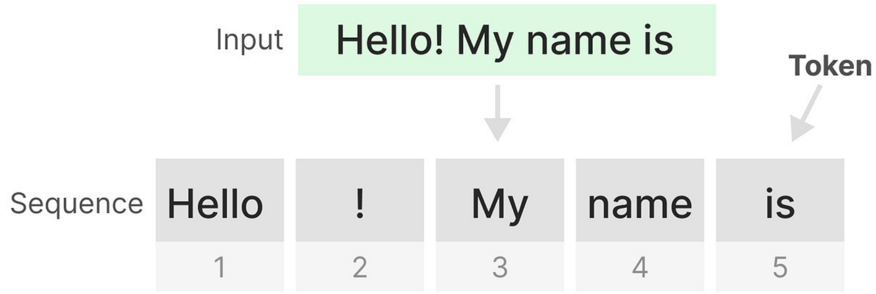
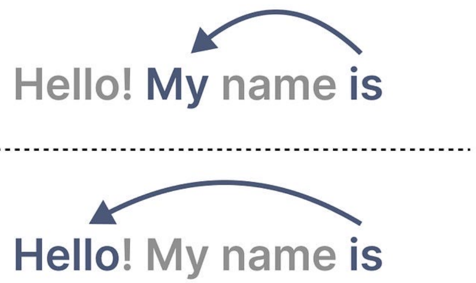
Mamba: Linear-Time Sequence Modeling with Selective State Spaces

The problem of transformer

The Core Concepts of Transformer

Prerequisites: What is Transformer (know the code is better), tokenizer, embedding layer





Transformer can selectively and individually look at previous tokens

The Core Concepts of Transformer

How Transformer-based model make an inference

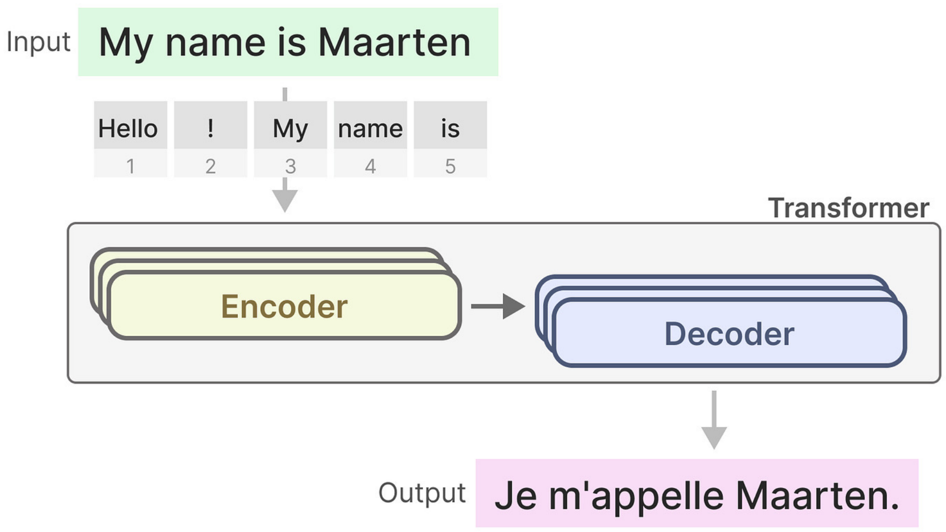


Fig Encoder-Decoder Model

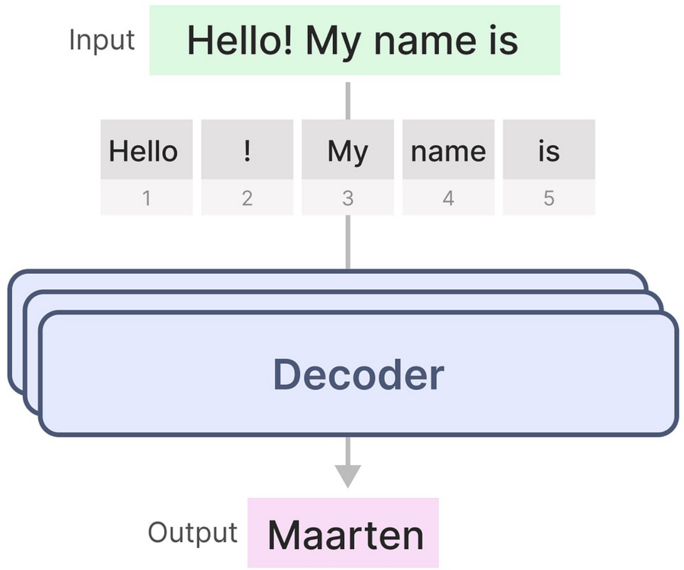
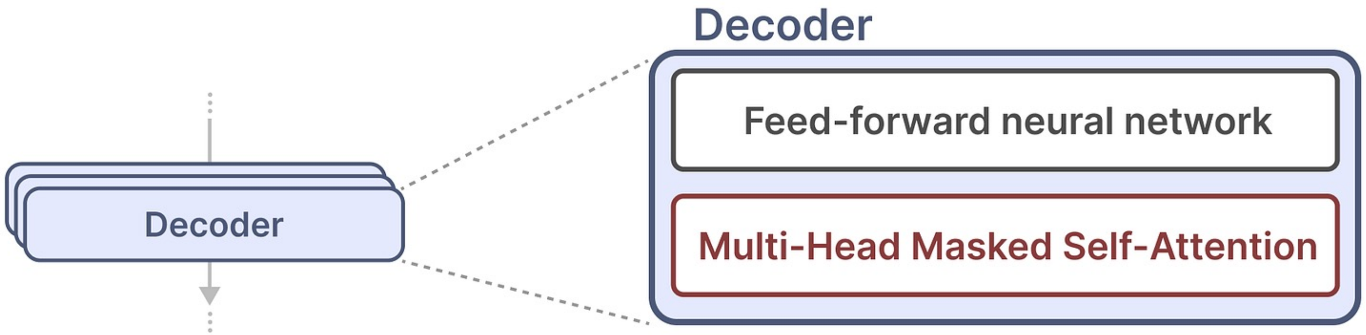


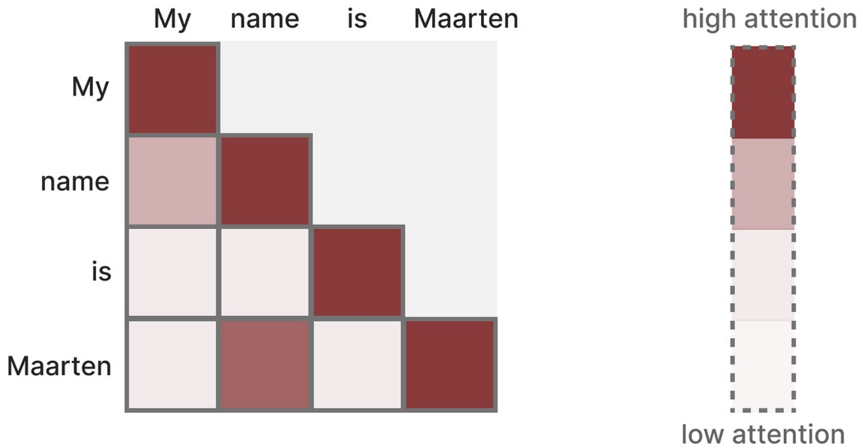
Fig Decoder-only Model

Nice when Training

Components of Transformer block(Decoder only):



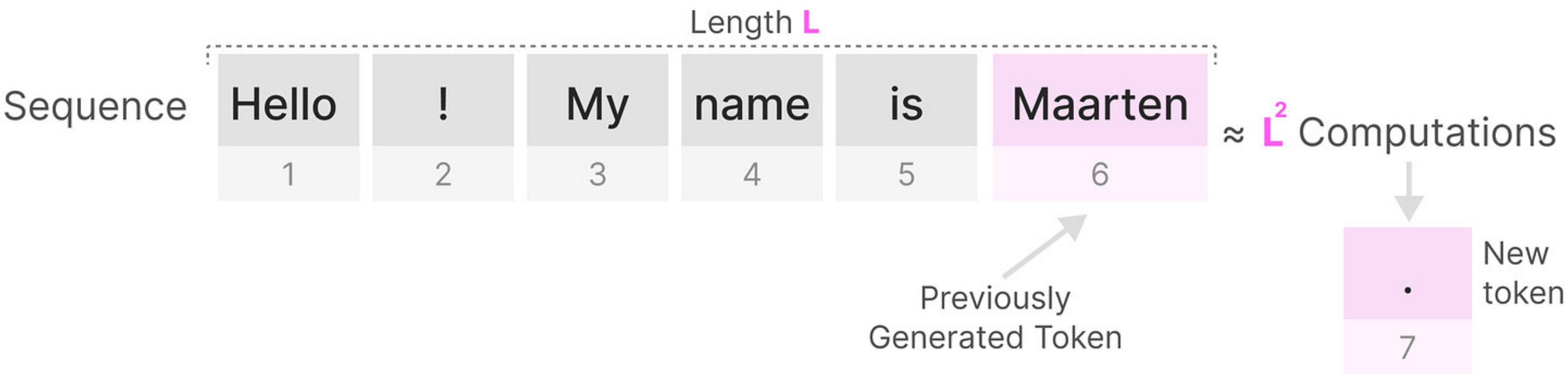
Muti-head masked self-attention:



Nice **parallelization**

Too bad when Testing

When generate the next token, we need to recalculate the attention matrix for the whole sequence (without considering Kv_cache.

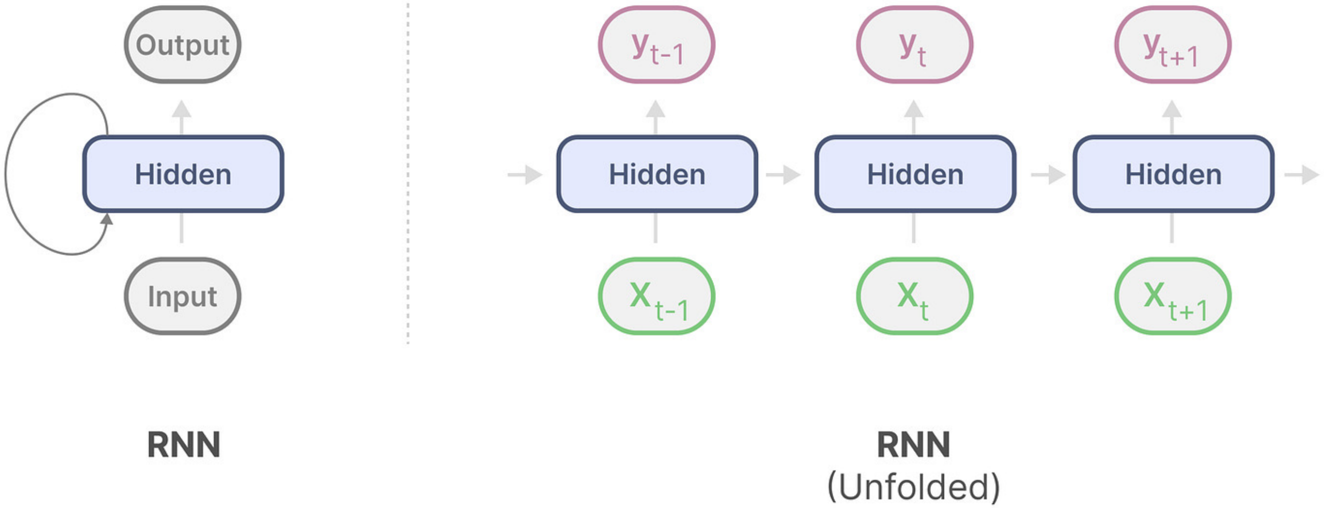


Flaws: Generating tokens for a sequence of L needs L^2 computations which is costly if sequence length increases

	Training	Inference
Transformer	Fast (parallelizable)	Slow (scales quadratically with sequence length)

RNN is a solution???

RNN model (Model input: x_t, hidden_states)



Training:
For any block, like block_t, it needs x_t and $hidden_{t-1}$ as inputs.
$hidden_{t-1}$ is passed by block_(t-1)
Thus, it cannot be trained in a parallelizable way.

Inference:
Nearly linearly inference.

Summary

	Training	**Inference**
Transformer	Fast (**parallelizable**)	Slow (scales **quadratically** with sequence length)
RNN	Slow (**not parallelizable**)	Fast (scales **linearly** with sequence length)

Can we somehow find an architecture that does parallelize training like Transformers while still performing inference that scales linearly with sequence length?
(Yes! That is what SSM do)

For the next video, we will talk about the basis of SSM and introduce the first Mamba series model S4.