

Shaping the future of AI from the history of Transformer

Hyung Won Chung

OpenAI

Twitter: [@hwchung27](https://twitter.com/hwchung27)

AI is advancing so fast that it is hard to keep up

People spend a lot of time and energy catching up with the latest developments

But not enough attention goes to the old things

It is more important to study the change itself

What does it mean to study the change itself?

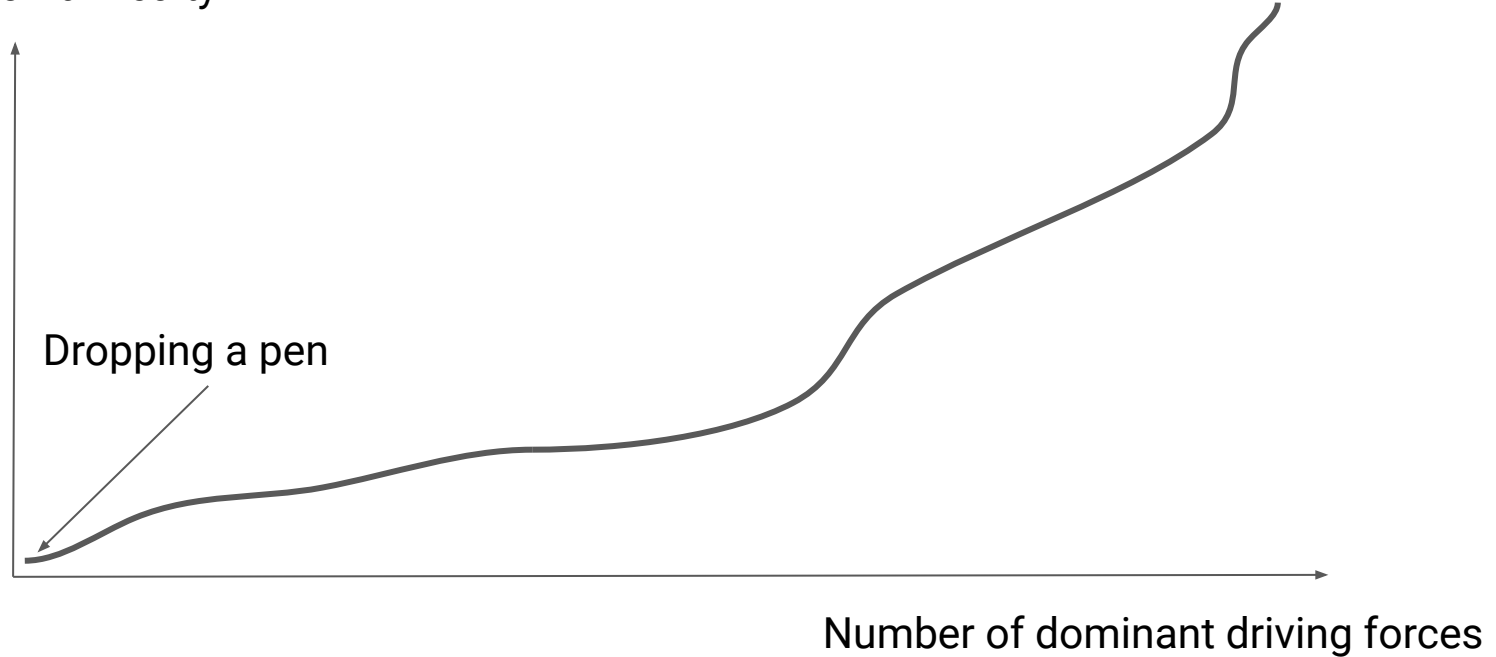
- 1 Identify the dominant driving forces behind the change
- 2 Understand the dominant driving forces
- 3 Predict the future trajectory

Toy experiment: dropping a pen

- 1 Identify the dominant driving forces: gravity
- 2 Understand gravity: Newtonian mechanics provides a good model
- 3 Predict the future trajectory of the pen $y(t) = \frac{1}{2}gt^2$

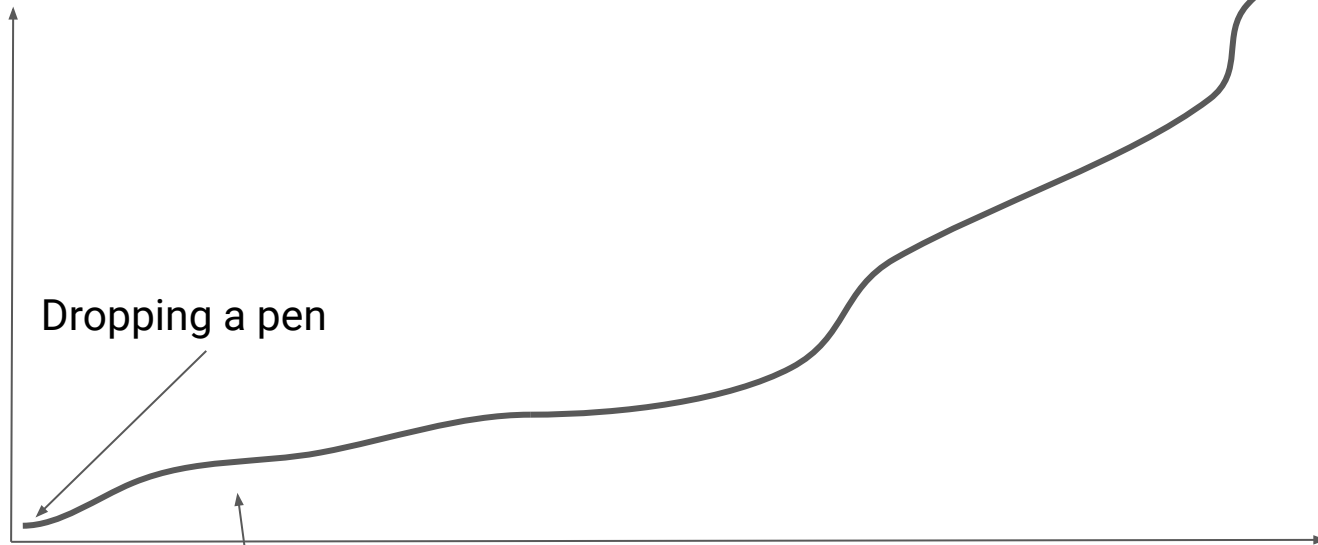
Predicting the future trajectory is difficult because there are many driving forces and the complexity of their interactions

Prediction difficulty



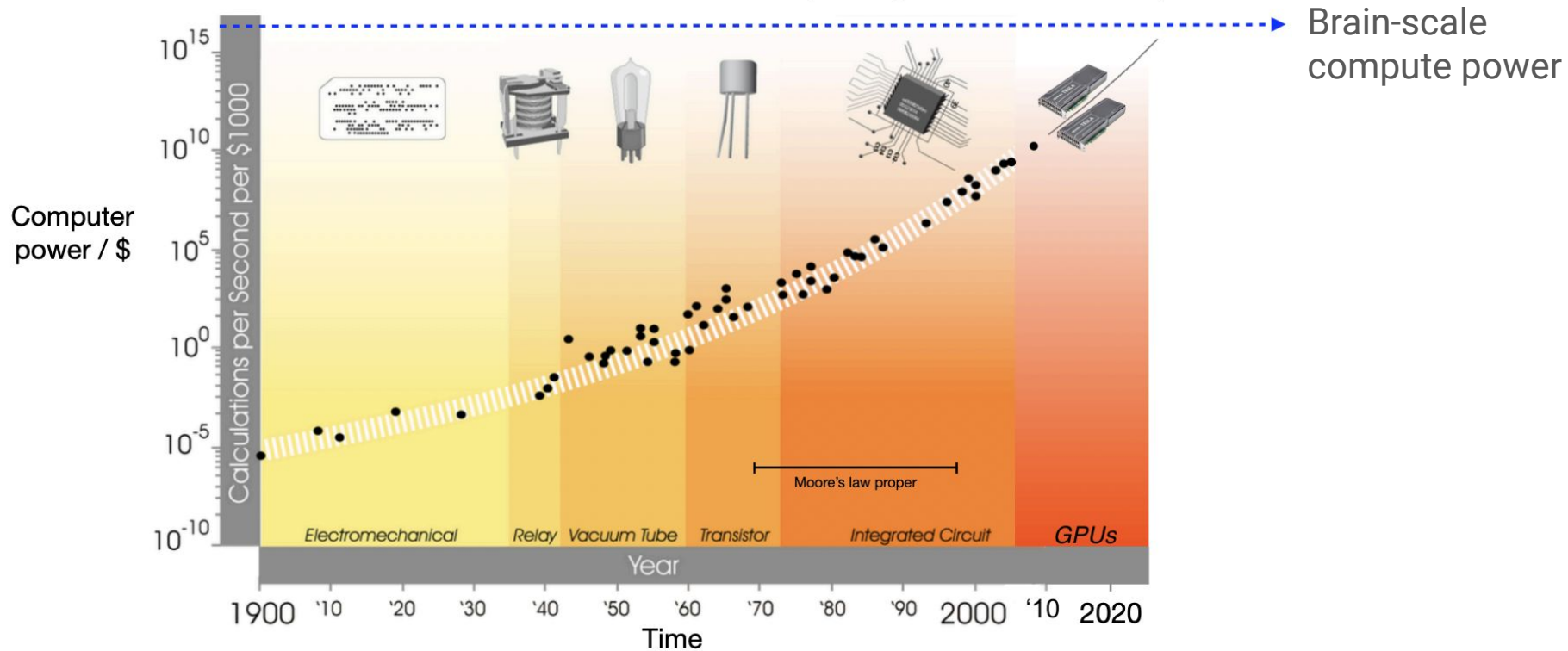
Predicting the future trajectory is difficult because there are many driving forces and the complexity of their interactions

Prediction difficulty



Number of dominant driving forces

AI research is closer to the left than we feel



Roughly, 10x more compute every 5 years

The job of AI researchers is to teach machines how to “think”

One (unfortunately common) approach

Teach the machines how *we think* we think

This approach poses structures to the problem, which can become the limitation when scaled up

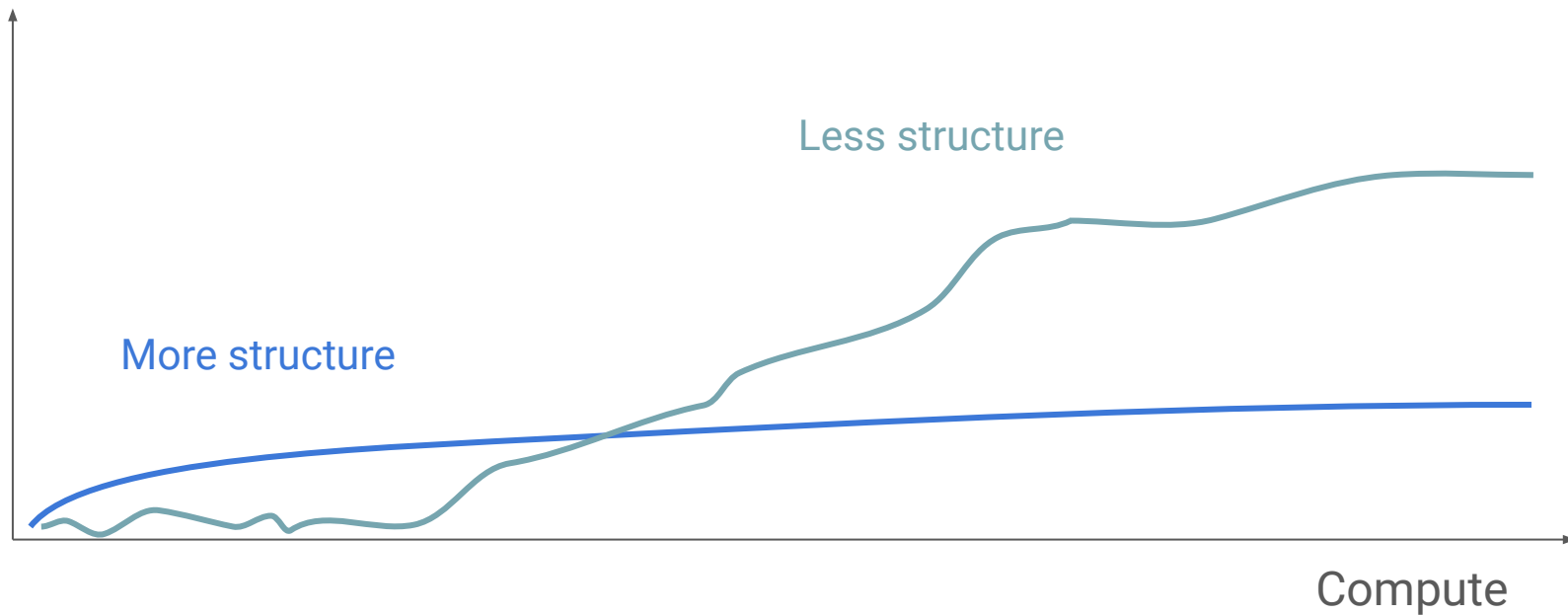
The job of AI researchers is to teach machines how to “think”

Bitter lesson: progress of AI in the past 70 years boils down to

- Develop progressively more general methods with weaker modeling assumptions
- Add more data and computation (i.e. scale up)

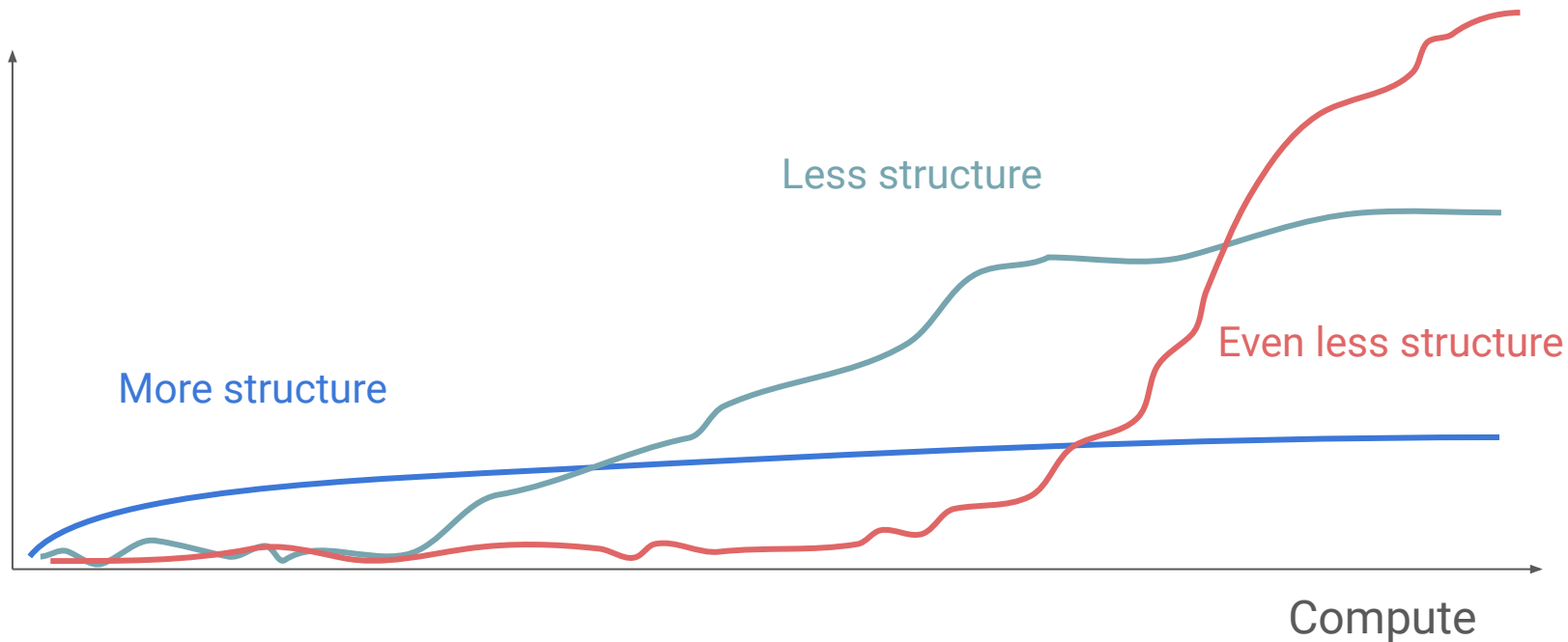
The more structure, the less scalable the method is

Performance



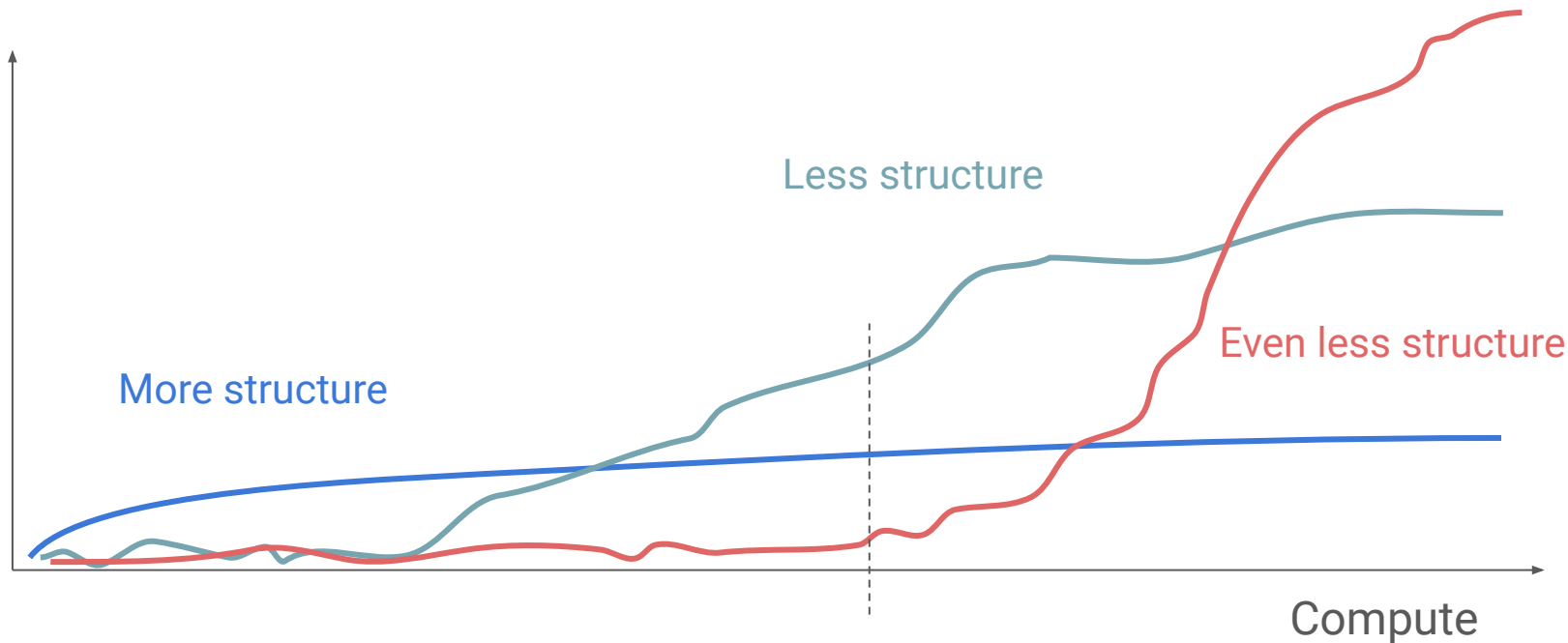
However, we can't just go for the most general method

Performance



However, we can't just go for the most general method

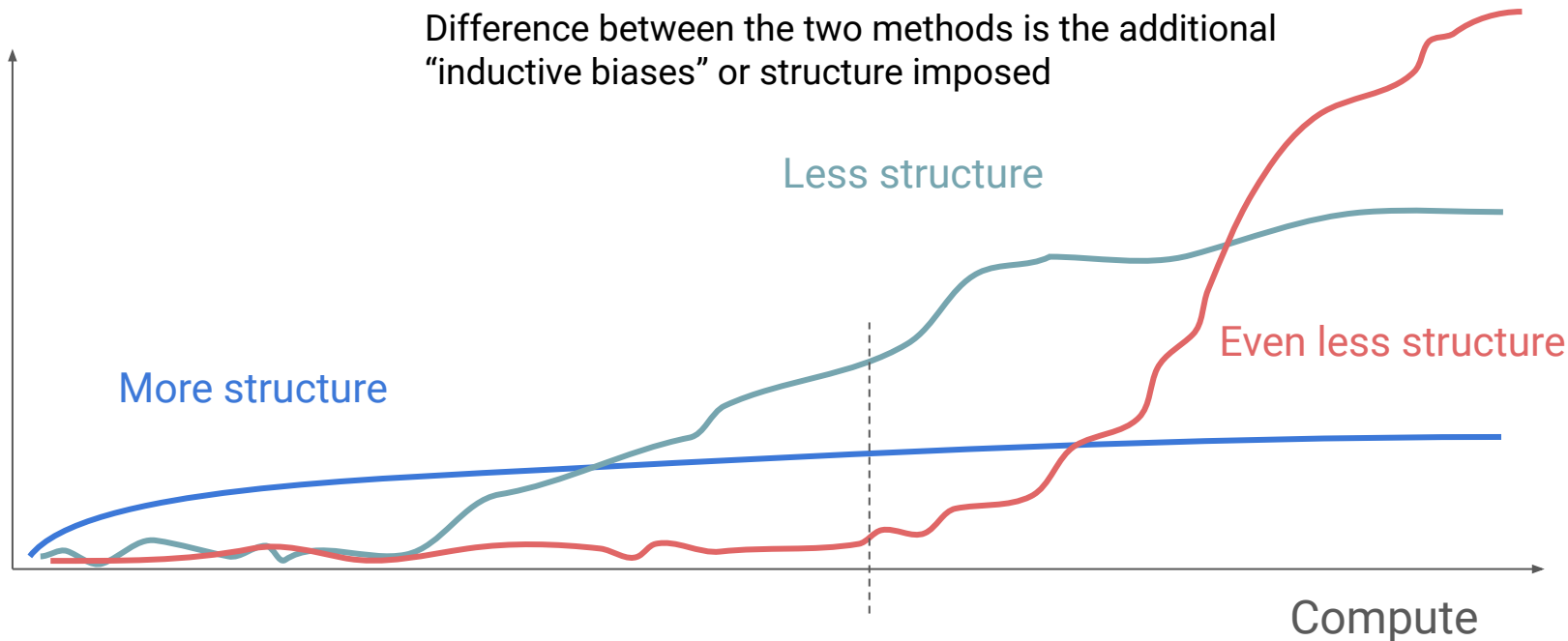
Performance



If we are here, we should choose “Less structure”. But remember to undo later

However, we can't just go for the most general method

Performance



If we are here, we should choose "Less structure". But remember to undo later

Adding optimal inductive bias for a given level of compute, data, algorithmic development and architecture is critical

These are shortcuts that will hinder further scaling later on. Remove them later.

As a community we do the former well but not the latter

Implications of bitter lesson

What is better in the long term almost always looks worse now

This is somewhat unique to the AI research. If clever modeling techniques and fancy math were the driving force, it would have been completely different story

Summary

We identified the dominant driving force: exponentially cheaper compute and scaling

1

Now we need to understand it better

2

For that we will go to back to early history of Transformer and analyze key structures added by researchers and their motivations.

Then we will see how these structures became less relevant with now that more compute and better algorithm is available

Transformer architectures variants

1. Encoder-decoder
2. Encoder-only
3. Decoder-only (least structure)

Process

Shape

“Unicode characters like emojis may be split.”

[]

Process

Shape

Unicode characters like emojis may be split.

[7085, 2456, 836, 470, 3975, 284, 530, 11241, 25, 773, 452, 12843, 13]



[Tokenization](#)

“Unicode characters like emojis may be split.”

[length]



[]

Process

Shape

2.3	-3.2	8.3	5.4	2.1	3.9	-8.9	3.8	3.9	3.3
4.5	5.9	4.5	7.1	1.0	5.3	5.0	3.1	0.7	5.0
...
3.8	1.2	3.8	9.0	9.3	3.1	4.2	0.8	9.2	5.8

[d_model, length]

↑ Embedding



Unicode characters like emojis may be split.

[length]

[7085, 2456, 836, 470, 3975, 284, 530, 11241, 25, 773, 452, 12843, 13]

↑ [Tokenization](#)



“Unicode characters like emojis may be split.”

[]

Process

3.2	-2.3	3.8	4.5	1.2	9.3	-9.8	8.3	9.3	3.3
5.4	9.5	5.4	1.7	0.1	3.5	0.5	1.3	7.0	0.5
...
8.3	2.1	8.3	0.9	3.9	1.3	2.4	8.0	2.9	8.5

↑ Sequence model

2.3	-3.2	8.3	5.4	2.1	3.9	-8.9	3.8	3.9	3.3
4.5	5.9	4.5	7.1	1.0	5.3	5.0	3.1	0.7	5.0
...
3.8	1.2	3.8	9.0	9.3	3.1	4.2	0.8	9.2	5.8

↑ Embedding

Unicode characters like emojis may be split.

[7085, 2456, 836, 470, 3975, 284, 530, 11241, 25, 773, 452, 12843, 13]

↑ [Tokenization](#)

“Unicode characters like emojis may be split.”

Shape

[d_model, length]



[d_model, length]



[length]

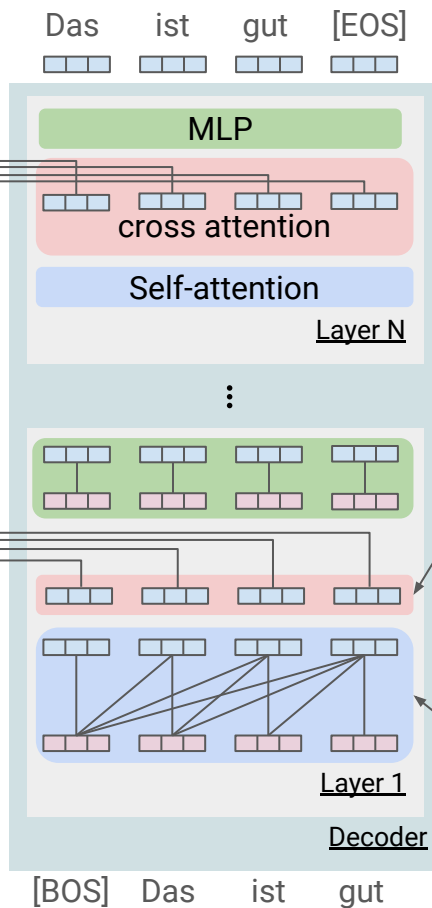
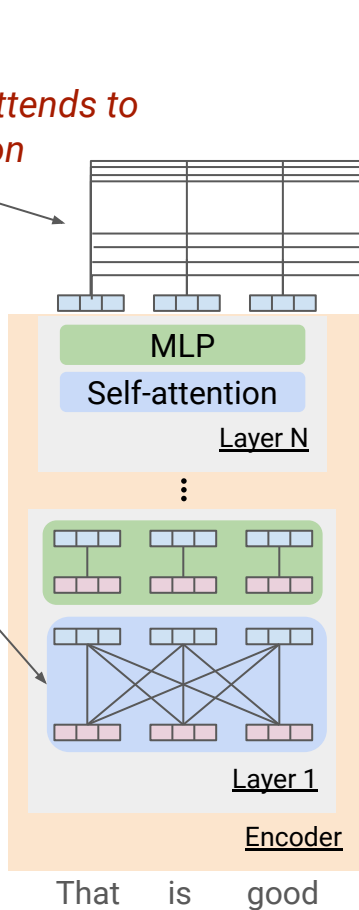


[]

Encoder-decoder

Cross attention only attends to the final layer activation

Bidirectional self-attention



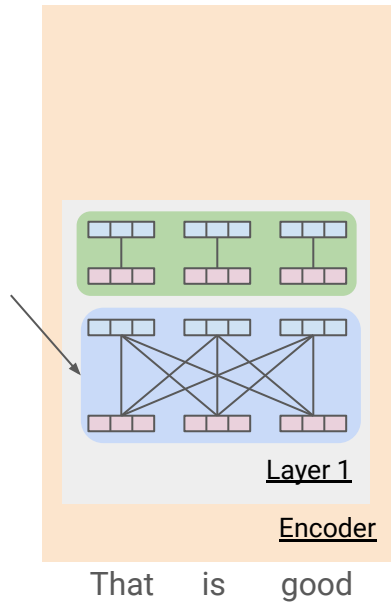
Output is a sequence

Cross-attention connects encoder and decoder

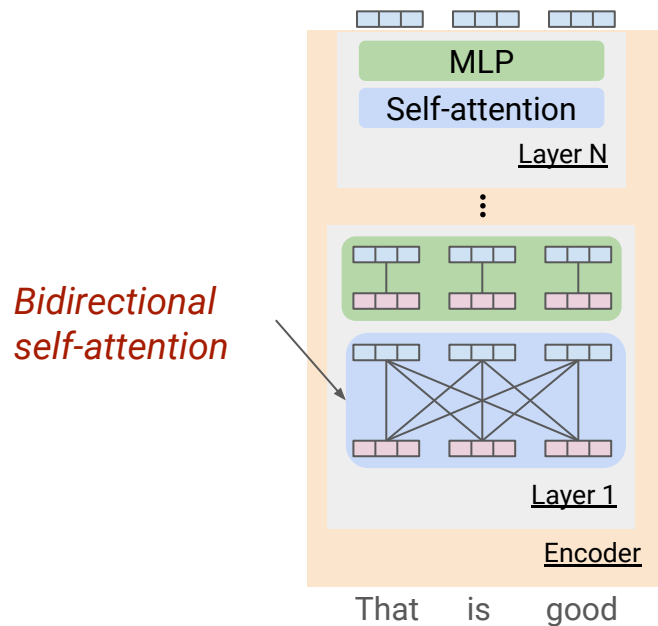
Causal self-attention

Encoder-decoder

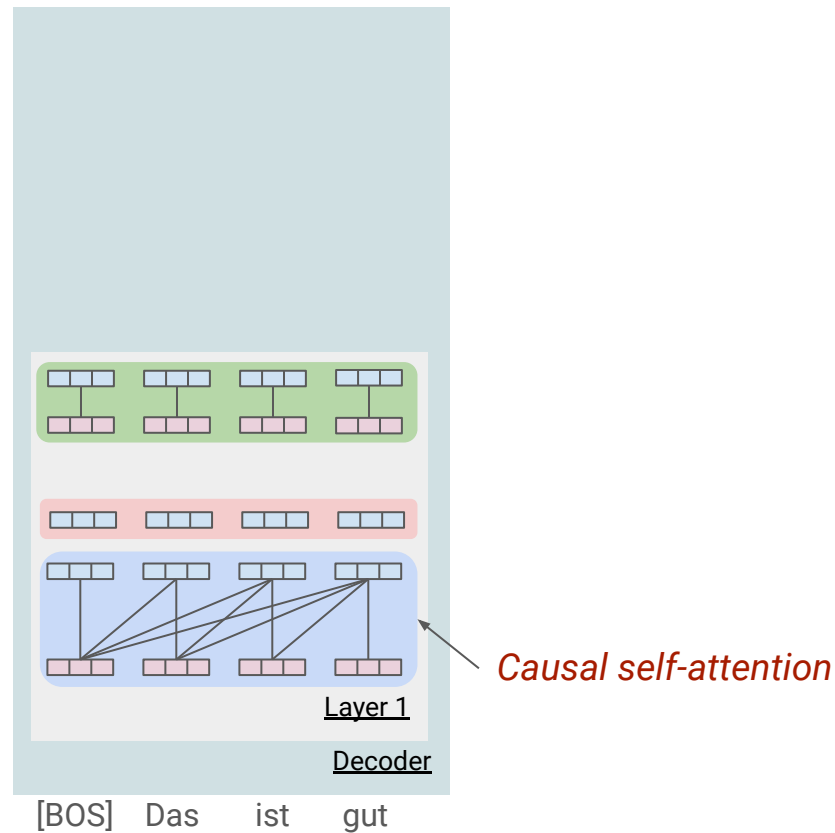
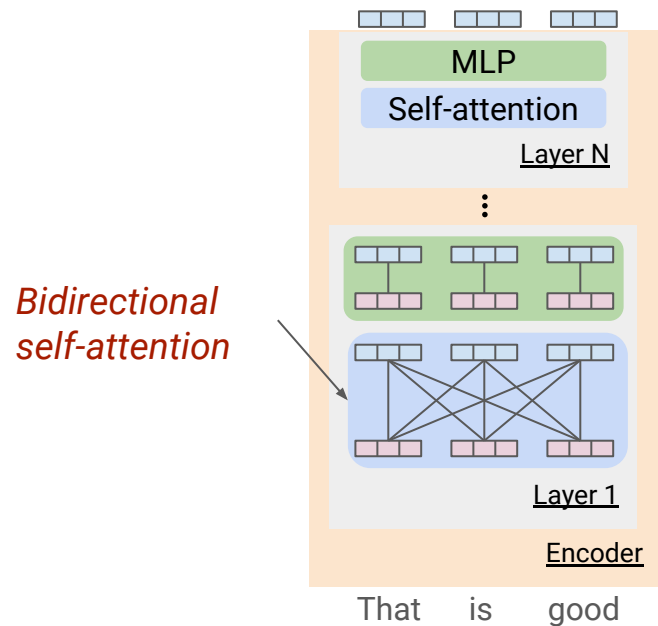
*Bidirectional
self-attention*



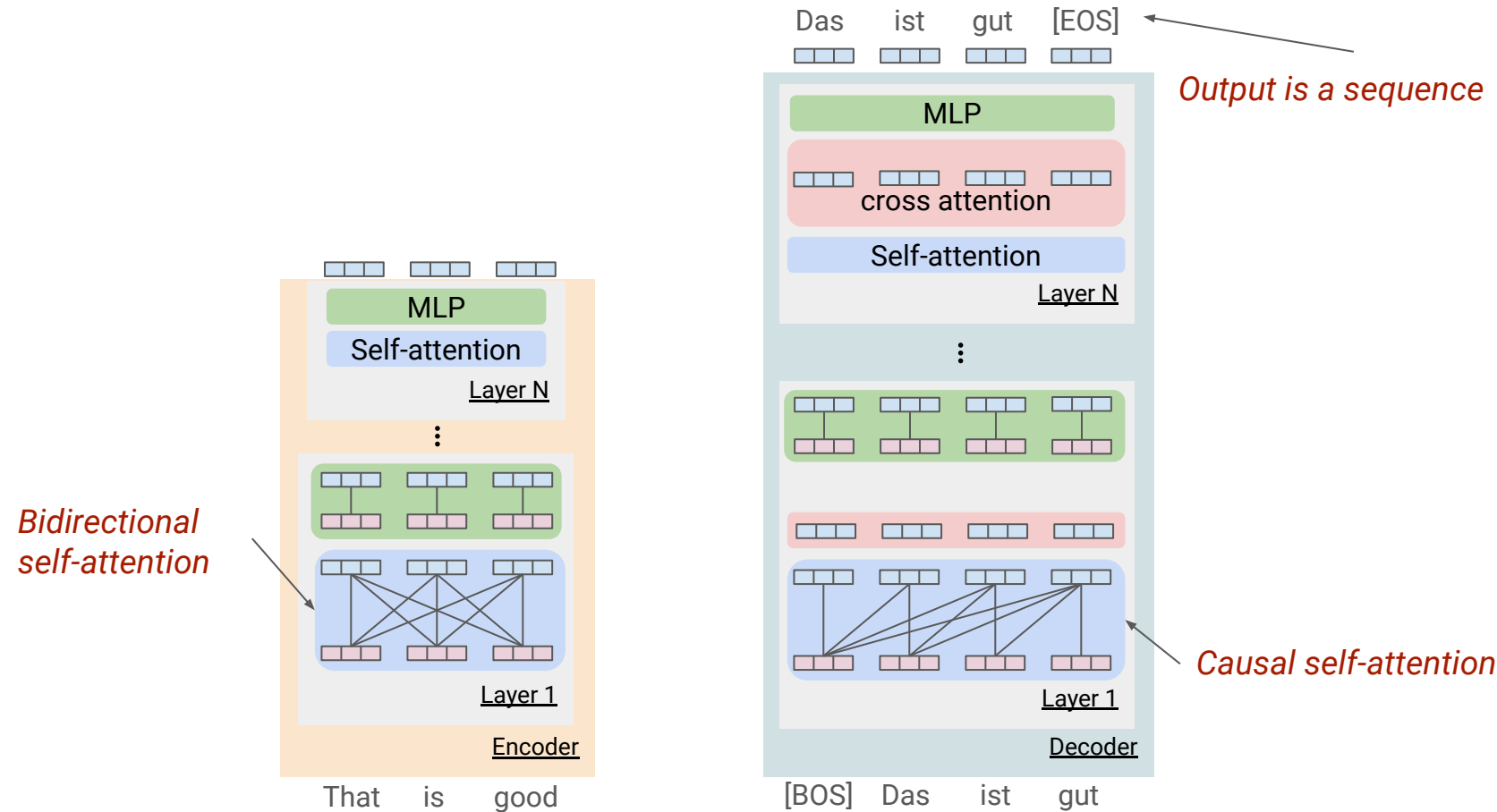
Encoder-decoder



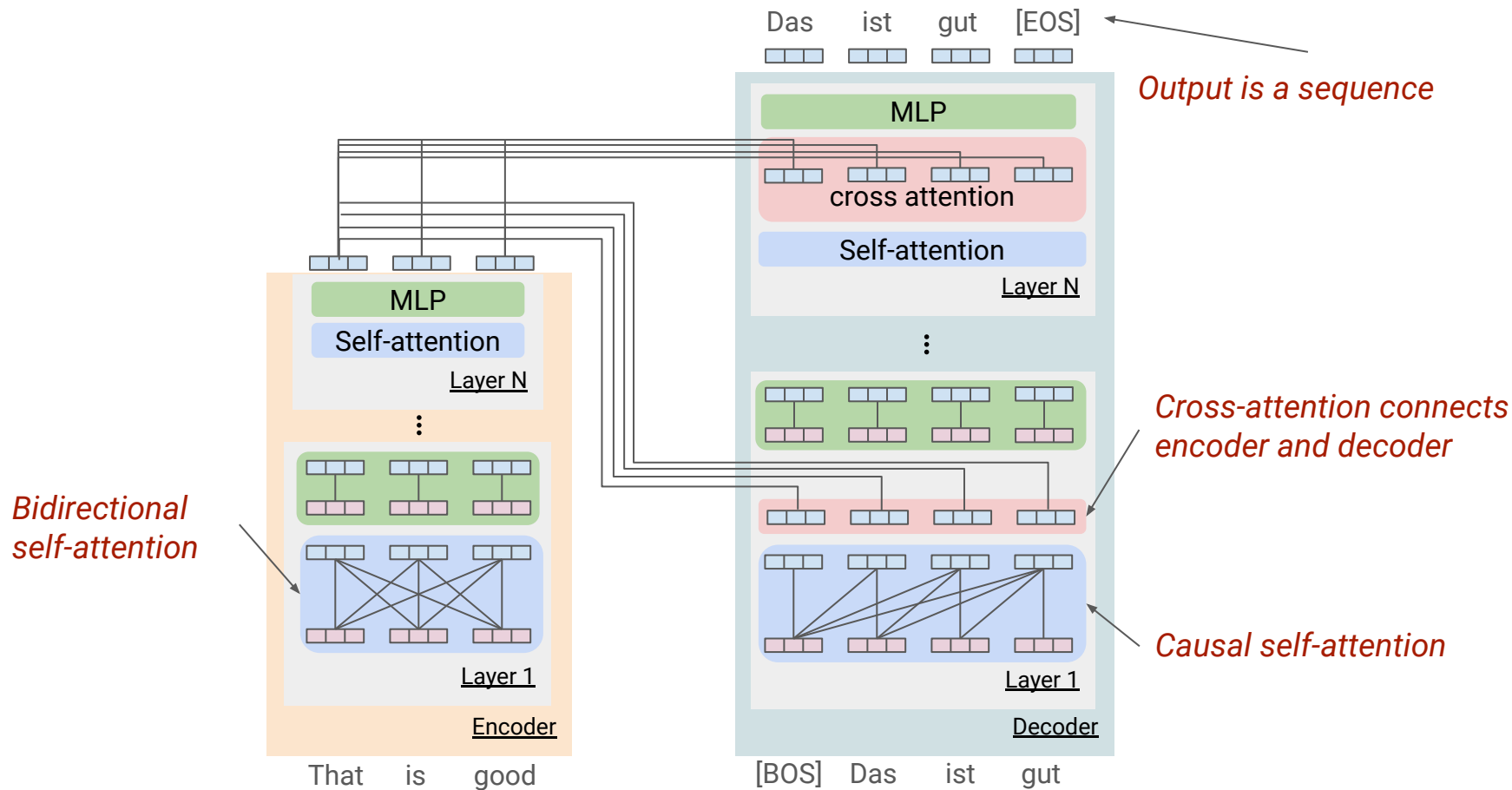
Encoder-decoder



Encoder-decoder



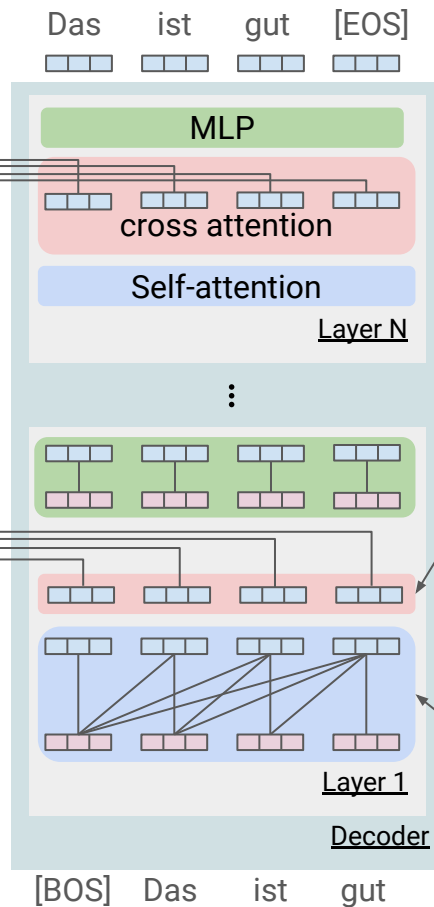
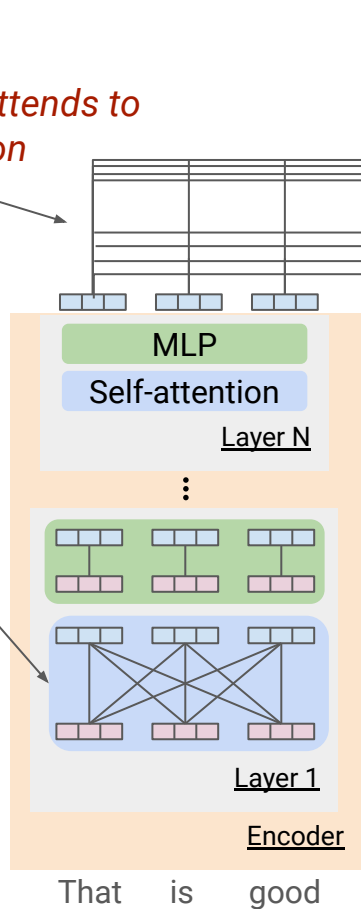
Encoder-decoder



Encoder-decoder

Cross attention only attends to the final layer activation

Bidirectional self-attention

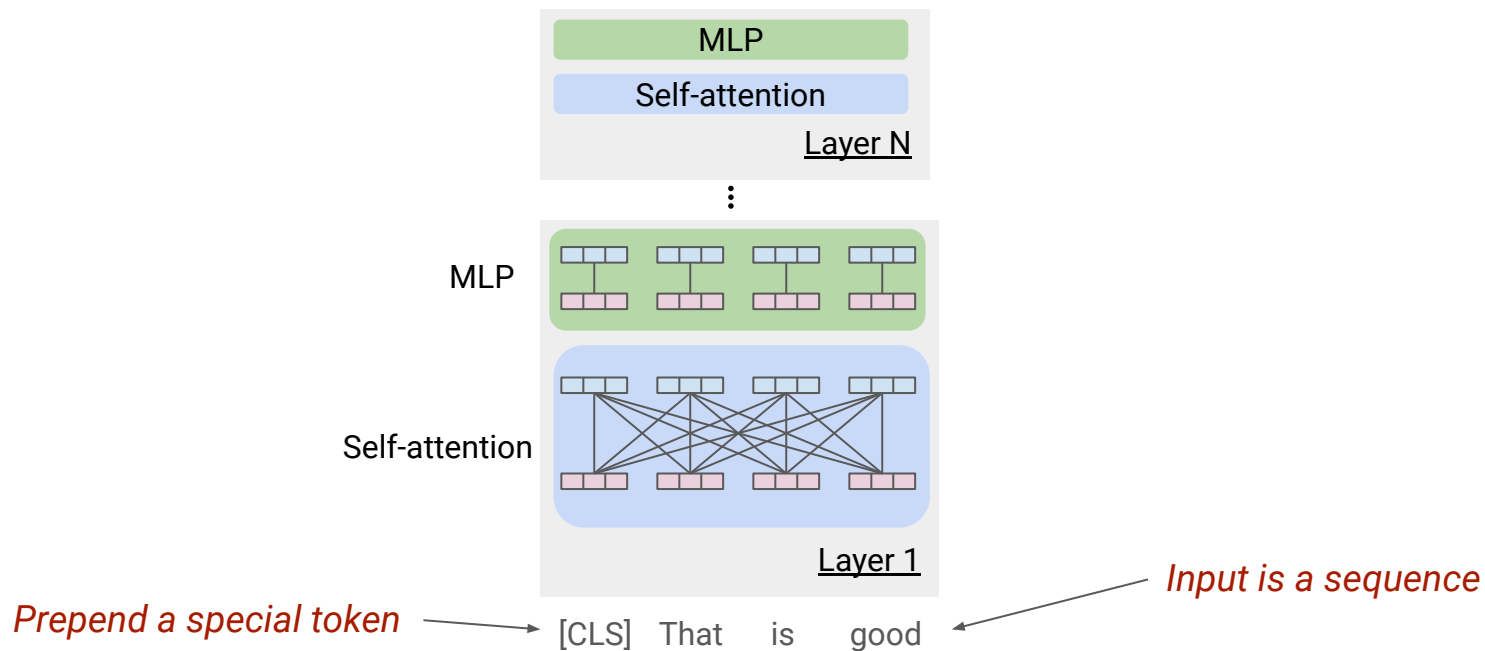


Output is a sequence

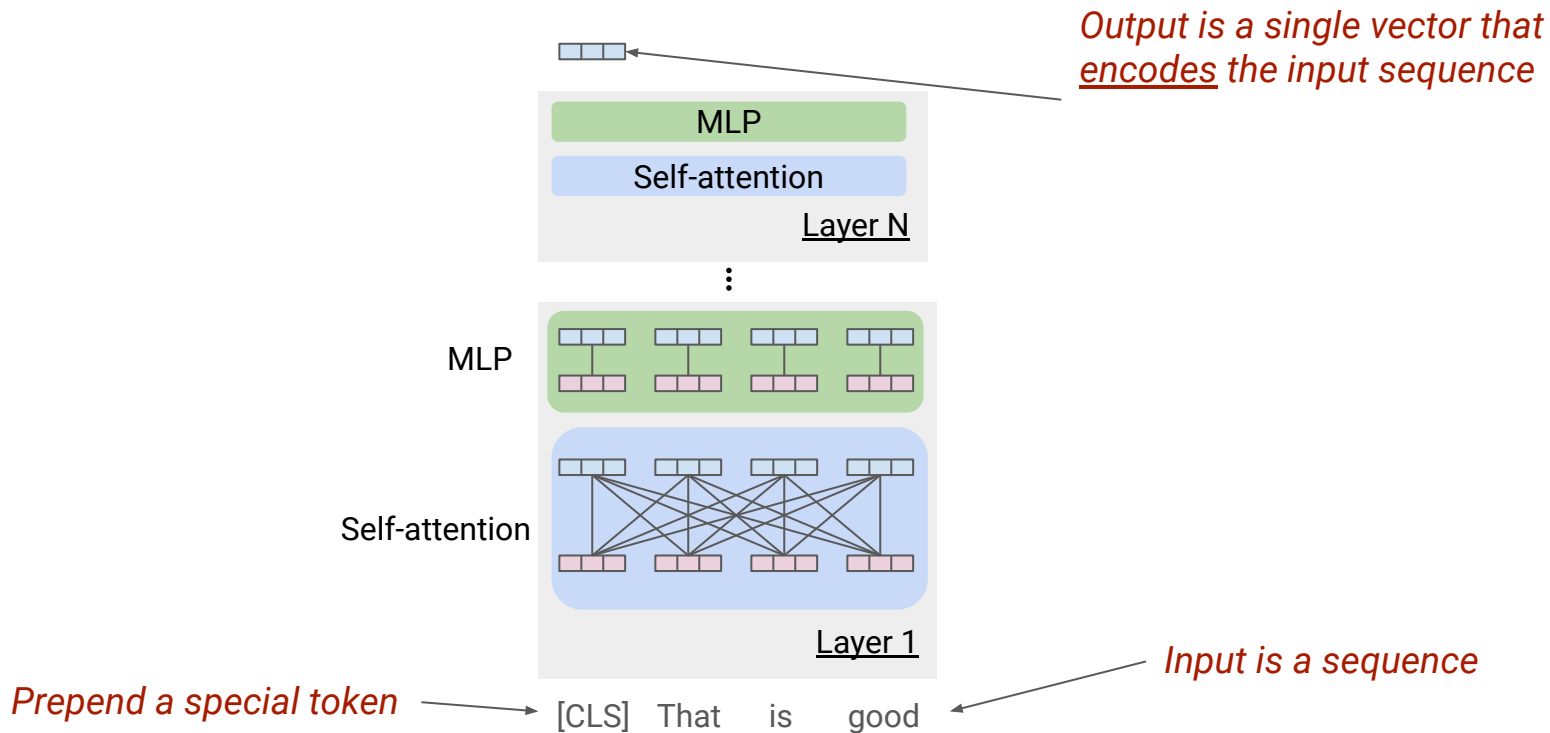
Cross-attention connects encoder and decoder

Causal self-attention

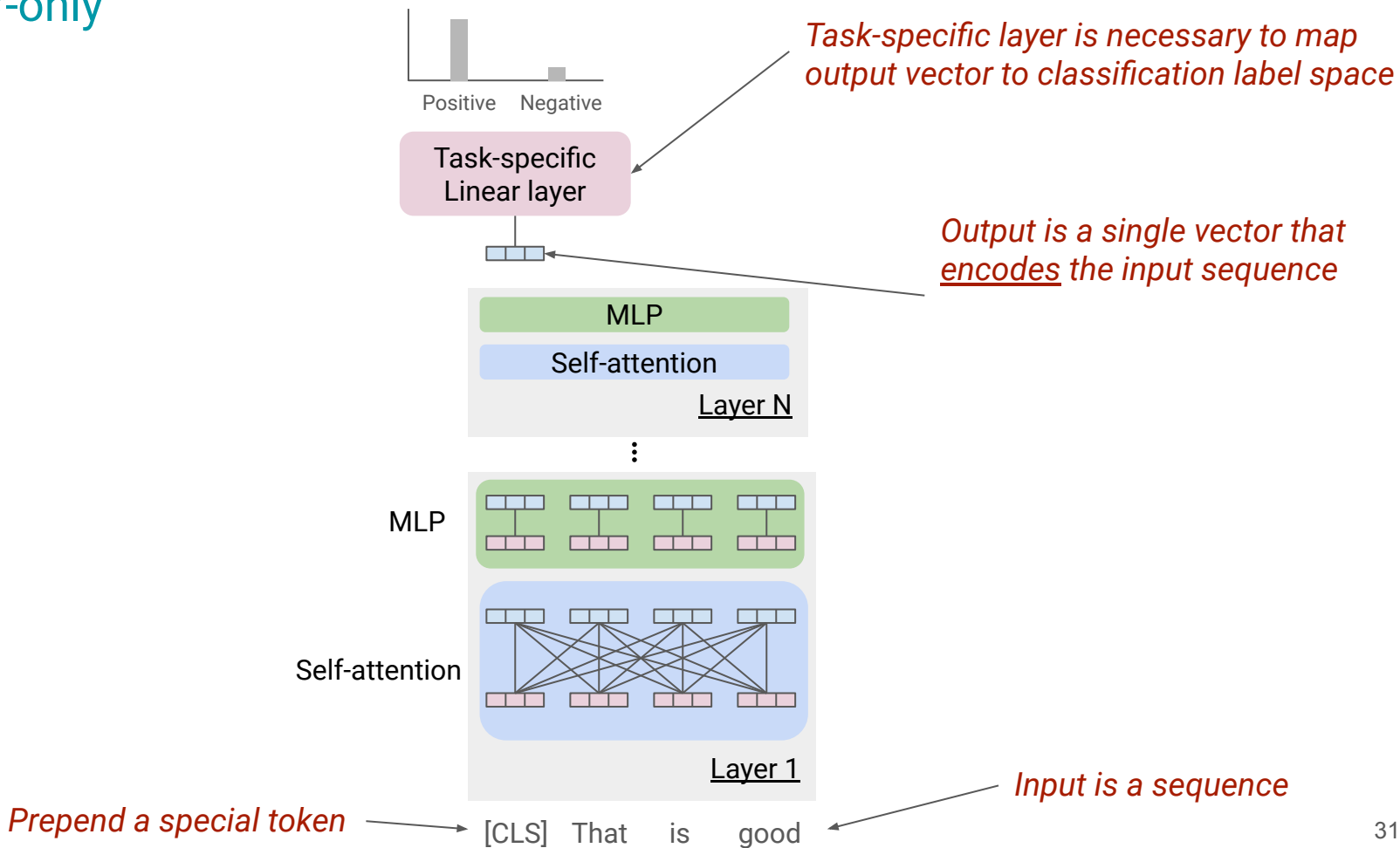
Encoder-only



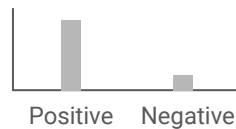
Encoder-only



Encoder-only



Encoder-only



Task-specific layer is necessary to map output vector to classification label space

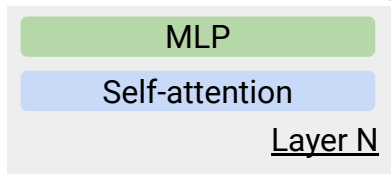
Task-specific
Linear layer



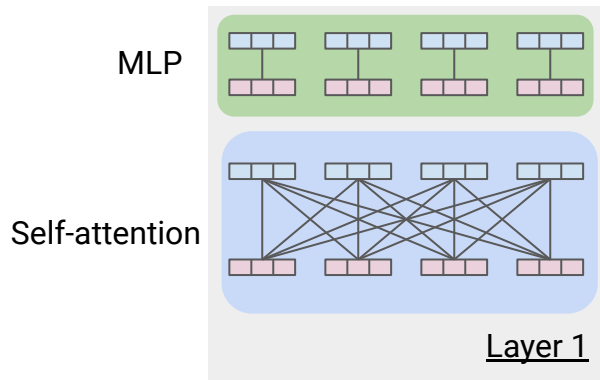
Output is a single vector that encodes the input sequence

Can't generate a sequence!!

Deal breaker for general use case



⋮

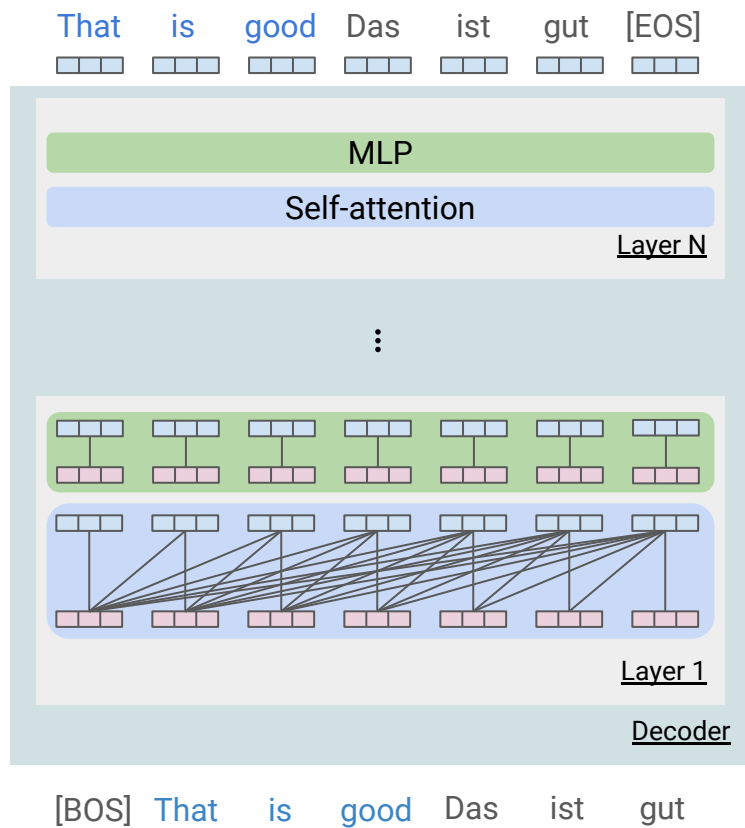


Prepend a special token

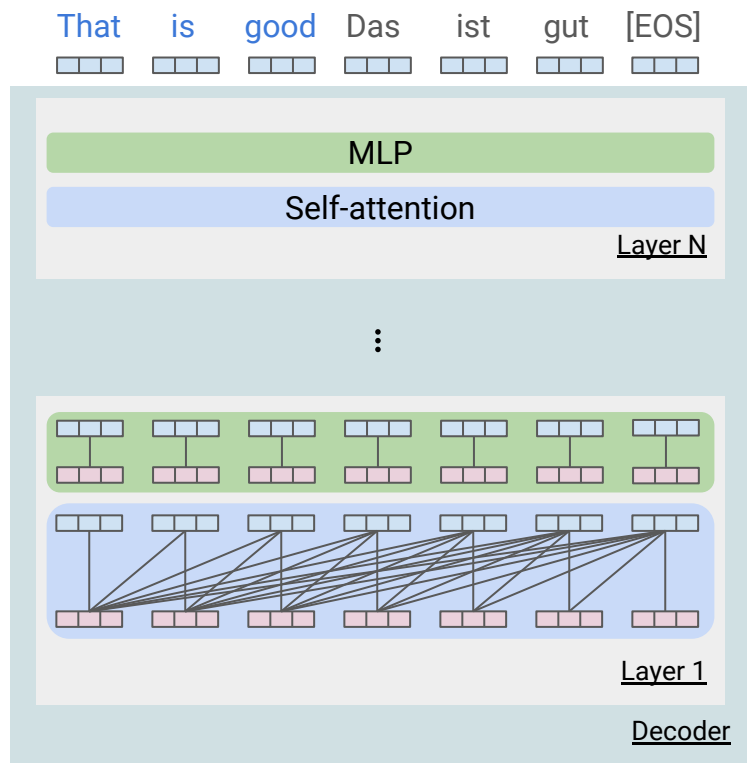
[CLS] That is good

Input is a sequence

Decoder-only



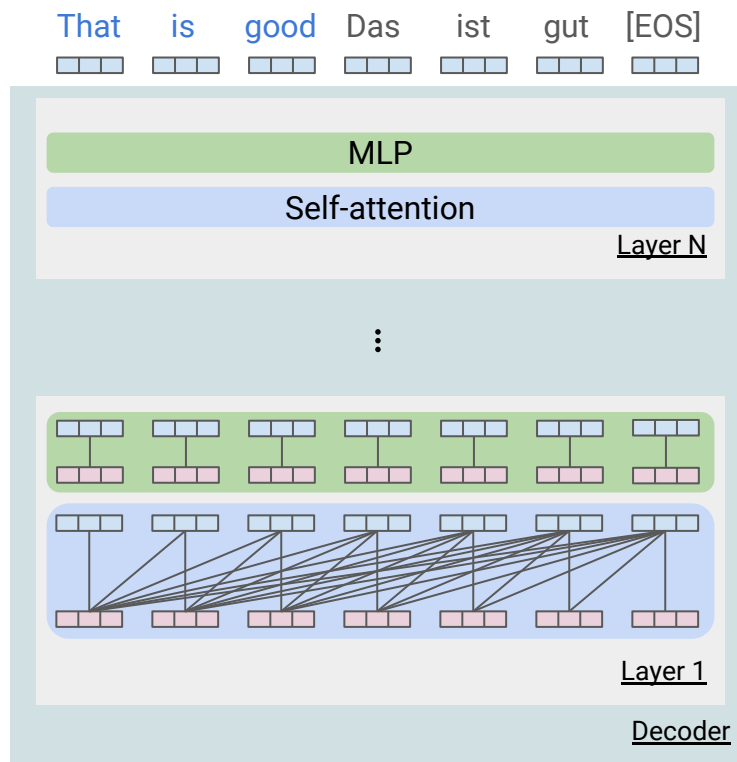
Decoder-only



Input and target are concatenated

[BOS] That is good Das ist gut

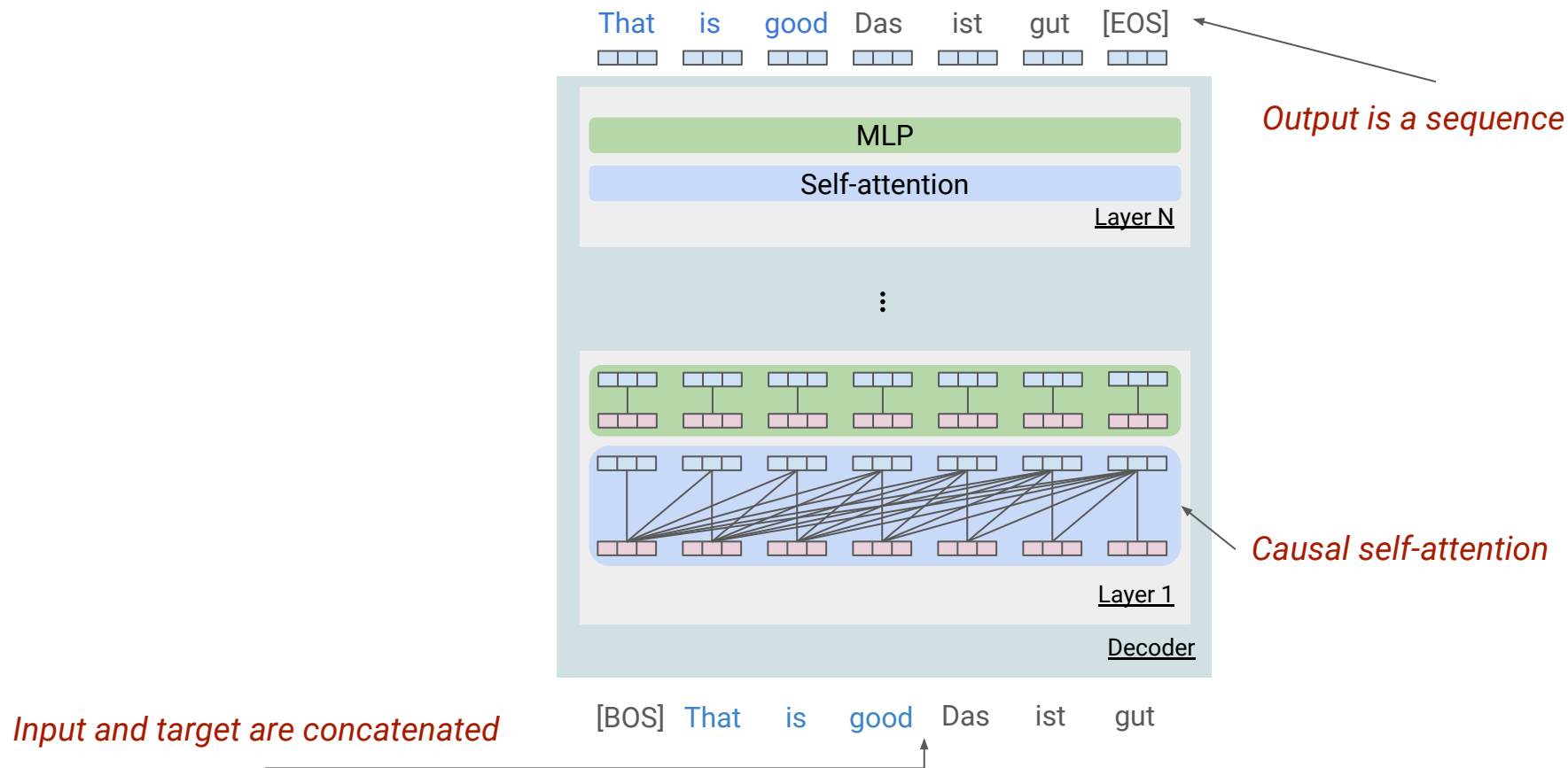
Decoder-only



Input and target are concatenated

[BOS] That is good Das ist gut

Decoder-only



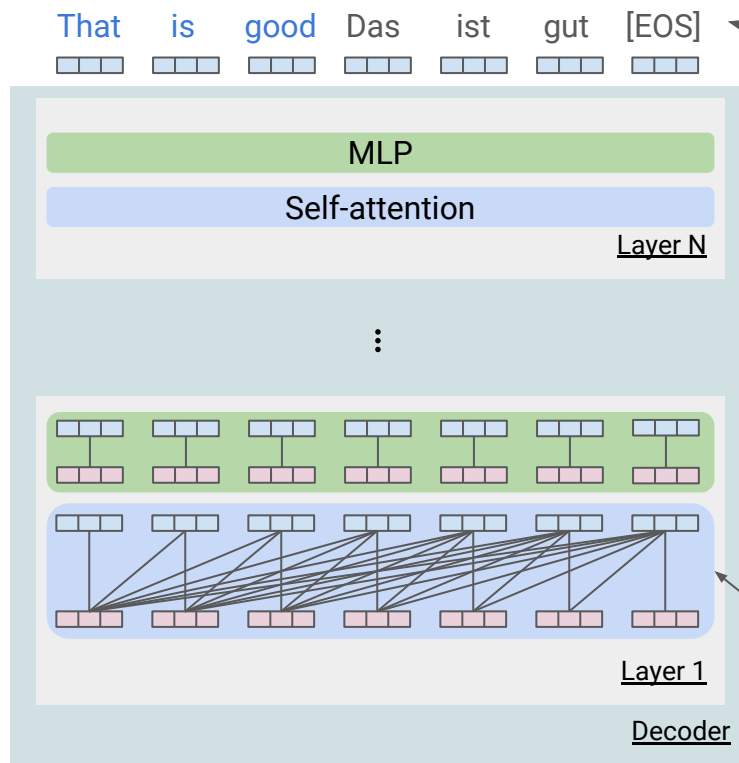
Decoder-only

Key design features

Self attention also serves the role of cross-attention

Same set of parameters apply to both input and target sequences

Input and target are concatenated



Output is a sequence

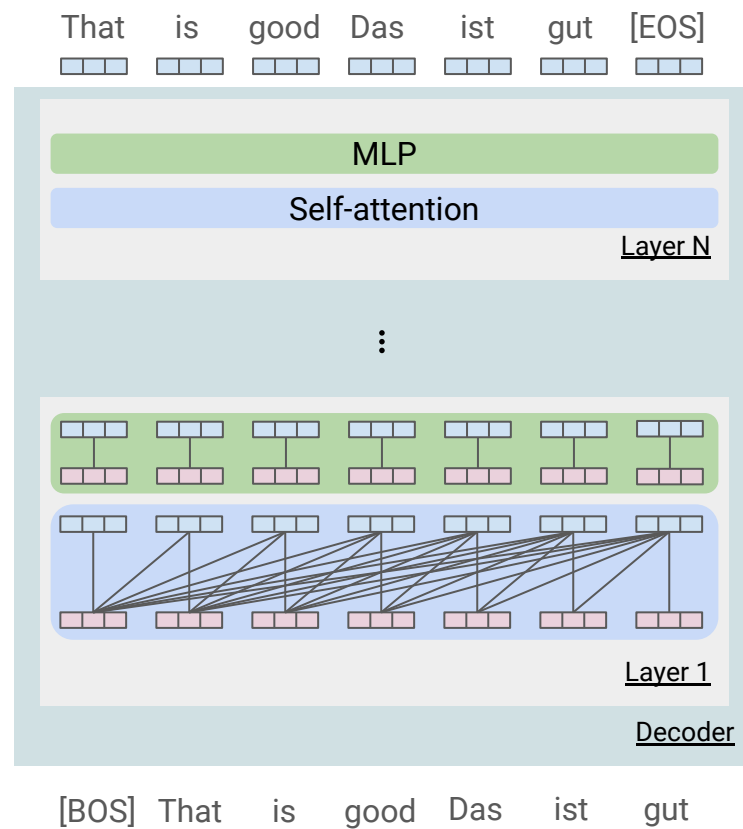
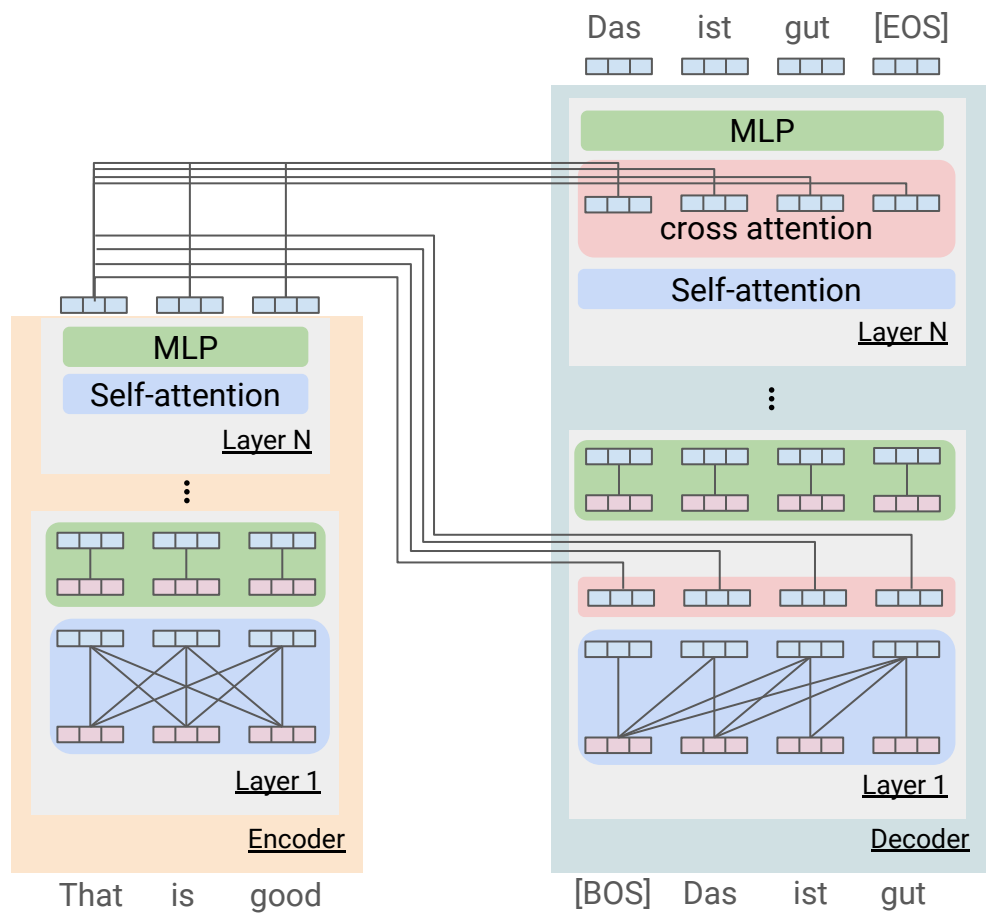
Causal self-attention

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

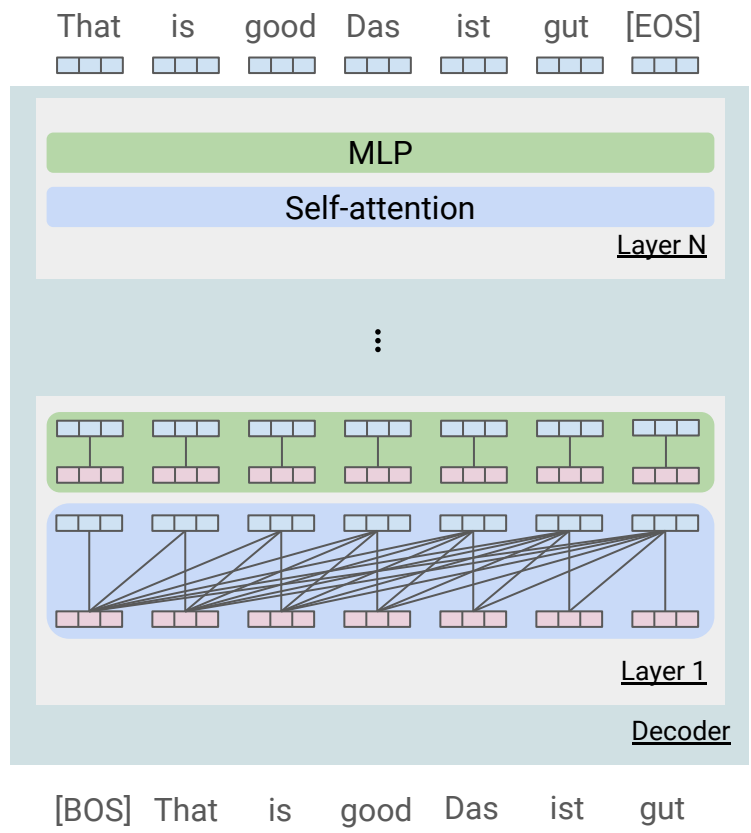
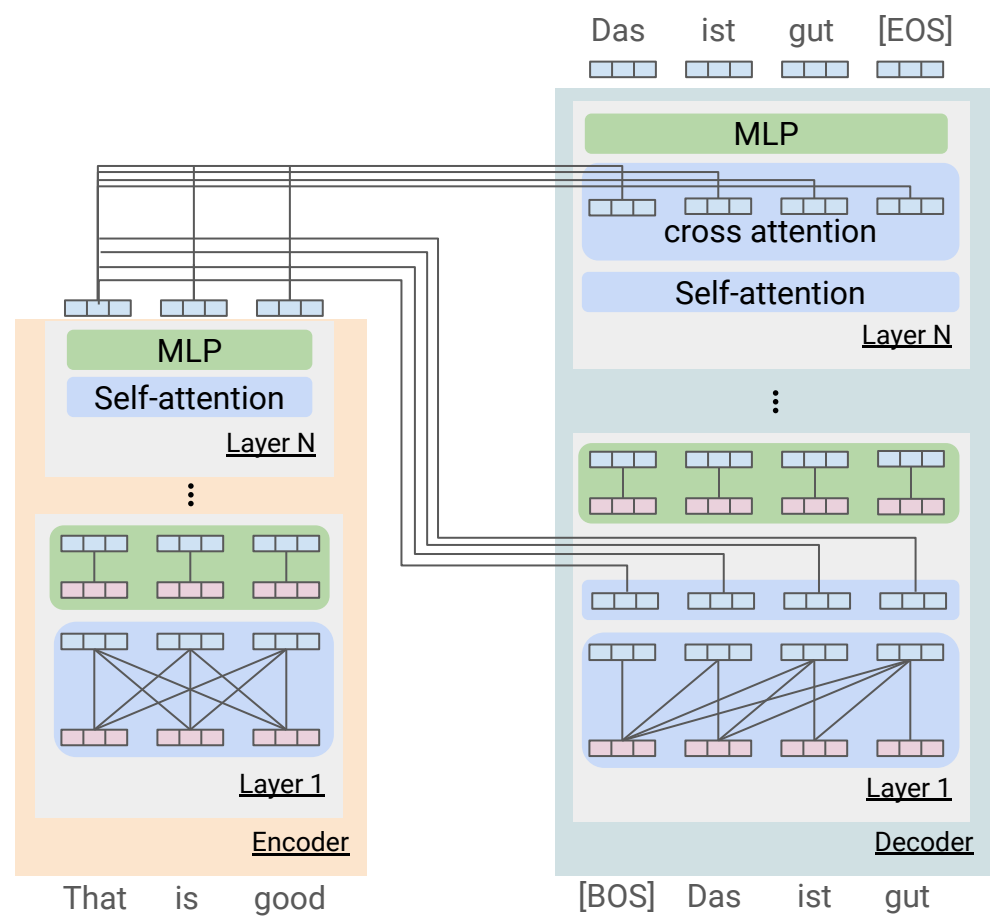
Let's try to transform encoder-decoder into decoder-only

Summary of the differences

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

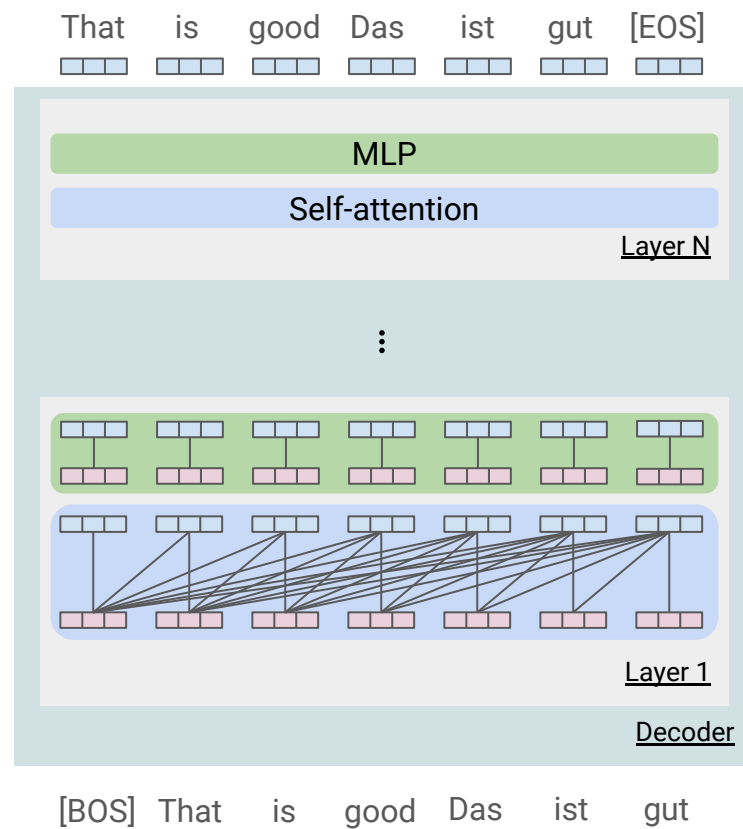
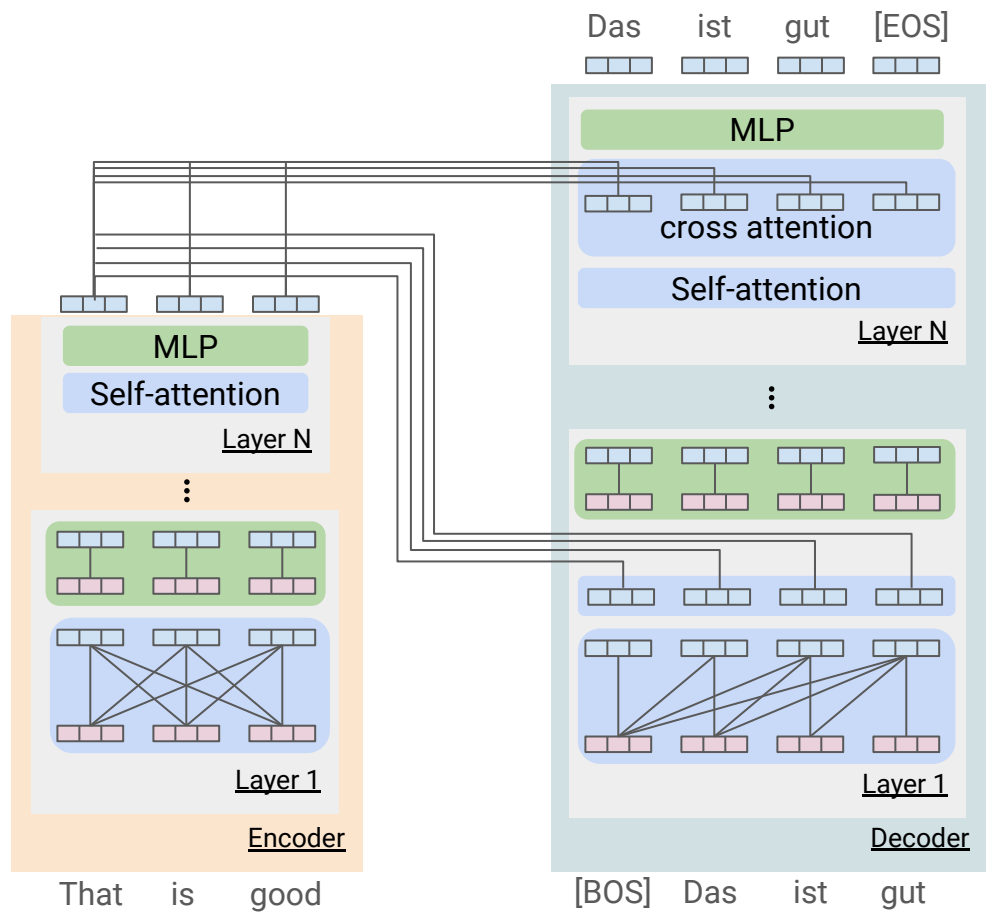


1. Share cross and self-attention parameters

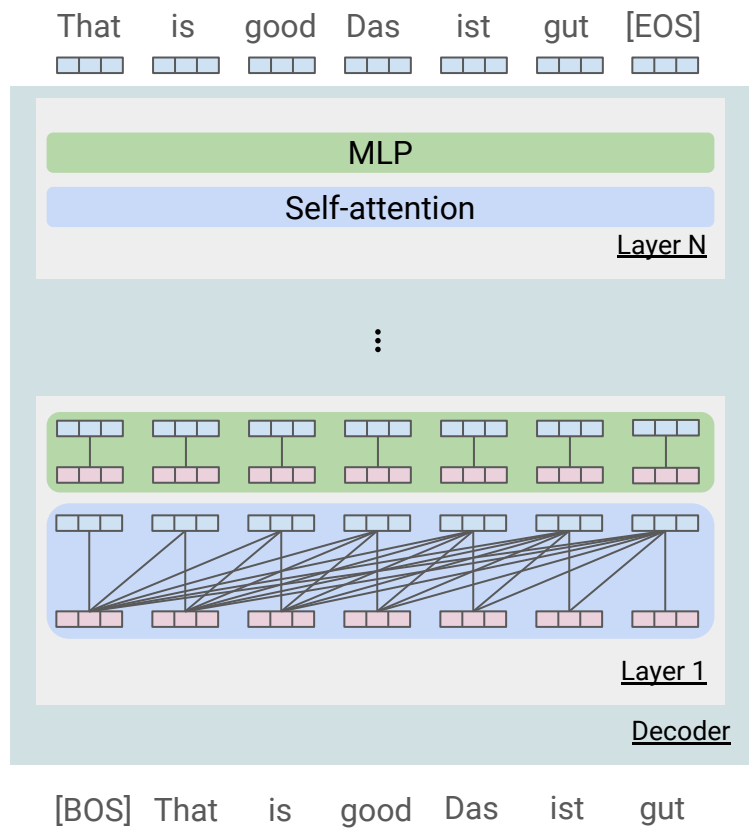
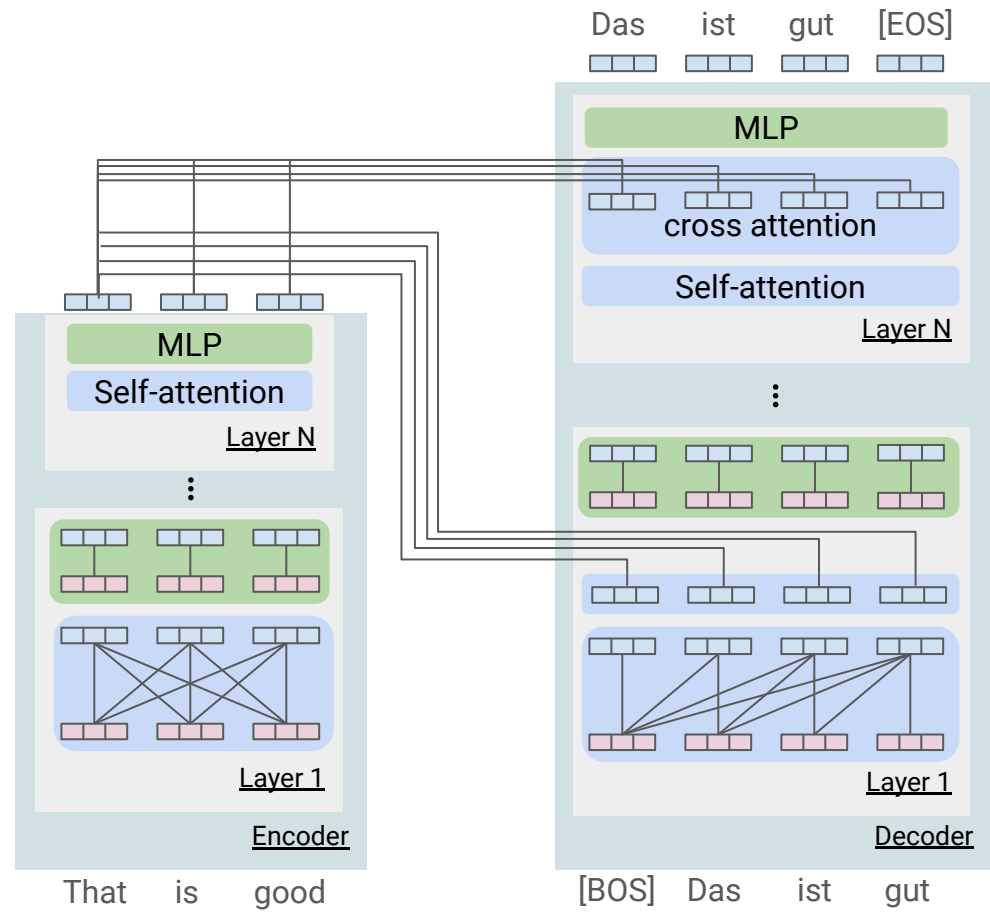


Summary of the differences

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

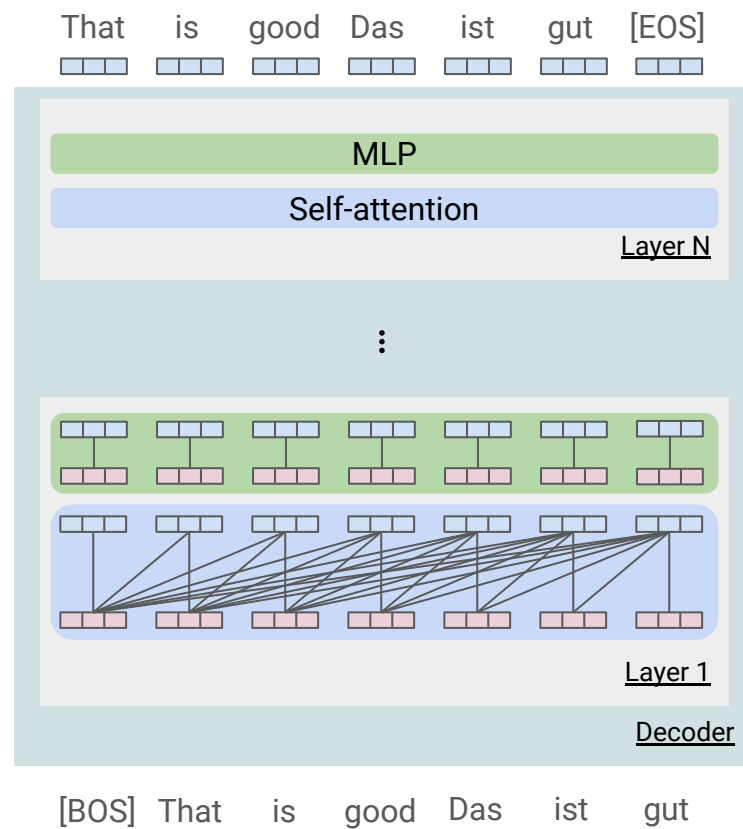
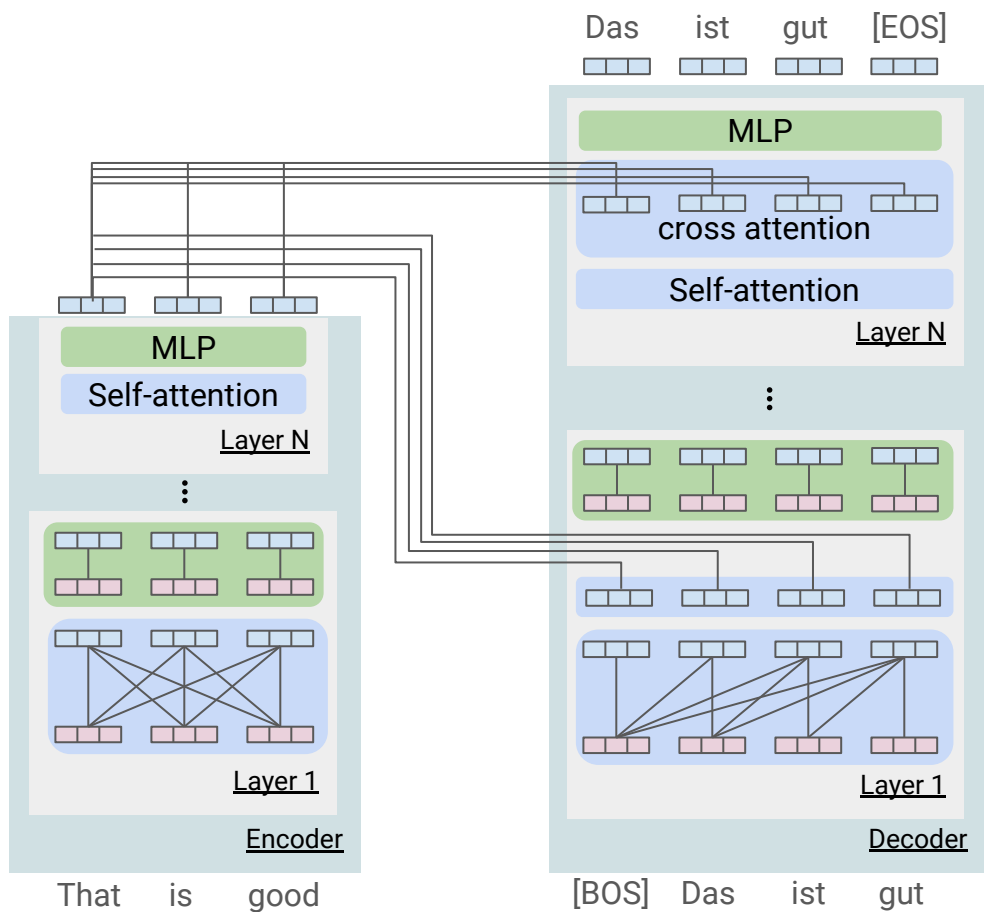


2. Share encoder and decoder parameters

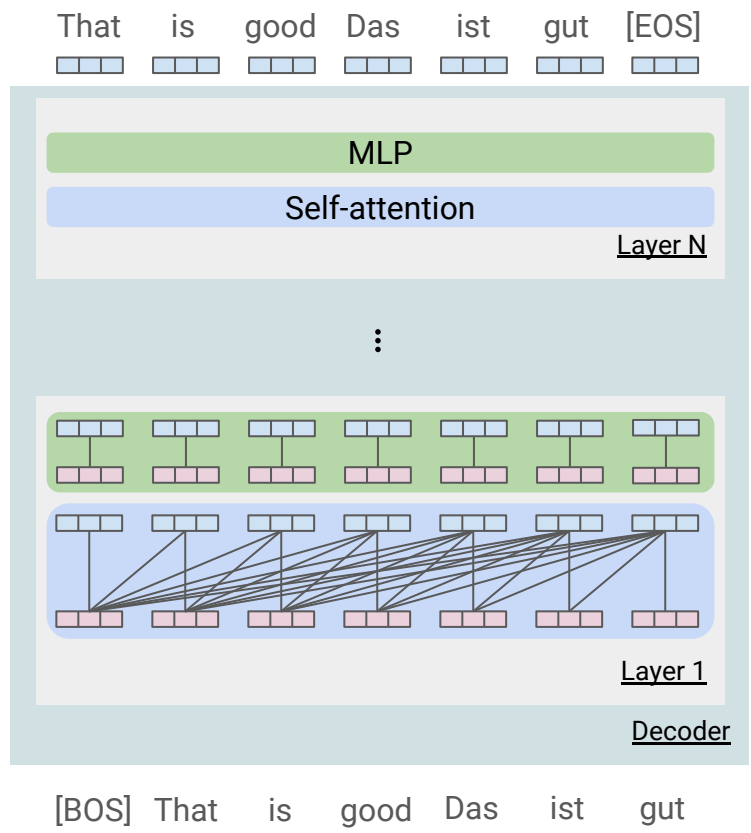
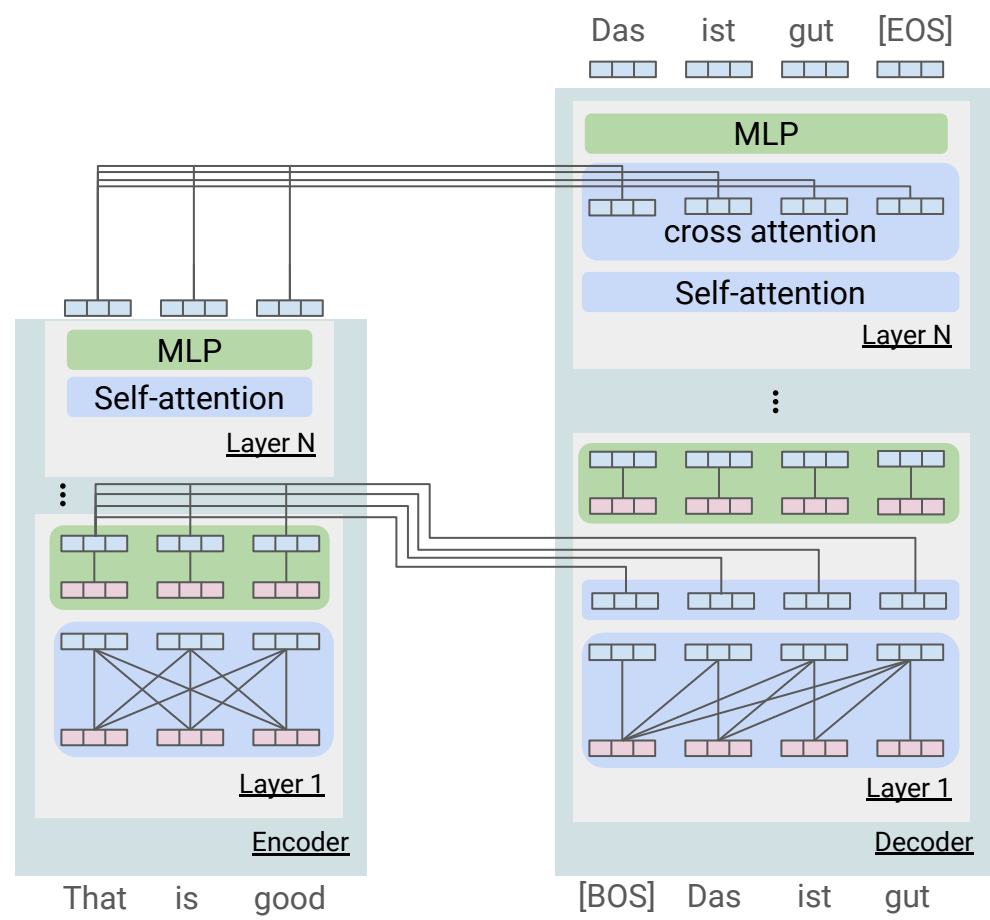


Summary of the differences

	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		



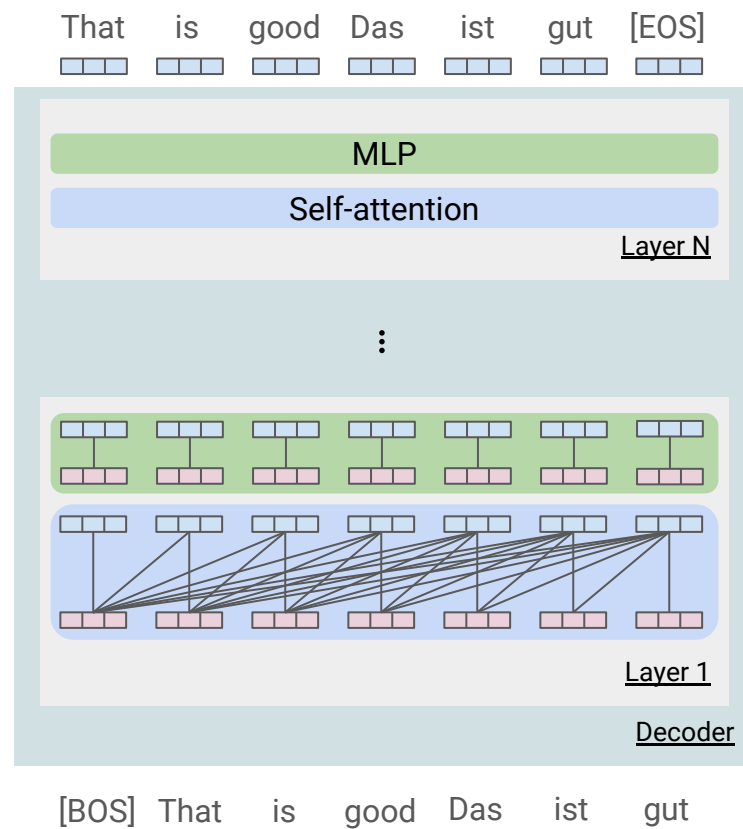
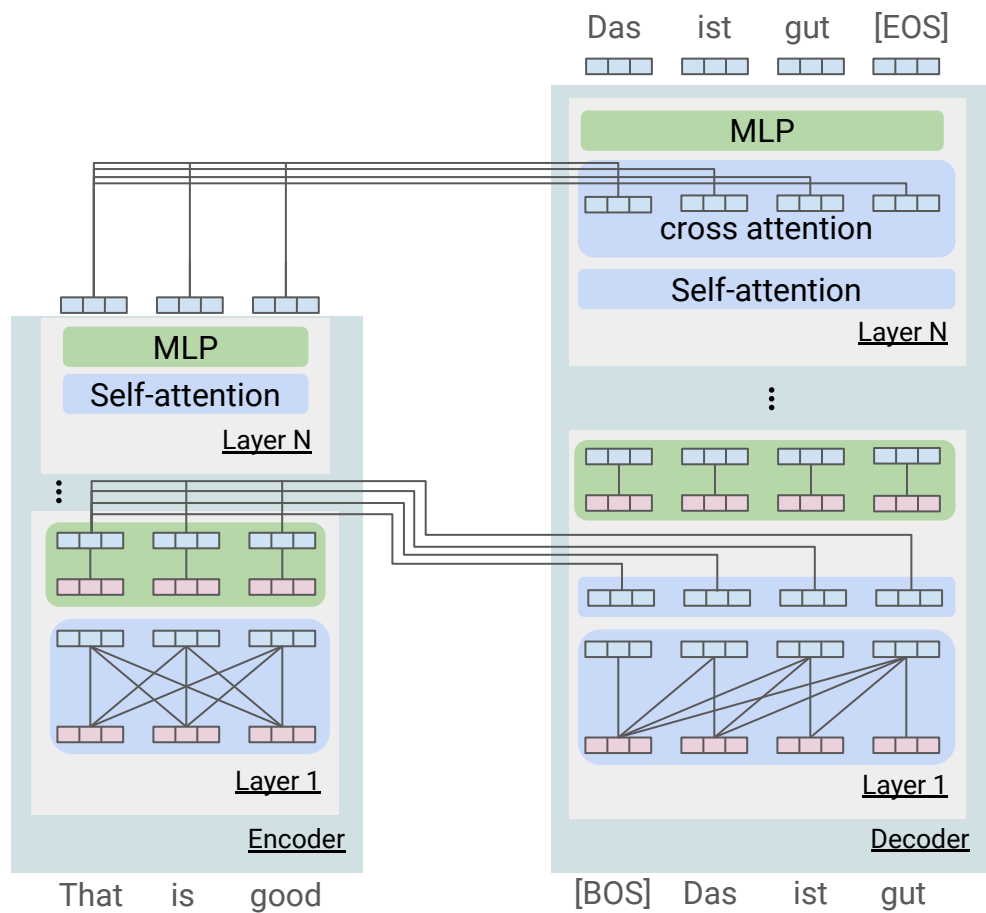
3. Decoder layer 1 attends to encoder layer 1



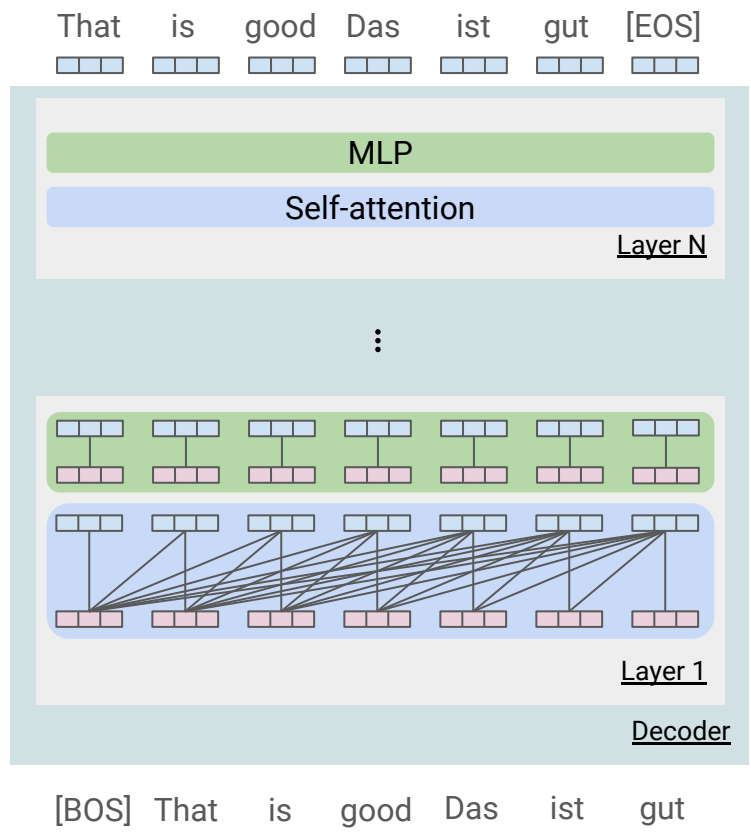
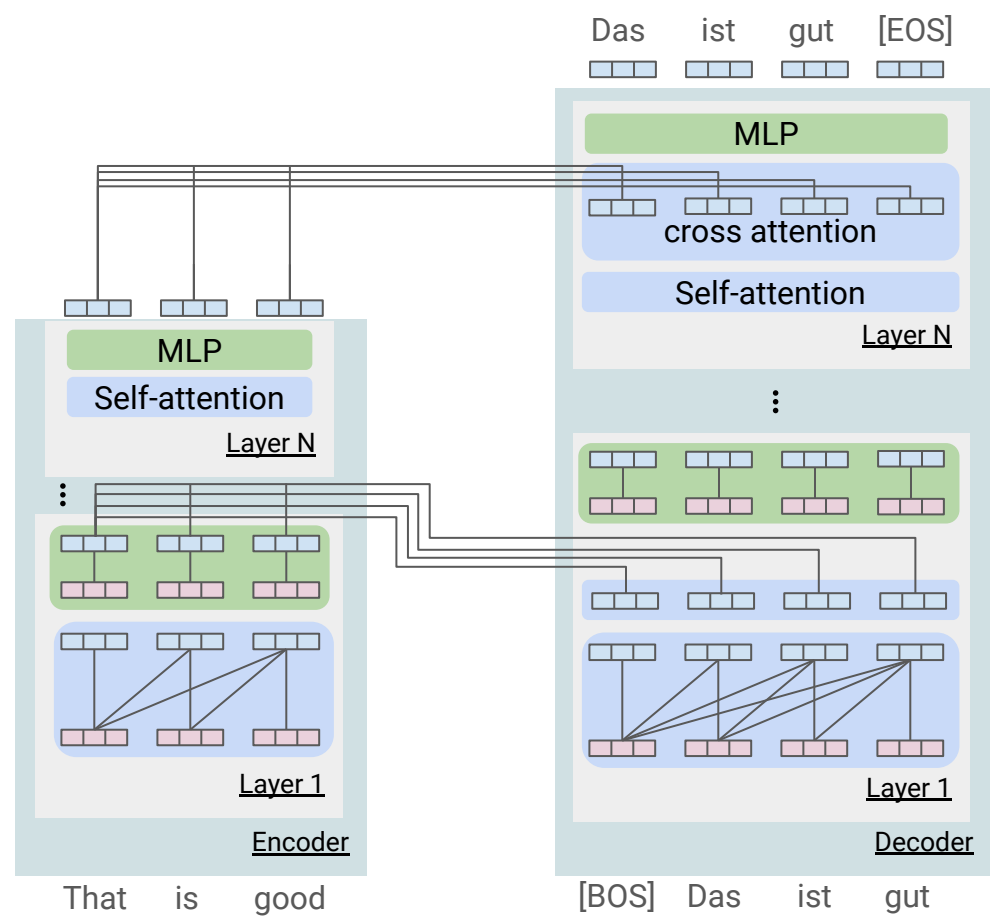
Summary of the differences

	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 1 attends to layer 1)
Input attention		

* input attention can be bidirectional



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 1 attends to layer 1)
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
2. The target element can attend to the fully encoded representation of the input
3. When encoding the input sequence, all-to-all interaction among the sequence elements is preferred

Additional structures in encoder-decoder compared to decoder-only

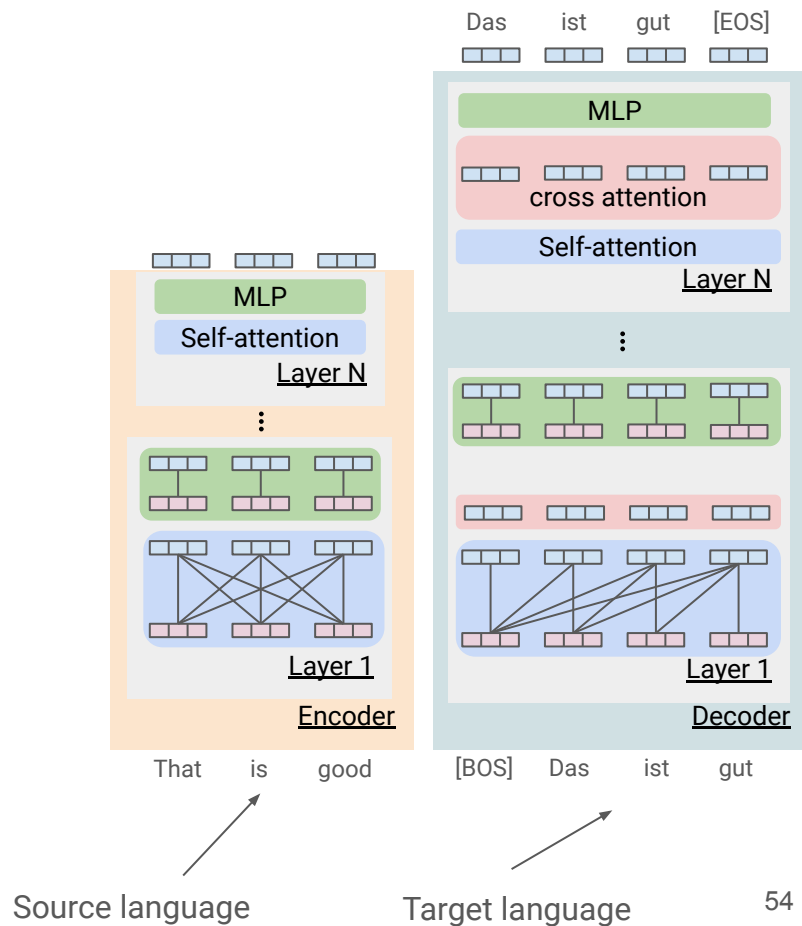
1. Input and target sequences are sufficiently different that using separate parameters can be effective
2. The target element can attend to the fully encoded representation of the input
3. When encoding the input sequence, all-to-all interaction among the sequence elements is preferred

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

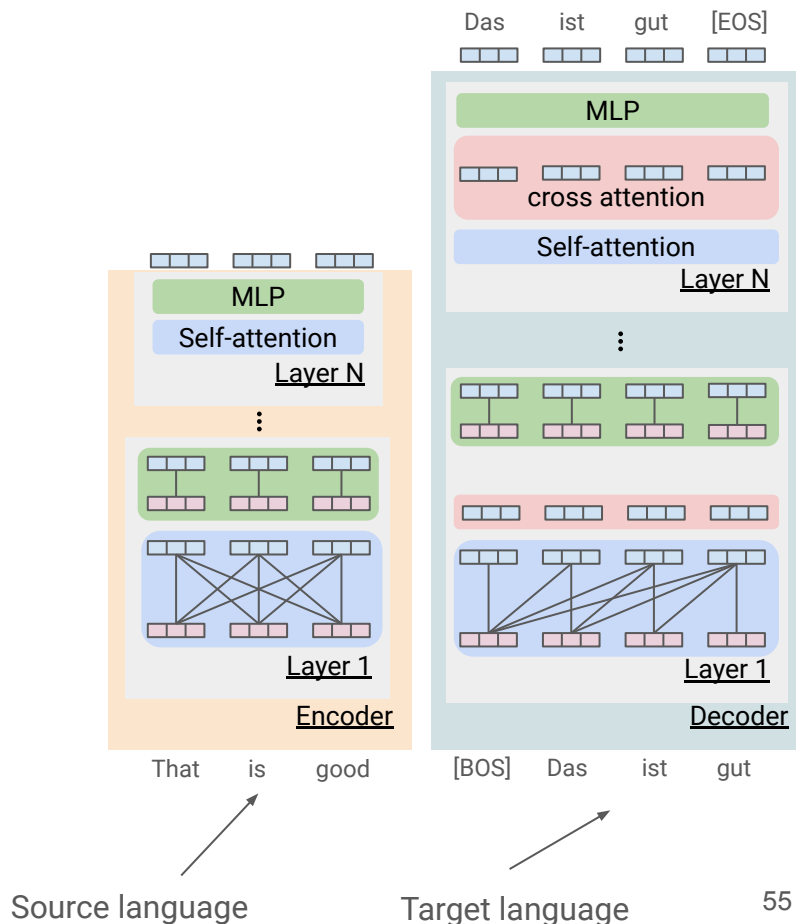
When Transformer was introduced in 2017, translation was a popular and difficult task

Input and target are in different languages

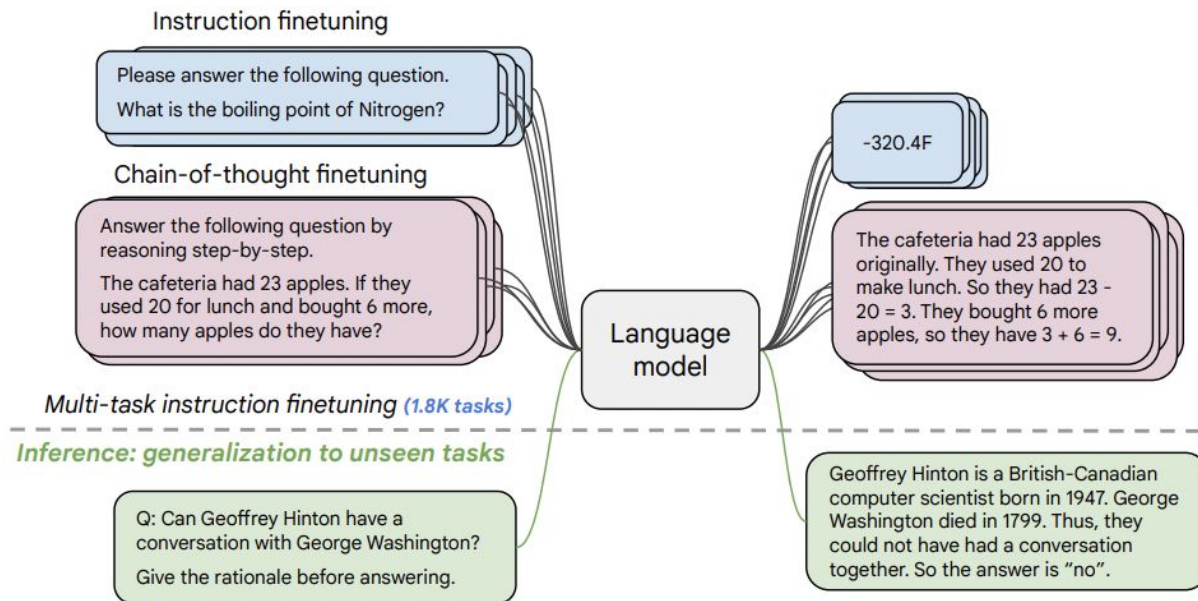
If the only goal is to learn translation, separate parameters make sense

Modern language models learn the world knowledge

Separate parameters for knowledge just expressed in different languages? 🤔



Example 2: instruction finetuning with academic datasets



Example 2: instruction finetuning with academic datasets

Encoder-decoder

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)

Encoder-decoder models had much bigger gain!

Decoder-only

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)

Example 2: instruction finetuning with academic datasets

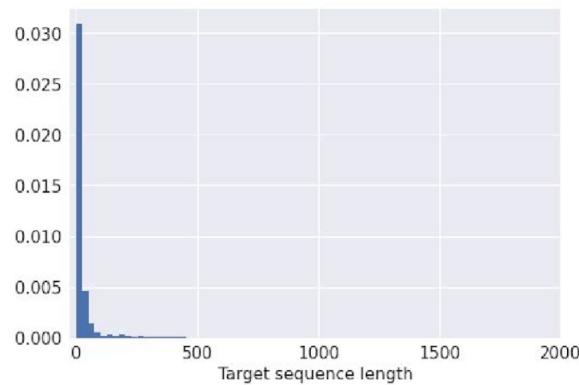
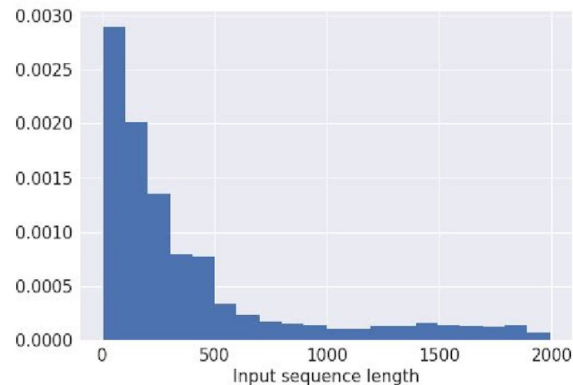
Academic datasets have distinctive length distribution: long input and short target

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

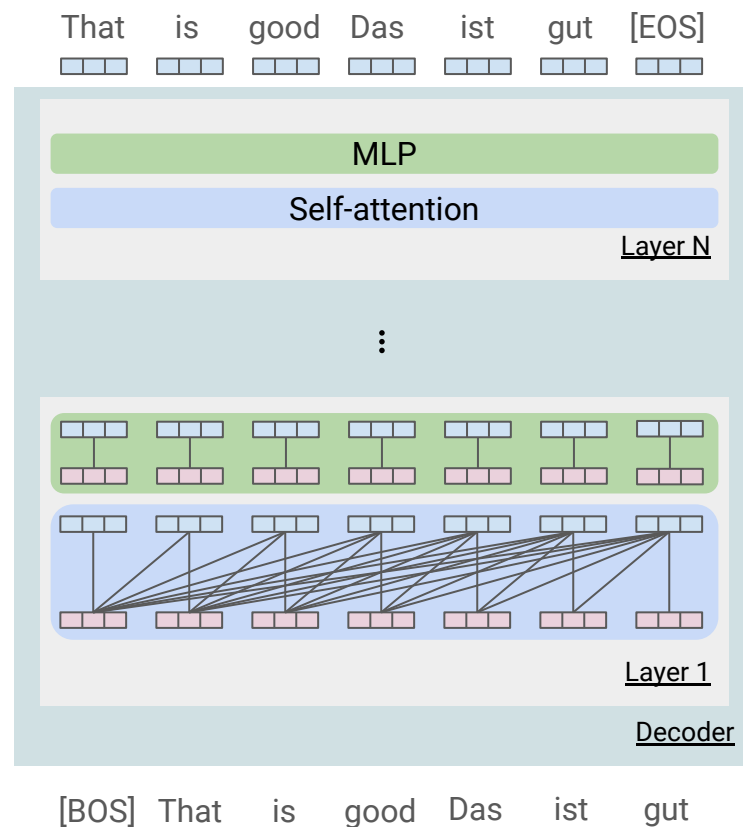
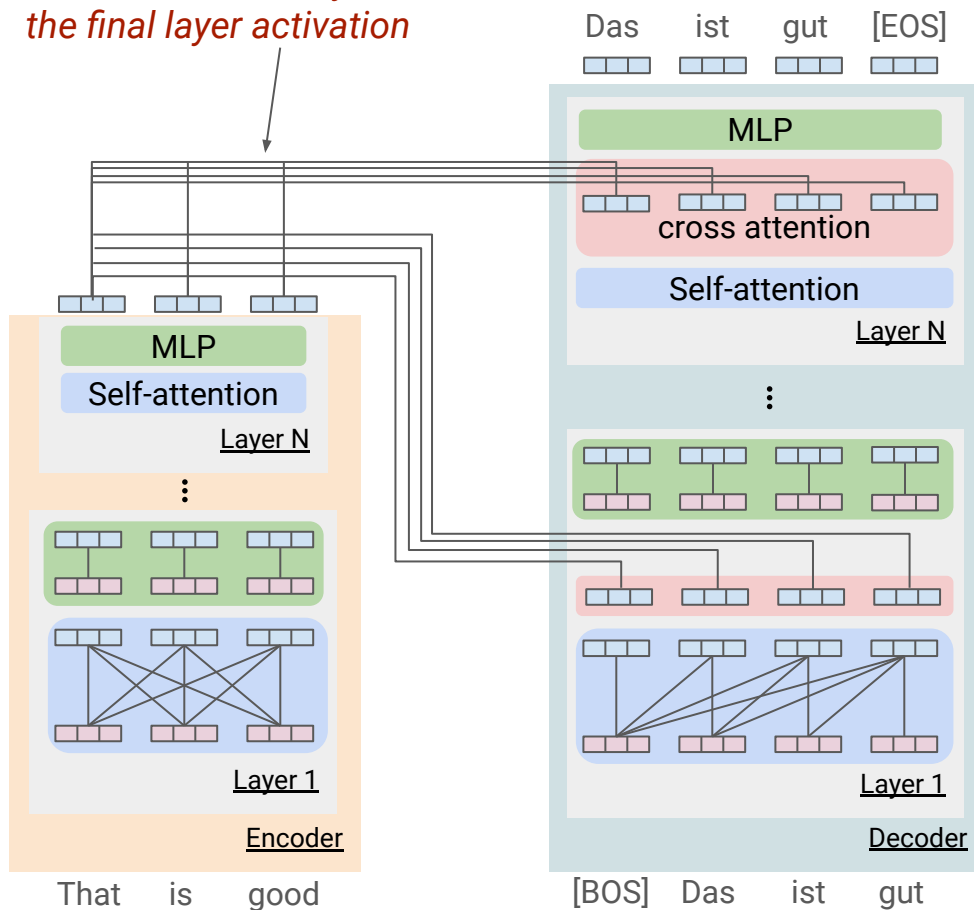


Length distribution of finetuning datasets

Additional structures in encoder-decoder compared to decoder-only

1. Input and target sequences are sufficiently different that using separate parameters can be effective
2. The target element can attend to the fully encoded representation of the input
3. When encoding the input sequence, all-to-all interaction among the sequence elements is preferred

Cross attention only attends to the final layer activation



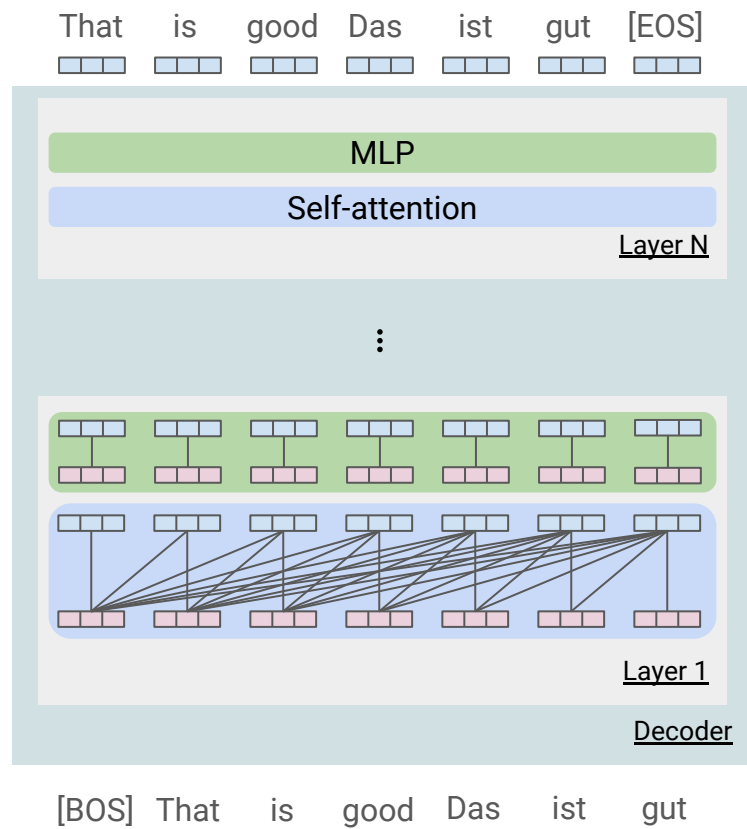
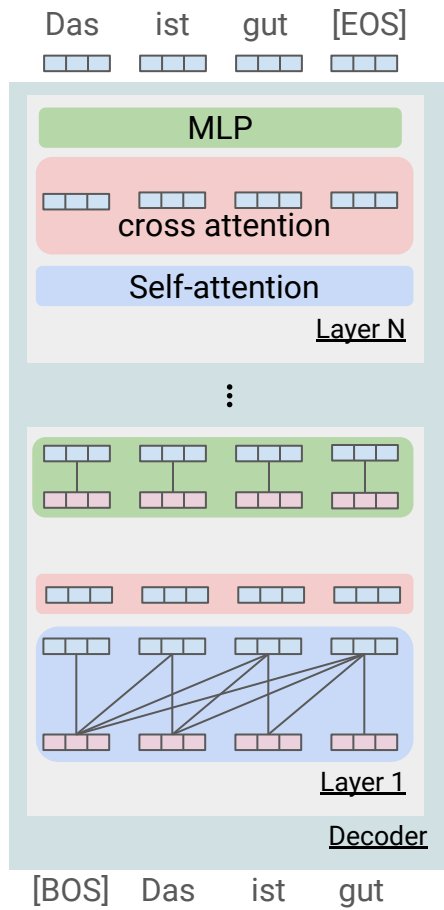
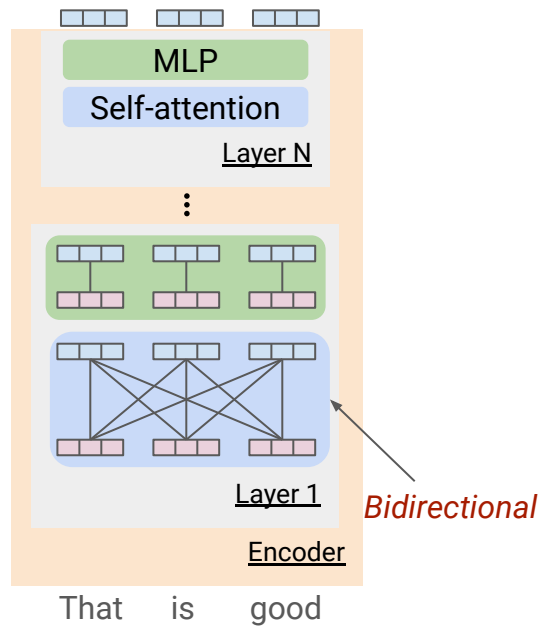
In deep neural networks, the bottom and top layers encode information at a different level of granularity

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
2. The target element can attend to the fully encoded representation of the input
3. When encoding the input sequence, all-to-all interaction among the sequence elements is preferred



(Highly anecdotal) At sufficient scale, bidirectionality doesn't seem to matter much

Bidirectionality brings in engineering challenges for multi-turn chat application

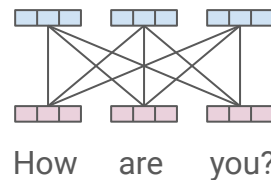
At every turn, the new input has to be encoded again; for unidirectional attention, only the newly added message needs to be encoded.

Input attention pattern for multi-turn conversation

Bidirectional

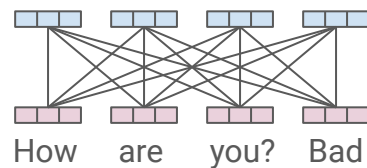
USER

How are you?



ASSISTANT

Bad



USER

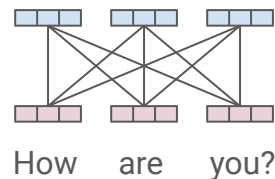
Why?

Input attention pattern for multi-turn conversation

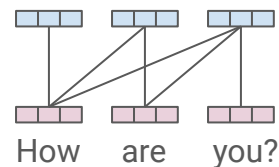
USER

How are you?

Bidirectional

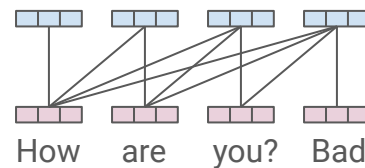
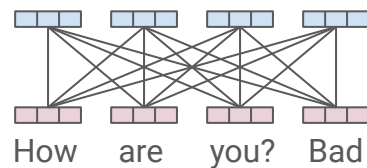


Unidirectional



ASSISTANT

Bad



USER

Why?

Conclusion

Identify the dominant driving force as exponentially cheaper compute and associated scaling

Analyzed additional structure of encoder-decoder from the perspective of scaling

Hopefully this perspective and analysis can be useful for understanding what is happening today and predict the future trajectory