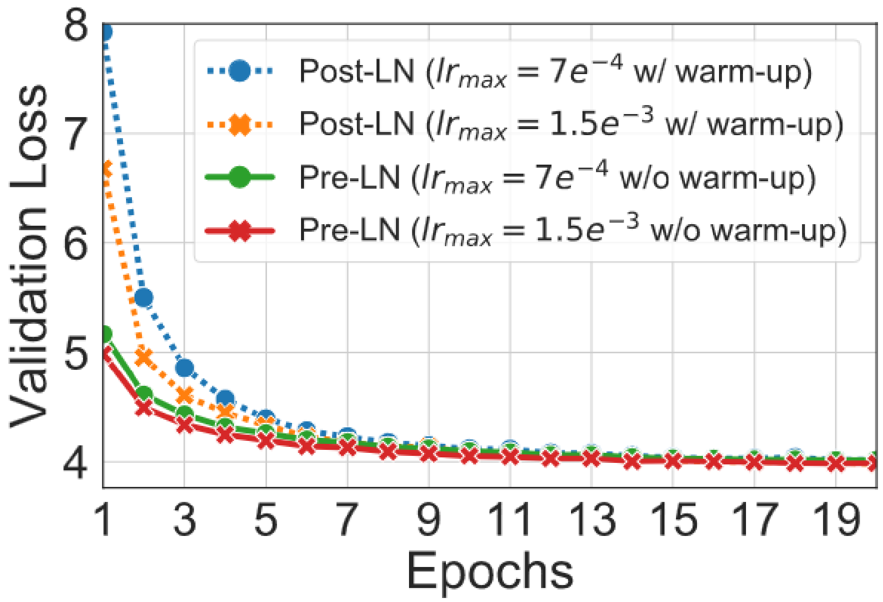
- Measure empirical performance on machine translation.



(a) Validation Loss (IWSLT)

PRE-LAYER NORMALIZATION

- Measure empirical performance on machine translation.



(c) Validation Loss (WMT)

RMS NORMALIZATION

- Zhang and Sennrich, 2019, proposed a simpler alternative to layer normalization, called root mean square layer normalization, or RMSNorm.

$$\text{LayerNorm}(x_i) = \frac{x_i - avg(x_i)}{\sqrt{var(x_i) + \varepsilon}} \circ \gamma + \beta$$

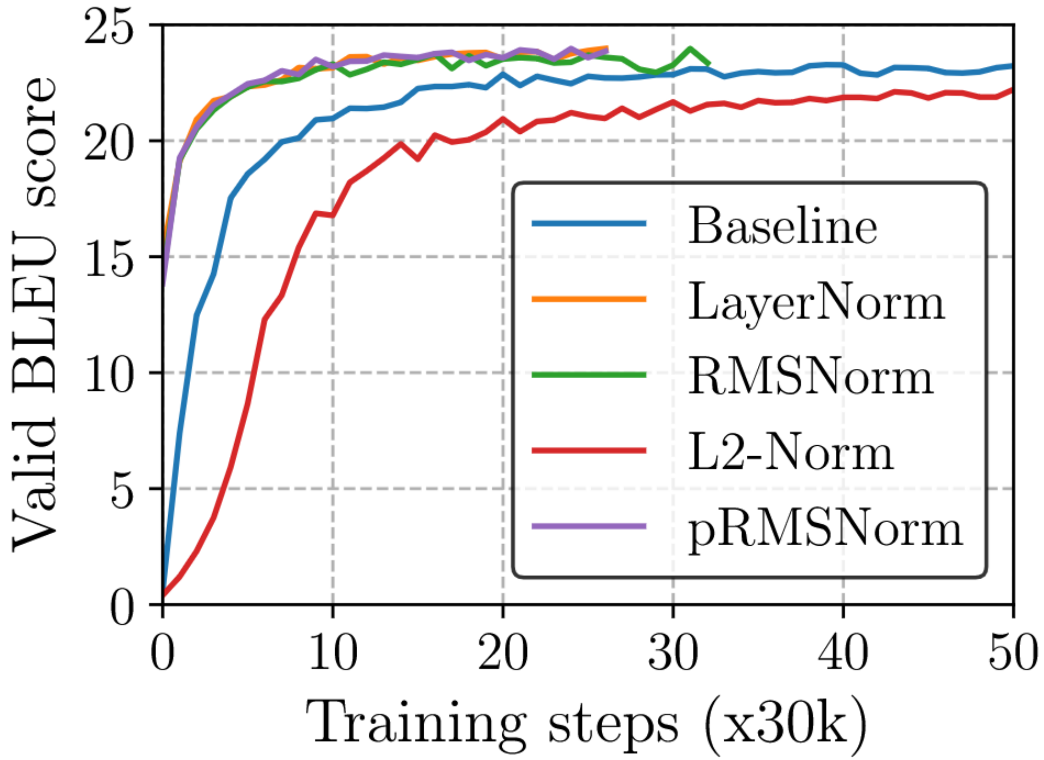
where ε is a small fixed constant, γ and β are vectors of learnable weights.

$$\text{RMSNorm}(x_i) = \frac{x_i}{RMS(x_i)} \circ \gamma \text{ where } RMS(x_i) = \sqrt{\frac{1}{n}\sum_{j=1}^{n} x_{i,j}^2}$$

- Note that RMSNorm is faster to compute.

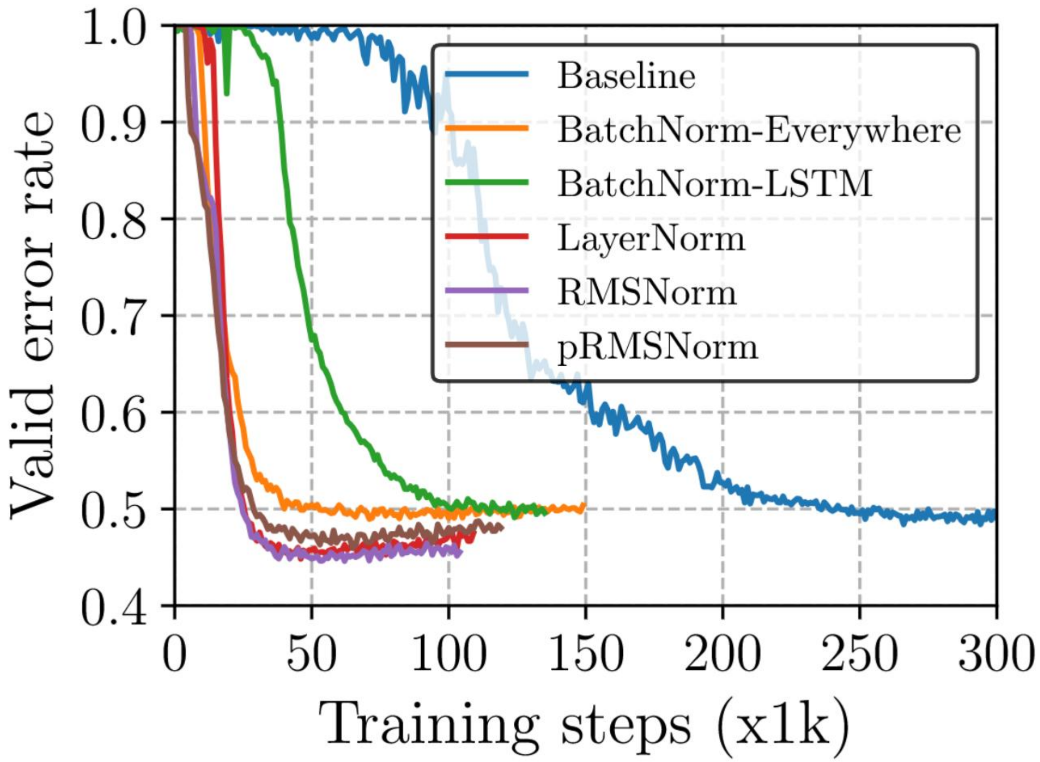
RMS NORMALIZATION

- Measure empirical performance on machine translation.



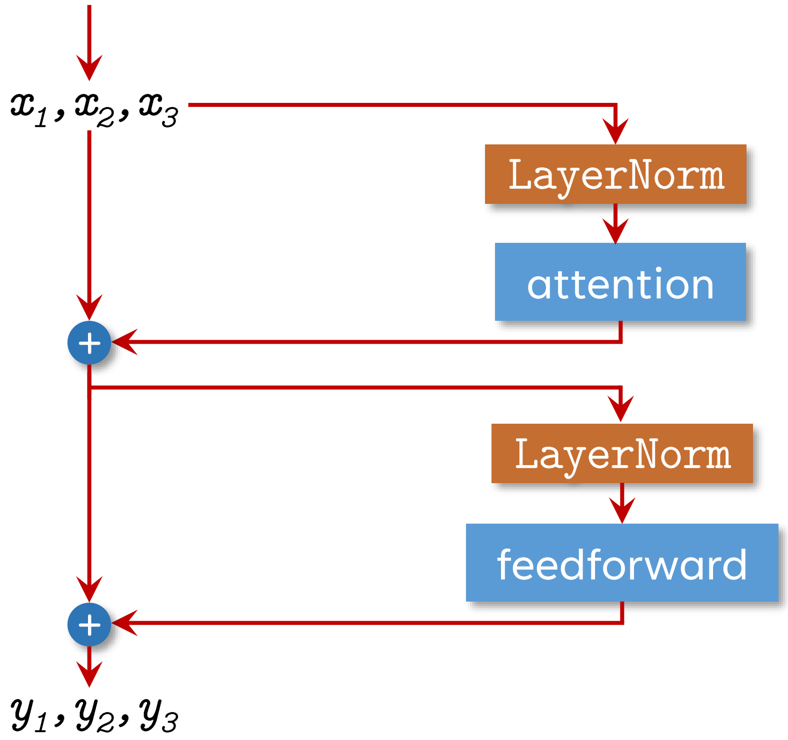
RMS NORMALIZATION

- Measure empirical performance on question answering.



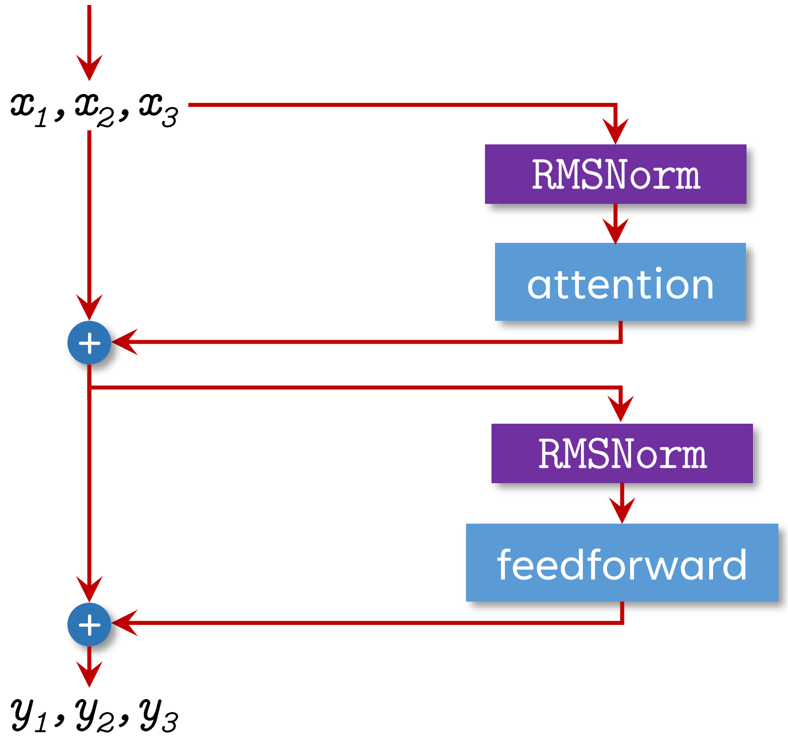
PRE-LAYER NORMALIZATION AND RMSNORM

- GPT-2 used pre-layer normalization.



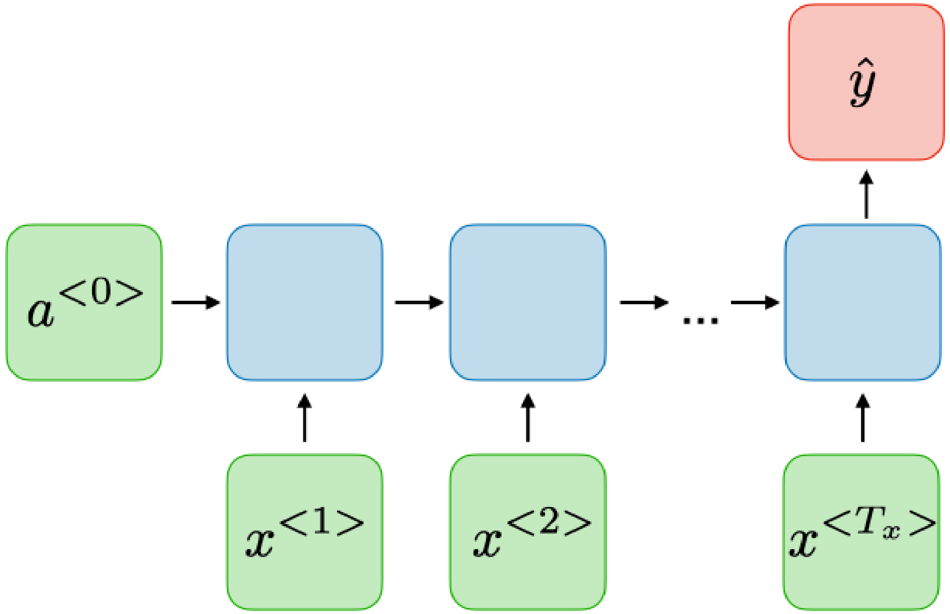
PRE-LAYER NORMALIZATION AND RMSNORM

- GPT-2 used pre-layer normalization.
- Recent models (e.g., LLaMA and DeepSeek) typically use pre-layer normalization with RMSNorm.



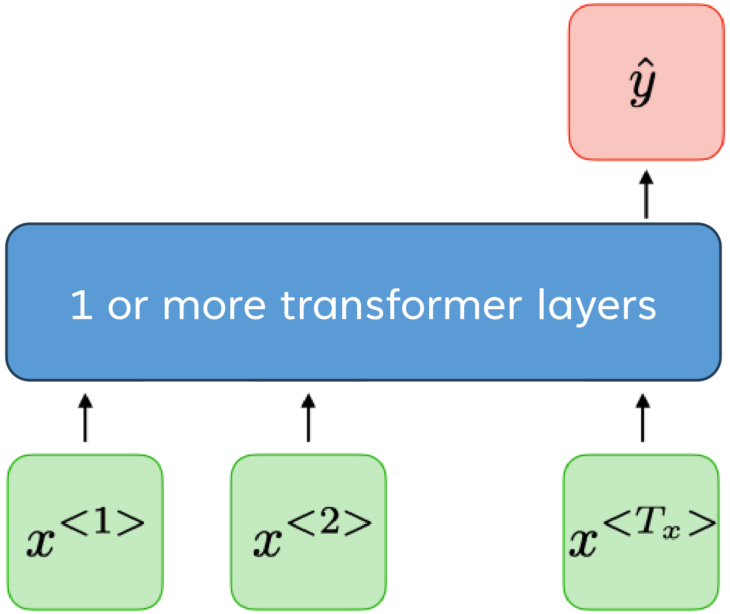
TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.



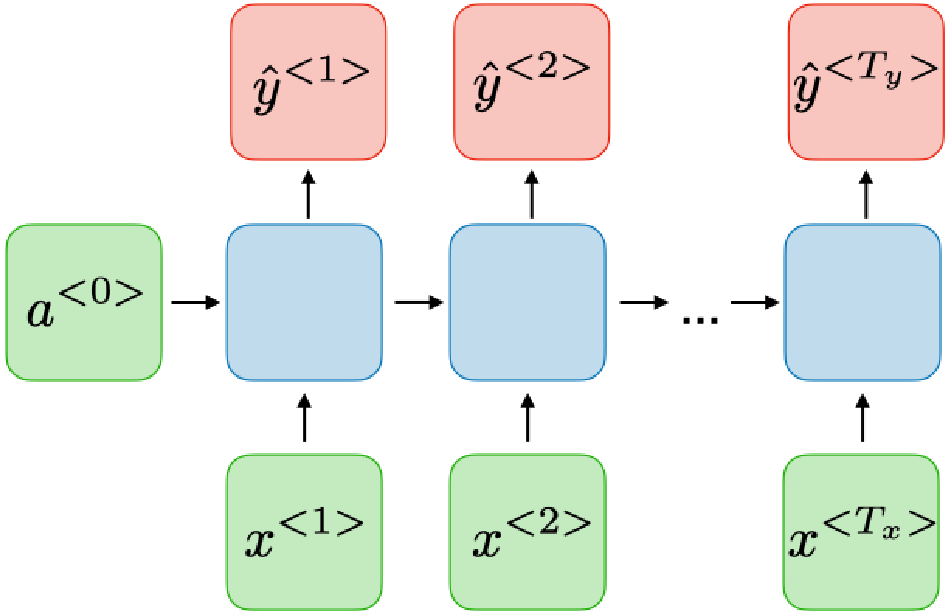
TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



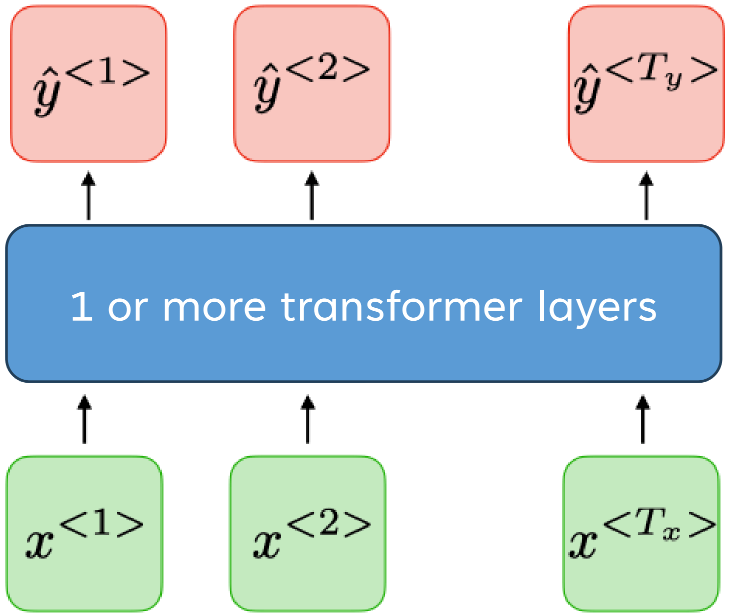
TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



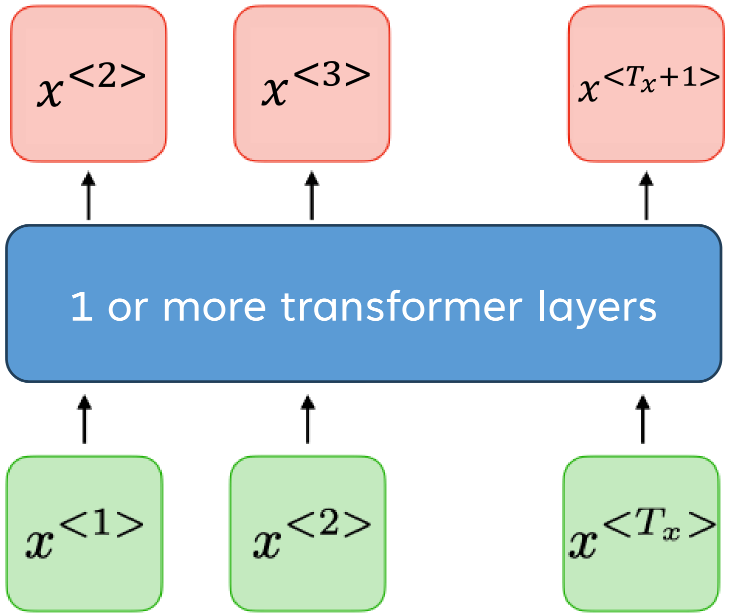
TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



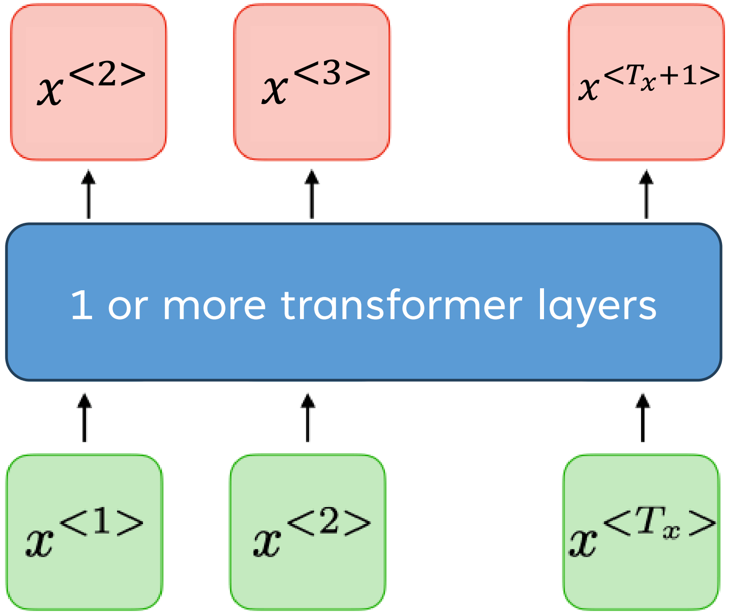
In the language modeling task, we often predict n tokens, where the i^{th} output token corresponds to the $i+1^{th}$ input token.

The loss is computed as the sum of the losses of all output tokens.

Often called autoregressive or causal language modeling.

WHY THE CAUSAL MASK?

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



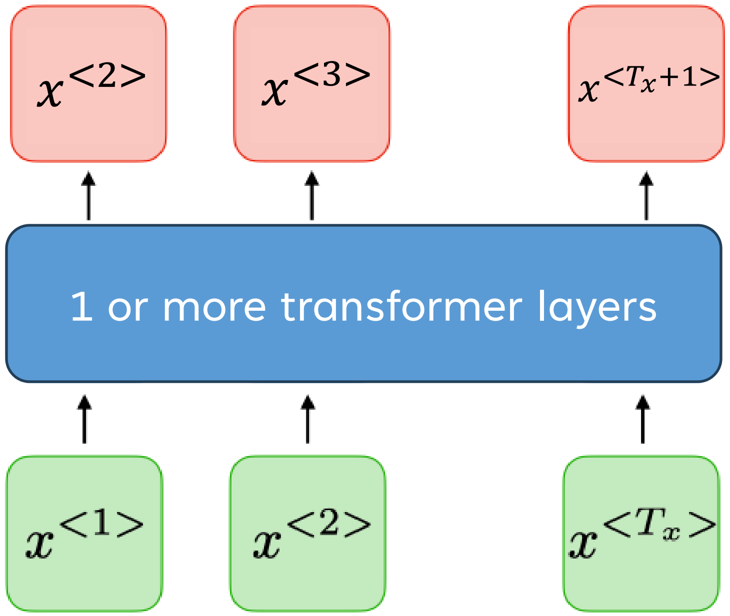
This motivates the causal mask.

We want $x^{<i>}$ to only attend to tokens that come before it.

Otherwise, it can simply cheat by copying $x^{<i+1>}$.

WHY THE CAUSAL MASK?

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.

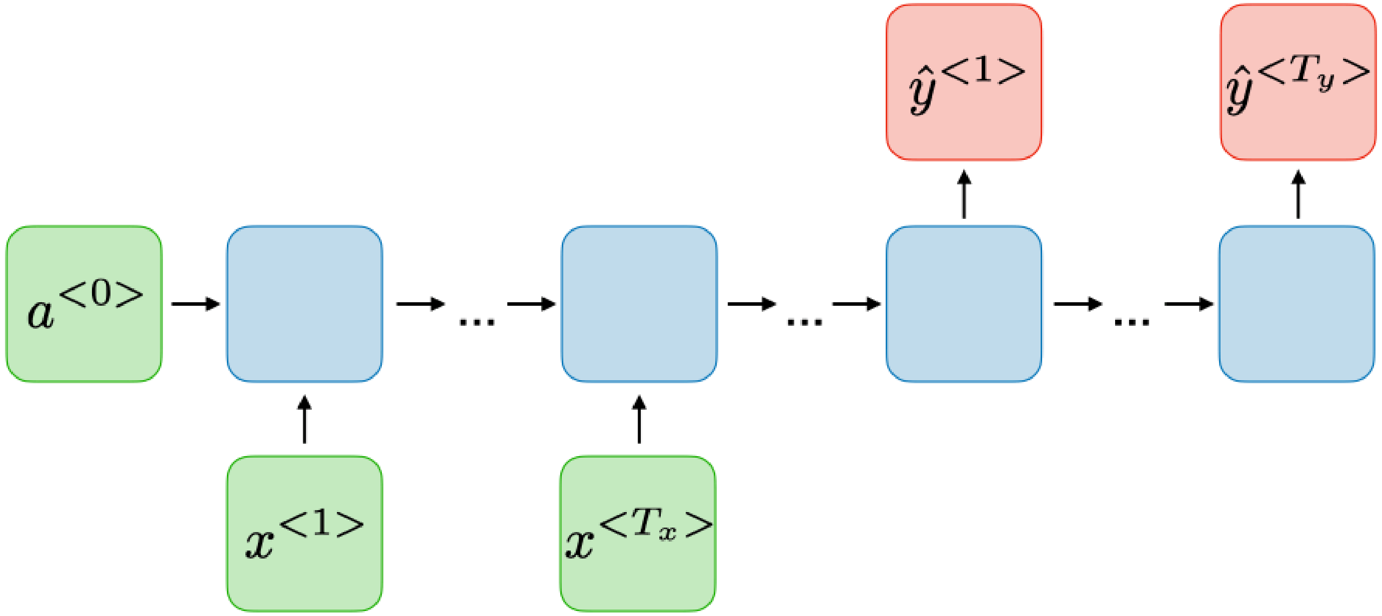


With or without the causal mask, the model still must predict the token after the last token.

The model can’t “cheat” here by attending to future tokens.

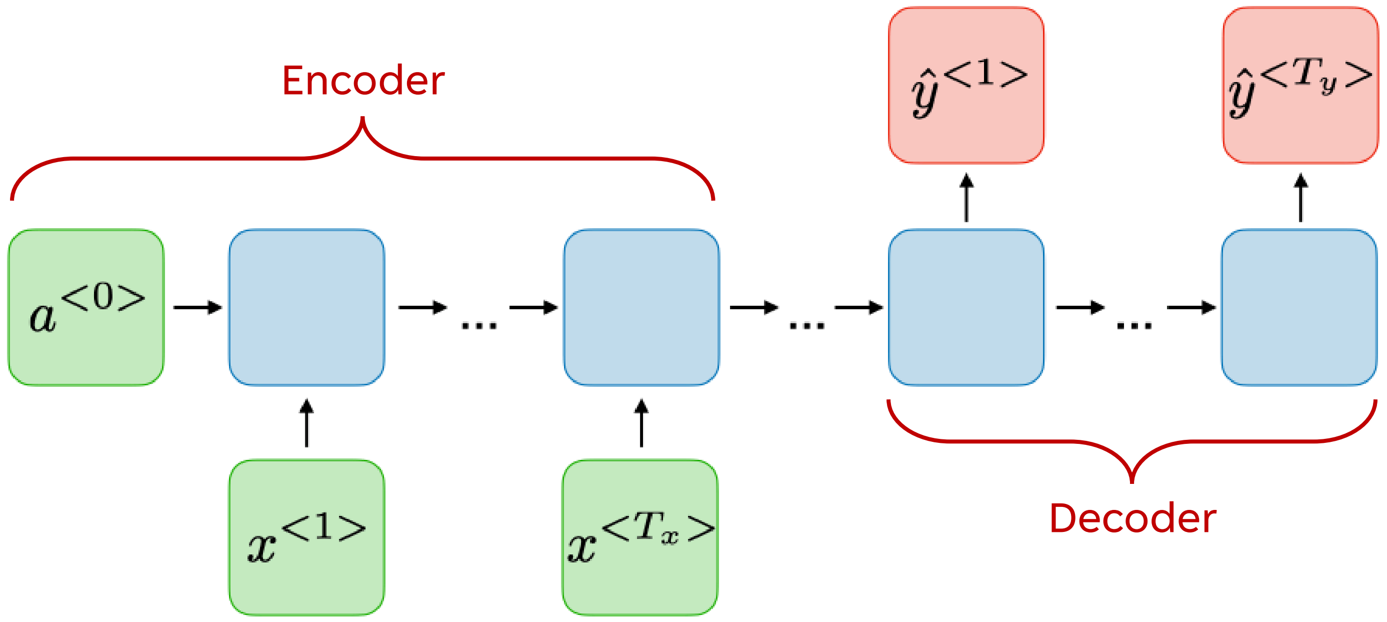
TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.

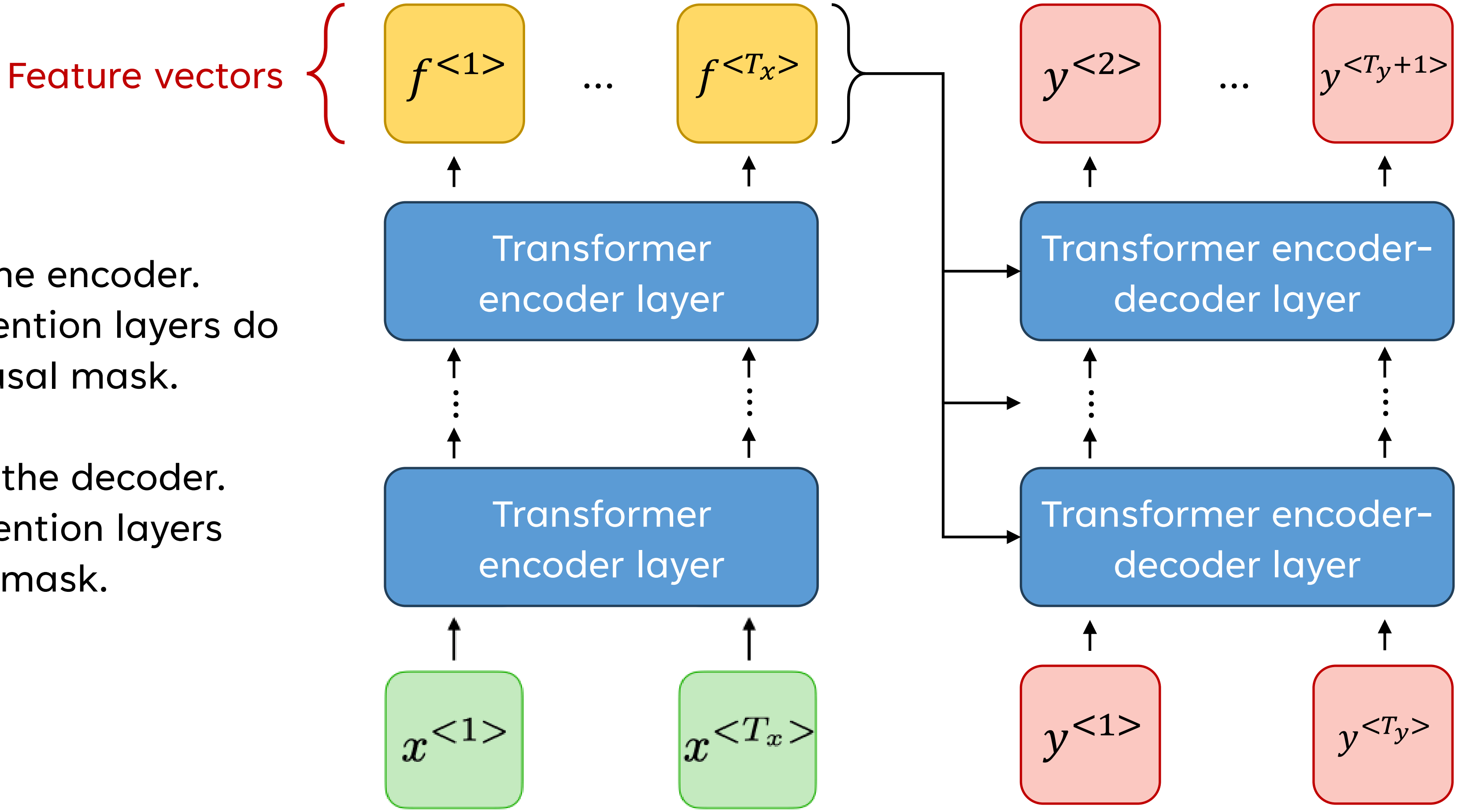


TRANSFORMER APPLICATIONS

- Recall that RNNs can be used in a wide variety of architectures.
- Transformers can similarly be a component in lots of different models.



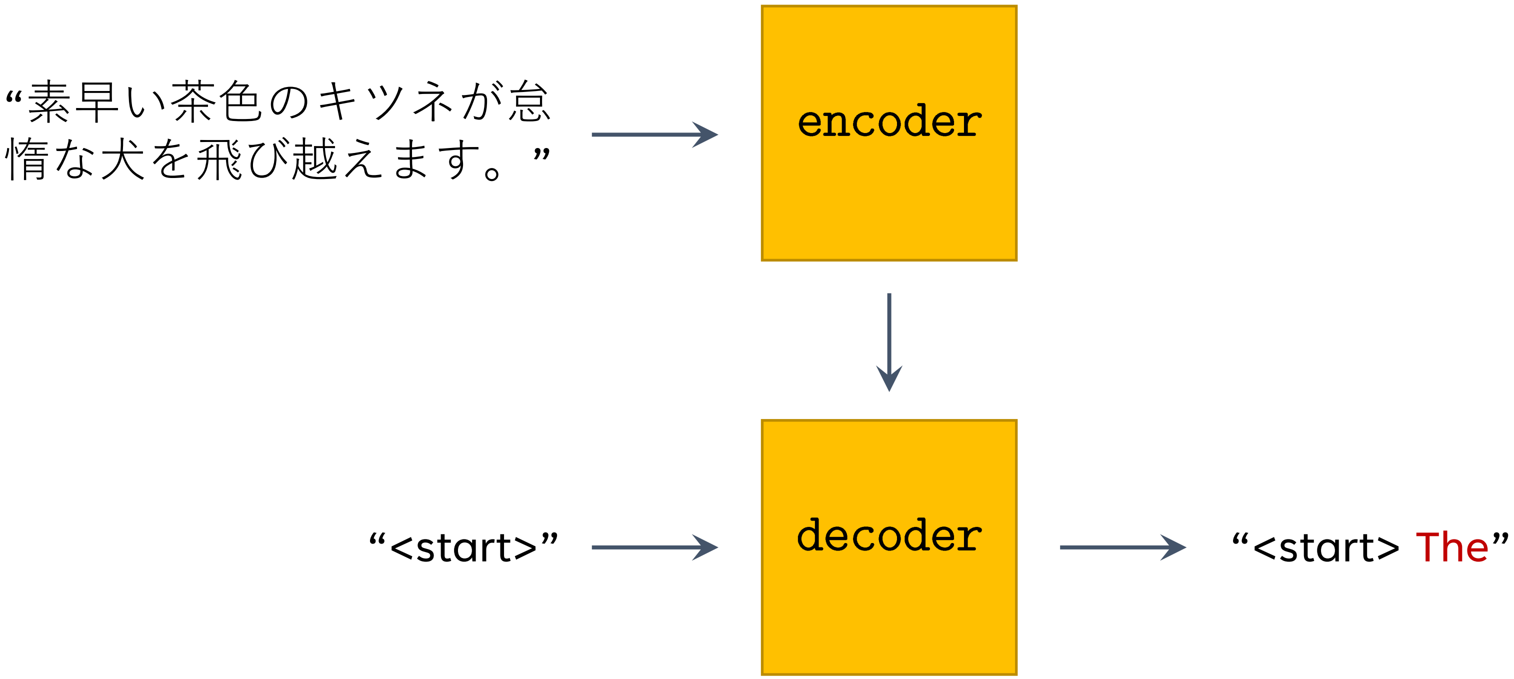
TRANSFORMER ENCODER-DECODER



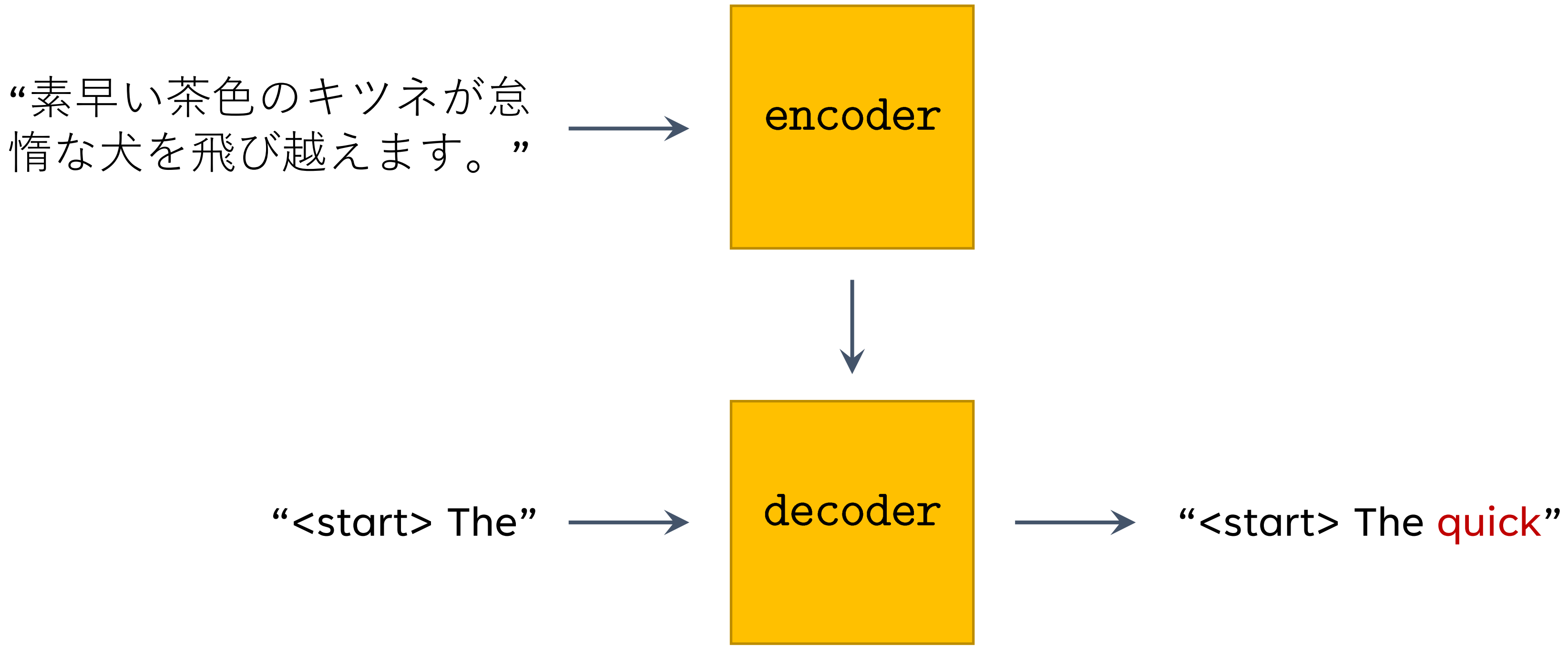
The left side is the encoder.
The encoder attention layers do not have the causal mask.

The right side is the decoder.
The decoder attention layers have the causal mask.

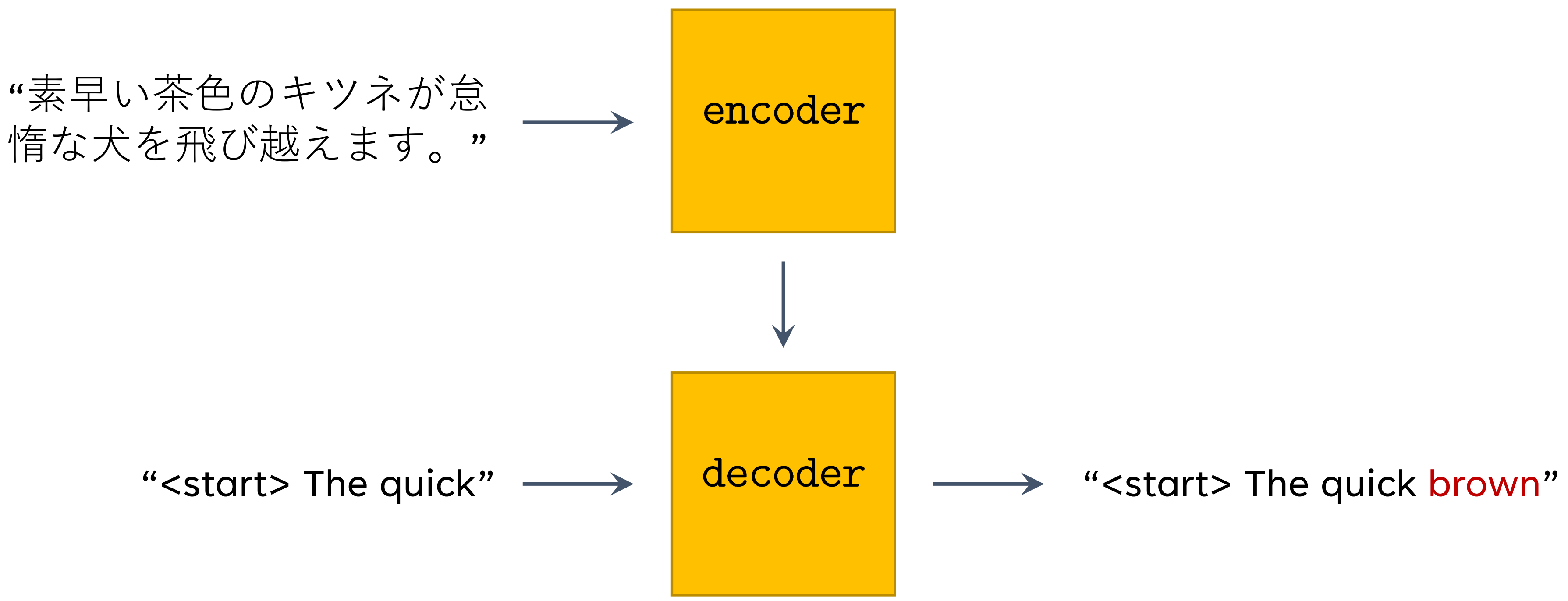
ENCODER-DECODER



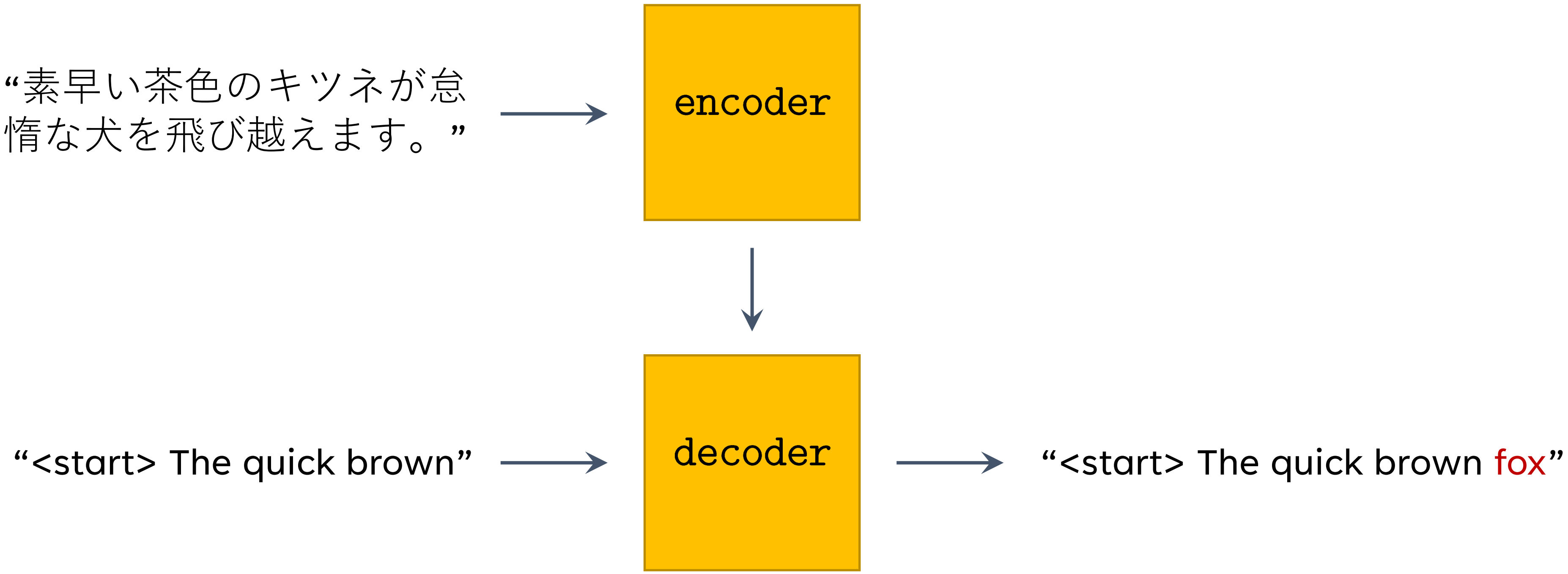
ENCODER-DECODER



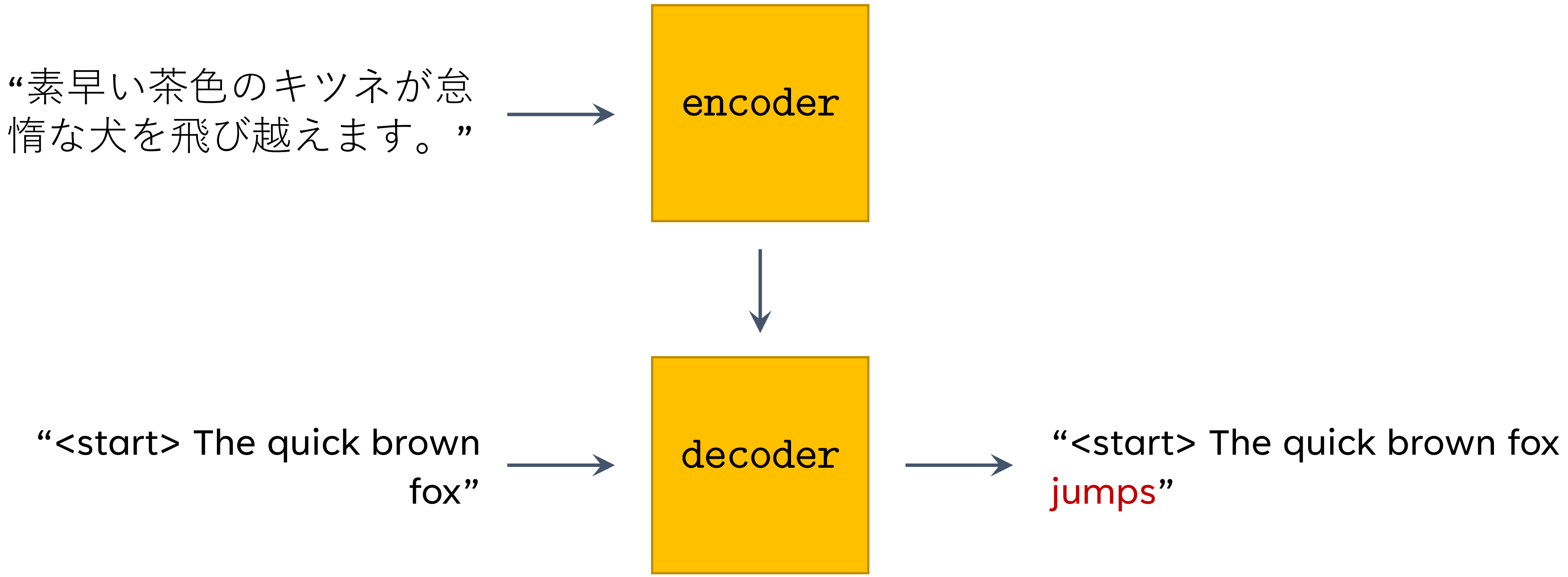
ENCODER-DECODER



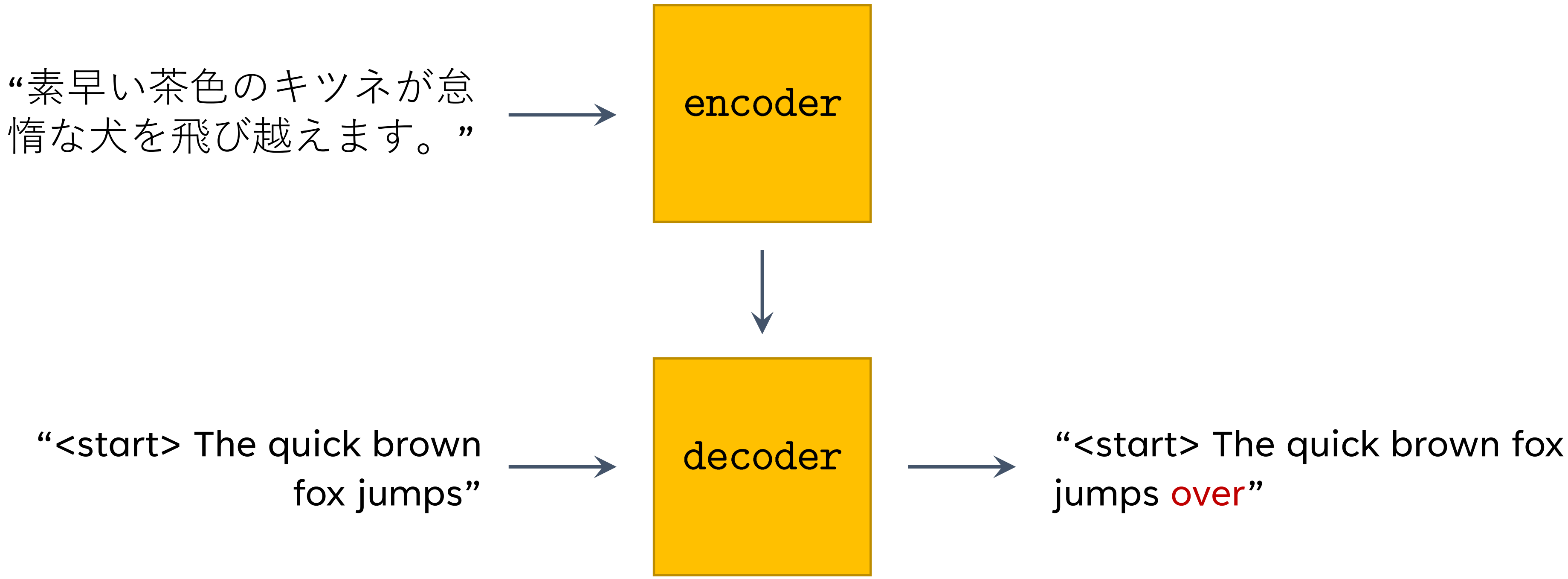
ENCODER-DECODER



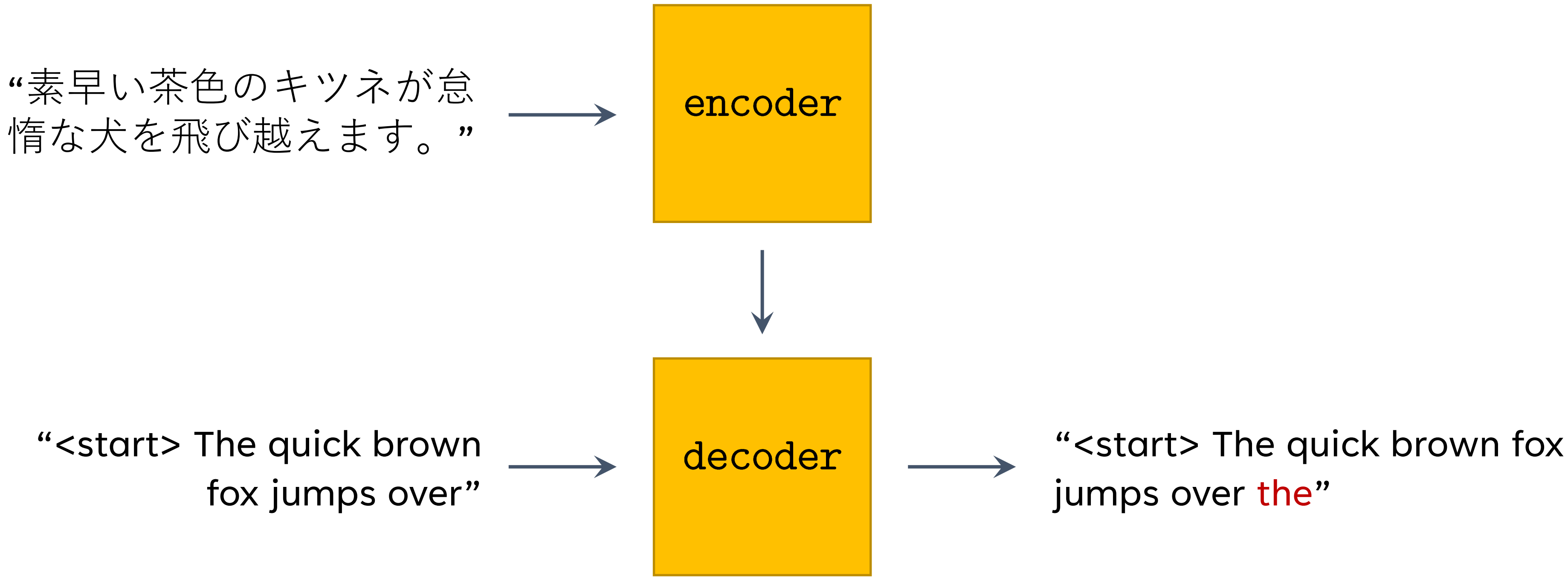
ENCODER-DECODER



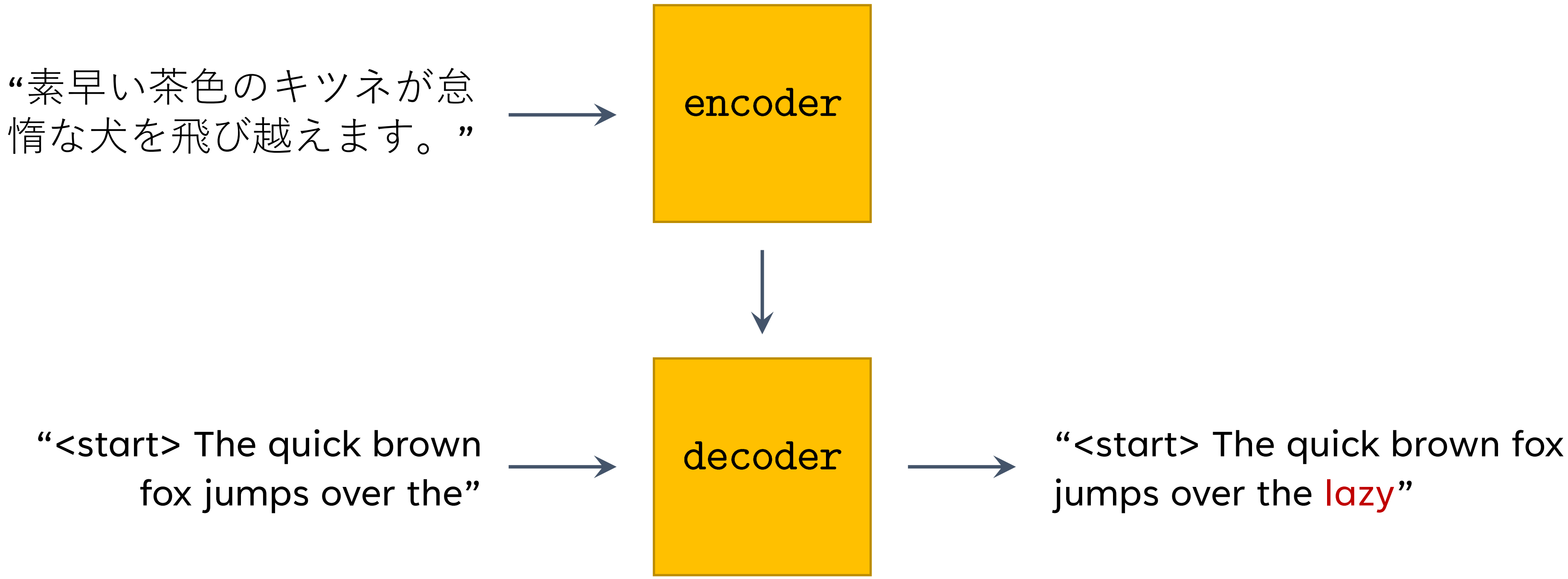
ENCODER-DECODER



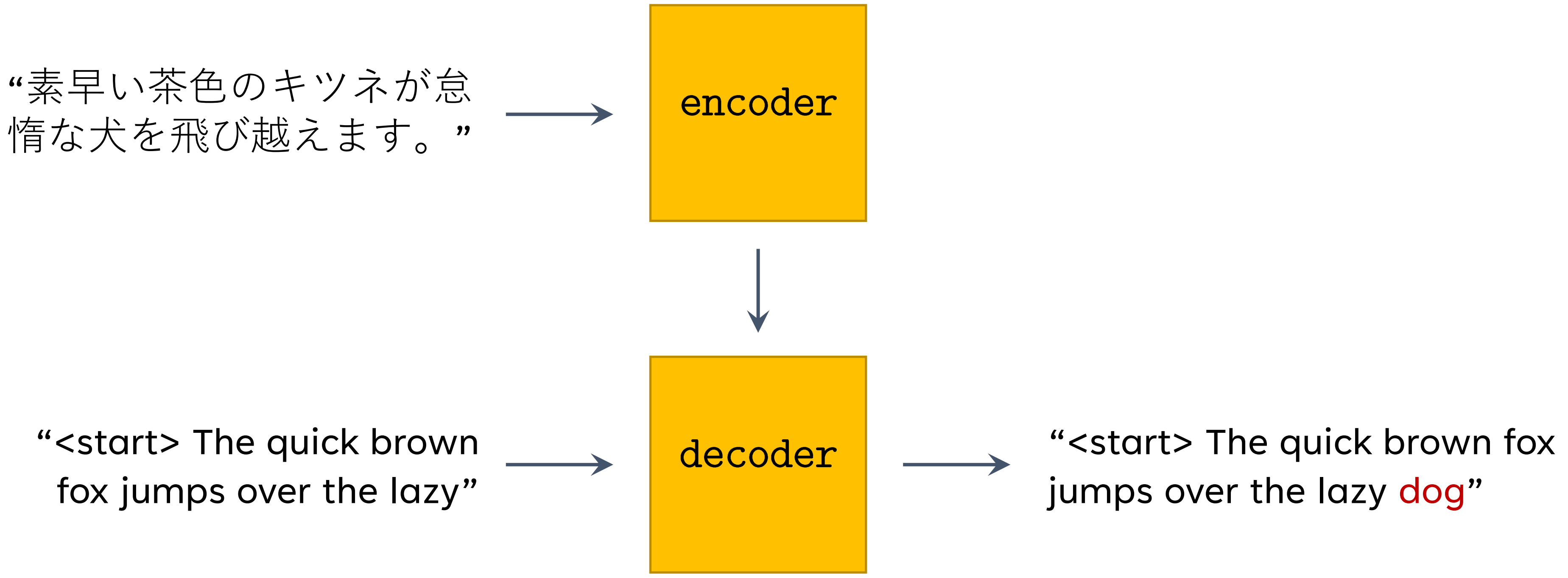
ENCODER-DECODER



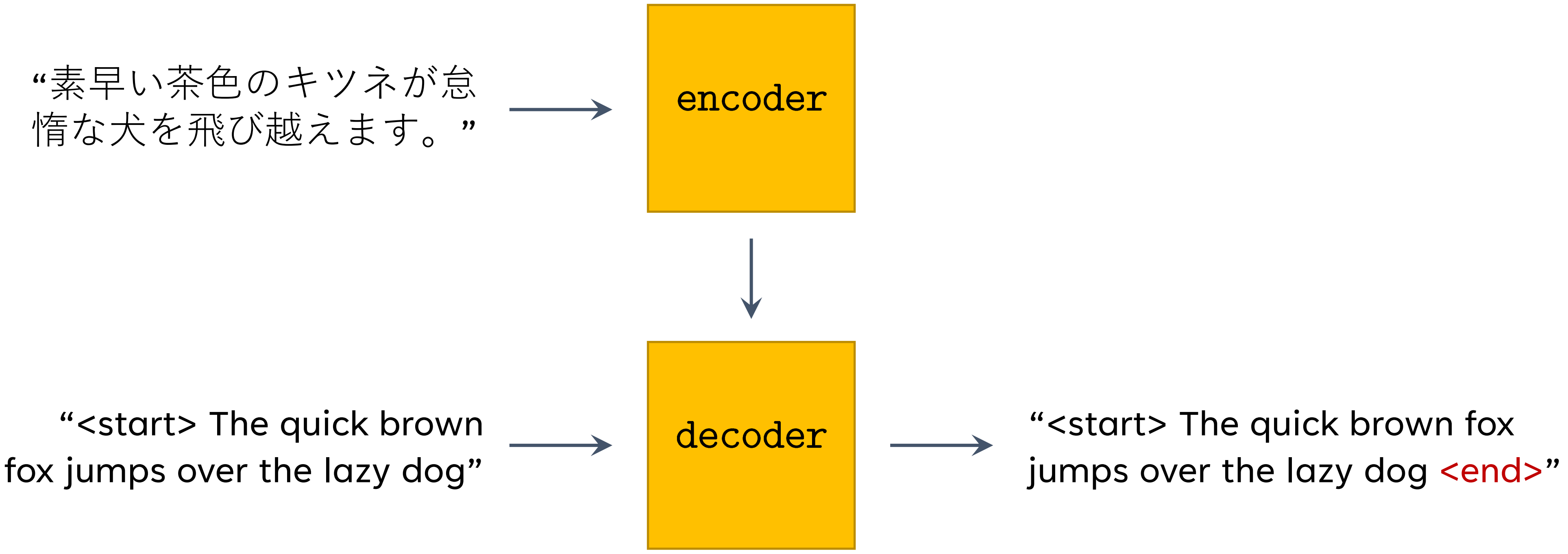
ENCODER-DECODER



ENCODER-DECODER

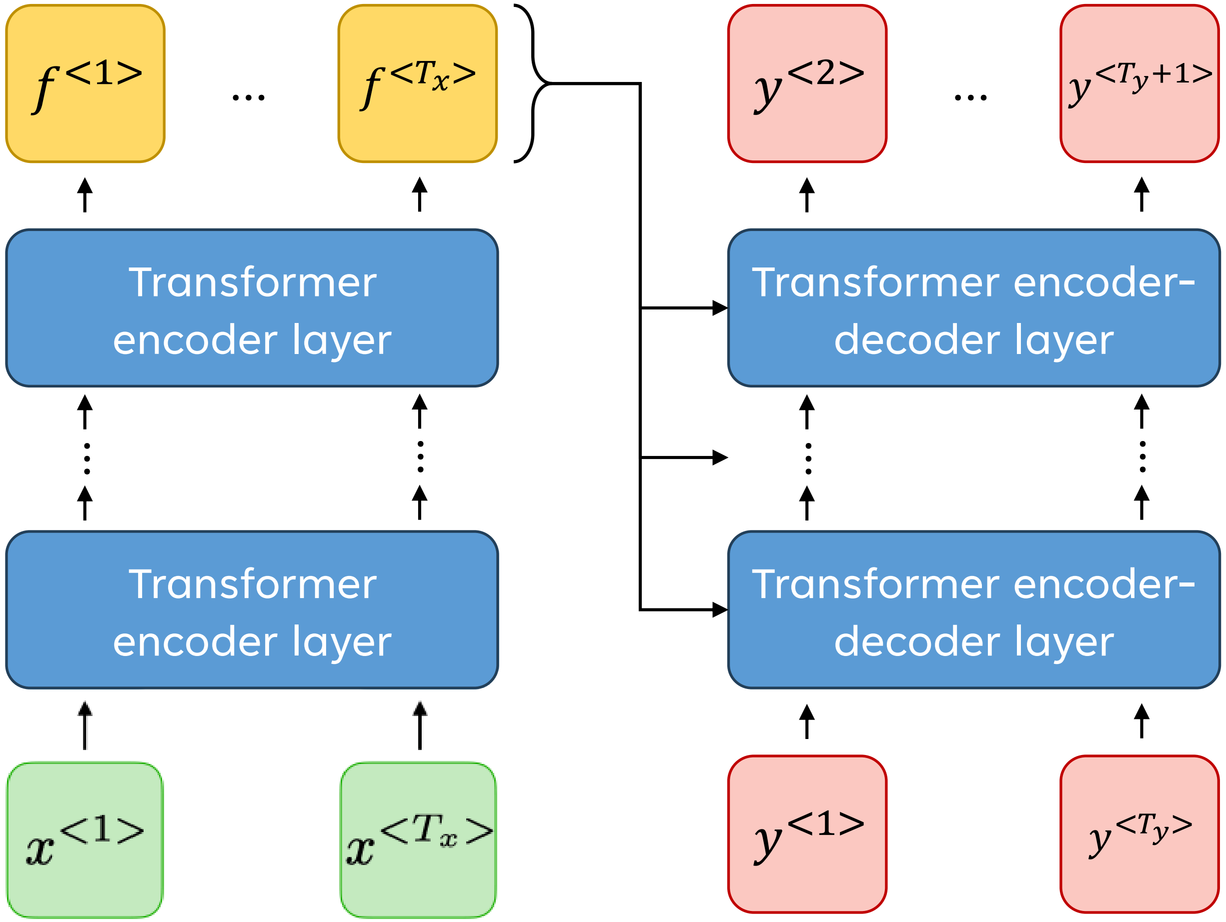


ENCODER-DECODER

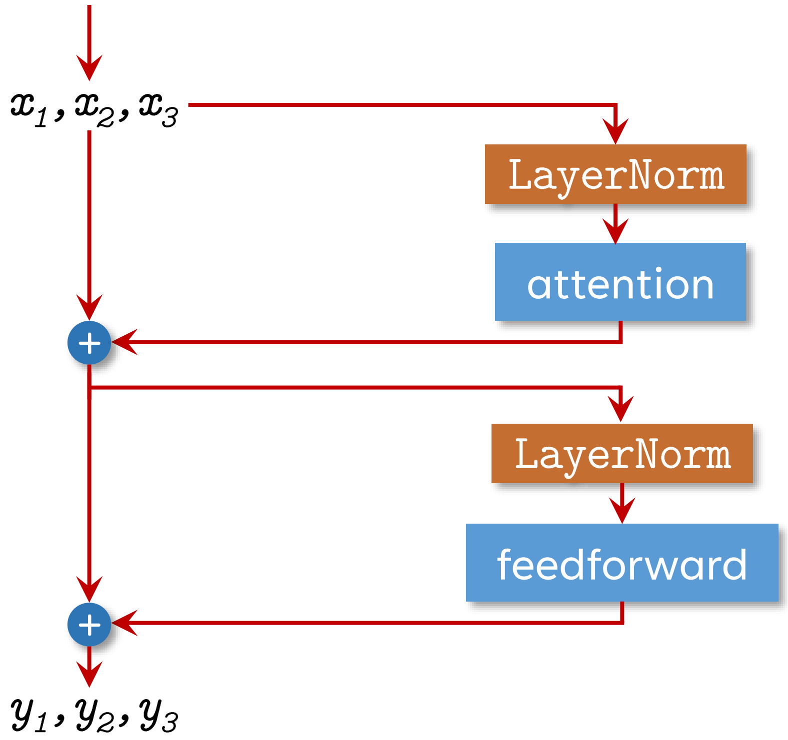


TRANSFORMER ENCODER-DECODER

How does the transformer encoder-decoder layer incorporate information from the encoder (i.e., features)?



TRANSFORMER LAYER

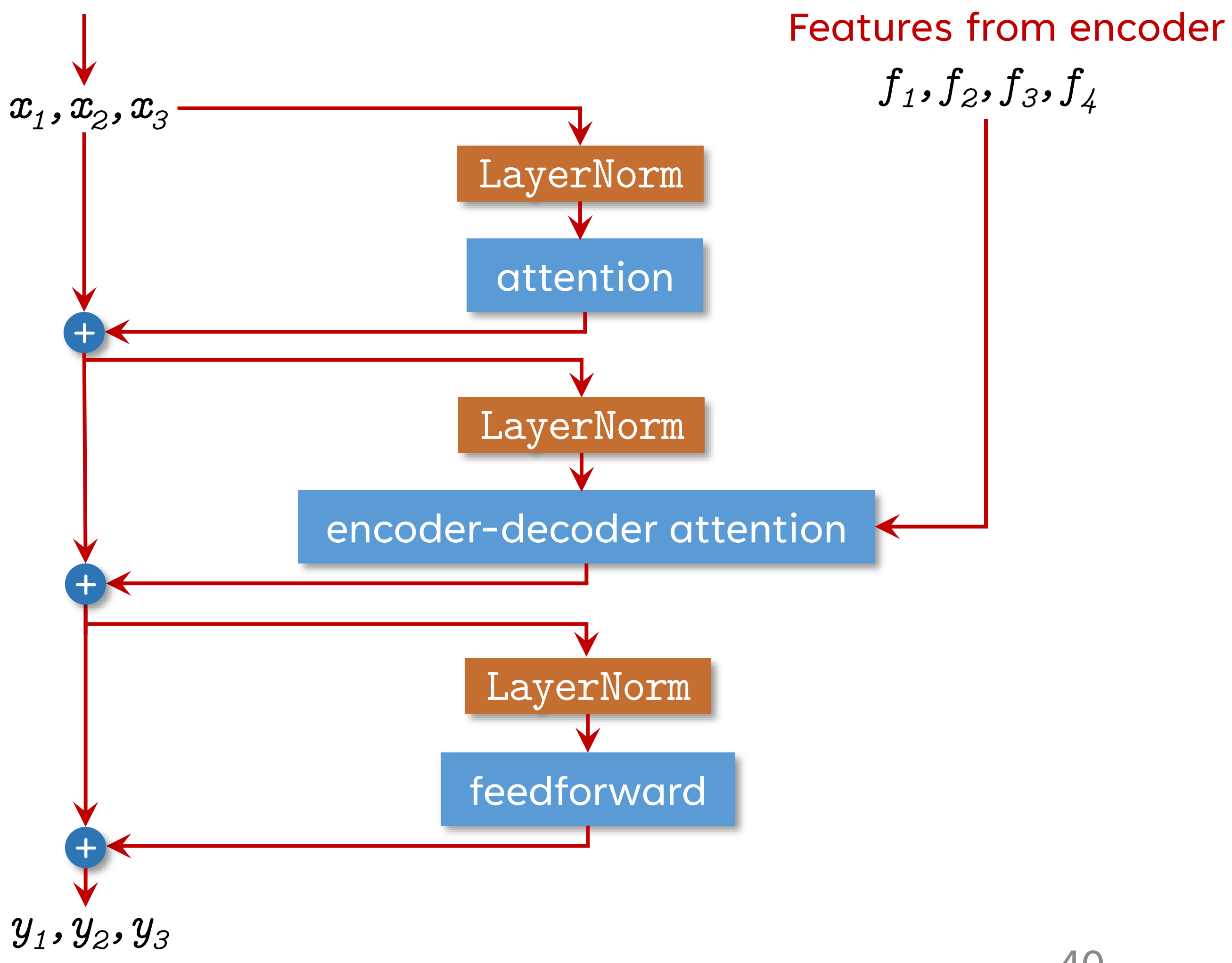


TRANSFORMER ENCODER-DECODER LAYER

The first attention layer has a causal mask.

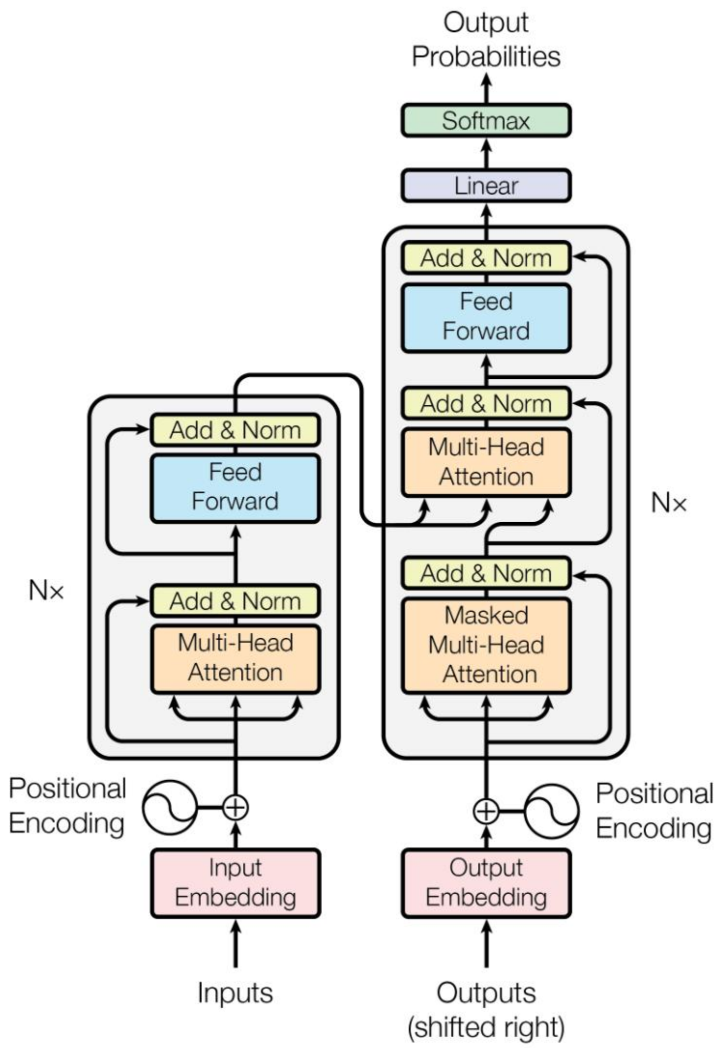
The encoder-decoder attention layer does not.

What does the encoder-decoder attention layer look like?



TRANSFORMER ENCODER-DECODER LAYER

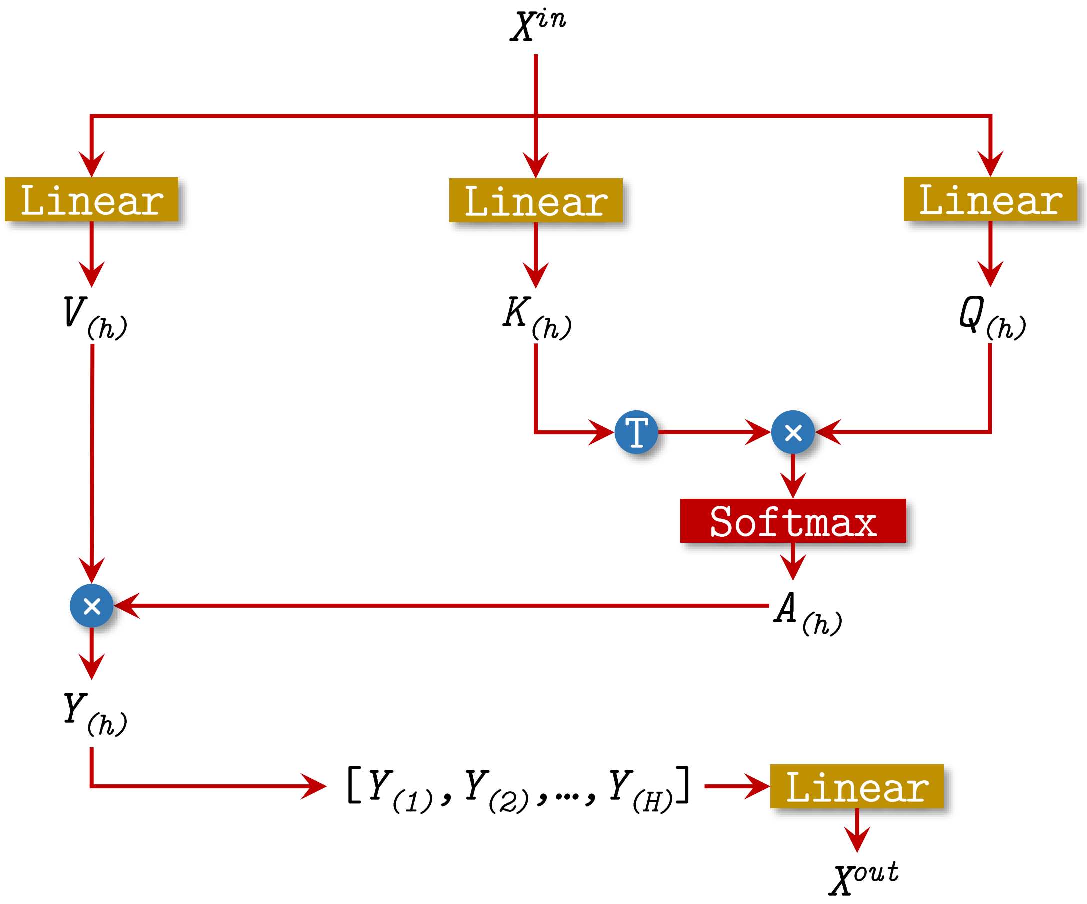
This is Figure 1 from the original transformer paper.



[Vaswani et al., 2017]

ATTENTION LAYER

Recall this is the circuit diagram for the multi-head attention component.

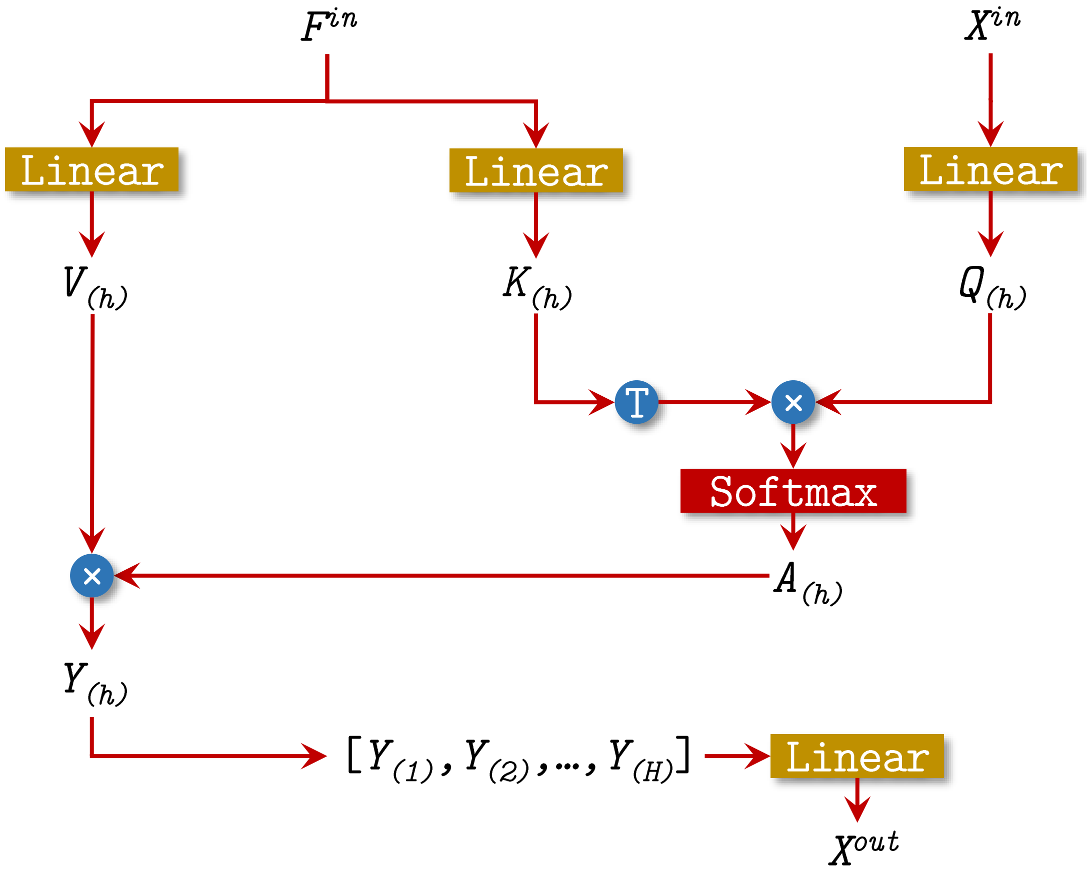


ENCODER-DECODER ATTENTION LAYER

X^{in} are the inputs from the previous decoder layer.

F^{in} are the inputs from the encoder (i.e., features).

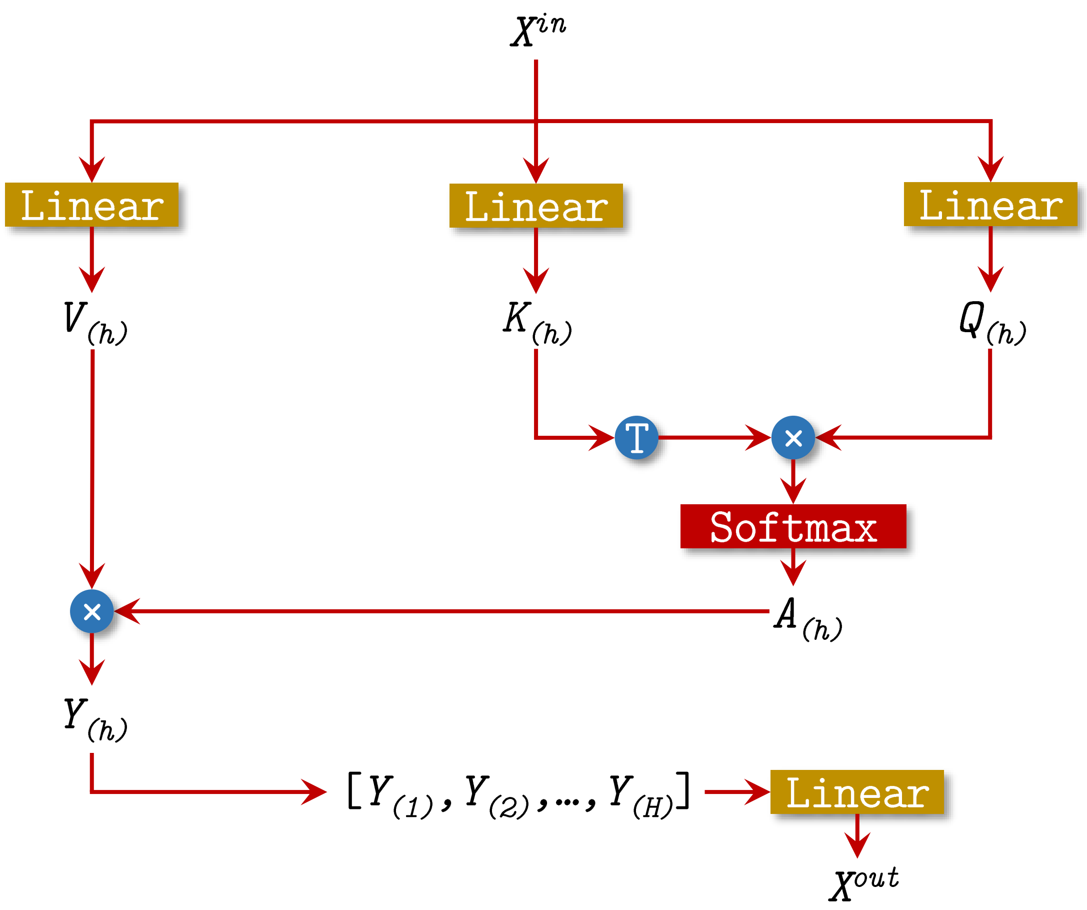
X^{in} is $n \times d_{model}$.
F^{in} is $m \times d_{model}$.
$V_{(h)}$ is $m \times d_{attn}$.
$K_{(h)}$ is $m \times d_{attn}$.
$Q_{(h)}$ is $n \times d_{attn}$.
$A_{(h)}$ is $n \times m$.
$Y_{(h)}$ is $n \times d_{attn}$.
X^{out} is $n \times d_{model}$.



ENCODER-ONLY AND DECODER-ONLY ATTENTION LAYER

So the attention layer where there is only 1 input is referred to as encoder-only or decoder-only.

Encoder-only attention layers do not have a causal mask.
Decoder-only attention layers have a causal mask.

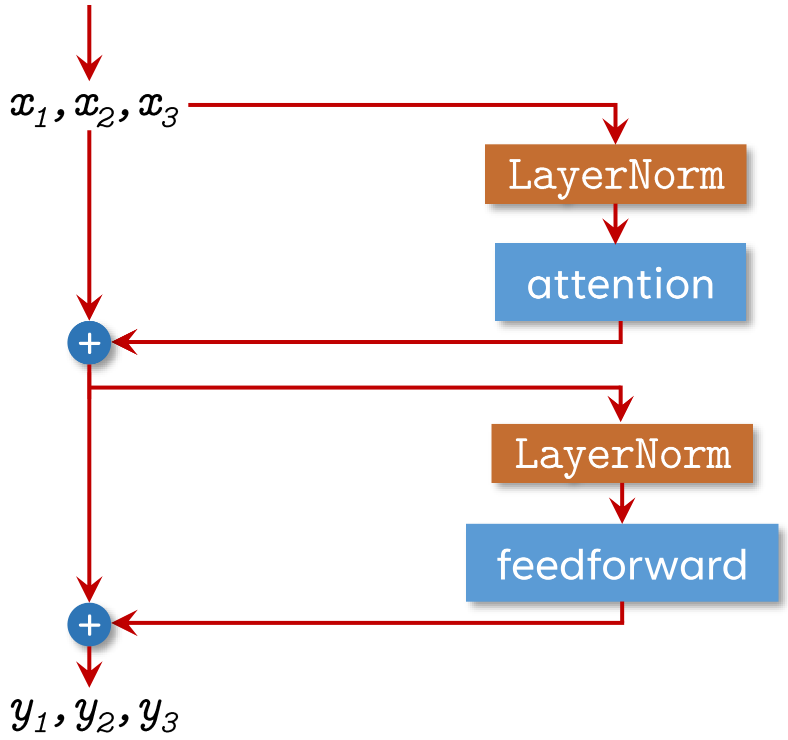


ENCODER-ONLY AND DECODER-ONLY TRANSFORMER LAYER

Similarly, a transformer layer without encoder-decoder attention is called an encoder-only or decoder-only transformer layer,

depending on whether it has a causal mask.

Encoder-only transformers are also called bidirectional transformers.



EXAMPLE ENCODER-DECODER TRANSFORMER MODELS

- Some example encoder-decoder models: (trivia)
  - BART (Lewis et al., 2019)
    - Up to 400M parameters.
  - T5 (Raffel et al., 2020)
    - Up to 11B parameters.
  - UnifiedQA (Khashabi et al., 2020)

EXAMPLE ENCODER-ONLY TRANSFORMER MODELS

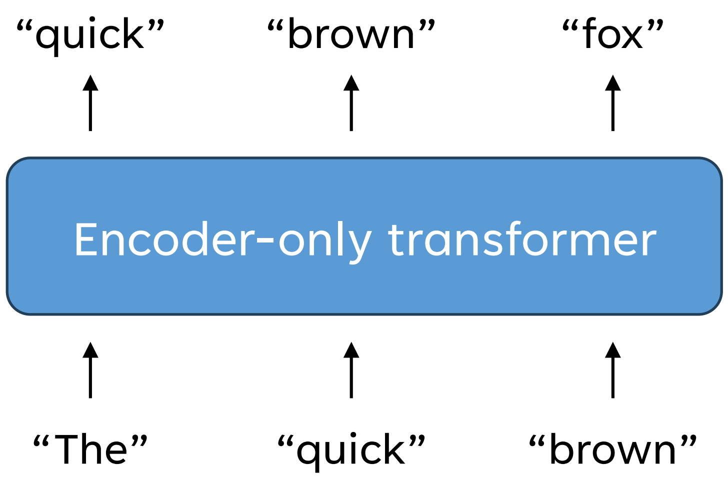
- Some example encoder-only models: (trivia)
  - BERT (Devlin et al., 2018)
    - Up to 355M parameters.
  - RoBERTa (Liu et al., 2019)
    - Up to 355M parameters.
  - DeBERTa (He et al., 2021)
    - Up to 1.5B parameters.

EXAMPLE DECODER-ONLY TRANSFORMER MODELS

- Modern language models are decoder-only models. (trivia)
  - GPT-2 (Radford et al., 2019)
    - Up to 1.5B parameters.
  - GPT-3 (Brown et al., 2020)
    - Up to 175B parameters.
  - GPT-4 (OpenAI, 2023)
    - (unofficial) Up to 1.7T parameters (111B per expert).
  - LLaMA 3 (Meta, 2024)
    - Up to 405B parameters.
  - DeepSeek-V3 (DeepSeek-AI, 2024)
    - Up to 671B parameters (37B per expert).

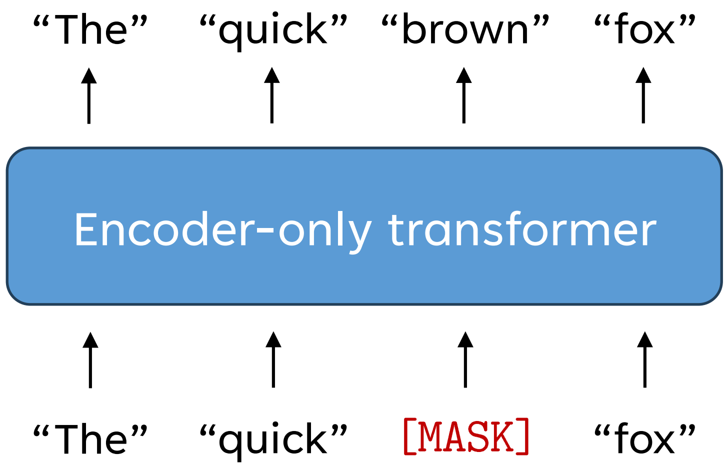
TRAINING ENCODER-ONLY MODELS

- How are encoder-only models like BERT trained?
- We can’t use autoregressive language modeling since without the causal mask, encoder-only transformers can “cheat” by copying the next token.



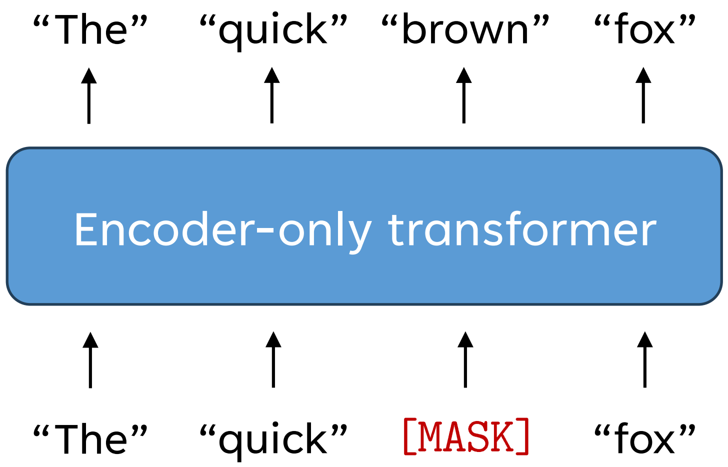
TRAINING ENCODER-ONLY MODELS

- How are encoder-only models like BERT trained?
- Instead, we use the masked language modeling task.
- Randomly replace some percentage of the input words with [MASK].
- The model's task is to fill in the masked words/tokens.



TRAINING ENCODER-ONLY MODELS

- How are encoder-only models like BERT trained?
- Measure the loss function only on the masked tokens.



QUESTIONS?