On Transformers and Bitter Lesson

15.09.2025 – Ivan Rubachev
Yandex Research & HSE

Brain-dump lecture

But with some technical bits

Links would be at the end

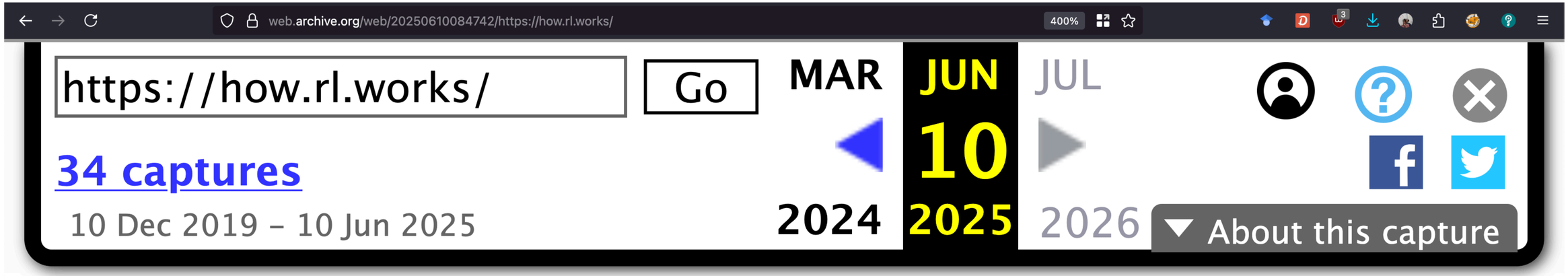
- resources
- blog posts, papers
- sources of info

I hope they are useful

How to look at research?

Has Transformer changed?

Let’s discuss/debate*


web.archive.org/web/20250610084742/https://how.rl.works/
400%
https://how.rl.works/
Go
34 captures
10 Dec 2019 - 10 Jun 2025
MAR
JUN
10
2025
JUL
2026
2024
About this capture

Badly


Firefox File Edit View History Bookmarks Tools Window Help
Colemak
Sun 14 Sep 00:35
How RL works
how.rl.works
500%

well

Is Attention All You Need?

Current Status: Yes

Time Remaining: 474d 12h 32m 12s

Proposition:

On January 1, 2027, a Transformer-like model will continue to hold the state-of-the-art position in most benchmarked tasks in natural language processing.

For the Motion

Jonathan Frankle
@jefrankle
Harvard Professor
Chief Scientist Mosaic ML

Against the Motion

Sasha Rush
@srush_nlp
Cornell Professor
Research Scientist Hugging Face 🤗

https://www.isattentionallyouneed.com/

The Bitter Lesson

Not Secure http://incompleteideas.net/IncIdeas/BitterLesson.html

The Bitter Lesson

Rich Sutton

March 13, 2019

The biggest lesson that can be read from 70 years of AI research is that general methods that leverage computation are ultimately the most effective, and by a large margin. The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation. Most AI research has been conducted as if the computation available to the agent were constant (in which case leveraging human knowledge would be one of the only ways to improve performance) but, over a slightly longer time than a typical research project, massively more computation inevitably becomes available. Seeking an improvement that makes a difference in the shorter term, researchers seek to leverage their human knowledge of the domain, but the only thing that matters in the long run is the leveraging of computation. These two need not run counter to each other, but in practice they tend to. Time spent on one is time not spent on the other. There are psychological commitments to investment in one approach or the other. And the human-knowledge approach tends to complicate methods in ways that make them less suited to taking advantage of general methods leveraging computation. There were many examples of AI researchers' belated learning of this bitter lesson, and it is instructive to review some of the most prominent.

In computer chess, the methods that defeated the world champion, Kasparov, in 1997, were based on massive, deep search. At the time, this was looked upon with dismay by the majority of computer-chess researchers who had pursued methods that leveraged human understanding of the special structure of chess. When a simpler, search-based approach with special hardware and software proved vastly more effective, these human-knowledge-based chess researchers were not good losers. They said that ``brute force" search may have won this time, but it was not a general strategy, and anyway it was not how people played chess. These researchers wanted methods based on human input to win and were disappointed when they did not.

A similar pattern of research progress was seen in computer Go, only delayed by a further 20 years. Enormous initial efforts went into avoiding search by taking advantage of human knowledge, or of the special features of the game, but all those efforts proved irrelevant, or worse, once search was applied effectively at scale. Also important was the use of learning by self play to learn a value function (as it was in many other games and even in chess, although learning did not play a big role in the 1997 program that first beat a world champion). Learning by self play, and learning in general, is like search in that it enables massive computation to be brought to bear. Search and learning are the two most important classes of techniques for utilizing massive amounts of computation in AI research. In computer Go, as in computer chess, researchers' initial effort was directed towards utilizing human understanding (so that less search was needed) and only much later was much greater success had by embracing search and learning.

In speech recognition, there was an early competition, sponsored by DARPA, in the 1970s. Entrants included a host of special methods that took advantage of human knowledge---knowledge of words, of phonemes, of the human vocal tract, etc. On the other side were newer methods that were more statistical in nature and did much more computation, based on hidden Markov models (HMMs). Again, the statistical methods won out over the human-knowledge-based methods. This led to a major change in all of natural language processing, gradually over decades, where statistics and computation came to dominate the field. The recent rise of deep learning in speech recognition is the most recent step in this consistent direction. Deep learning methods rely even less on human knowledge, and use even more computation, together with learning on huge training sets, to produce dramatically better speech recognition systems. As in the games, researchers always tried to make systems that worked the way the researchers thought their own minds worked---they tried to put that knowledge in their systems---but it proved ultimately counterproductive, and a colossal waste of researcher's time, when, through Moore's law, massive computation became available and a means was found to put it to good use.

In computer vision, there has been a similar pattern. Early methods conceived of vision as searching for edges, or generalized cylinders, or in terms of SIFT features. But today all this is discarded. Modern deep-learning neural networks use only the notions of convolution and certain kinds of invariances, and perform much better.

This is a big lesson. As a field, we still have not thoroughly learned it, as we are continuing to make the same kind of mistakes. To see this, and to effectively resist it, we have to understand the appeal of these mistakes. We have to learn the bitter lesson that building in how we think we think does not work in the long run. The bitter lesson is based on the historical observations that 1) AI researchers have often tried to build knowledge into their agents, 2) this always helps in the short term, and is personally satisfying to the researcher, but 3) in the long run it plateaus and even inhibits further progress, and 4) breakthrough progress eventually arrives by an opposing approach based on scaling computation by search and learning. The eventual success is tinged with bitterness, and often incompletely digested, because it is success over a favored, human-centric approach.

One thing that should be learned from the bitter lesson is the great power of general purpose methods, of methods that continue to scale with increased computation even as the available computation becomes very great. The two methods that seem to scale arbitrarily in this way are *search* and *learning*.

The second general point to be learned from the bitter lesson is that the actual contents of minds are tremendously, irredeemably complex; we should stop trying to find simple ways to think about the contents of minds, such as simple ways to think about space, objects, multiple agents, or symmetries. All these are part of the arbitrary, intrinsically-complex, outside world. They are not what should be built in, as their complexity is endless; instead we should build in only the meta-methods that can find and capture this arbitrary complexity. Essential to these methods is that they can find good approximations, but the search for them should be by our methods, not by us. We want AI agents that can discover like we can, not which contain what we have discovered. Building in our discoveries only makes it harder to see how the discovering process can be done.

UNIVERSITY OF ALBERTA

Folio

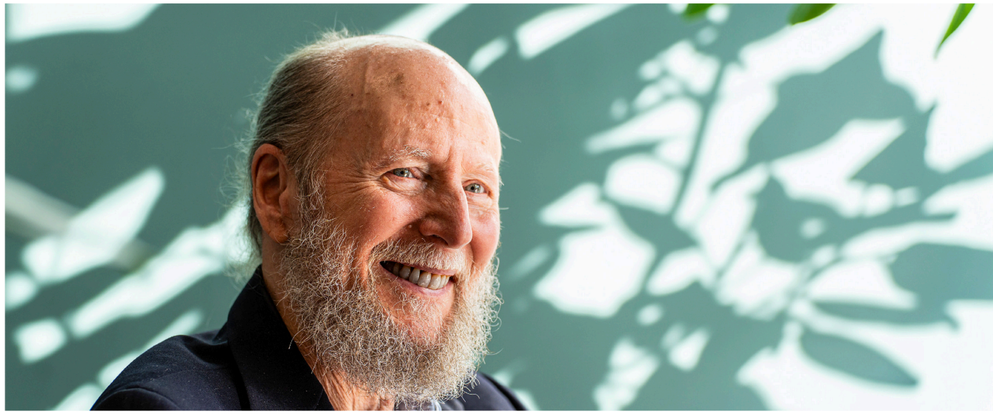
Science & Tech Health & Wellness Society & Culture Business Commentary

SCIENCE AND TECHNOLOGY, RESEARCH

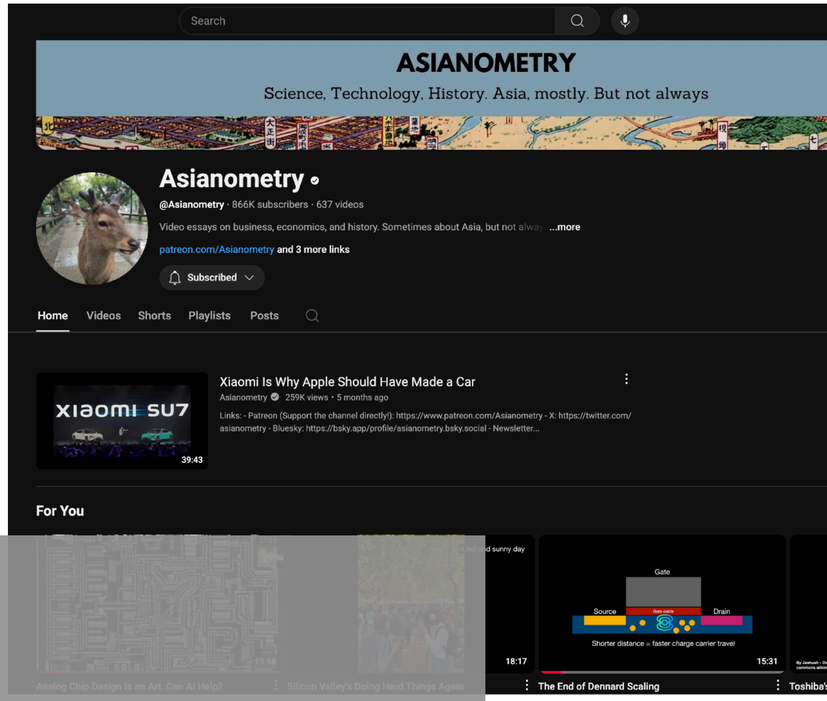
Computing science professor wins 'Nobel Prize in computing'

Richard Sutton is co-recipient of the 2024 A.M. Turing Award for his work as one of the founders of reinforcement learning.

MARCH 05, 2025 BY ADRIANNA MACPHERSON



http://incompleteideas.net/IncIdeas/BitterLesson.html

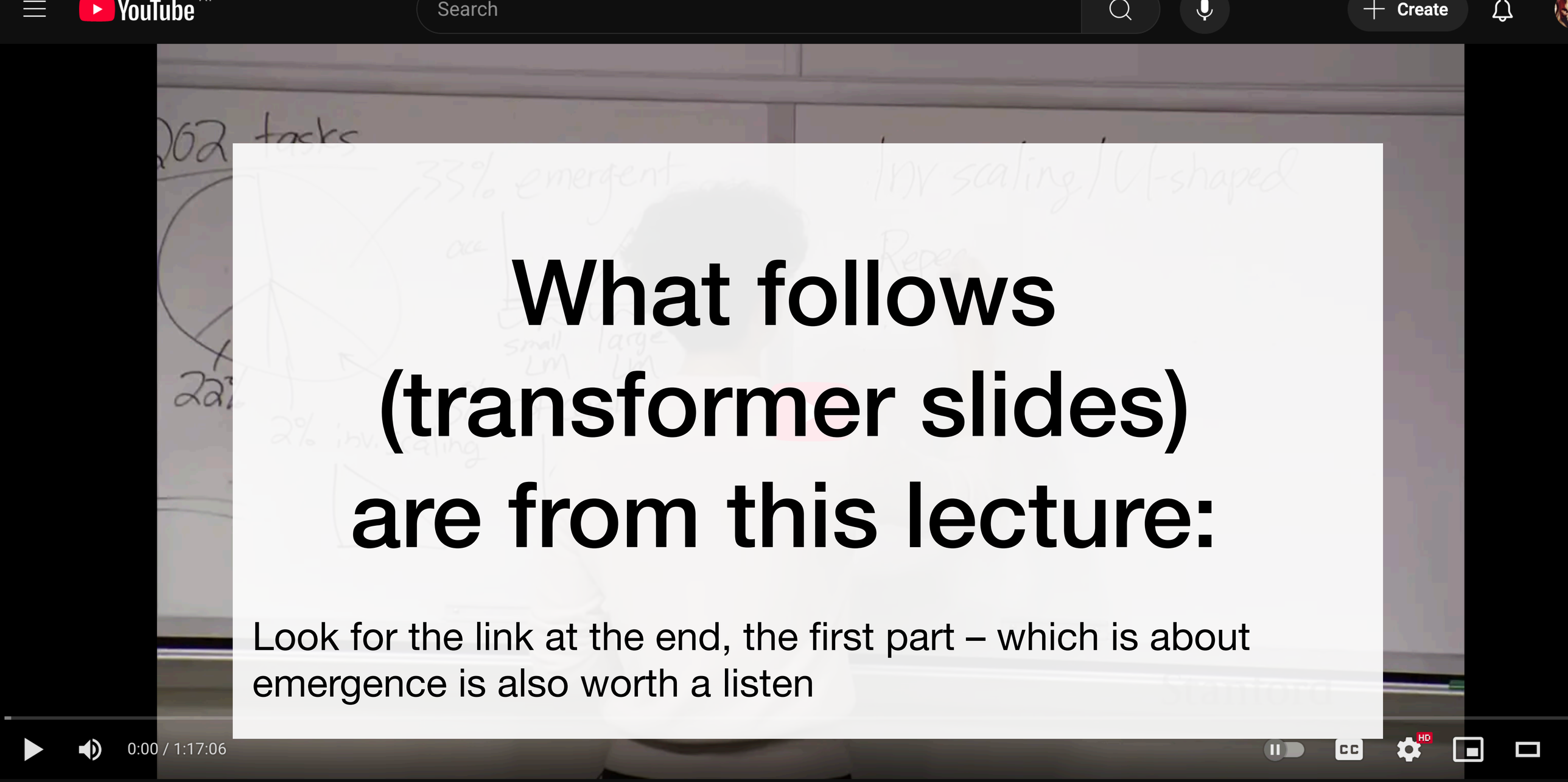


“

The biggest lesson that can be read from 70 years of AI research is that general **methods that leverage computation are ultimately the most effective**, and by a large margin.
The ultimate reason for this is Moore's law, or rather its generalization of continued exponentially falling cost per unit of computation.

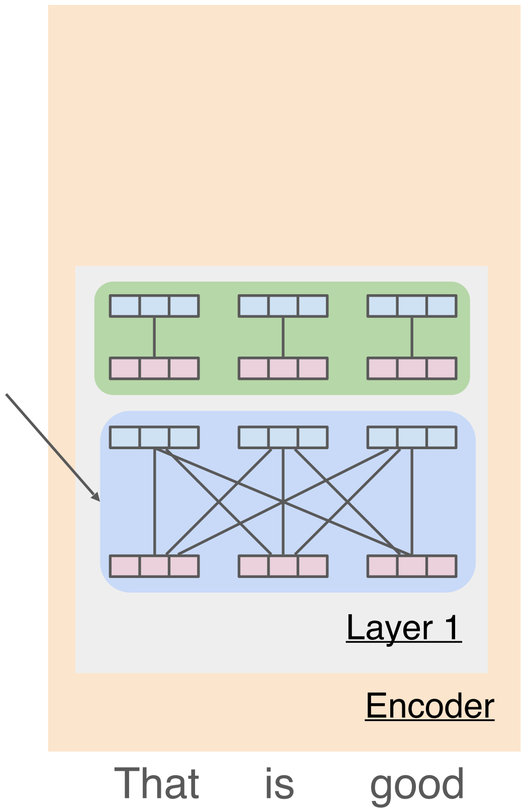
Transformers

Short recap and a note on structure


YouTube
Search
Create
202 tasks
33% emergent
for scaling / U-shaped
22%
2% increasing
What follows
(transformer slides)
are from this lecture:
Look for the link at the end, the first part – which is about emergence is also worth a listen
0:00 / 1:17:06
Stanford Cs25: V4 I Jason Wei & Hyung Won Chung of OpenAI
Stanford Online
851K subscribers
Subscribe
4.6K
Share
Save
All
From Stanford Online
AI
Computers

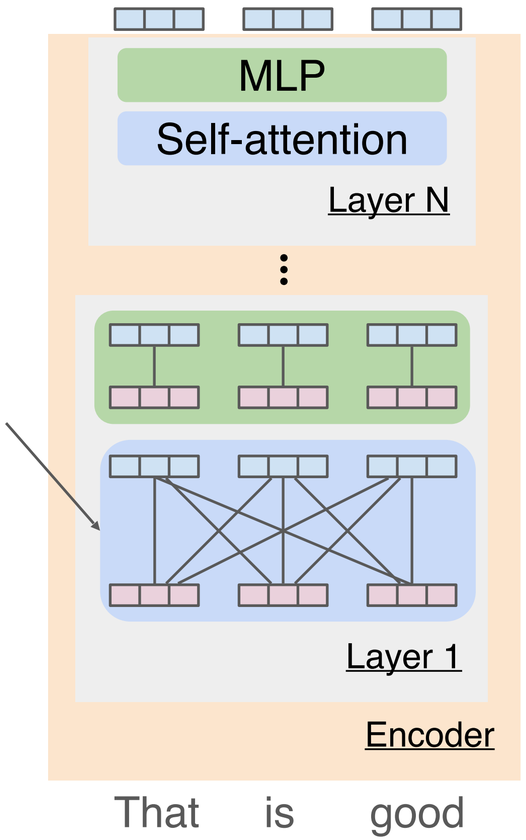
Encoder-decoder

Bidirectional self-attention

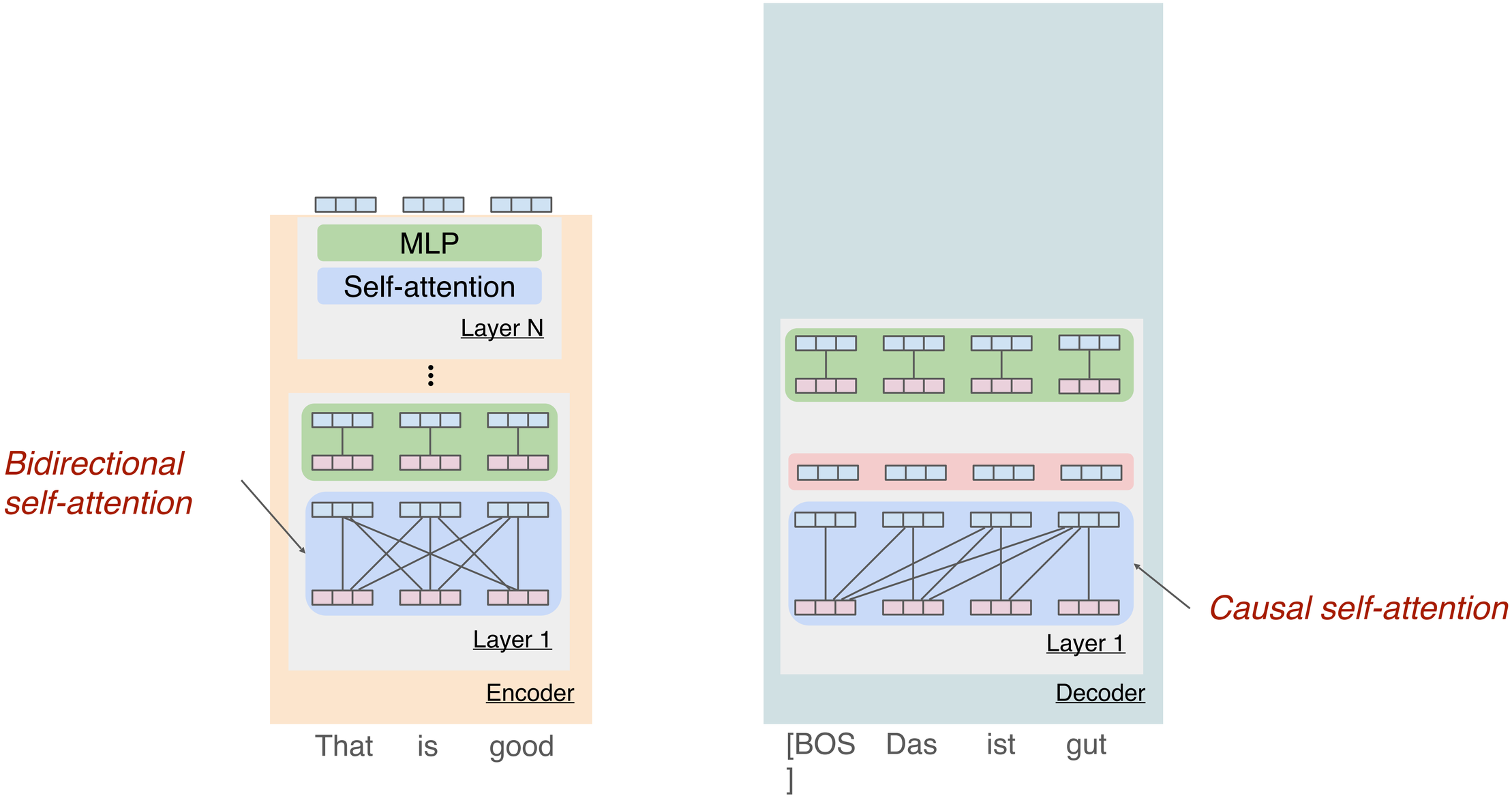


Encoder-decoder

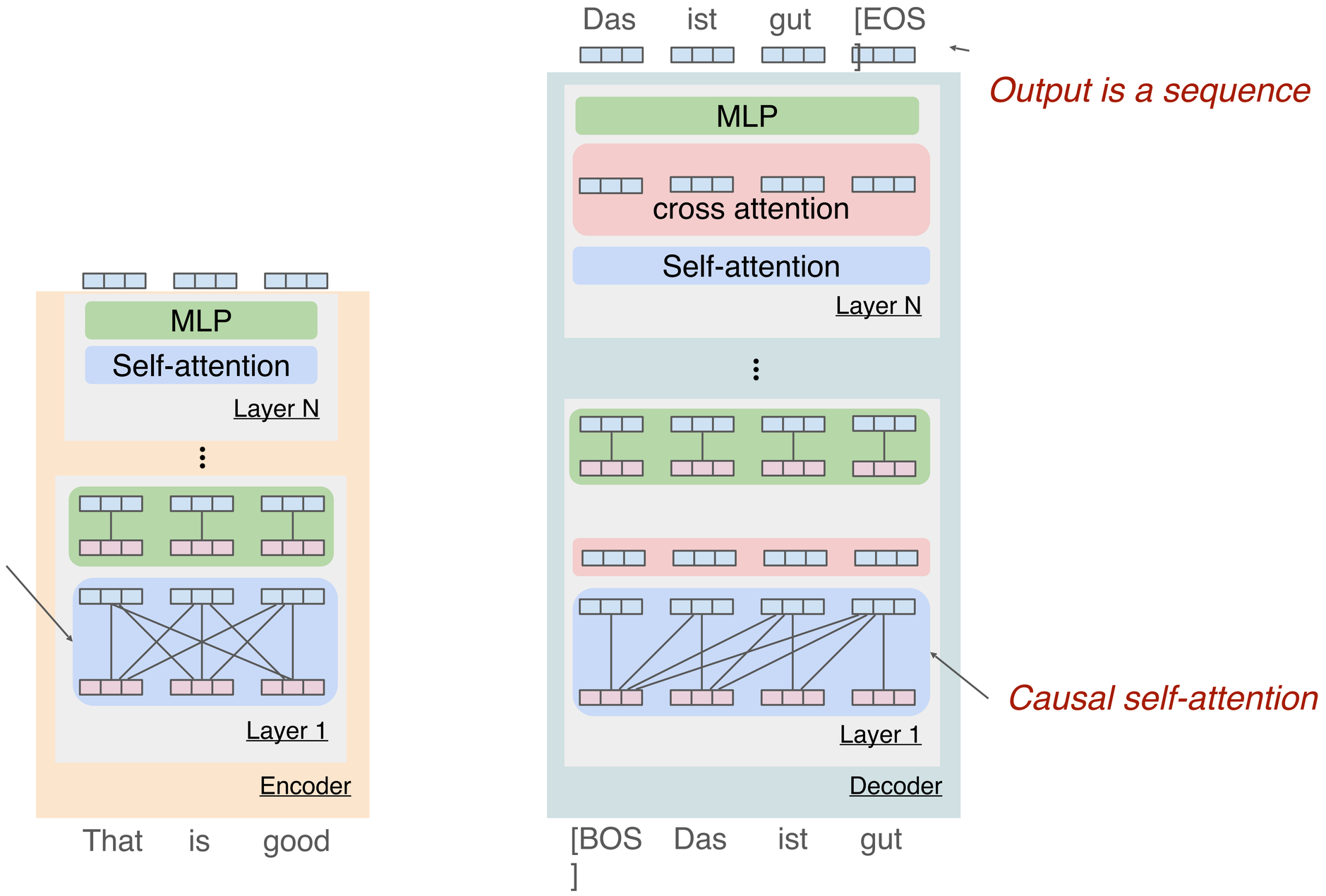
Bidirectional self-attention



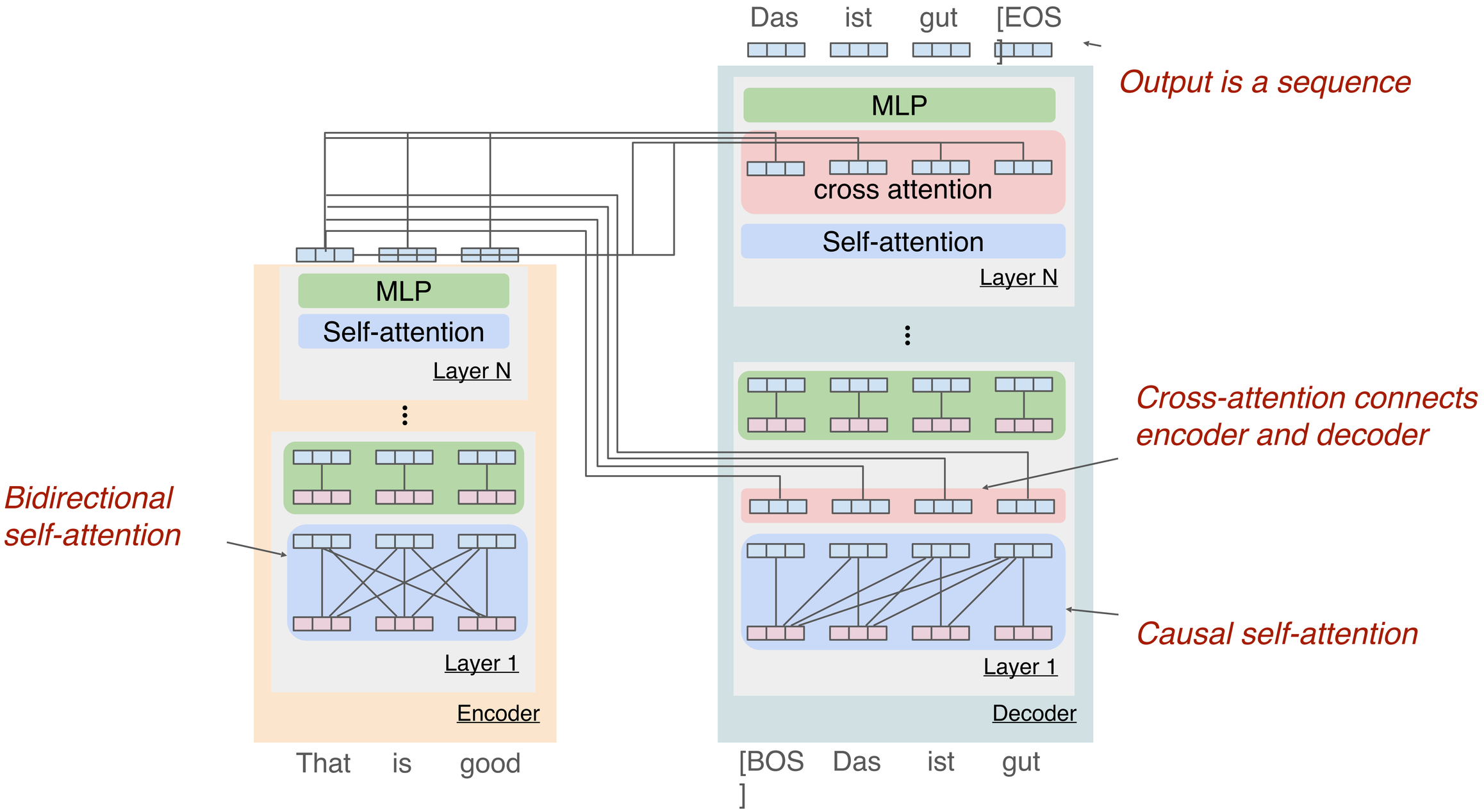
Encoder-decoder



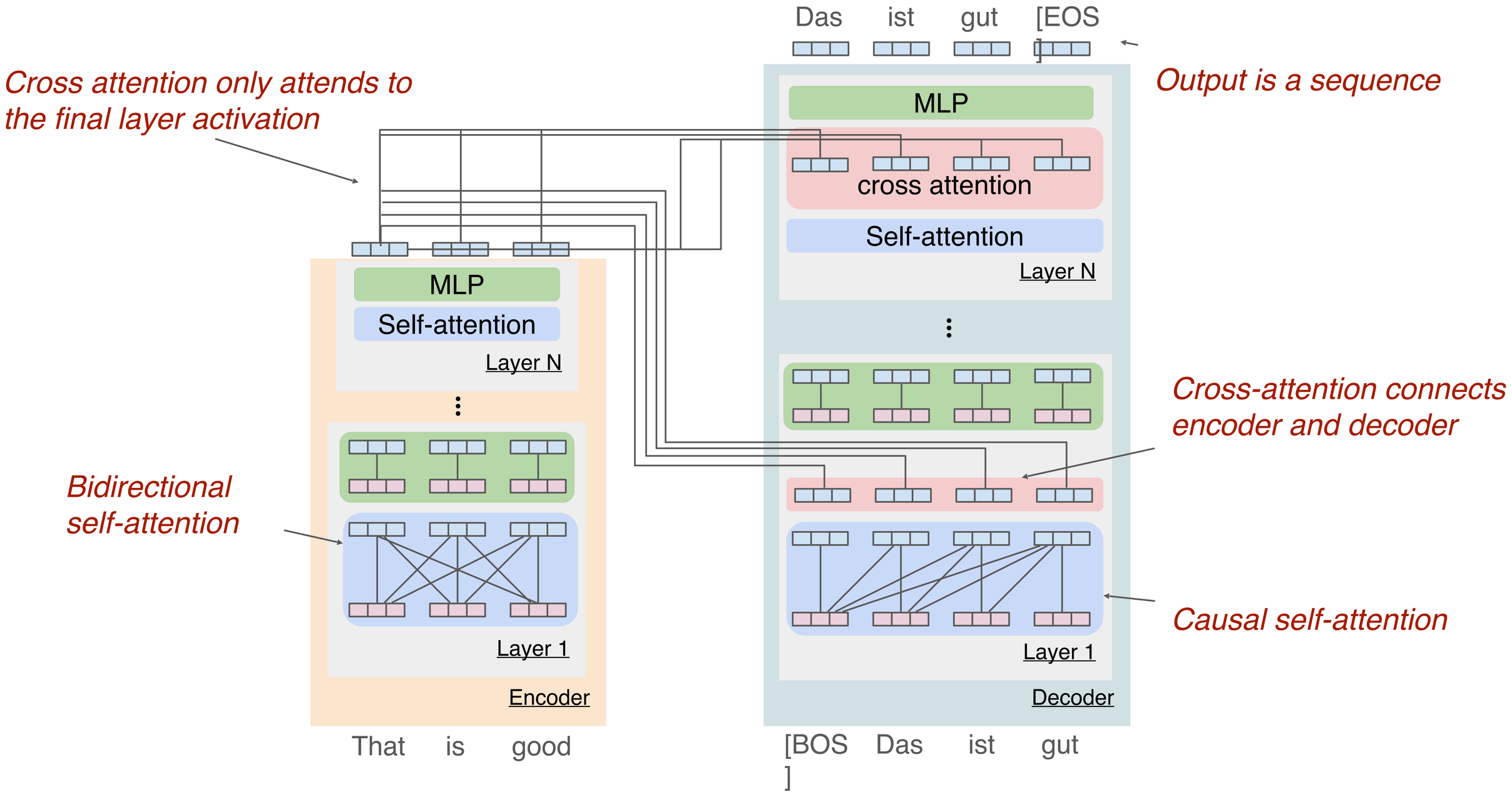
Encoder-decoder



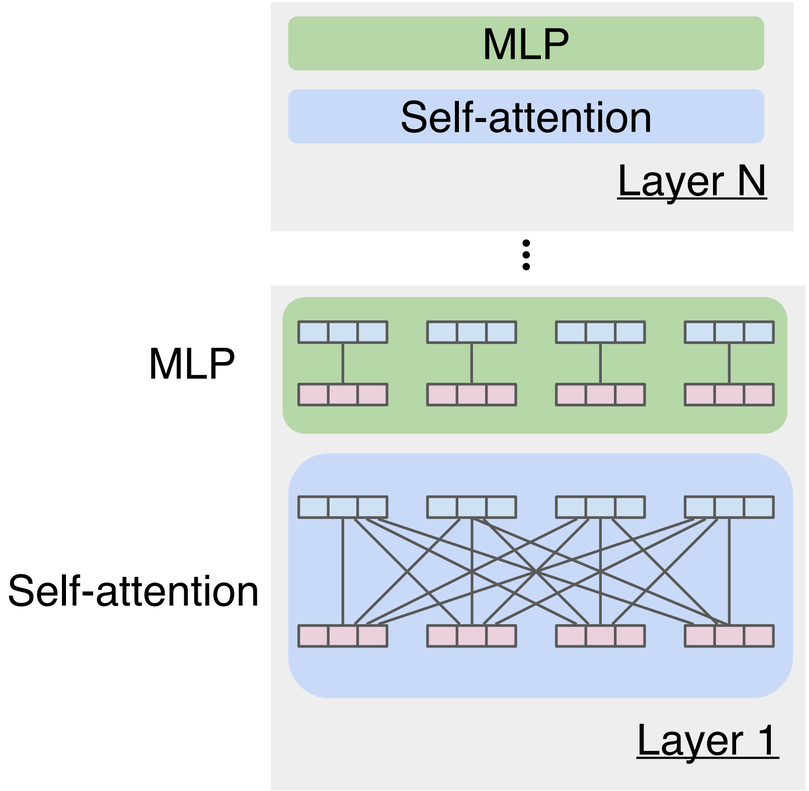
Encoder-decoder



Encoder-decoder



Encoder-only

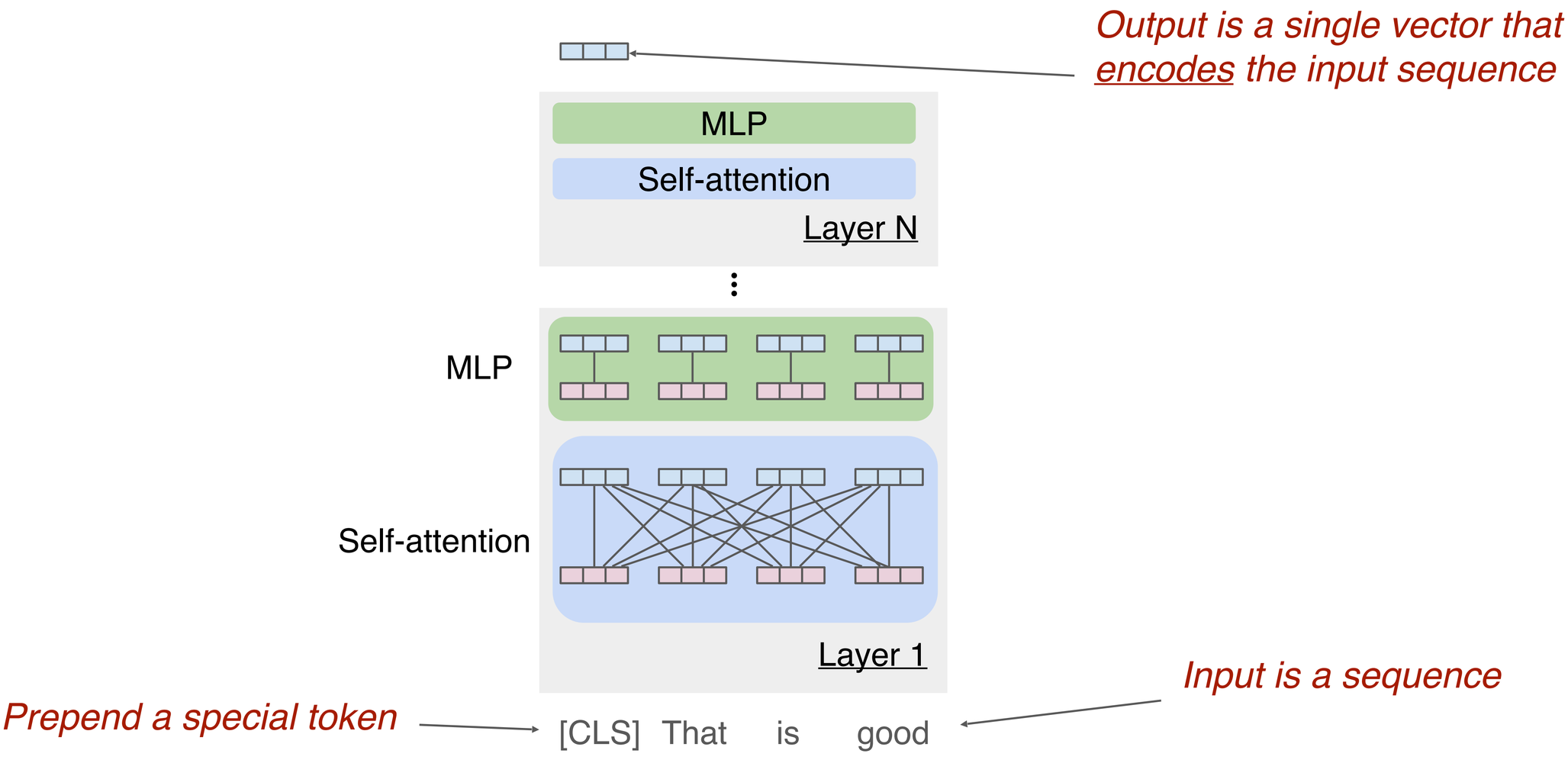


Prepend a special token

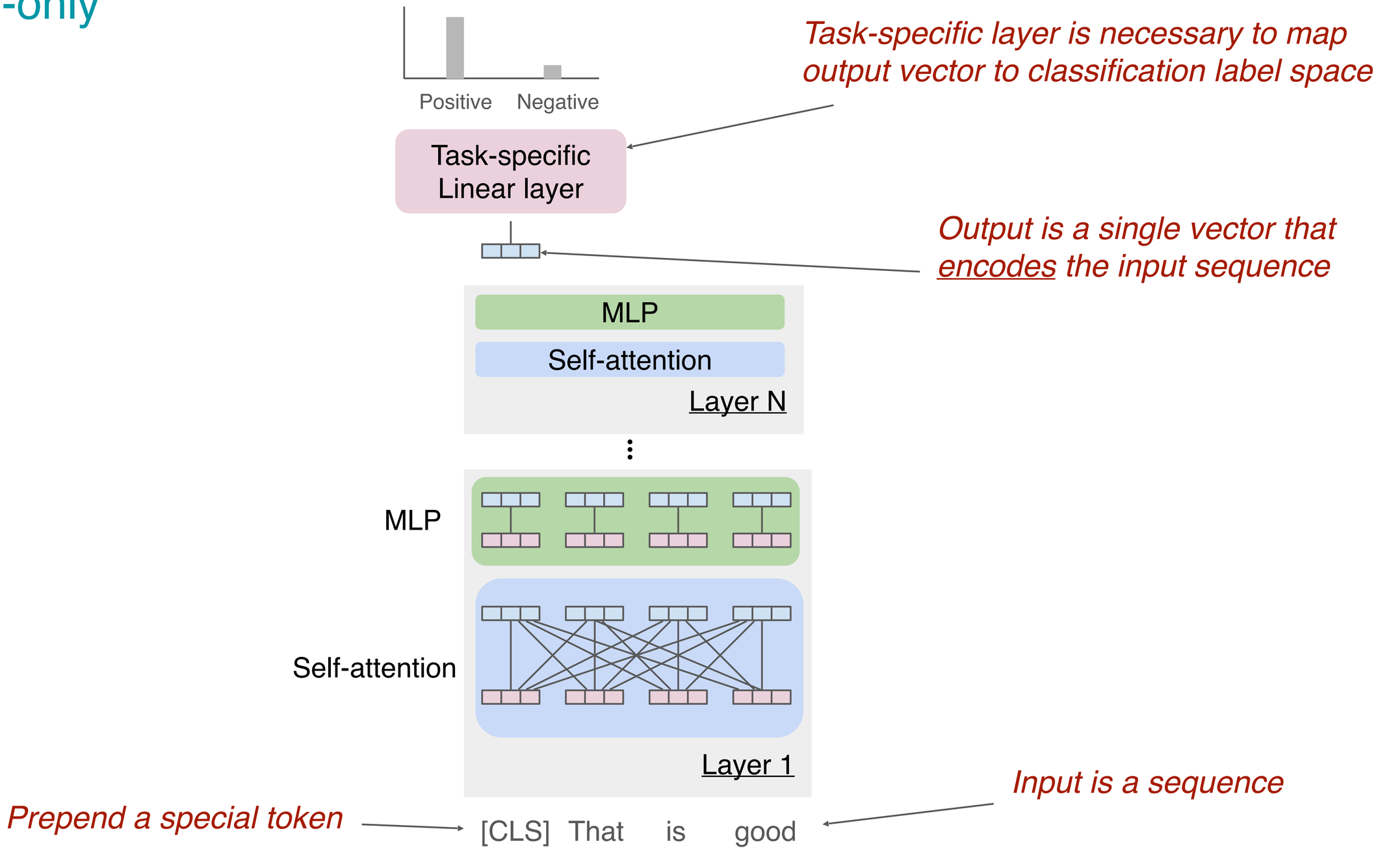
[CLS] That is good

Input is a sequence

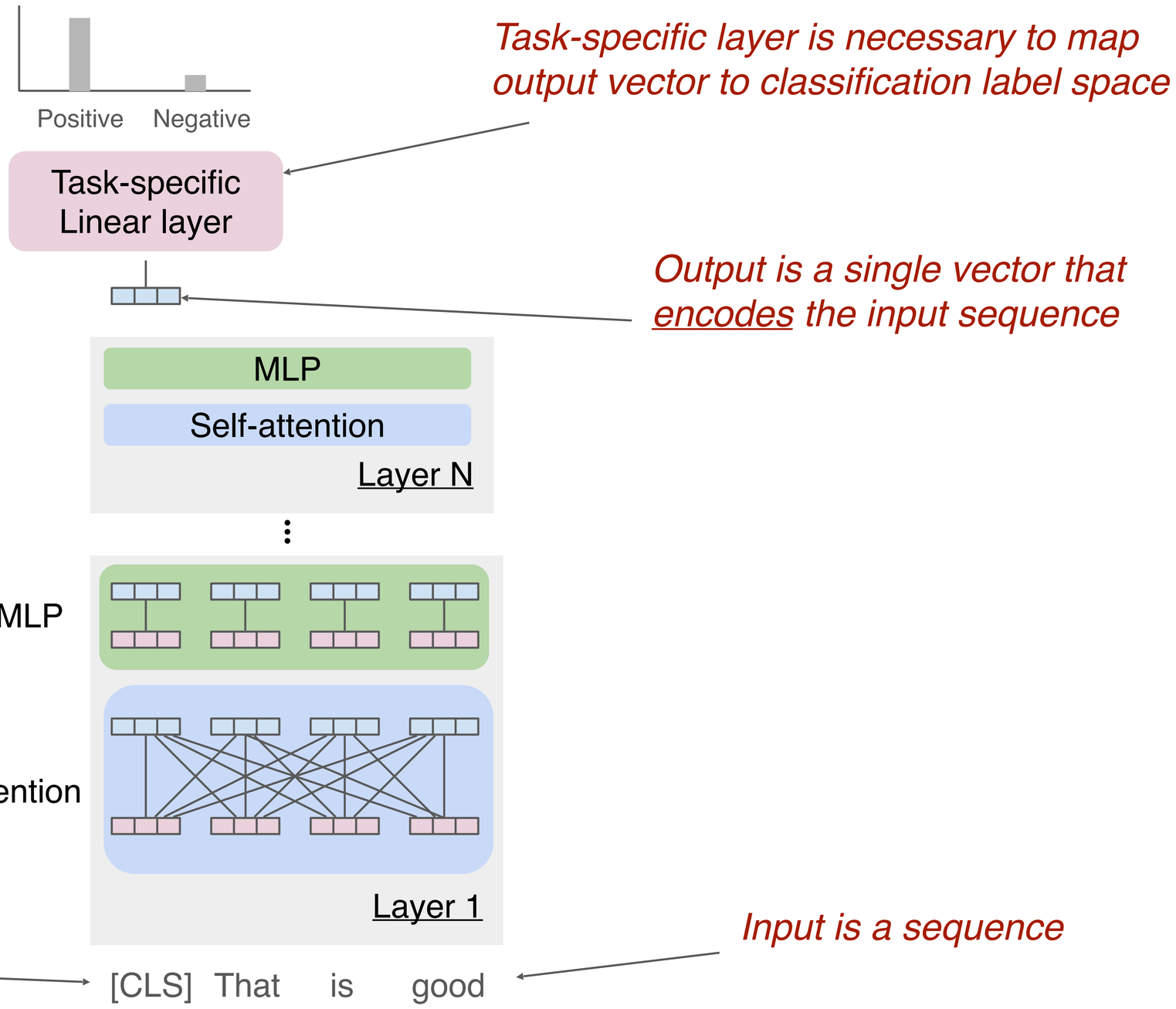
Encoder-only



Encoder-only



Encoder-only

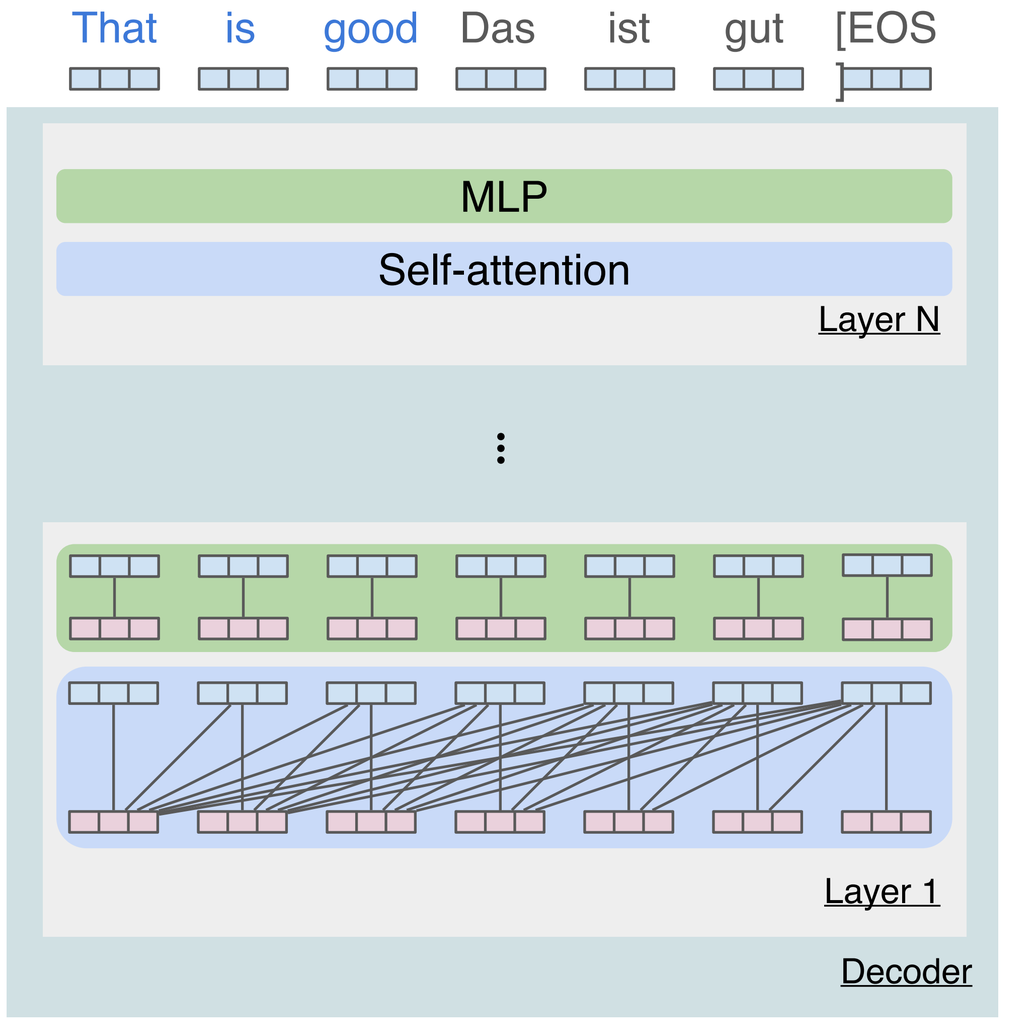


Can't generate a sequence!!

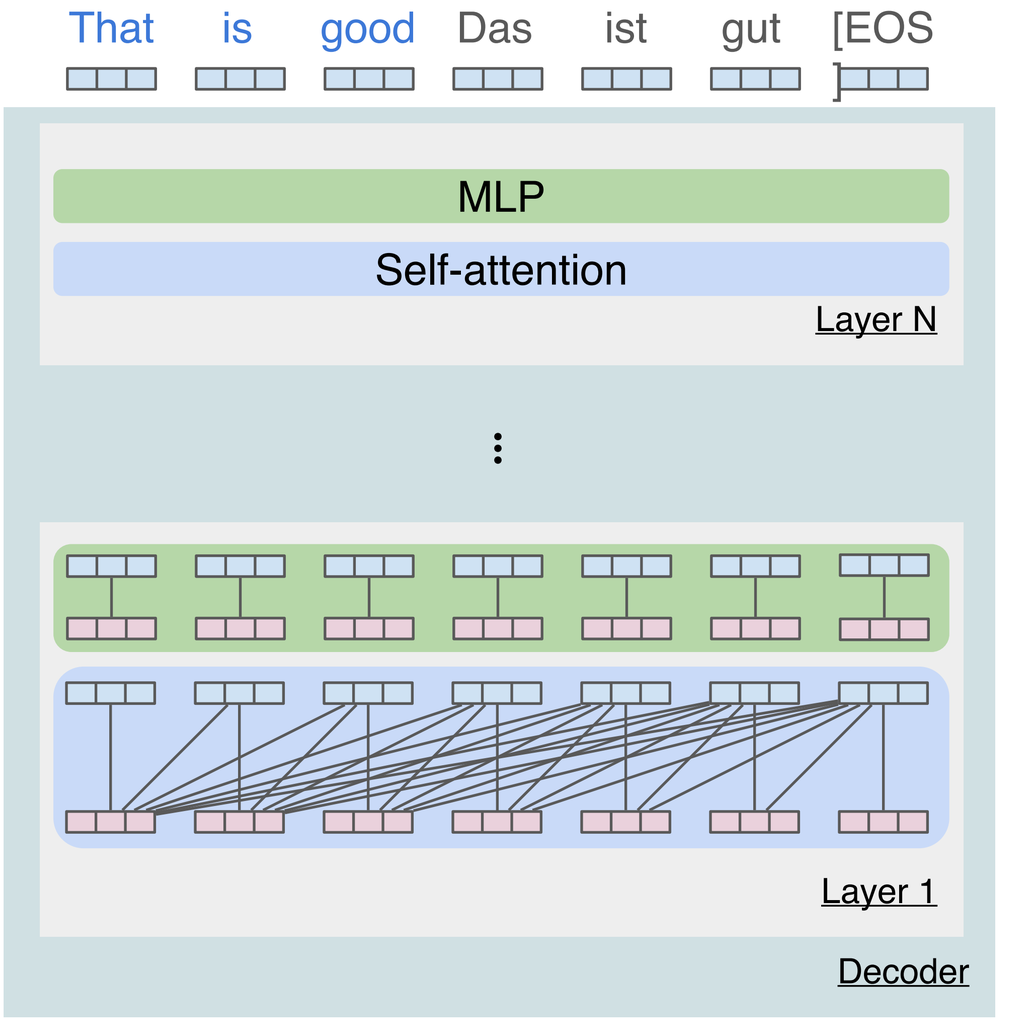
Deal breaker for general use case

(General use case here is text2text anything)

Decoder-only



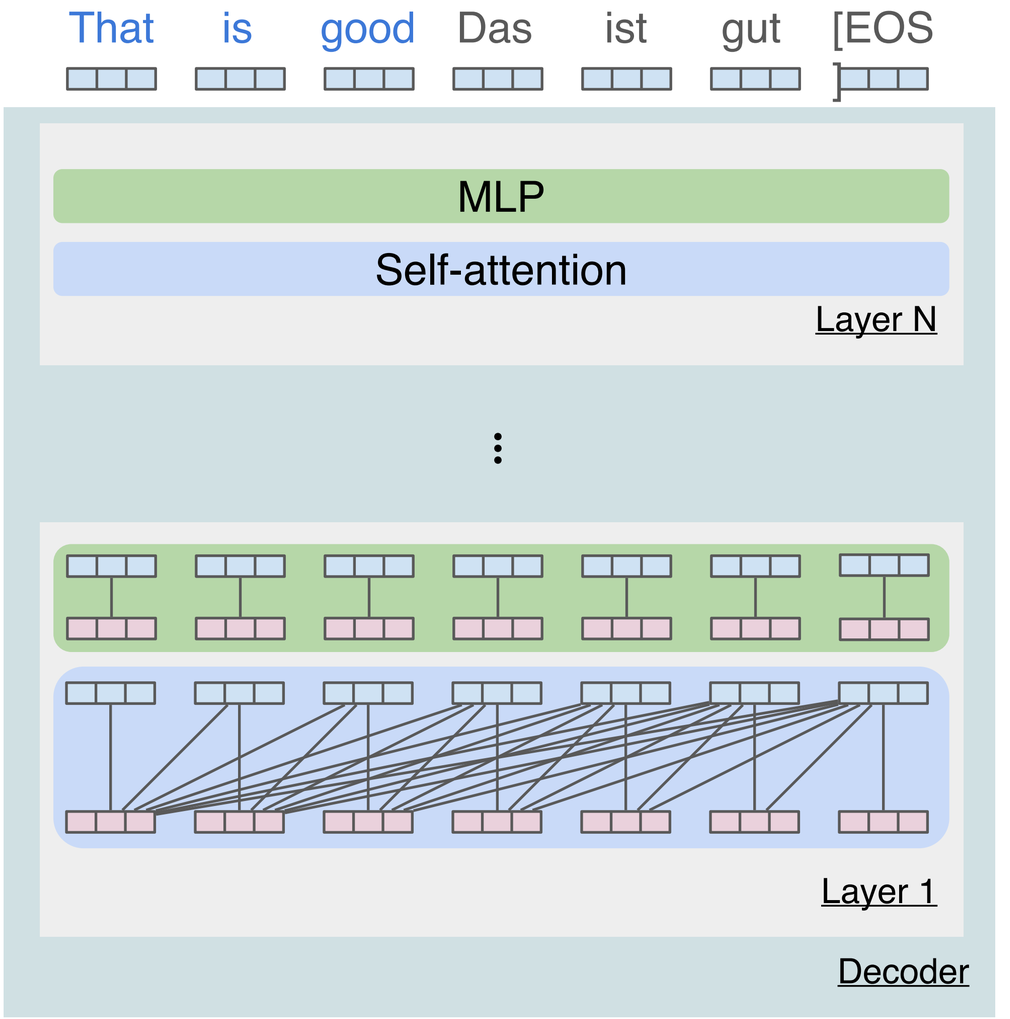
Decoder-only



Input and target are concatenated

[BOS That is good Das ist gut
]

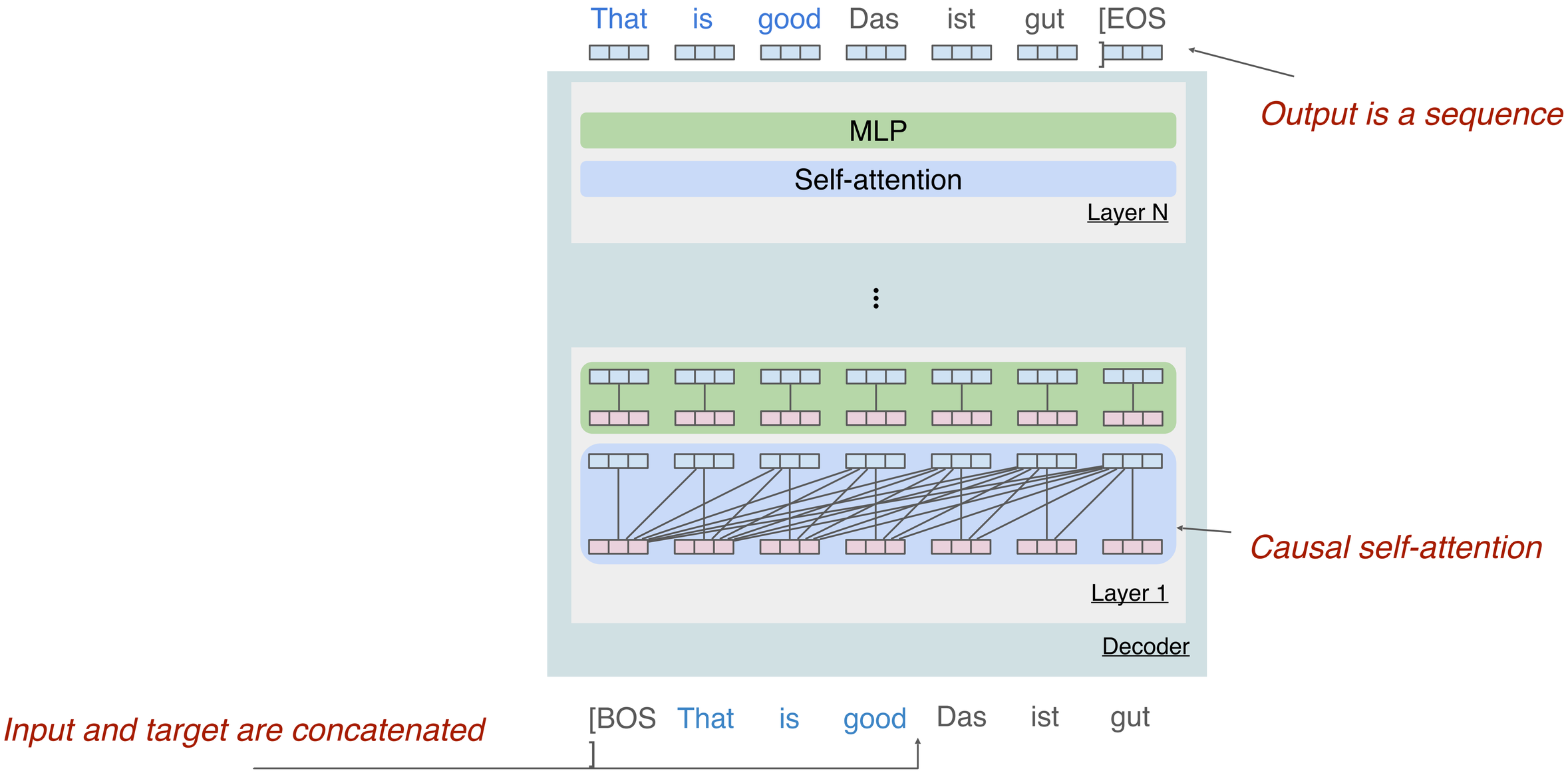
Decoder-only



Input and target are concatenated

[BOS That is good Das ist gut
]

Decoder-only

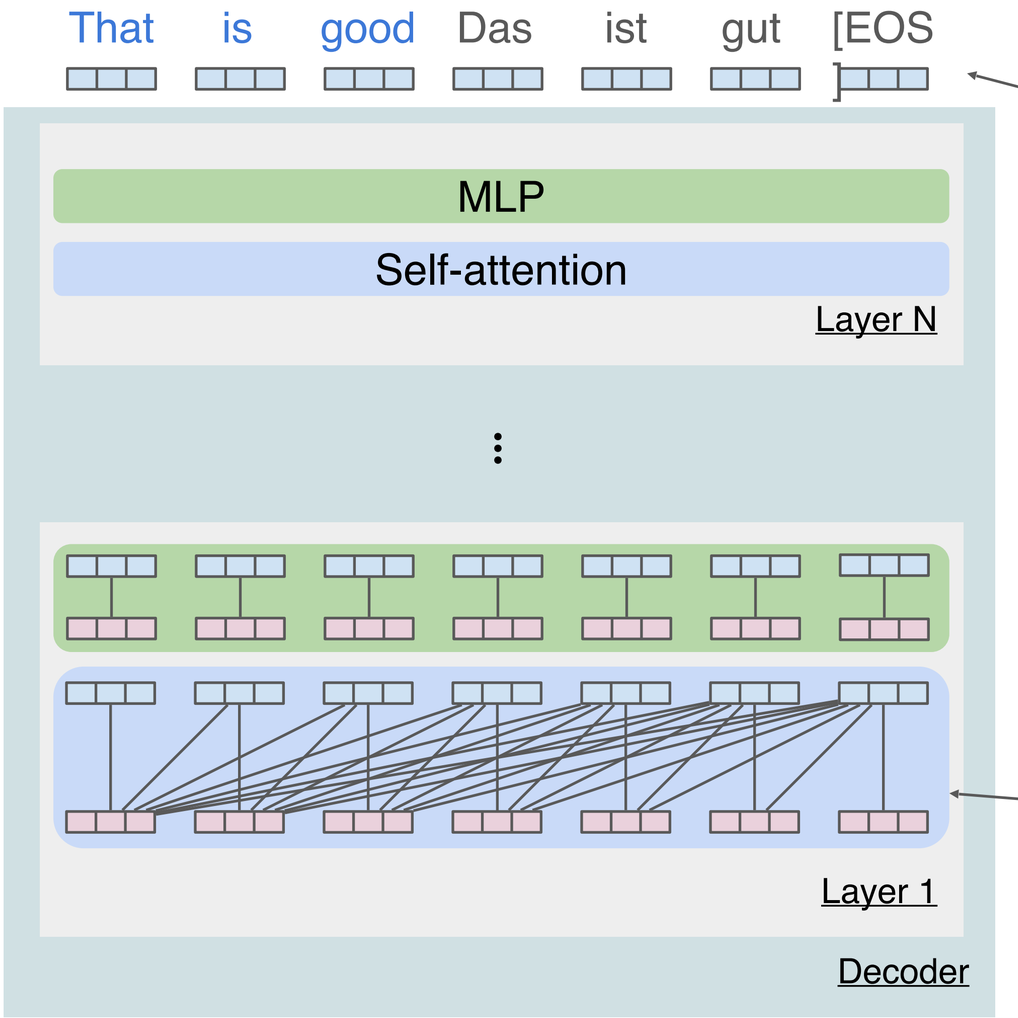


Decoder-only

Key design features

Self attention also serves the role of cross-attention

Same set of parameters apply to both input and target sequences



Output is a sequence

Causal self-attention

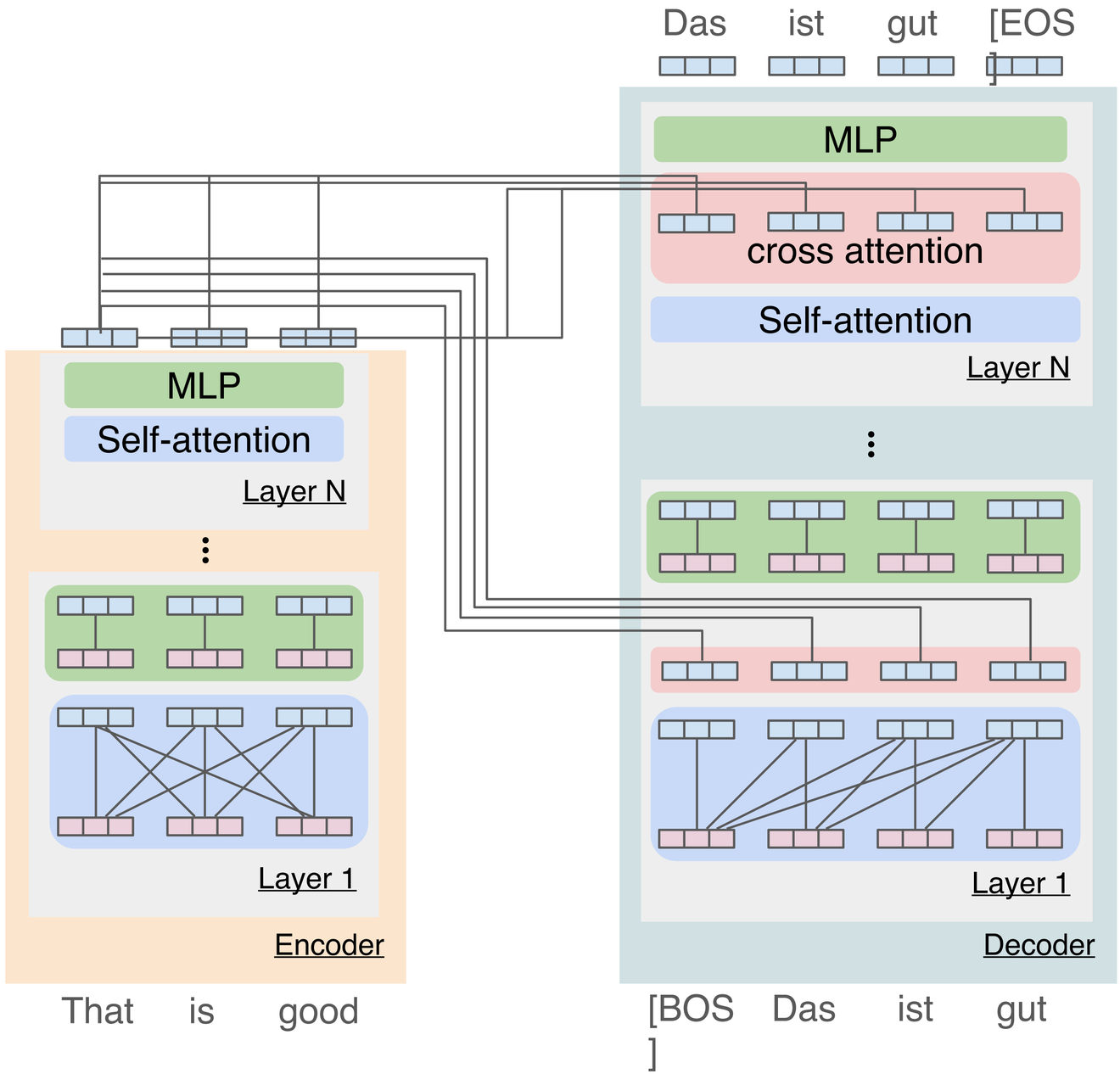
Input and target are concatenated

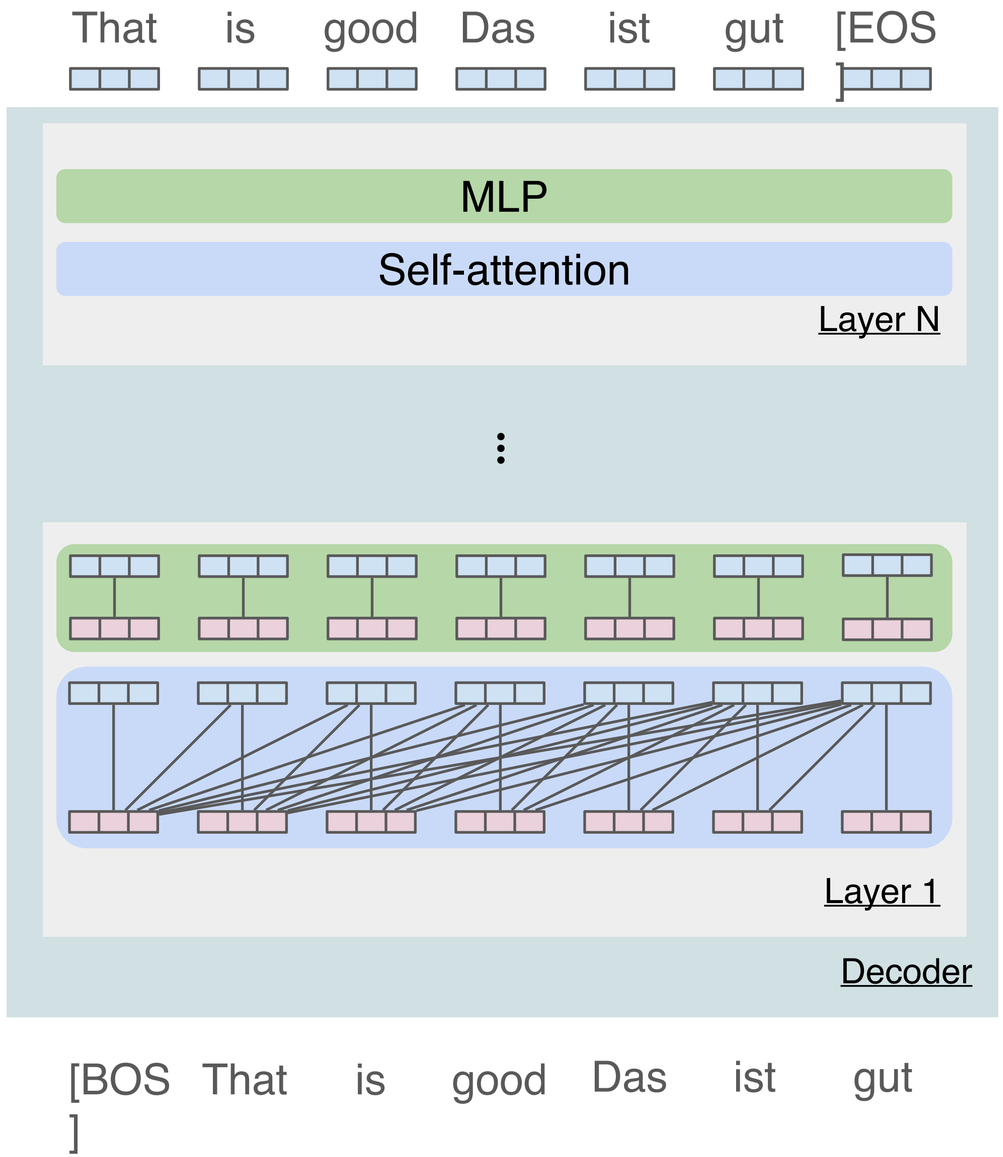
How different are encoder-decoder and decoder-only architectures?

Let’s try to transform encoder-decoder into decoder-only

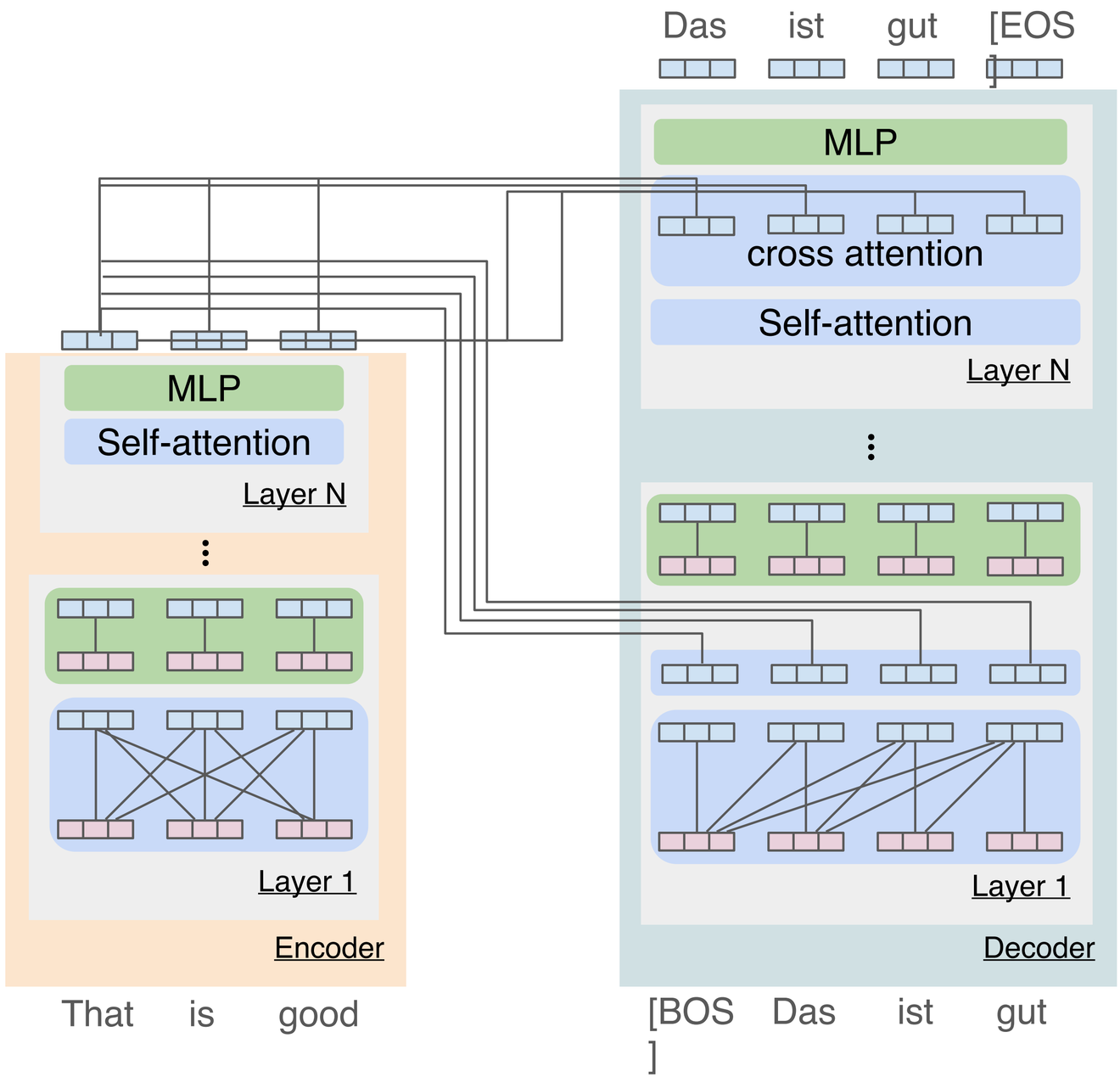
Summary of the differences

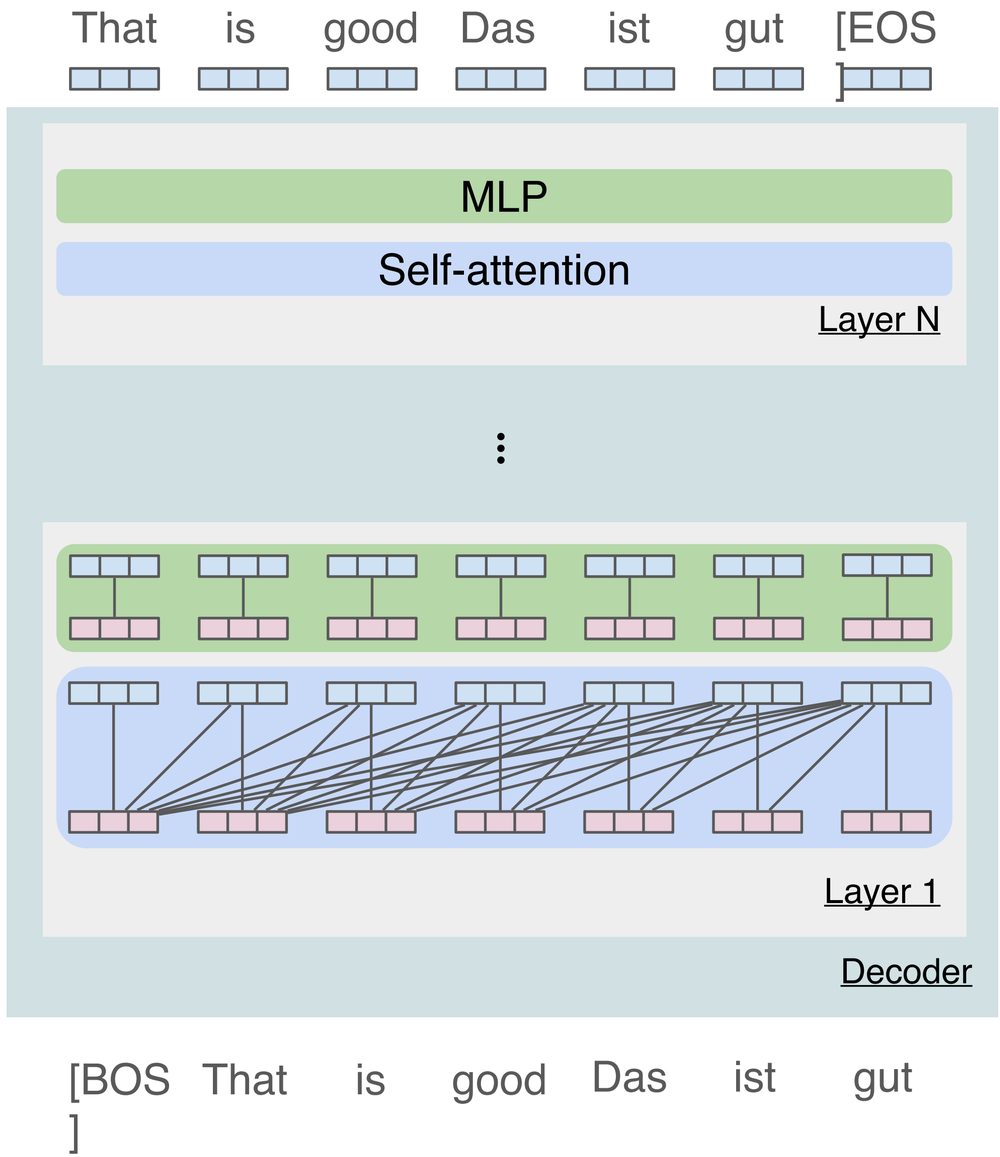
	Encoder-decoder	**Decoder-only**
Additional cross attention		
Parameter sharing		
Target-to-input attention pattern		
Input attention		


Das
ist
gut
[EOS
]
MLP
cross attention
Self-attention
Layer N
MLP
Self-attention
Layer N
Layer 1
Layer 1
Encoder
Decoder
That
is
good
[BOS
]
Das
ist
gut


That
is
good
Das
ist
gut
[EOS
]
MLP
Self-attention
Layer N
Layer 1
Decoder
[BOS
]
That
is
good
Das
ist
gut

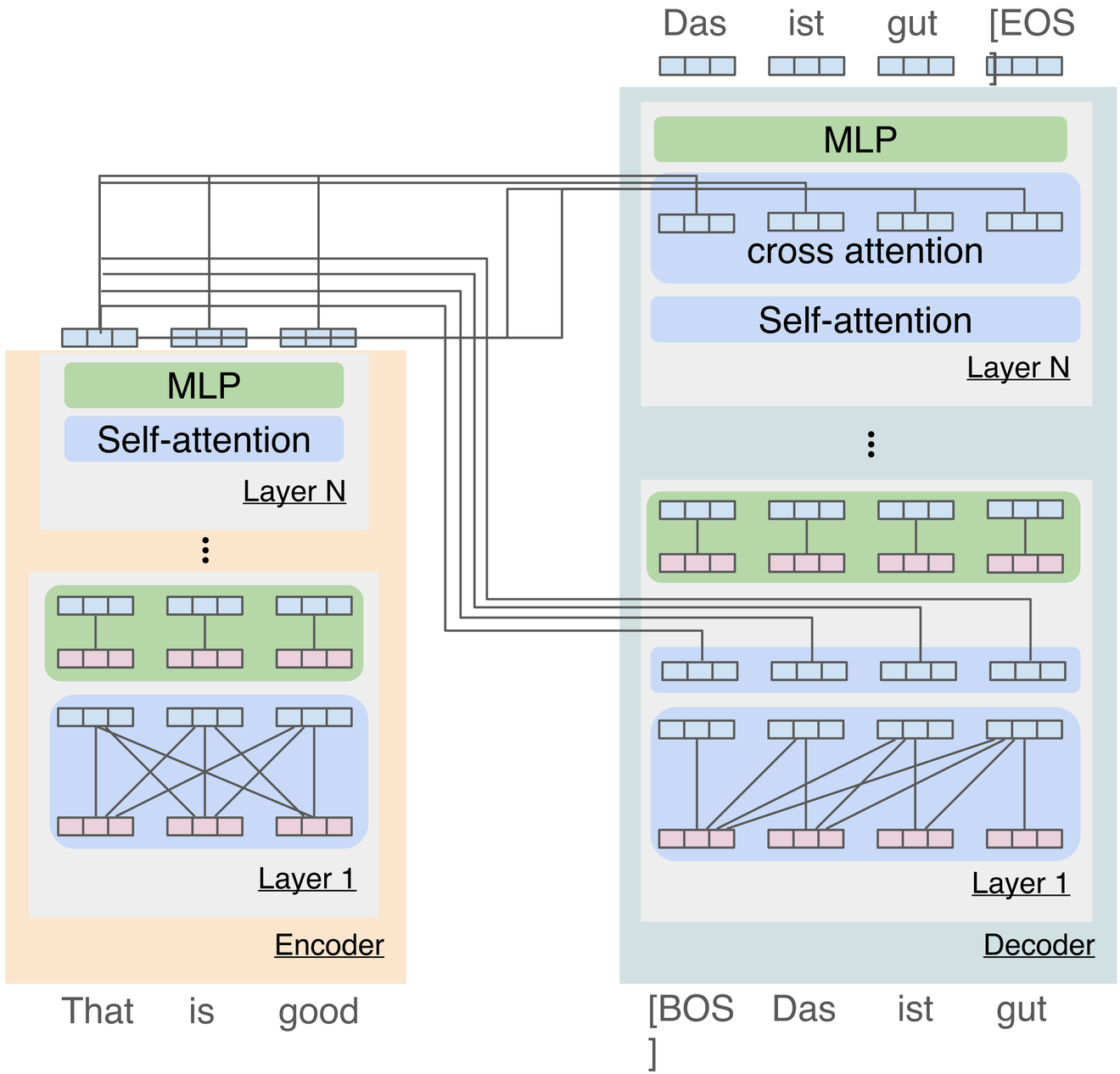
1. Share cross and self-attention parameters

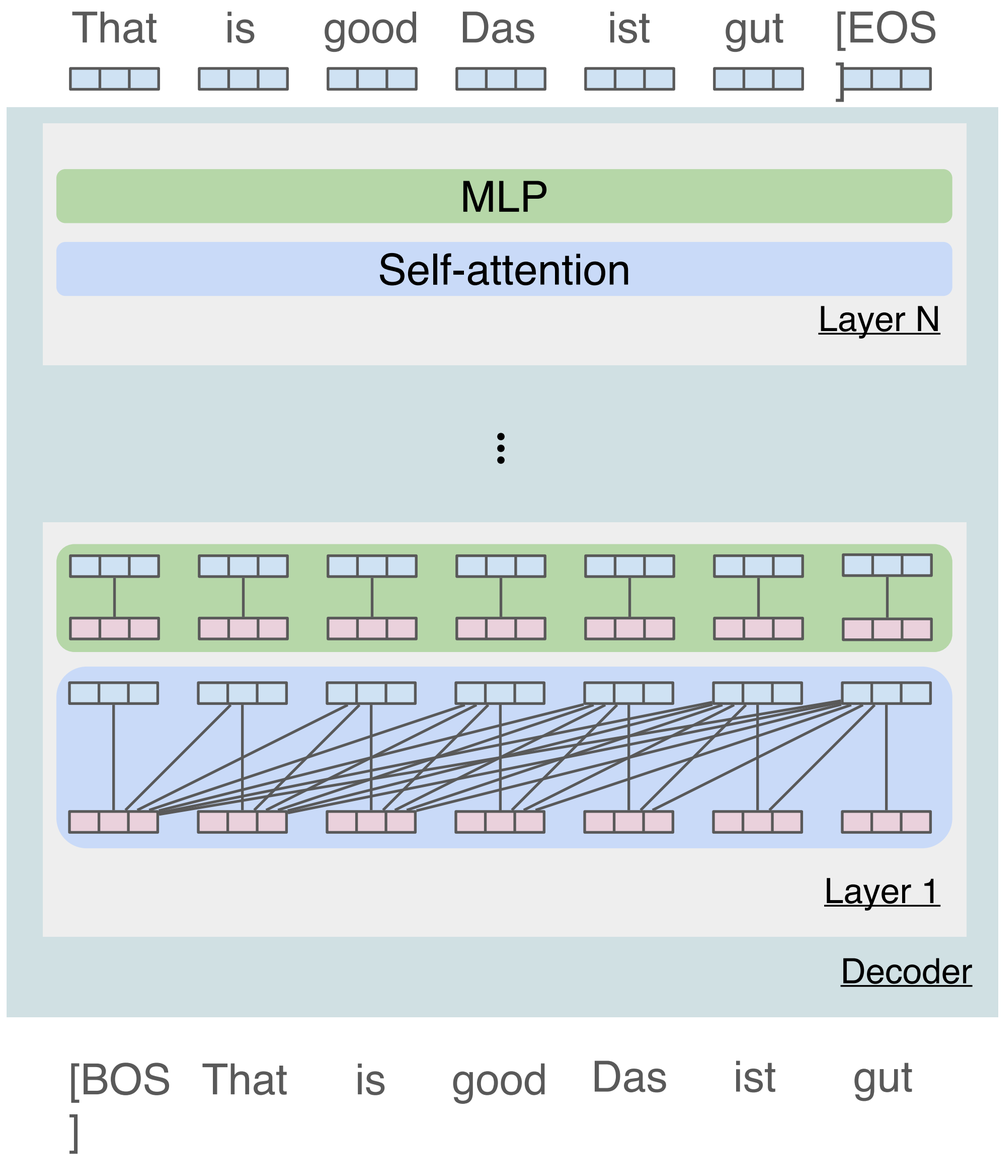




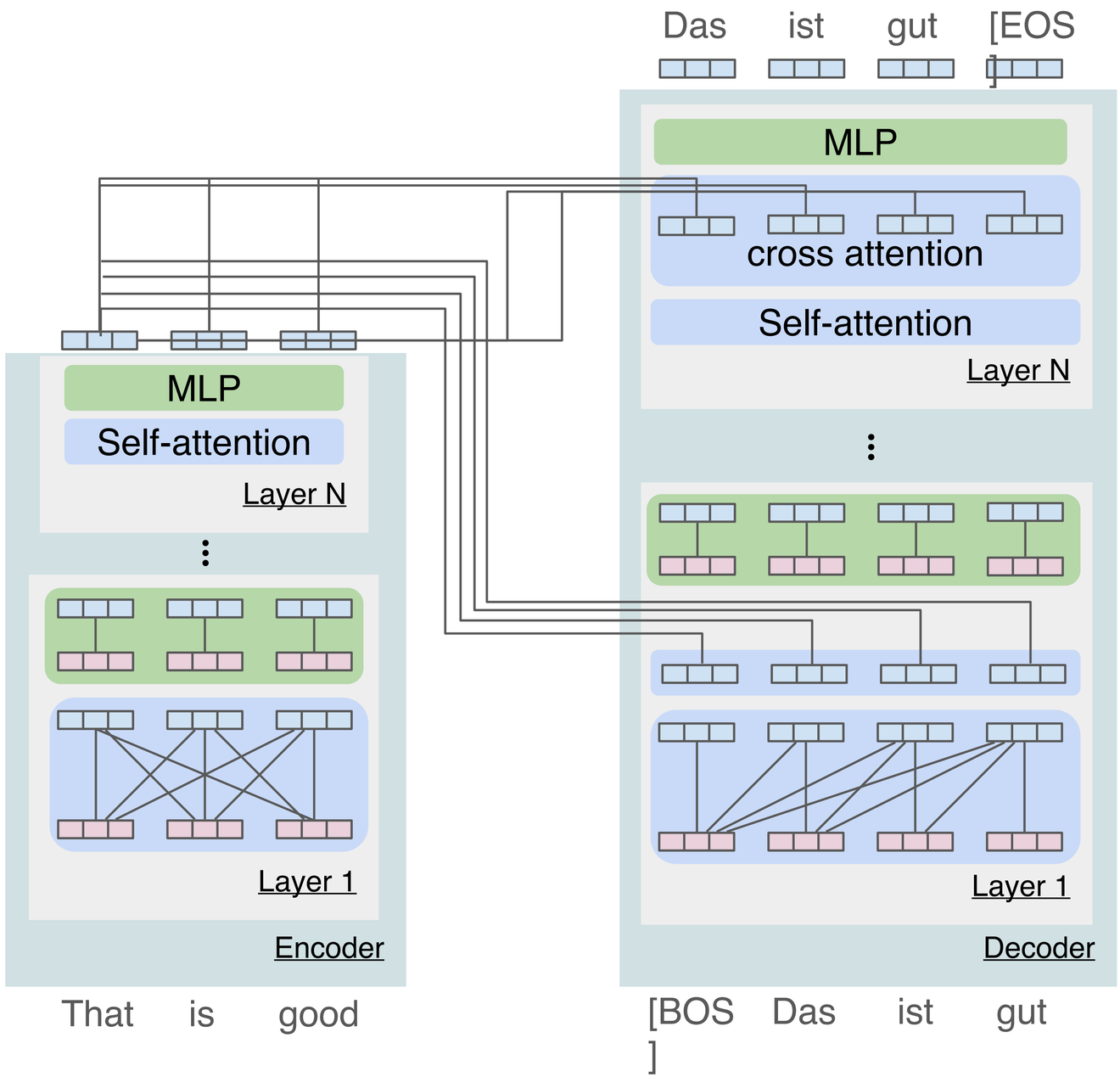
Summary of the differences

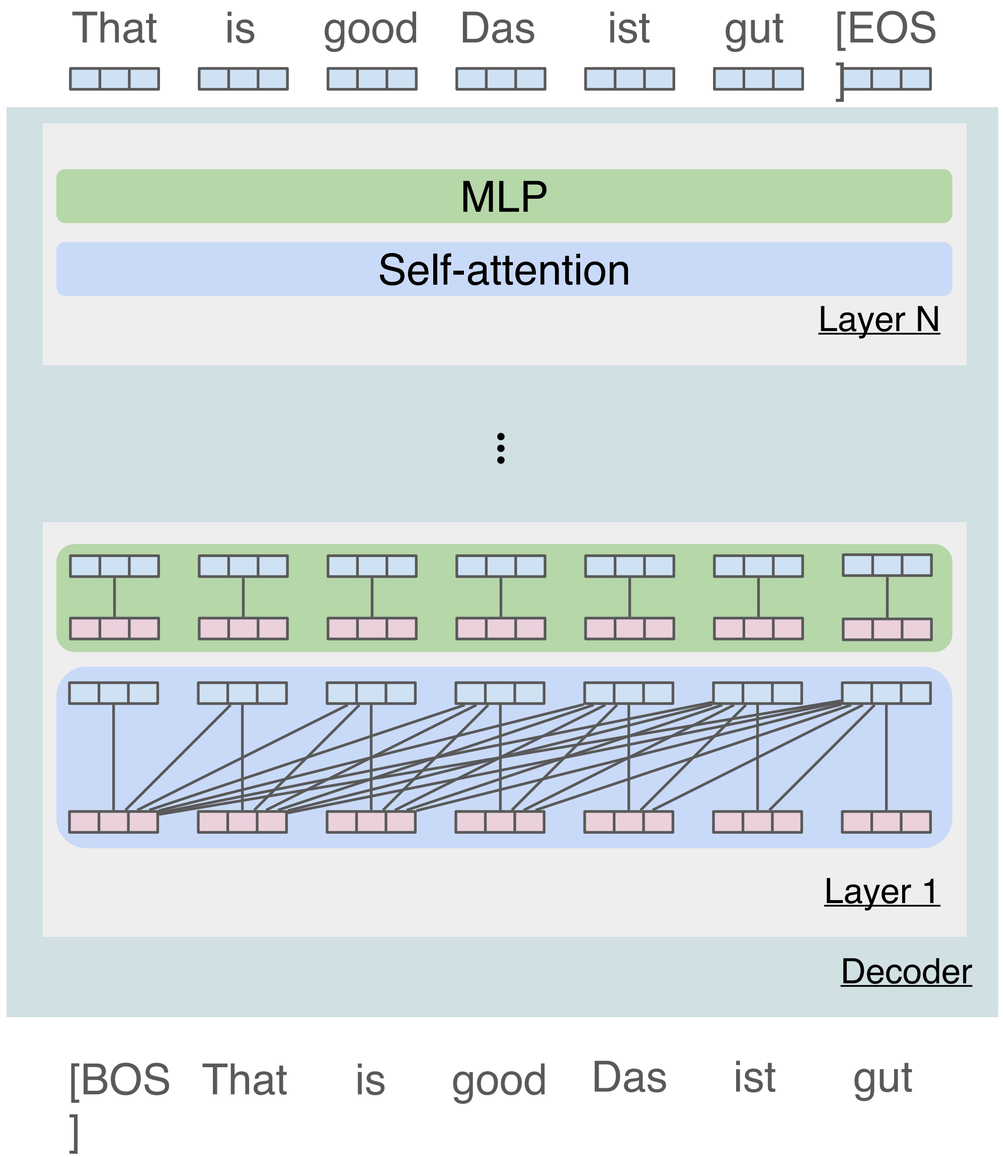
	Encoder-decoder	**Decoder-only**
Additional cross attention	Separate cross attention	Self-attention serving both roles
Parameter sharing		
Target-to-input attention pattern		
Input attention		


Das
ist
gut
[EOS
]
MLP
cross attention
Self-attention
Layer N
MLP
Self-attention
Layer N
Layer 1
Layer 1
Encoder
Decoder
That
is
good
[BOS
]
Das
ist
gut


That
is
good
Das
ist
gut
[EOS
]
MLP
Self-attention
Layer N
Layer 1
Decoder
[BOS
]
That
is
good
Das
ist
gut

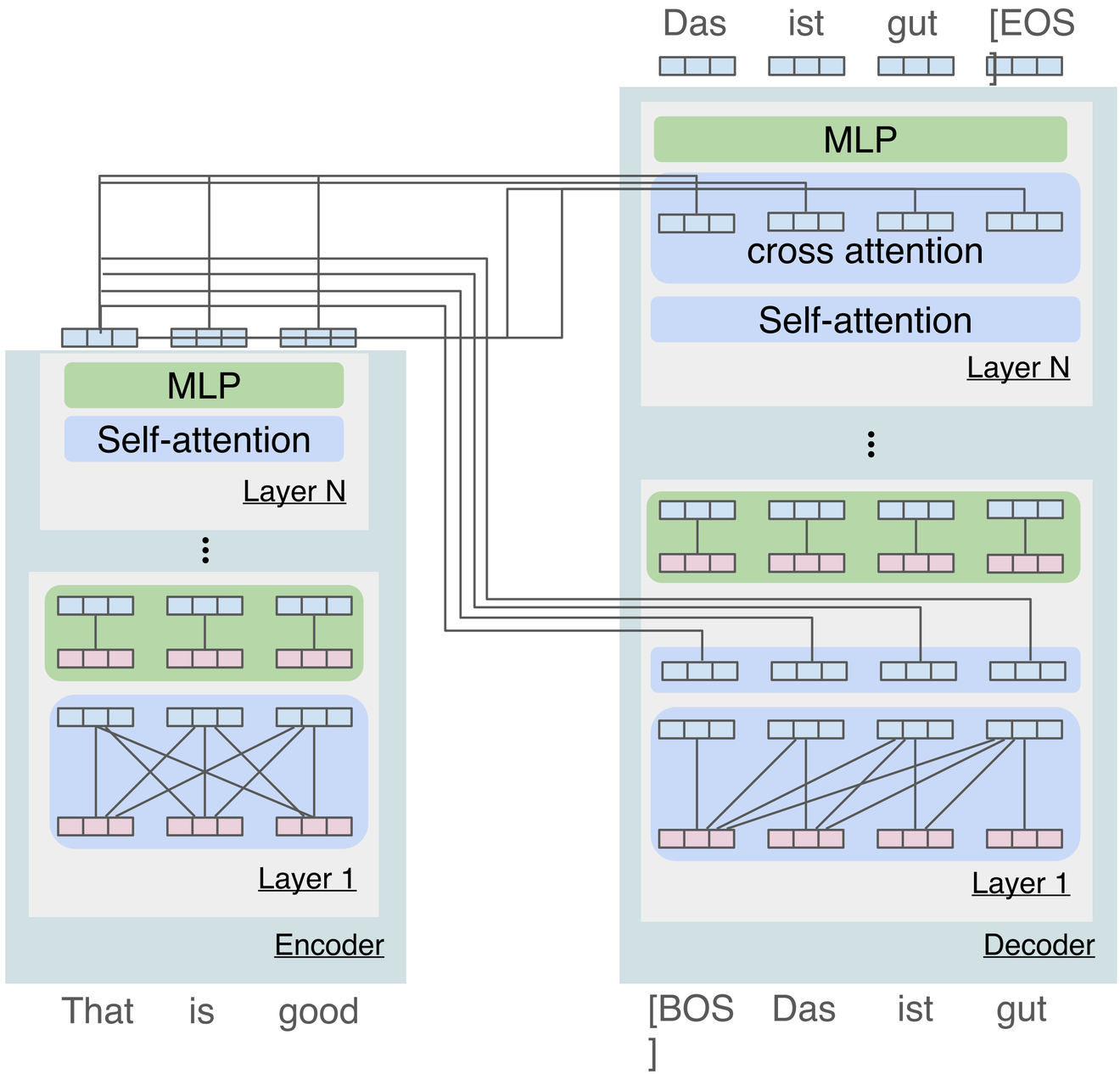
2. Share encoder and decoder parameters

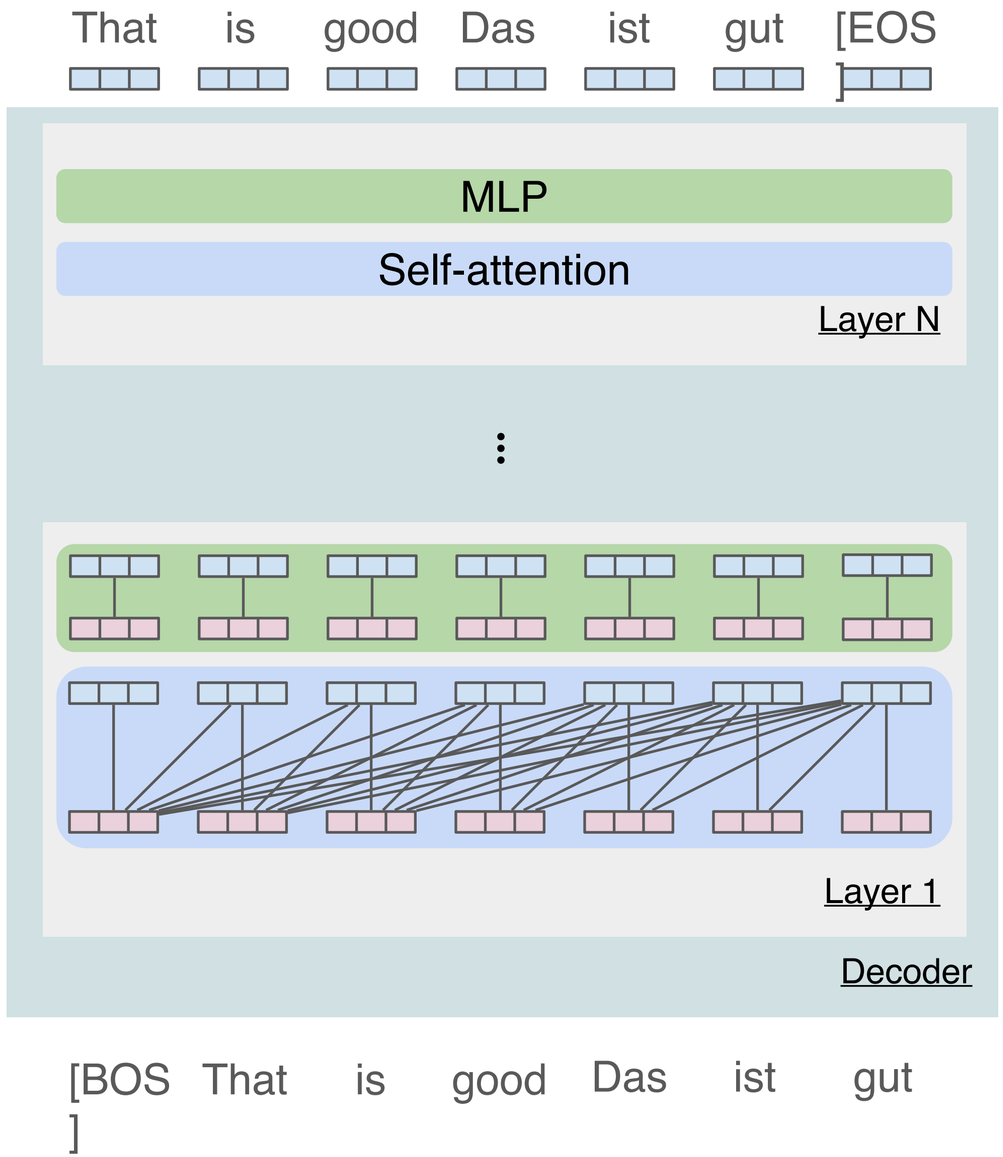




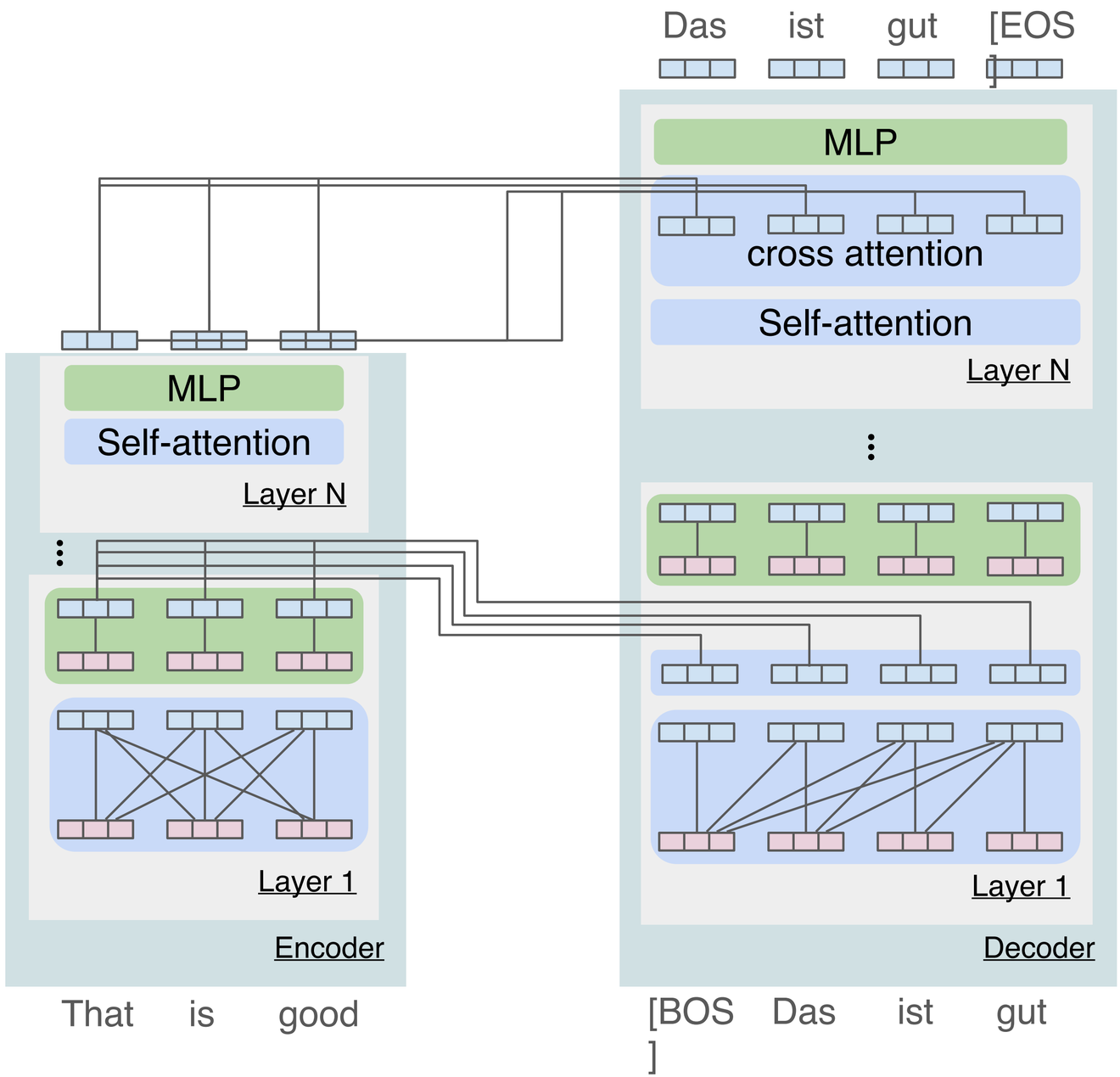
Summary of the differences

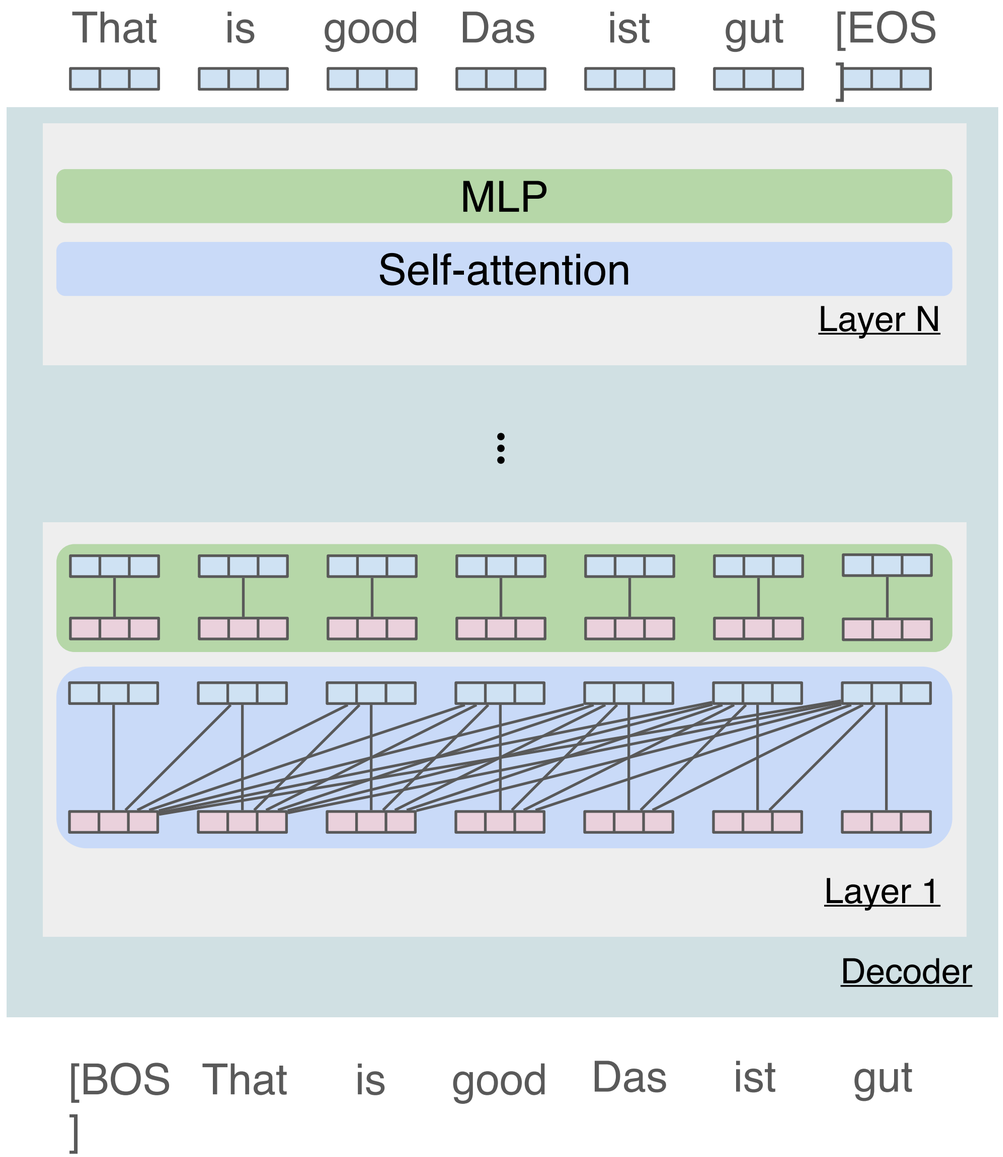
	Encoder-decoder	**Decoder-only**
Additional cross attention	Separate cross attention	Self-attention serving both roles
Parameter sharing	Separate parameters for input and target	Shared parameters
Target-to-input attention pattern		
Input attention		


Das
ist
gut
[EOS
]
MLP
cross attention
Self-attention
Layer N
MLP
Self-attention
Layer N
Layer 1
Layer 1
Encoder
Decoder
That
is
good
[BOS
]
Das
ist
gut


That
is
good
Das
ist
gut
[EOS
]
MLP
Self-attention
Layer N
Layer 1
Decoder
[BOS
]
That
is
good
Das
ist
gut

3. Decoder layer l attends to encoder layer l

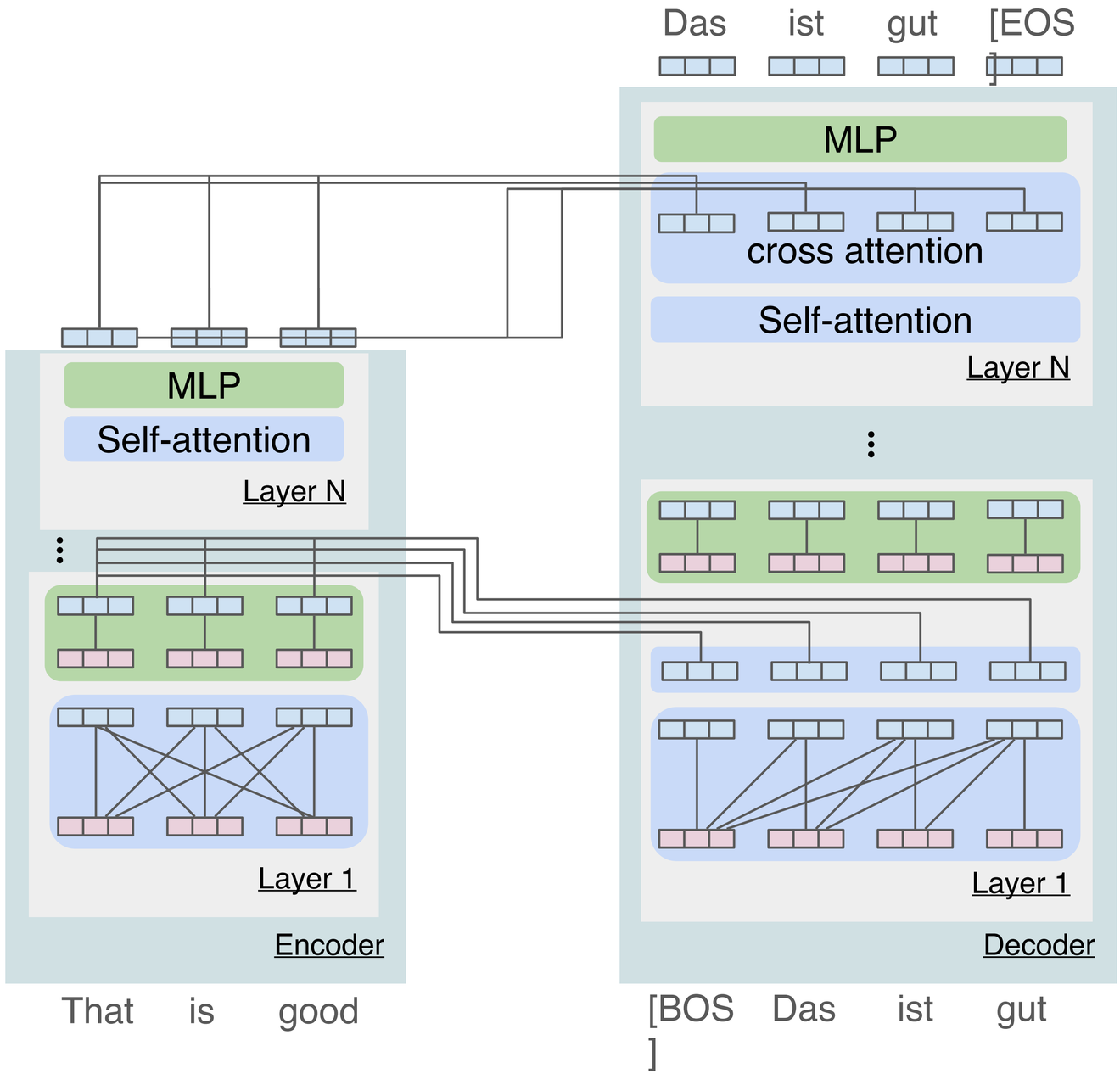


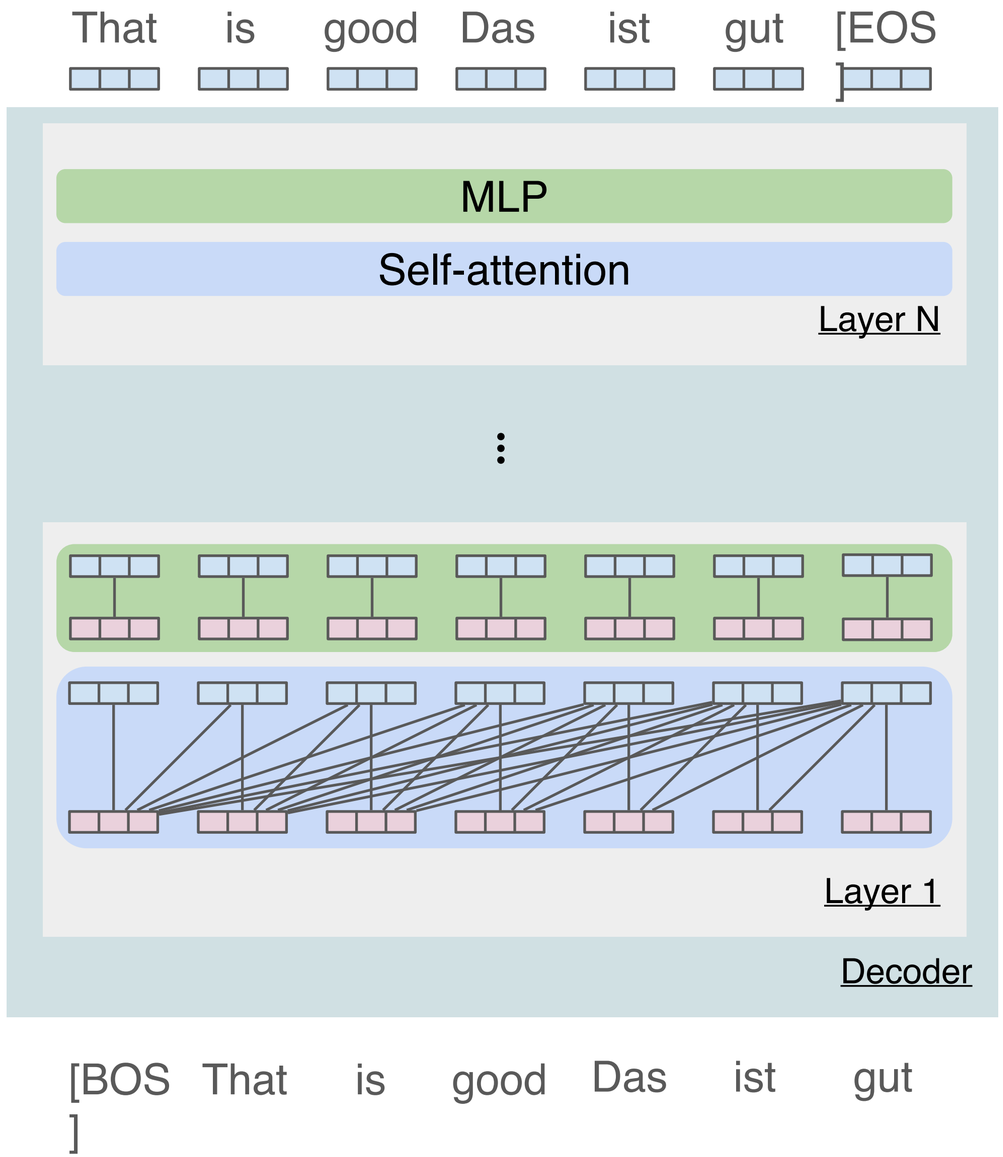


Summary of the differences

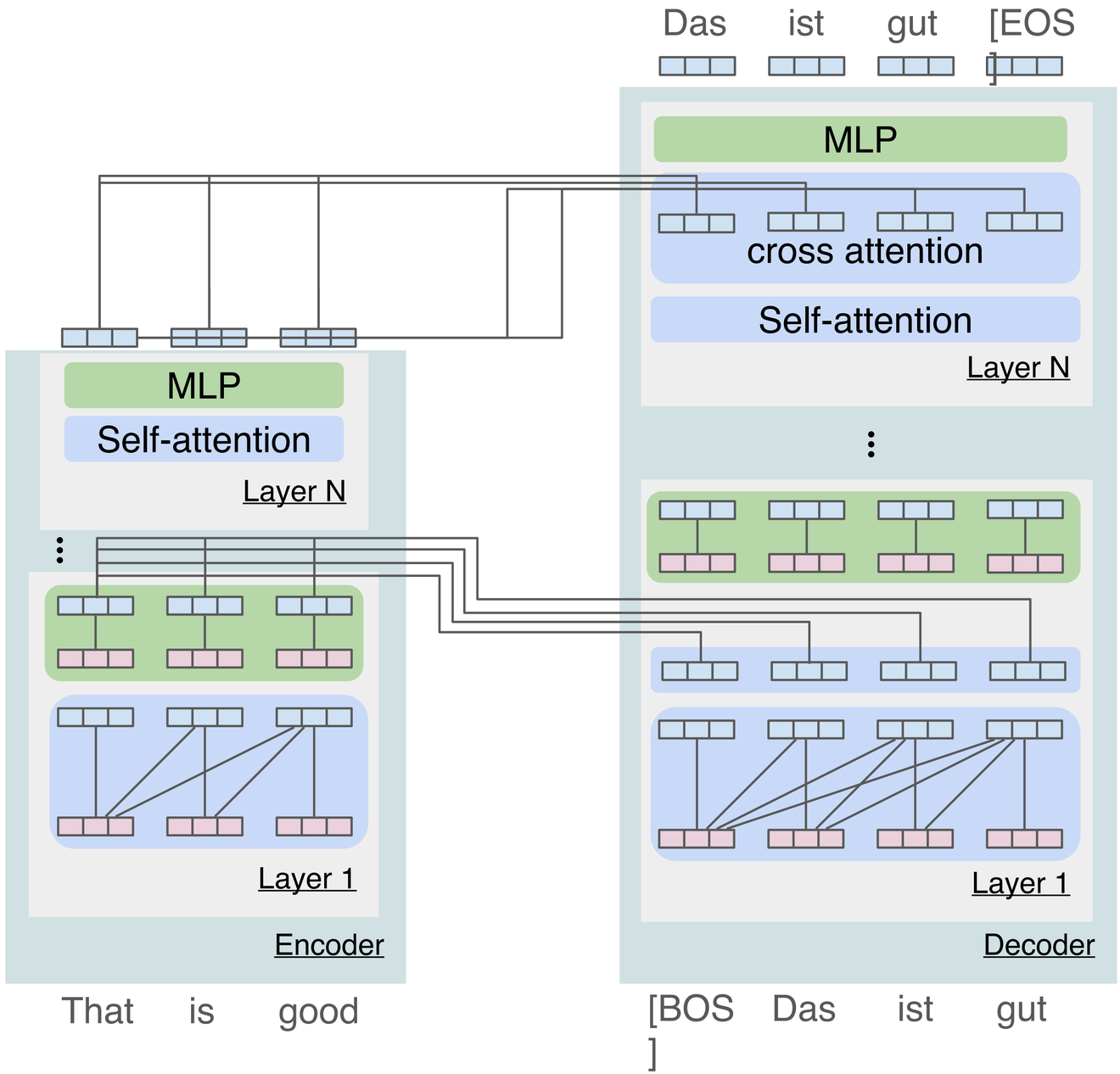
	Encoder-decoder	**Decoder-only**
Additional cross attention	Separate cross attention	Self-attention serving both roles
Parameter sharing	Separate parameters for input and target	Shared parameters
Target-to-input attention pattern	Only attends to the last layer of encoder's output	Within-layer (i.e. layer l attends to layer l)
Input attention		

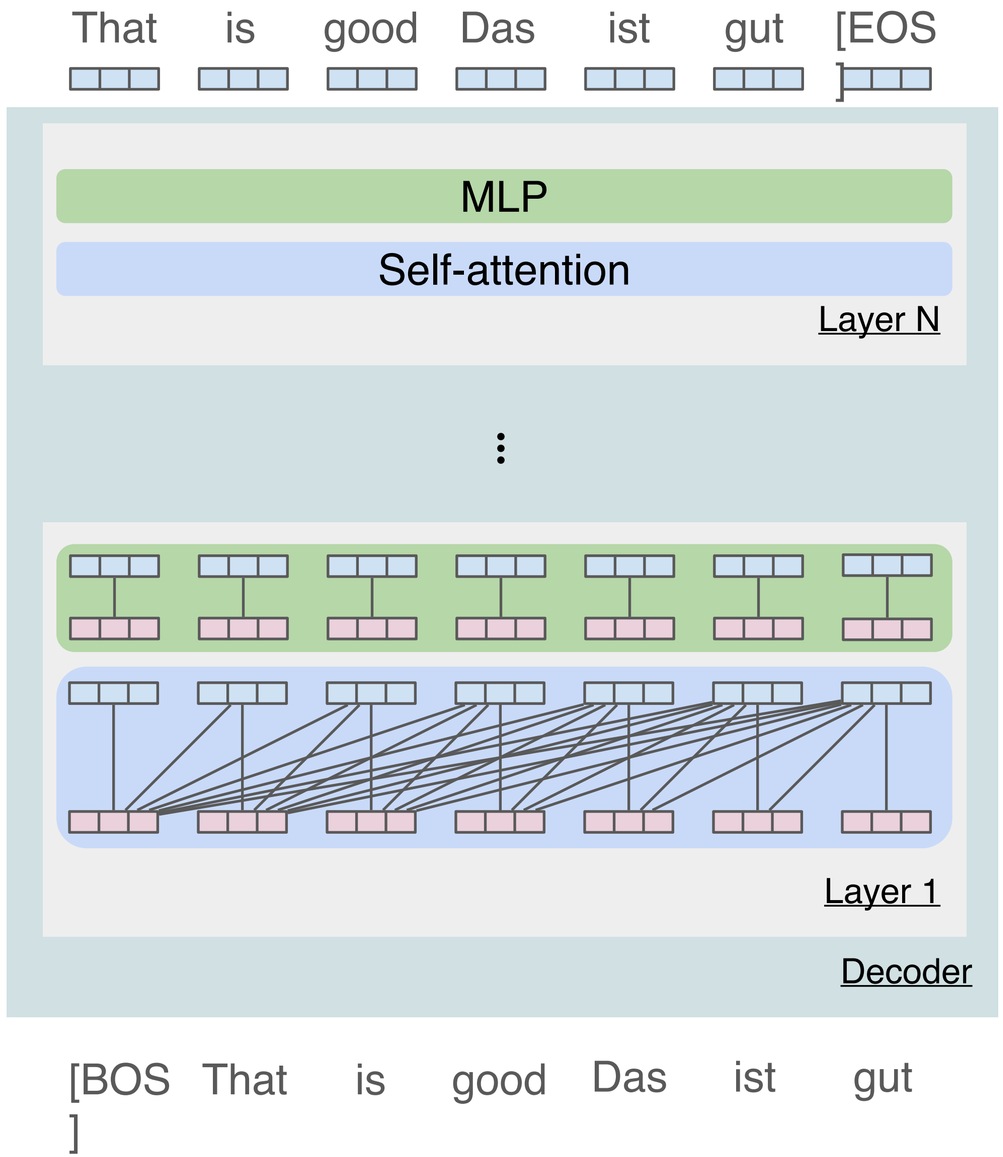
* input attention can be bidirectional


Das
ist
gut
[EOS
]
MLP
cross attention
Self-attention
Layer N
MLP
Self-attention
Layer N
Layer 1
Layer 1
Encoder
Decoder
That
is
good
[BOS
]
Das
ist
gut


That
is
good
Das
ist
gut
[EOS
]
MLP
Self-attention
Layer N
Layer 1
Decoder
[BOS
]
That
is
good
Das
ist
gut

4. Make encoder self-attention causal





Summary of the differences

	Encoder-decoder	**Decoder-only**
Additional cross attention	Separate cross attention	Self-attention serving both roles
Parameter sharing	Separate parameters for input and target	Shared parameters
Target-to-input attention pattern	Only attends to the last layer of encoder's output	Within-layer (i.e. layer l attends to layer l)
Input attention	Bidirectional	Unidirectional*

* input attention can be bidirectional

Additional structures in encoder-decoder compared to decoder-only

1. Input and target sequences are sufficiently different that using separate parameters can be effective

1. The target element can attend to the fully encoded representation of the input

1. When encoding the input sequence, all-to-all interaction among the sequence elements is preferred

Additional structures in encoder-decoder compared to decoder-only

1. Input and target sequences are sufficiently different that using separate parameters can be effective

1. The target element can attend to the fully encoded representation of the input

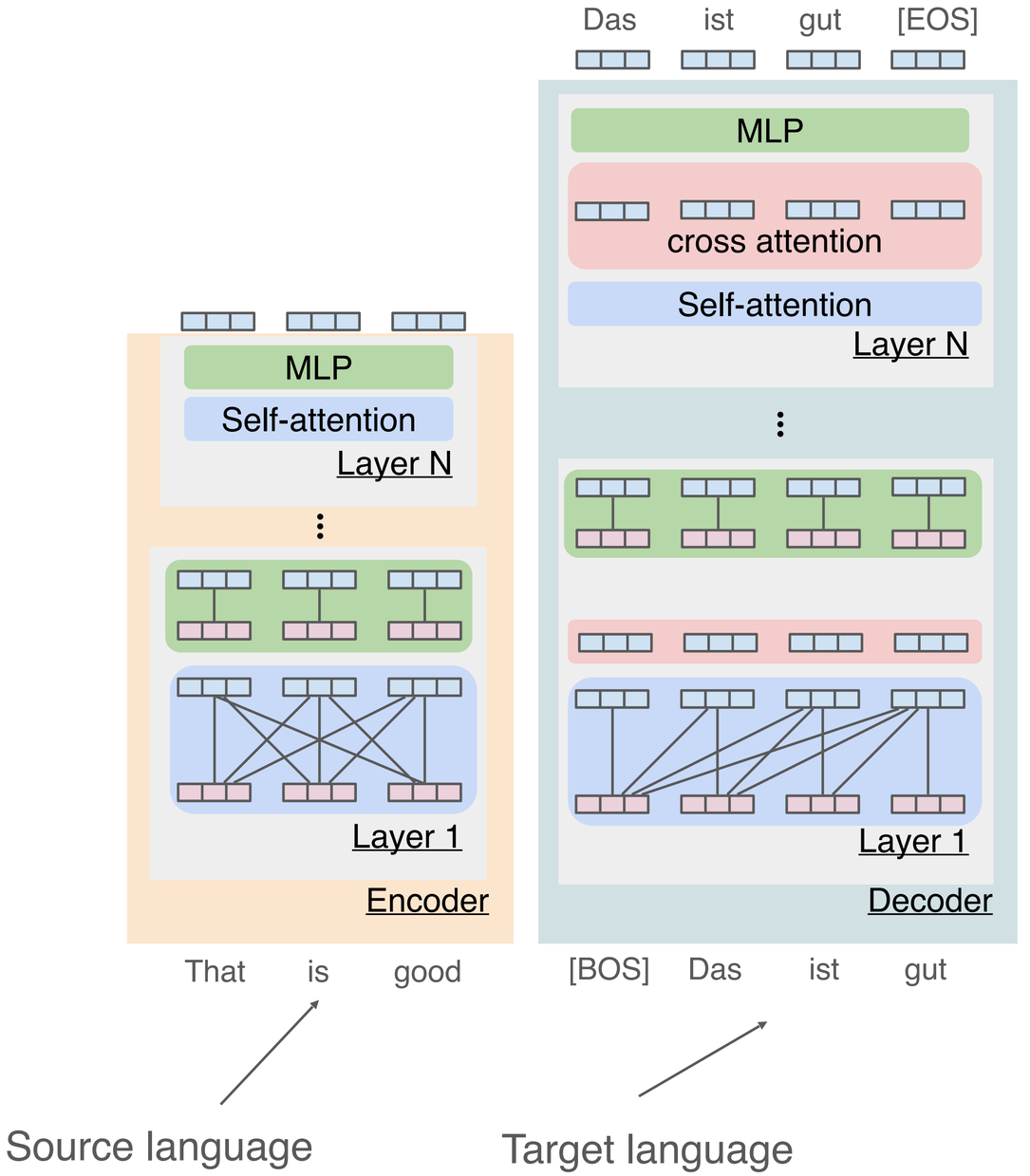
1. When encoding the input sequence, all-to-all interaction among the sequence elements is preferred

Example 1: machine translation

When Transformer was introduced in 2017, translation was a popular and difficult task

Input and target are in different languages

If the only goal is to learn translation, separate parameters make sense



Example 1: machine translation

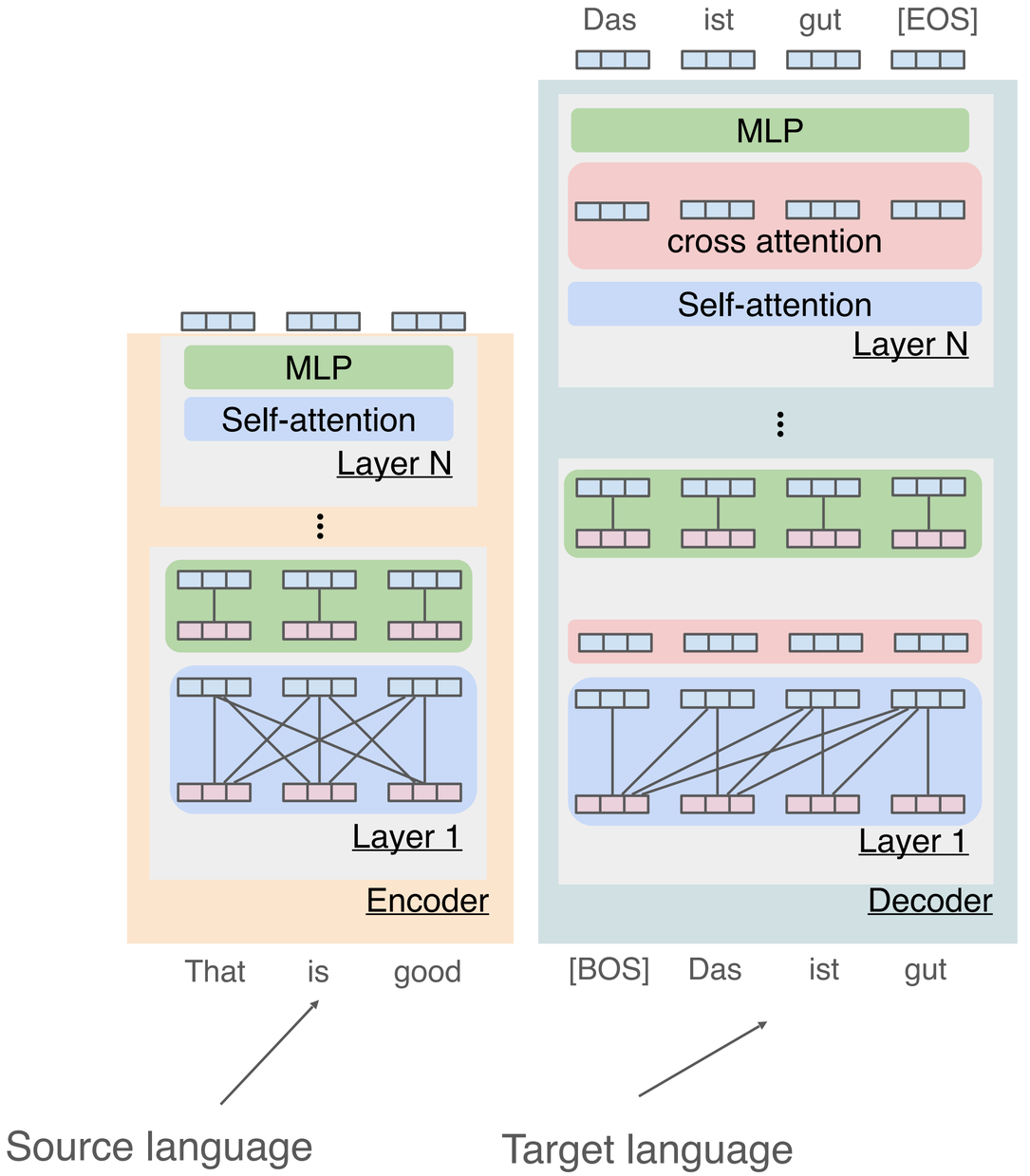
When Transformer was introduced in 2017, translation was a popular and difficult task

Input and target are in different languages

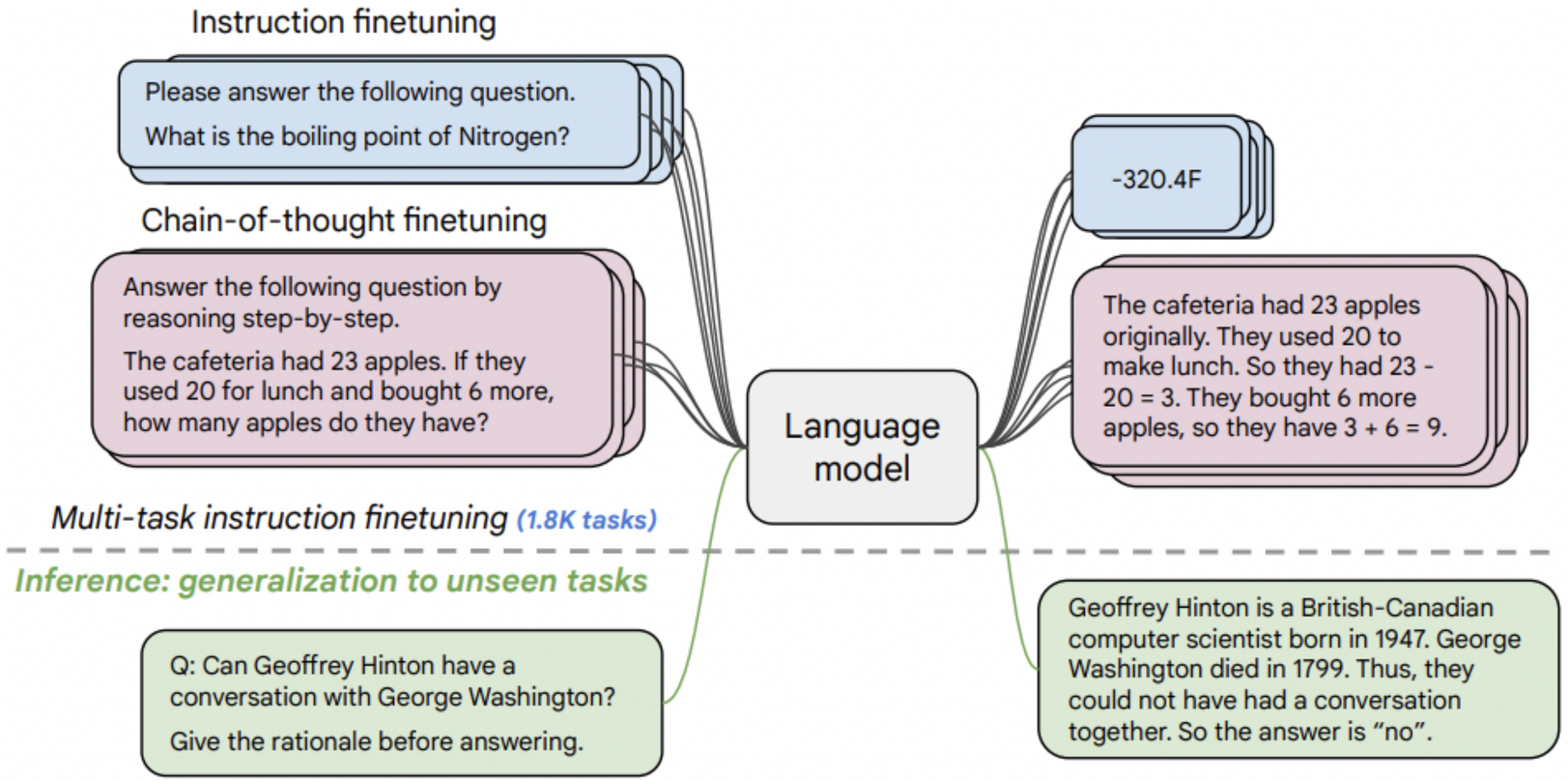
If the only goal is to learn translation, separate parameters make sense

Modern language models learn the world knowledge

Separate parameters for knowledge just expressed in different languages? 🤔



Example 2: instruction finetuning with academic datasets



Scaling Instruction-Finetuned Language Models
Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus et al. (2022)

Example 2: instruction finetuning with academic datasets

Encoder-decoder

Decoder-only

Params	Model	Norm. avg.
80M	T5-Small	-9.2
	Flan-T5-Small	-3.1 **(+6.1)**
250M	T5-Base	-5.1
	Flan-T5-Base	6.5 **(+11.6)**
780M	T5-Large	-5.0
	Flan-T5-Large	13.8 **(+18.8)**
3B	T5-XL	-4.1
	Flan-T5-XL	19.1 **(+23.2)**
11B	T5-XXL	-2.9
	Flan-T5-XXL	23.7 **(+26.6)**
8B	PaLM	6.4
	Flan-PaLM	21.9 **(+15.5)**
62B	PaLM	28.4
	Flan-PaLM	38.8 **(+10.4)**
540B	PaLM	49.1
	Flan-PaLM	58.4 **(+9.3)**
62B	cont-PaLM	38.1
	Flan-cont-PaLM	46.7 **(+8.6)**
540B	U-PaLM	50.2
	Flan-U-PaLM	59.1 **(+8.9)**

Encoder-decoder models had much bigger gain!

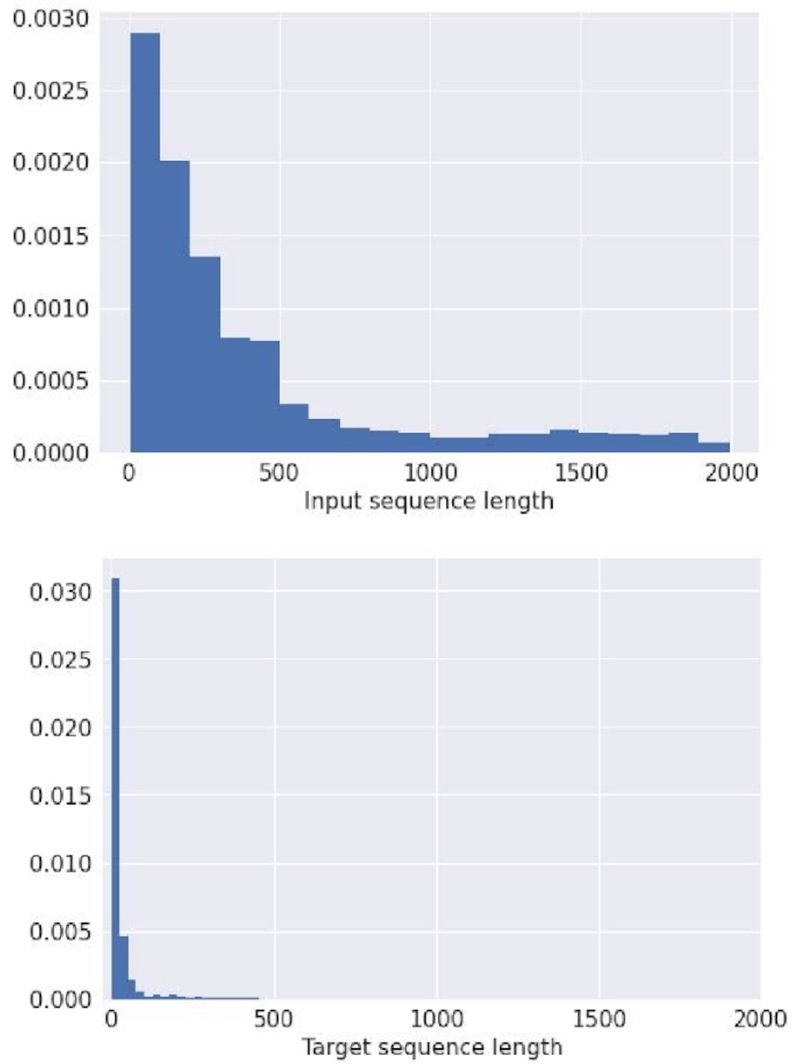
Example 2: instruction finetuning with academic datasets

Academic datasets have distinctive length distribution: long input and short target

This is due to inherent difficulty of grading long text

Hypothesis
Having separate parameters for longer text in input and shorter text in target was an effective *structure*

No longer a good structure as more interesting language model use cases have longer generation



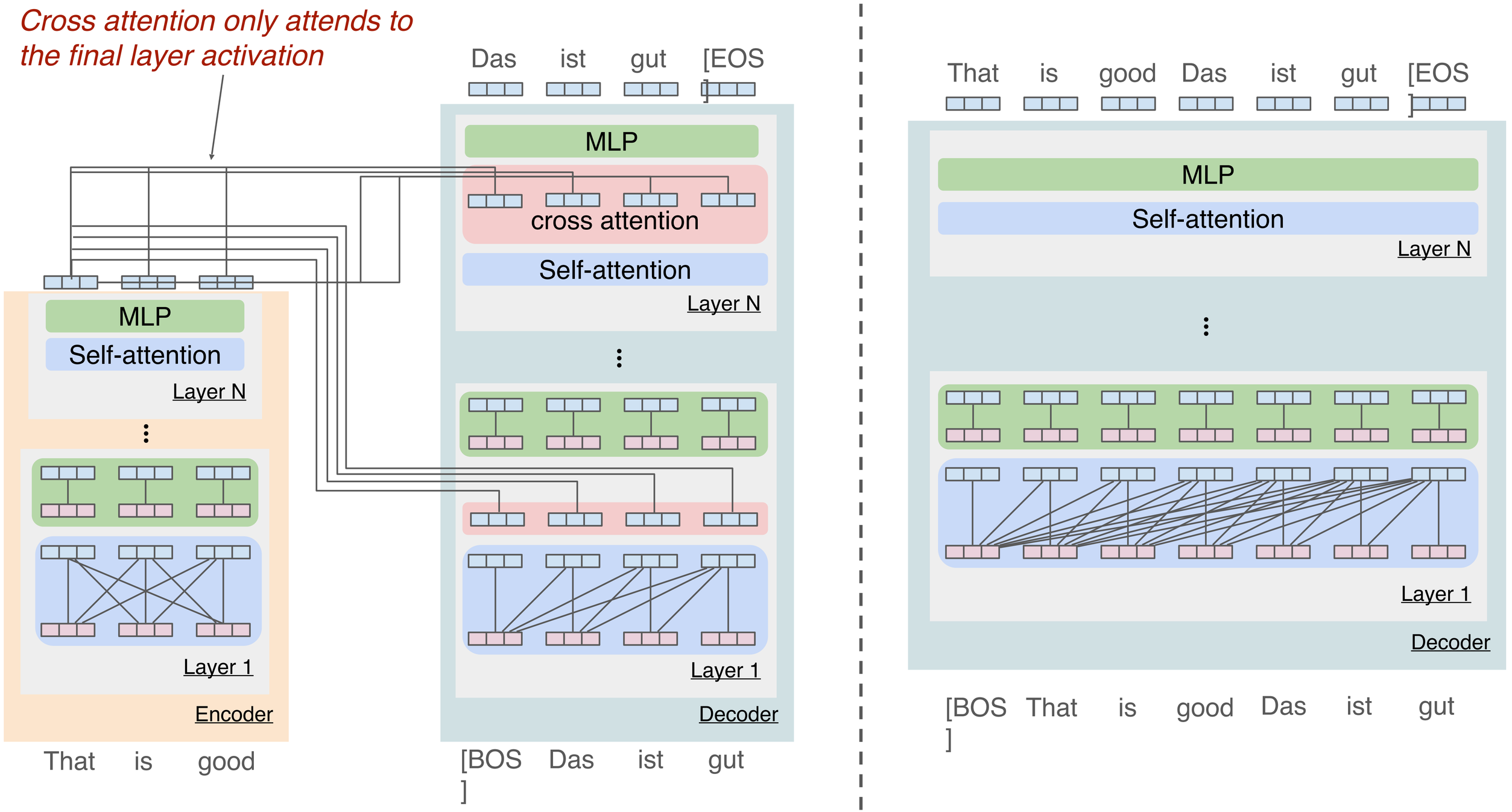
Length distribution of finetuning datasets

Additional structures in encoder-decoder compared to decoder-only

1. Input and target sequences are sufficiently different that using separate parameters can be effective

1. The target element can attend to the fully encoded representation of the input

1. When encoding the input sequence, all-to-all interaction among the sequence elements is preferred


Cross attention only attends to the final layer activation
Das ist gut [EOS]
MLP
cross attention
Self-attention
Layer N
MLP
Self-attention
Layer N
Layer 1
Encoder
Layer 1
Decoder
That is good
[BOS]
That is good Das ist gut [EOS]
MLP
Self-attention
Layer N
Layer 1
Decoder
[BOS] That is good Das ist gut

In deep neural networks, the bottom and top layers encode information at a different level of granularity

For example, in computer vision, the bottom layers learn to encode edges and the top layers learn higher level features (e.g. cat face)

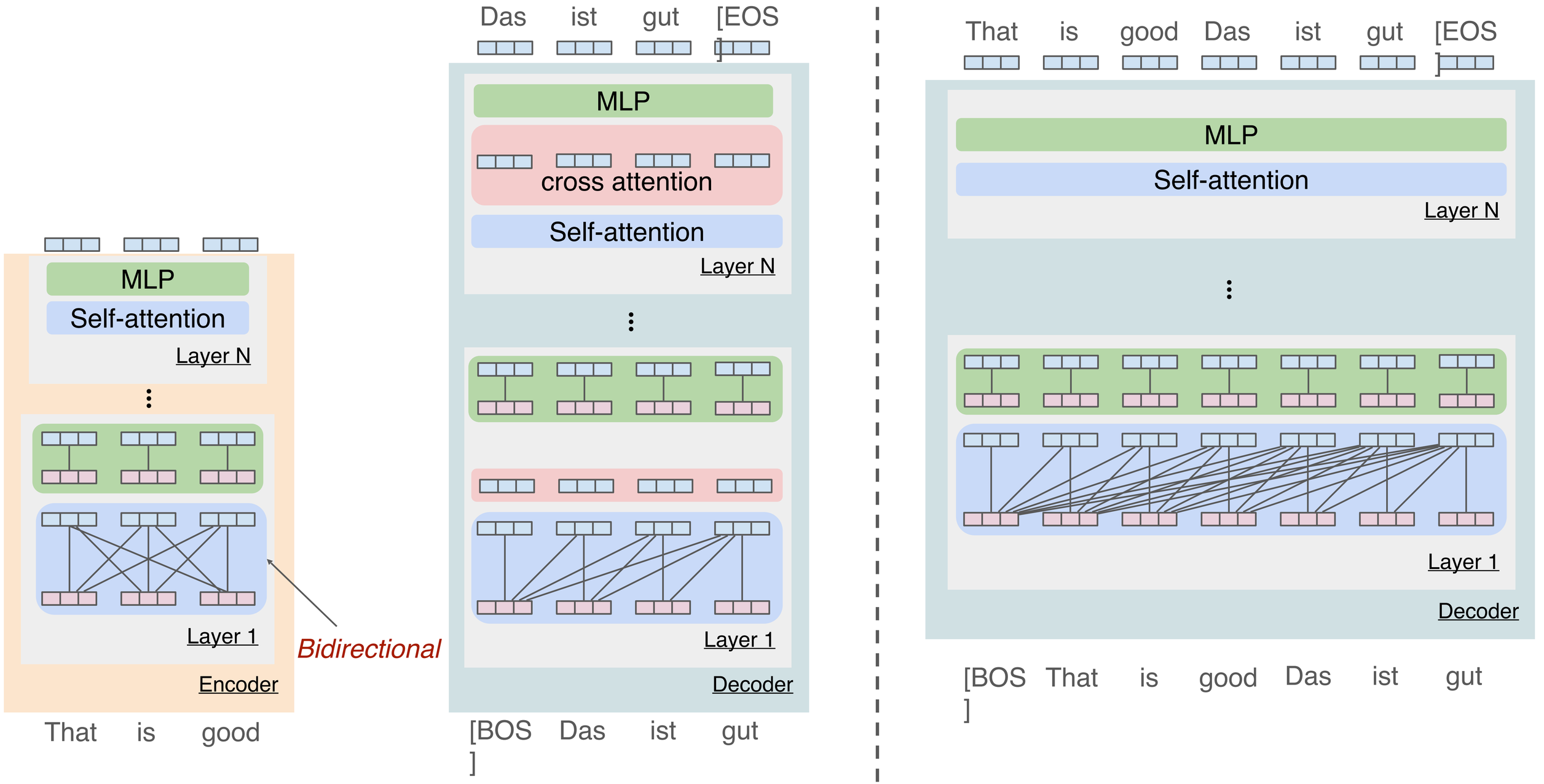
Can decoder only attending to the final layer of encoder be an *information bottleneck* if encoder is sufficiently deep? 🤔

Additional structures in encoder-decoder compared to decoder-only

1. Input and target sequences are sufficiently different that using separate parameters can be effective

1. The target element can attend to the fully encoded representation of the input

1. When encoding the input sequence, all-to-all interaction among the sequence elements is preferred


Das ist gut [EOS]
MLP
cross attention
Self-attention
Layer N
Layer 1
Decoder
MLP
Self-attention
Layer N
Layer 1
Encoder
Bidirectional
That is good
[BOS]
That is good Das ist gut [EOS]
MLP
Self-attention
Layer N
Layer 1
Decoder
[BOS] That is good Das ist gut

Remember this slide

(??Highly anecdotal?? – may be not true)
At sufficient scale, bidirectionality doesn’t seem to matter much

Bidirectionality brings in engineering challenges for multi-turn chat application

At every turn, the new input has to be encoded again; for unidirectional attention, only the newly added message needs to be encoded.

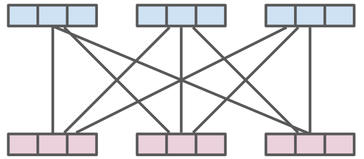
Input attention pattern for multi-turn conversation

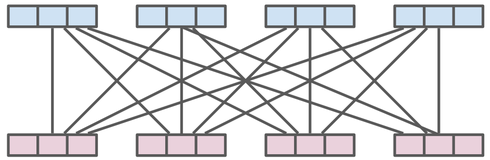
USER
How are you?

ASSISTANT
Bad

USER
Why?

Bidirectional





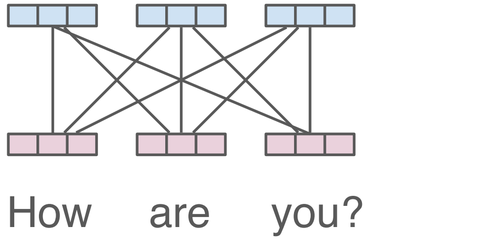
Input attention pattern for multi-turn conversation

USER
How are you?

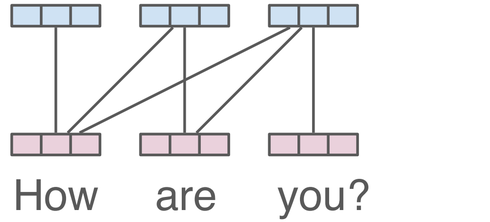
ASSISTANT
Bad

USER
Why?

Bidirectional



Unidirectional



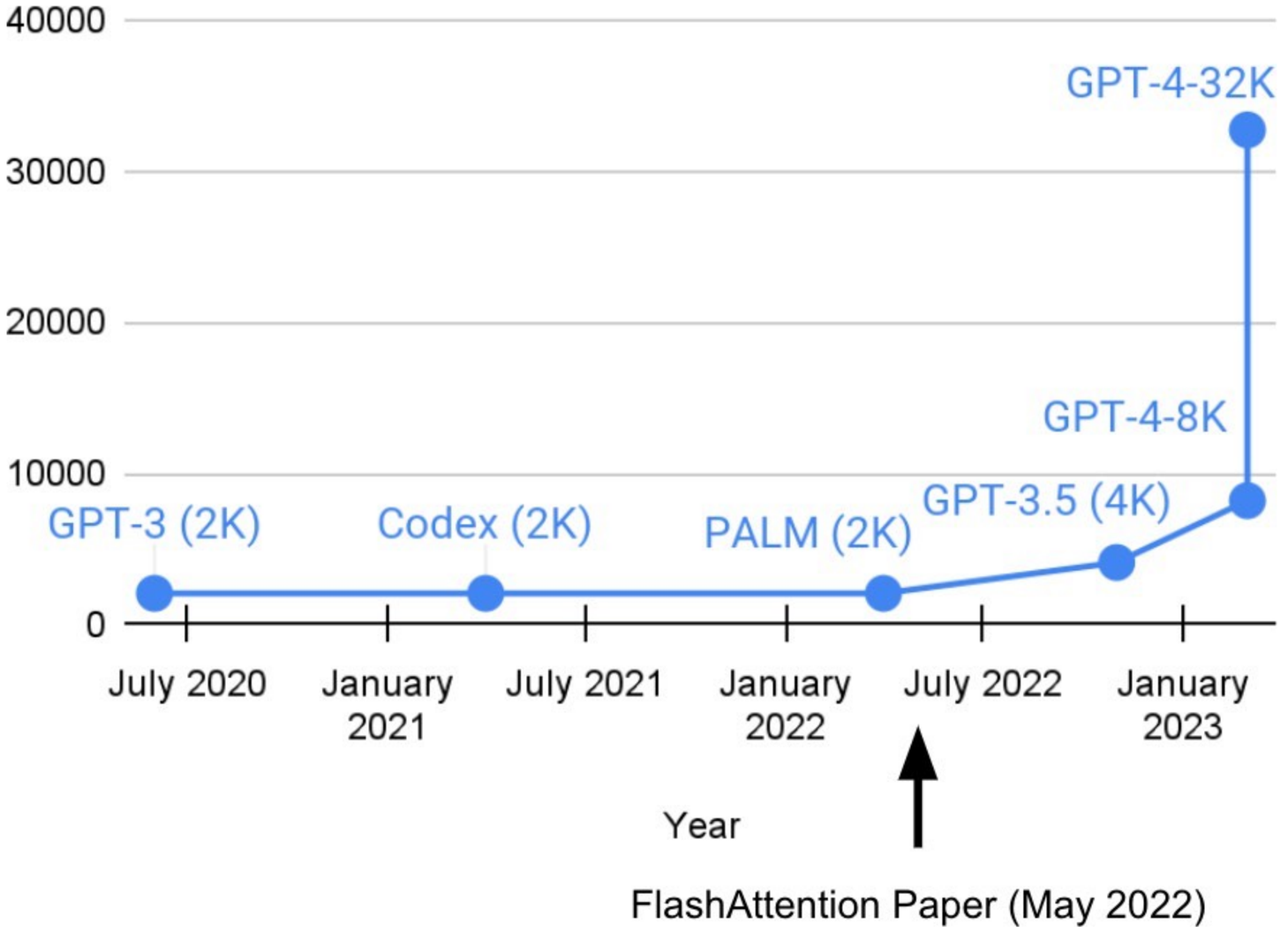
Linear Attentions and Beyond

What follows (linear attention explanations) is from this lecture:

Look for the link at the end of the slides – there are parts that I omit here like gated delta network parallelisation strategy and talk about expressivity

Linear Attention and Beyond (Interactive Tutorial with Songlin Yang)

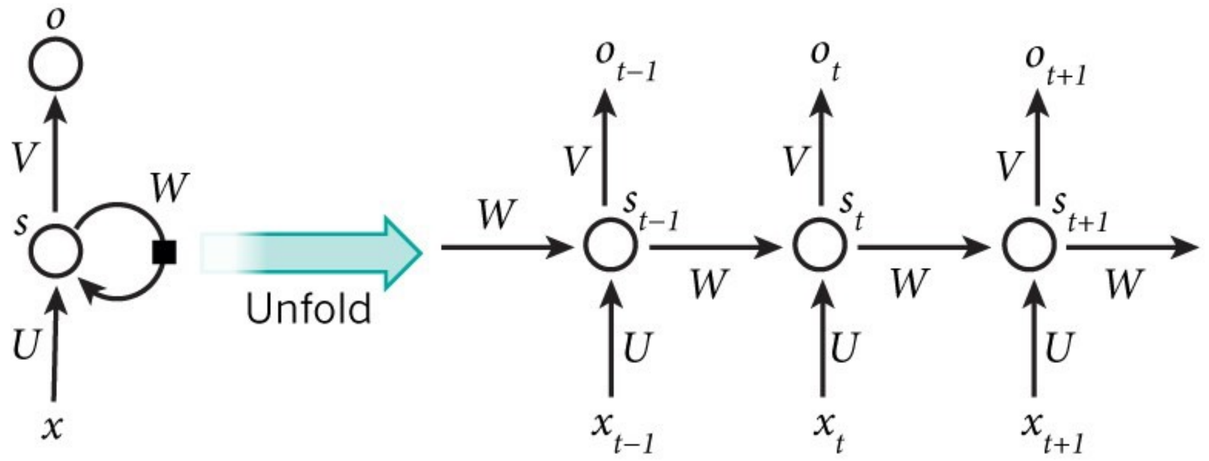
Foundation Model Context Length



Issues with Transformers

- Training: quadratic time complexity
  - Expensive for long sequence modeling (e.g., video or DNA modeling)
- Inference: linear memory complexity
  - Requires storing KV cache for each token
  - High memory burden.

Revisiting RNNs



- Training: linear complexity, however, traditional RNNs are not parallelizable.

- Inference: constant memory

Modern linear recurrent models

Use linear recurrence for parallel training

- Gated linear RNNs (HGRN, Griffin, ...)
- State-space models (S4, Mamba, ...)
- Linear attention (RetNet, GLA, xLSTM, DeltaNet, ...)

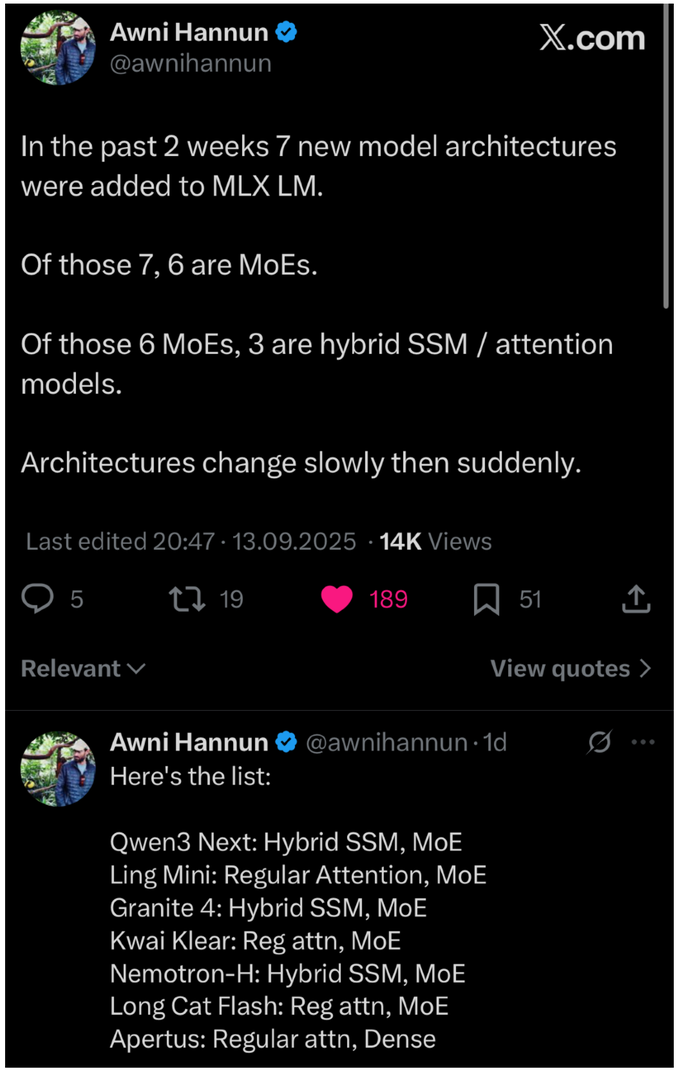
Modern linear recurrent models

Use linear recurrence for parallel training

- Gated linear RNNs (HGRN, Griffin, ...)
- State-space models (S4, Mamba, ...)
- **Linear attention (RetNet, GLA, xLSTM, DeltaNet, ...)**

Linear attention is the focus today


Tri Dao @tri_dao · Jan 15
Hybrid linear-softmax attention working very well at large scale and long-context! As we've seen with multiple models now, you only need a couple of (full) attention layers
MiniMax (official) @MiniMax__AI · Jan 14
MiniMax-01 is Now Open-Source: Scaling Lightning Attention for the AI Agent Era
We are thrilled to introduce our latest open-source models: the foundational language model MiniMax-Text-01 and the visual multi-...


Awni Hannun
@awnihannun
X.com
In the past 2 weeks 7 new model architectures were added to MLX LM.
Of those 7, 6 are MoEs.
Of those 6 MoEs, 3 are hybrid SSM / attention models.
Architectures change slowly then suddenly.
Last edited 20:47 · 13.09.2025 · 14K Views
5
19
189
51
Relevant
View quotes
Awni Hannun @awnihannun · 1d
Here's the list:
Qwen3 Next: Hybrid SSM, MoE
Ling Mini: Regular Attention, MoE
Granite 4: Hybrid SSM, MoE
Kwai Klear: Reg attn, MoE
Nemotron-H: Hybrid SSM, MoE
Long Cat Flash: Reg attn, MoE
Apertus: Regular attn, Dense

Linear Attention

Softmax Attention

Attention:

$$\text{Parallel training}: \quad \mathbf{O} = \text{softmax}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L\times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\exp(\mathbf{q}_t^\top \mathbf{k}_j)}{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)} \mathbf{v}_j \quad \in \mathbb{R}^{d}$$

where $\mathbf{M} \in \mathbb{R}^{L\times L}$ is the casual mask:

$$\mathbf{M}_{i,j} = \begin{cases} -\infty & \text{if } j > i \\ 1 & \text{if } j \leq i \end{cases}$$

Linear Attention

Linear attention (Katharopoulos et al. 2020):

$$\text{Parallel training}: \quad \mathbf{O} = \cancel{\text{softmax}}(\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \quad \in \mathbb{R}^{L\times d}$$

$$\text{Iterative inference}: \quad \mathbf{o_t} = \sum_{j=1}^{t} \frac{\cancel{\exp}(\mathbf{q}_t^\top \mathbf{k}_j)}{\cancel{\sum_{l=1}^{t} \exp(\mathbf{q}_t^\top \mathbf{k}_l)}} \mathbf{v}_j \quad \in \mathbb{R}^{d}$$

where $\mathbf{M}$ is the causal mask for linear attention:

$$\mathbf{M}_{i,j} = \begin{cases} 0 & \text{if } j > i \\ 1 & \text{if } j \leq i \end{cases}$$

Linear attention: Matrix-valued hidden state

$$
\begin{aligned}
\mathbf{o_t} &= \sum_{j=1}^{t} (\mathbf{q}_t^\top \mathbf{k}_j)\mathbf{v}_j \\
&= \sum_{j=1}^{t} \mathbf{v}_j(\mathbf{k}_j^\top \mathbf{q}_t) \quad \mathbf{k}_j^\top \mathbf{q}_t = \mathbf{q}_t^\top \mathbf{k}_j \in \mathbb{R} \\
&= \underbrace{(\sum_{j=1}^{t} \mathbf{v}_j \mathbf{k}_j^\top)}_{\mathbf{S}_t \in \mathbb{R}^{d \times d}} \mathbf{q}_t \qquad \text{By associativity}
\end{aligned}
$$

Challenges in training: the parallel form

Recall our part 1

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{M})\mathbf{V} \in \mathbb{R}^{L \times d}$$

The time complexity is still quadratic in sequence length, which is problematic for long sequences.

Challenges in training: the recurrent form

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \quad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \quad \in \mathbb{R}^{d}$$

Poor GPU utilization due to:

- Sequential computation limits parallelization opportunities across the sequence dimension.
- Rank-1 outer product updates and matrix-vector multiplications are not optimized for GPU tensor cores, which are designed for dense matrix-multiply operations (typically at least 16x16 matrix sizes).

Solution: chunk-wise parallel

Chunkwise Form:

- Interpolates between recurrent and parallel forms.
- Splits a sequence of length L into L/C chunks of size C:
  - When $C = 1$, it reduces to the recurrent form.
  - When $C = L$, it reduces to the parallel form.
- **Key Property:** Chunkwise form is **NOT an approximation**—it computes the exact same output as the original formulation.

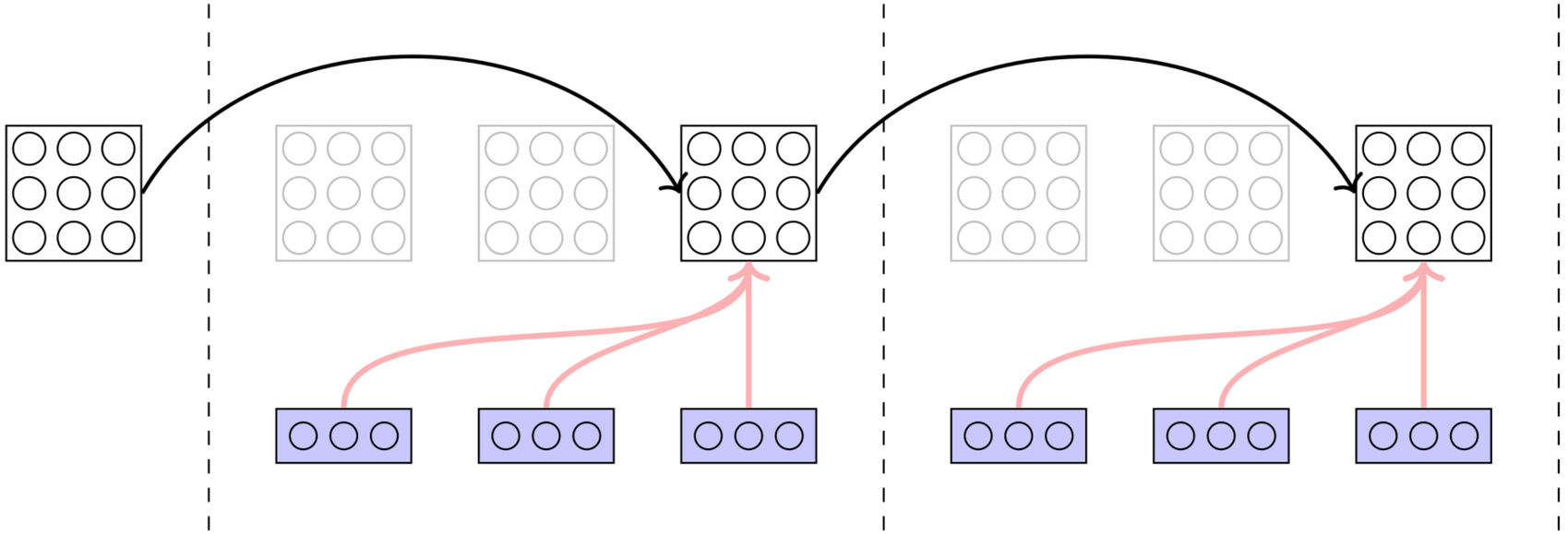
Notation

$$\mathbf{S}_{[i]} := \mathbf{S}_{iC} \in \mathbb{R}^{d \times d} \qquad \text{the last hidden state of chunk } i,$$

$$\mathbf{Q}_{[i]} = \mathbf{Q}_{iC+1:(i+1)C} \in \mathbb{R}^{C \times d} \qquad \text{the query block of chunk } i.$$

We define $\mathbf{K}_{[i]}, \mathbf{V}_{[i]}, \mathbf{O}_{[i]}$ in a similar way.

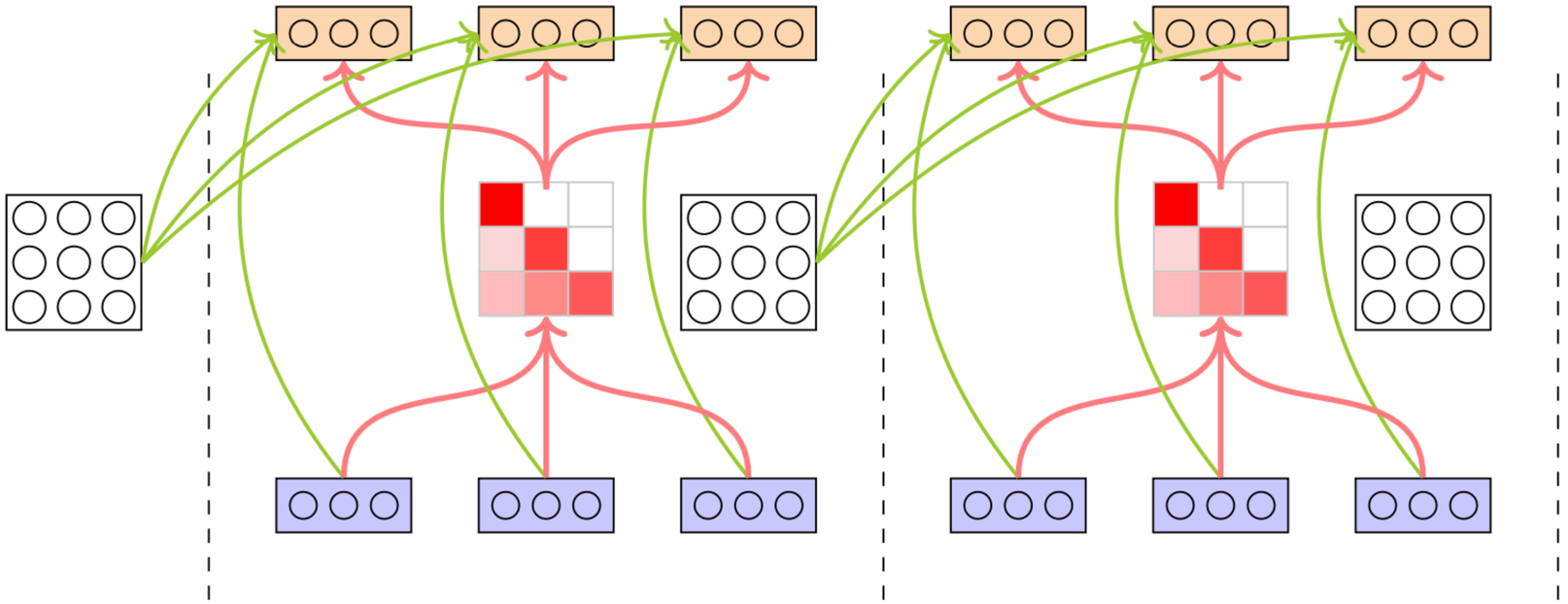
Sequential Chunk-Level State Passing:



$$\mathbf{S}_{[t+1]} = \underbrace{\mathbf{S}_{[t]}}_{\mathbb{R}^{d\times d}} + \underbrace{\mathbf{V}_{[t]}^{\top}}_{\mathbb{R}^{d\times C}} \underbrace{\mathbf{K}_{[t]}}_{\mathbb{R}^{C\times d}} \qquad \in \mathbb{R}^{d\times d}$$

Computational Complexity: $\mathcal{O}(Cd^2)$ per chunk and $\mathcal{O}(Ld^2)$ for the entire sequence.

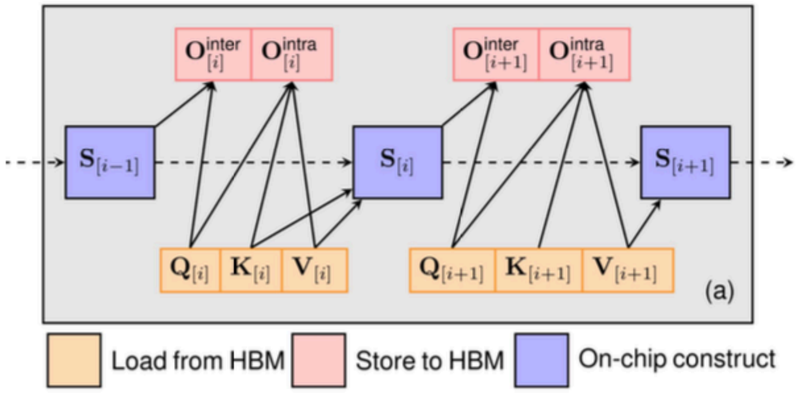
Parallel Output Computation:

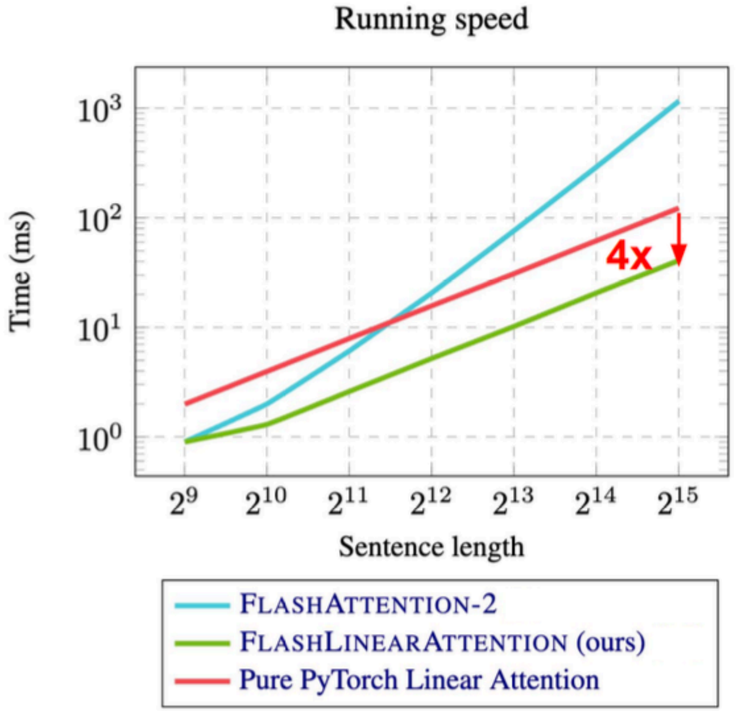


$$\mathbf{O}_{[t]} = \underbrace{\mathbf{Q}_{[t]}\mathbf{S}_{[t]}^{\top}}_{\text{inter-chunk:}\mathbf{O}_{[t]}^{\text{inter}}} + \underbrace{(\mathbf{Q}_{[t]}\mathbf{K}_{[t]}^{\top} \odot \mathbf{M})\mathbf{V}_{[t]}}_{\text{intra-chunk:}\mathbf{O}_{[t]}^{\text{intra}}} \in \mathbb{R}^{C\times d}$$

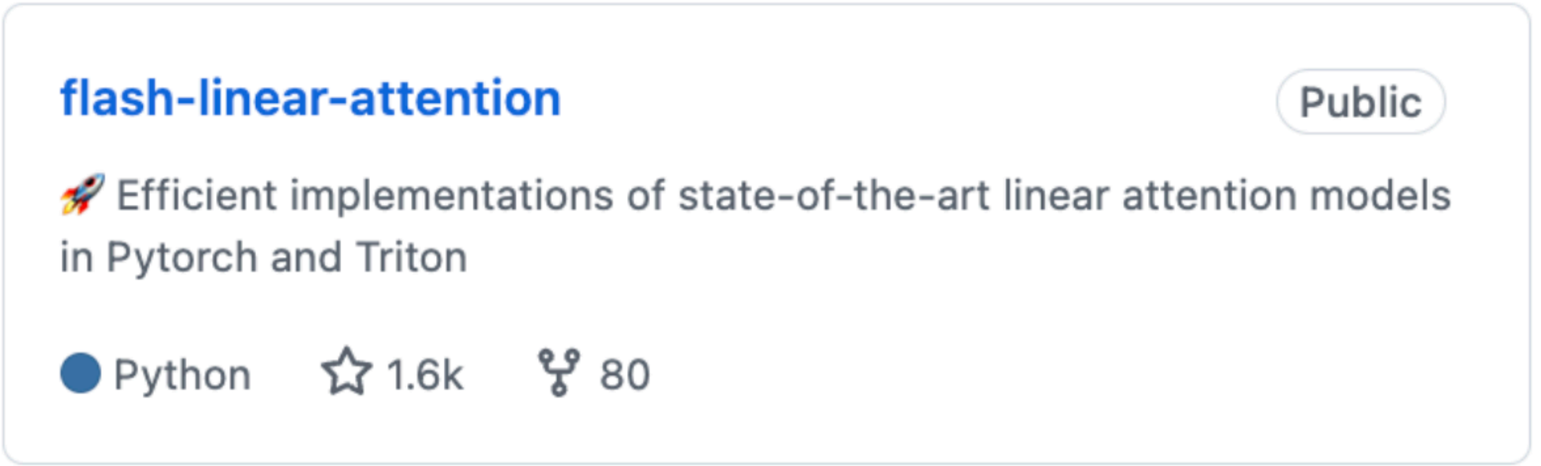
Computational Complexity: $\mathcal{O}(C^2d + Cd^2)$ per chunk.
$\mathcal{O}(Ld^2 + LCd)$ for the entire sequence.

- Total complexity: $\mathcal{O}(Ld^2 + LdC)$, achieving subquadratic complexity in sequence length when C is small.
- Practical settings: C is typically set to $\{64, 128, 256\}$.
- Extensibility: Can be generalized to linear attention with decay or delta rule (to be discussed later).
- Adoption: The de facto standard for training modern linear attention models, including:
  - Mamba2, Based, GLA, DeltaNet, Lightning Attention, mLSTM, and others.





I/O optimization significantly improves the wall-clock time.



The Flash Linear Attention library provides hardware-efficient implementation of various linear attention models.

- RetNet, GLA, Based, HGRN2, RWKV6, GSA, Mamba2, DeltaNet, Gated DeltaNet, RWKV7 ...

- Linear attention = Softmax attention - softmax.
- Linear attention = Linear RNN + matrix-valued hidden state.
- The chunkwise parallel form is more hardware-friendly than the recurrent and parallel forms.
- Flash Linear Attention is an I/O-aware implementation of the chunkwise parallel form.

Linear Attention is not enough

$$\mathbf{S}_t = \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

Vanilla linear attention performs poorly in language modeling.

- Only stores key-value pairs in memory.
- Has no mechanism to forget old memories.
- Leads to memory saturation and degradation in performance since the memory size is fixed.

$$\mathbf{S}_t = \gamma \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- γ is a constant exponential decay factor $0 < \gamma < 1$.
- This mechanism weighs recent tokens more than distant tokens, and language modeling has a strong recency bias.
- Works well in practice: RetNet (Sun et al. 2023), Lightning Attention (Qin et al. 2024b).

$$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$

$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

- $\gamma_t \in (0, 1)$ is a data-dependent decay term that is a function of $\mathbf{x}_t$.
- Enables dynamic control of memory retention/forgetting based on input data.
- Examples: Mamba2 (Dao and Gu 2024), mLSTM (Beck et al. 2024), Gated Retention (Sun et al. 2024b).

The parallel form for linear attention with decay

$$\mathbf{S}_t = \gamma_t \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \in \mathbb{R}^{d}$$

Linear attention with decay has the following parallel form:

$$\mathbf{O} = (\mathbf{Q}\mathbf{K}^\top \odot \mathbf{D})\mathbf{V} \in \mathbb{R}^{L \times d}$$

$$\mathbf{D}_{i,j} = \begin{cases} \prod_{m=j+1}^{i} \gamma_m & \text{if } i > j \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

The duality between recurrent and parallel forms is referred to as state space duality (SSD) in Mamba2 (Dao and Gu 2024).

Linear attention with more fine-grained decay

$$\mathbf{S}_t = \mathbf{G}_t \odot \mathbf{S}_{t-1} + \mathbf{v}_t \mathbf{k}_t^\top \qquad \in \mathbb{R}^{d \times d}$$
$$\mathbf{o}_t = \mathbf{S}_t \mathbf{q}_t \qquad \in \mathbb{R}^{d}$$

Condition for the matmul-based (chunkwise) parallel form (Yang et al. 2023):

$$\mathbf{G}_t = \boldsymbol{\beta}_t \boldsymbol{\alpha}_t^\top \in \mathbb{R}^{d \times d}, \quad \boldsymbol{\alpha}_t, \boldsymbol{\beta}_t \in \mathbb{R}^{d}$$

- Mamba-1 (Gu and Dao 2023) does not conform to this structure, preventing the use of tensor cores.
- It is common to set $\boldsymbol{\beta}_t = \mathbf{1}$ in practice, examples: GLA (Yang et al. 2023), RWKV6 (Peng et al. 2024), GSA (Zhang et al. 2024), HGRN2 (Qin et al. 2024a).

Summary

- Language modeling has a strong recency bias.
- Decay helps bridge the perplexity gap between linear and softmax attention.
- Fine-grained decay enhances performance but poses scaling challenges.
- Outer-product structure enables efficient chunkwise training with fine-grained decay.