Lecture 3

CME 295: Transformers & Large Language Models

 **Afshine Amidi & Shervine Amidi** 

Stanford University

ICME

Recap of last episode…

Recap of last episode...

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5

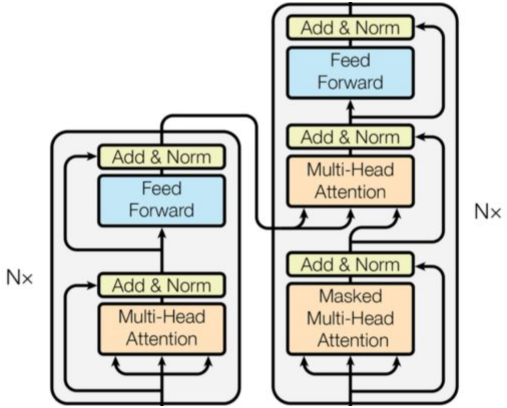


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Recap of last episode...

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa

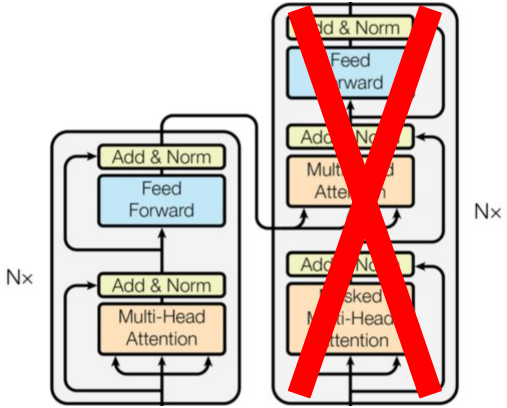


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Recap of last episode...

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series

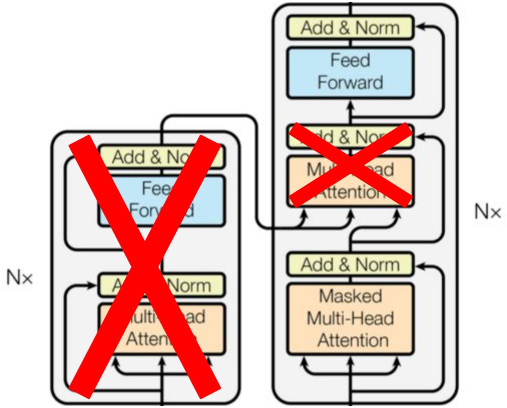


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Recap of last episode...

3 categories of Transformer-based models:

Encoder-decoder	Text to text	T5, mT5, ByT5
Encoder-only	Projection of embedding for class prediction (e.g. sentiment extraction)	BERT, DistilBERT, RoBERTa
Decoder-only	Text to text	GPT series

Transformers & Large Language Models

LLM overview

MoE-based LLMs

Response generation

Prompting strategies

Inference optimizations

Terminology

LLM = **L**arge **L**anguage **M**odel

Terminology

L**L****M** = **L**arge $\boxed{\textbf{L}\text{anguage } \textbf{M}\text{odel}}$

Terminology

LLM = Large $\boxed{\textbf{L}\text{anguage }\textbf{M}\text{odel}}$

"A **language model** is a statistical or machine learning **model** that assigns **probabilities** to sequences of **tokens**."

Terminology

Large **L**anguage **M**odel

Terminology

LLM = $\boxed{\text{Large}}$ Language Model

- **Model size**: billions of parameters or more
- **Training data**: 100s of billions of tokens or more
- **Compute**: a lot of GPUs

Characteristics

Decoder-only **Transformer**-based model

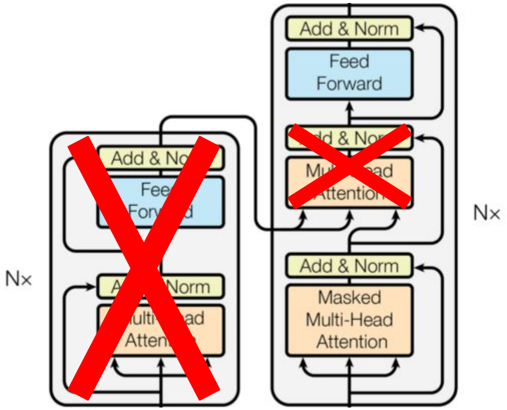


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Characteristics

Decoder-only **Transformer**-based model

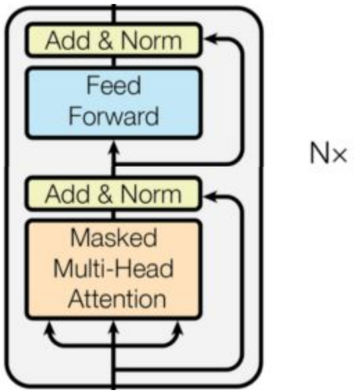
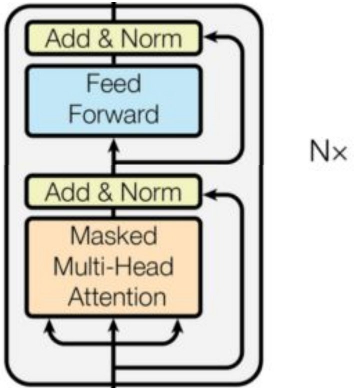


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Characteristics

Decoder-only **Transformer**-based model



Examples: GPT series, LLaMA, Gemma, DeepSeek, Mistral, Qwen, ...

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Transformers & Large Language Models

LLM overview

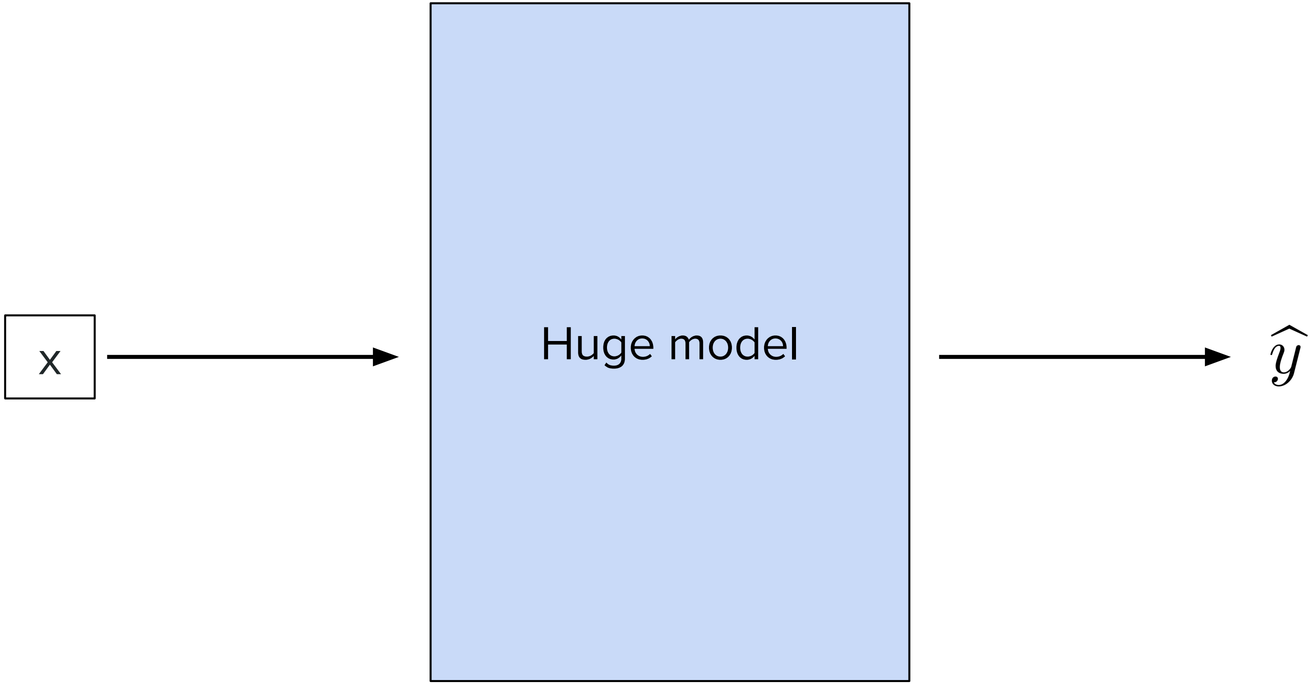
MoE-based LLMs

Response generation

Prompting strategies

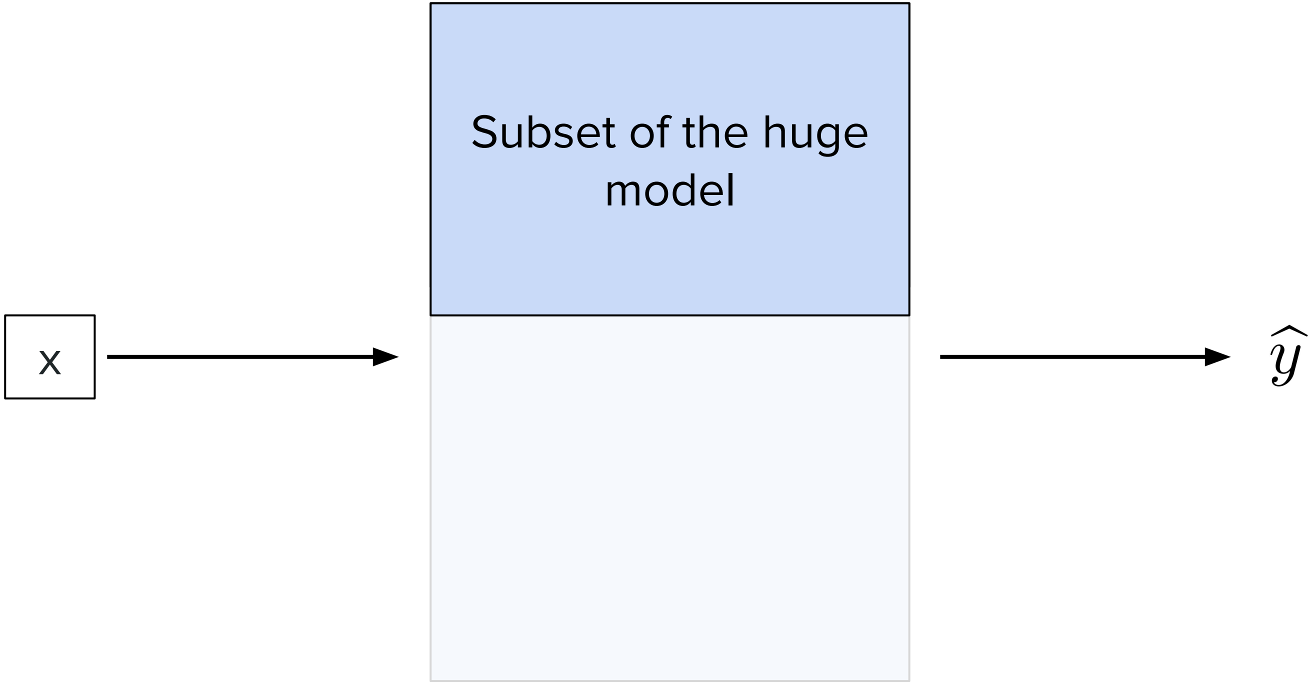
Inference optimizations

Motivation

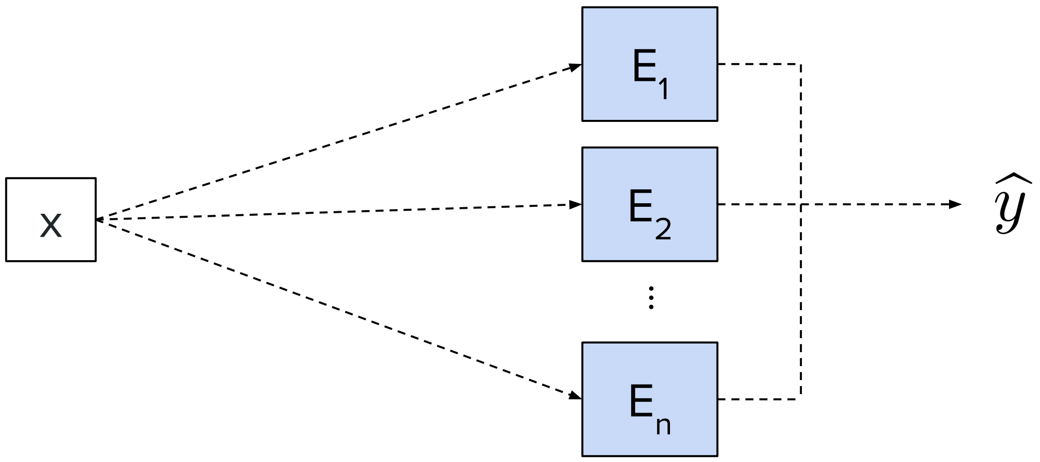


Motivation

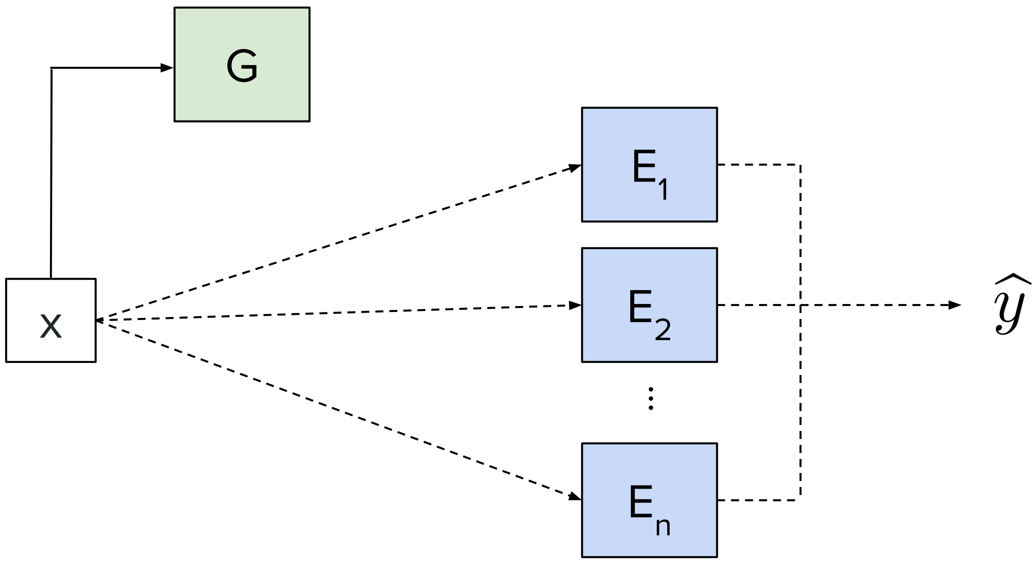
Idea. Not all weights are useful in the forward pass



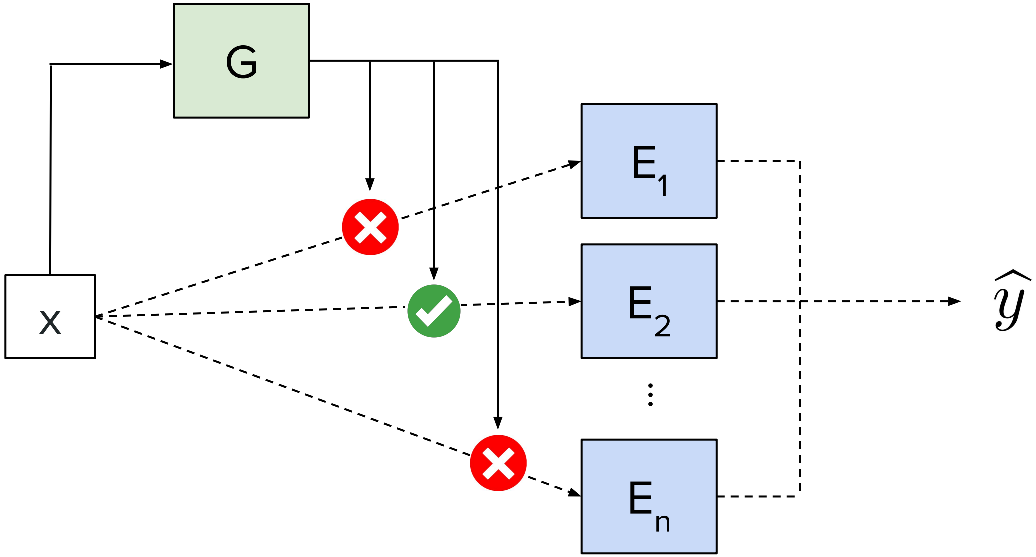
Motivation



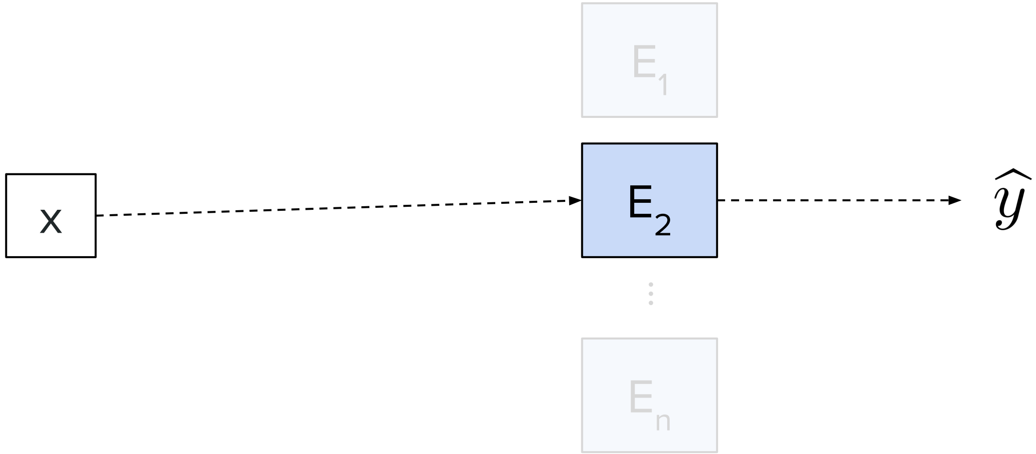
Motivation



Motivation

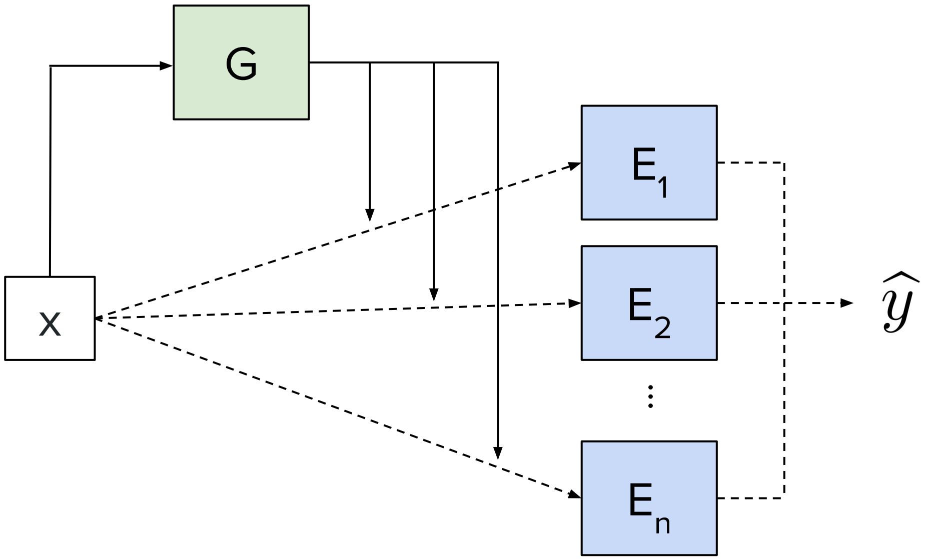


Motivation



Overview of MoEs

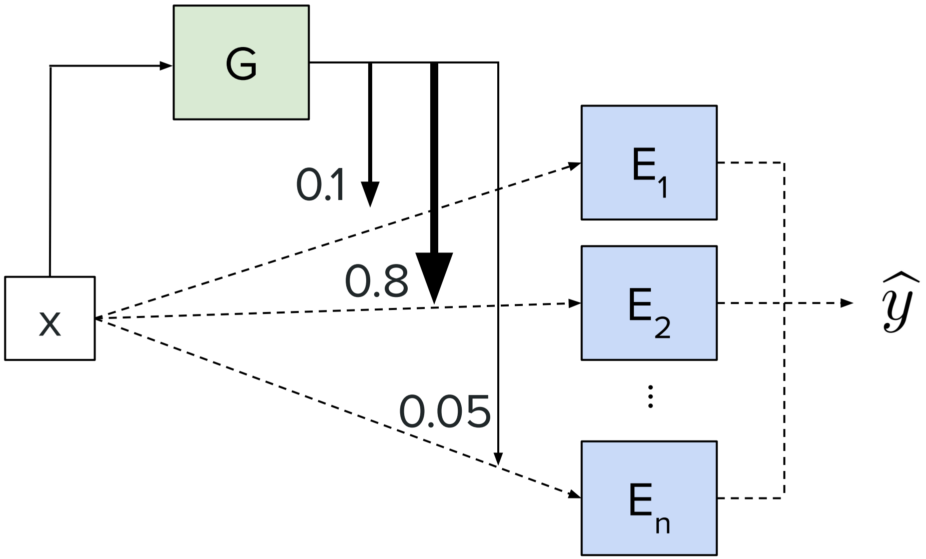
MoE = **M**ixture **of** **E**xperts



$$\widehat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$$

Overview of MoEs

MoE = **M**ixture **of** **E**xperts

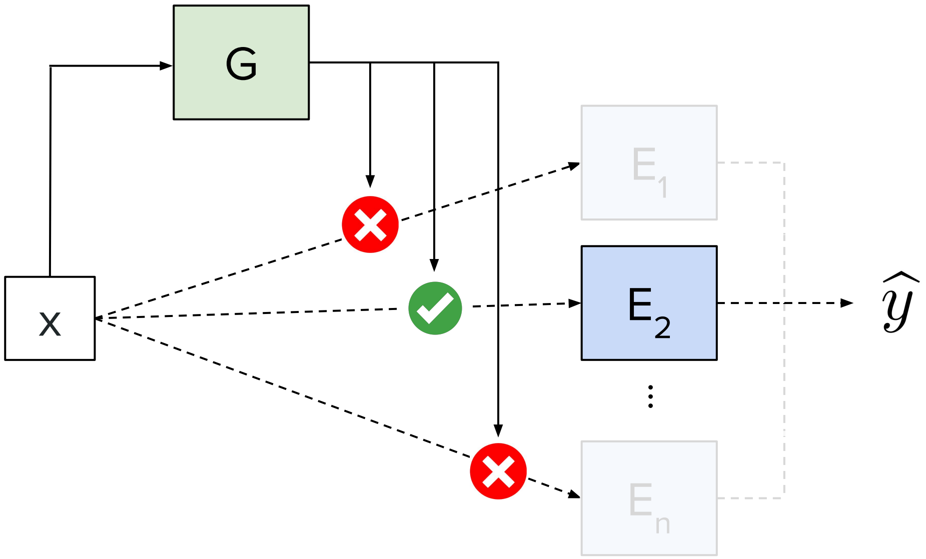


Dense MoE. Output is weighted average of **all** expert outputs.

$$\boxed{\widehat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)}$$

Overview of MoEs

MoE = **M**ixture **o**f **E**xperts



Sparse MoE. Output is weighted average of **selected** expert outputs.

$$\boxed{\widehat{y} = \sum_{i \in \mathcal{I}_k} G(x)_i E_i(x)}$$

Via **top-k** selection

 "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer", Shazeer et al., 2017.

MoE in Transformer-based models

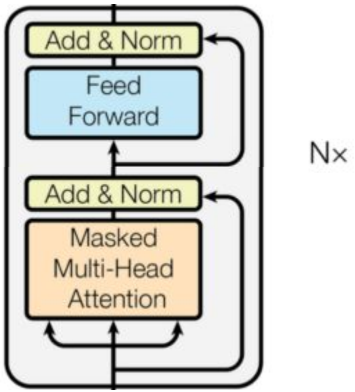


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

MoE in Transformer-based models

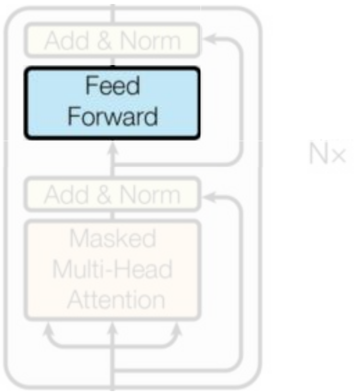
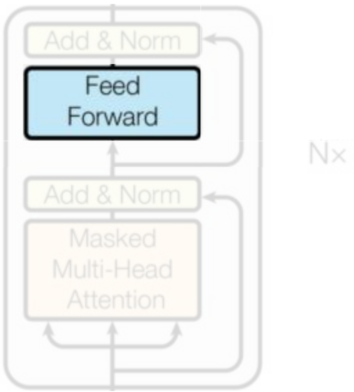


 Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

MoE in Transformer-based models



Routing done **for each token**!

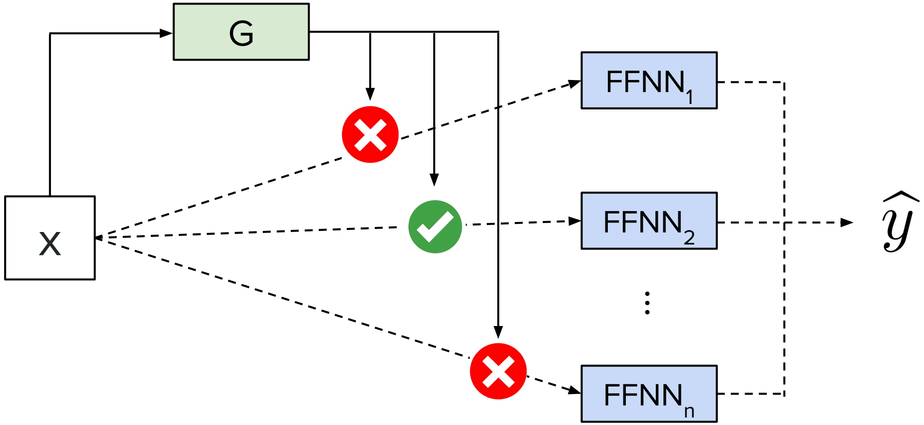


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Training challenges include routing collapse

Symptom. Same expert gets selected most of the time.

"routing collapse"

"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021.

Training challenges include routing collapse

Symptom. Same expert gets selected most of the time.

"routing collapse"

Remedy. Force other experts to be "part of the game" too via auxiliary loss:

$$\boxed{\text{loss} = \alpha \cdot N \cdot \sum_{i=1}^{N} f_i \cdot P_i}$$

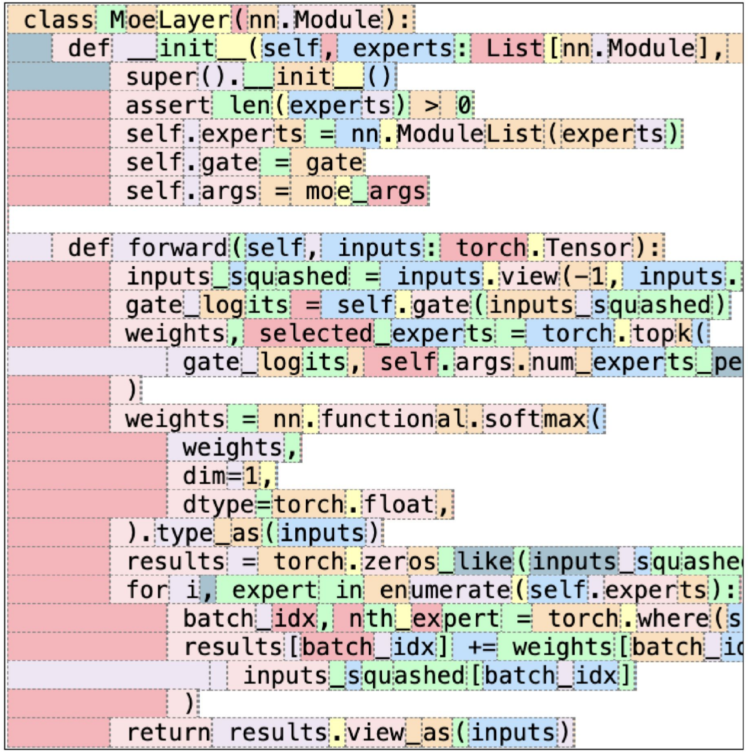
f_i = fraction of tokens routed to expert i

P_i = average routing probability for expert i

 "Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity", Fedus et al., 2021.

Interpreting experts

Layer 0

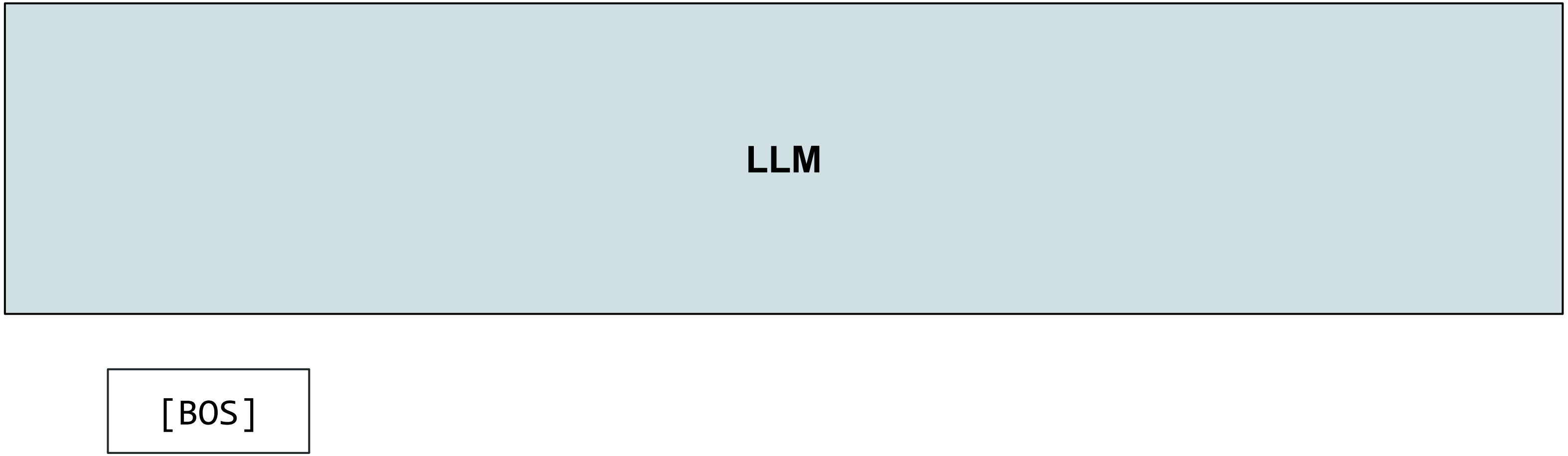


Each color represent an expert

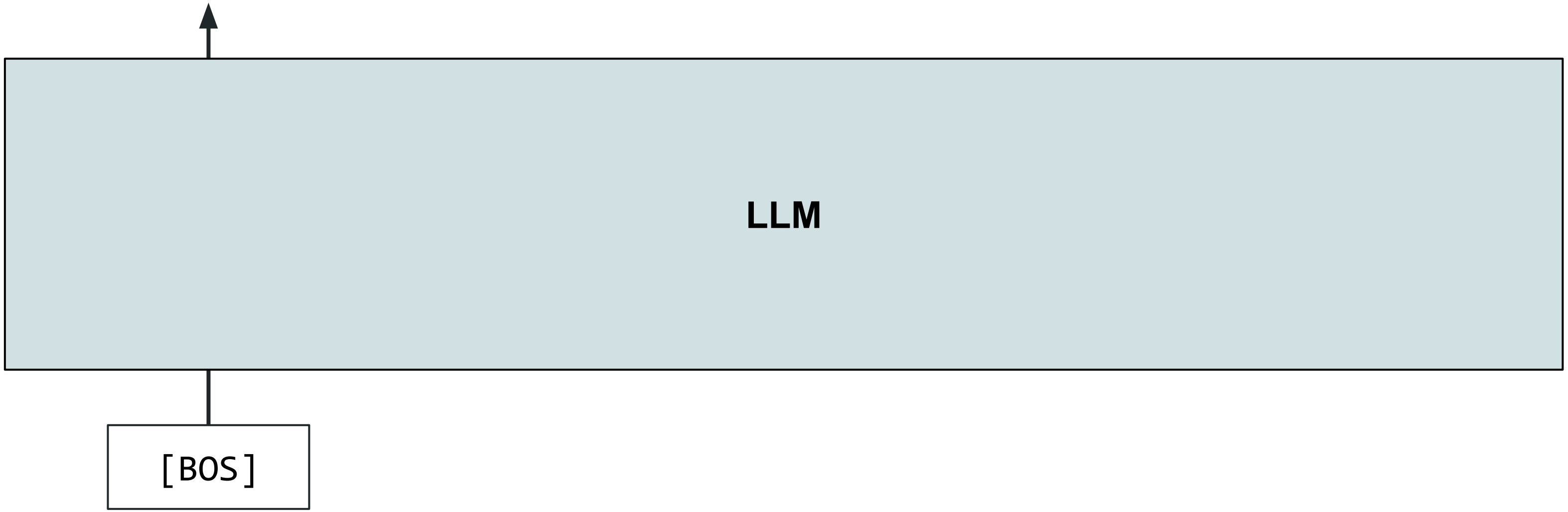
Transformers & Large Language Models

- LLM overview
- MoE-based LLMs
- **Response generation**
- Prompting strategies
- Inference optimizations

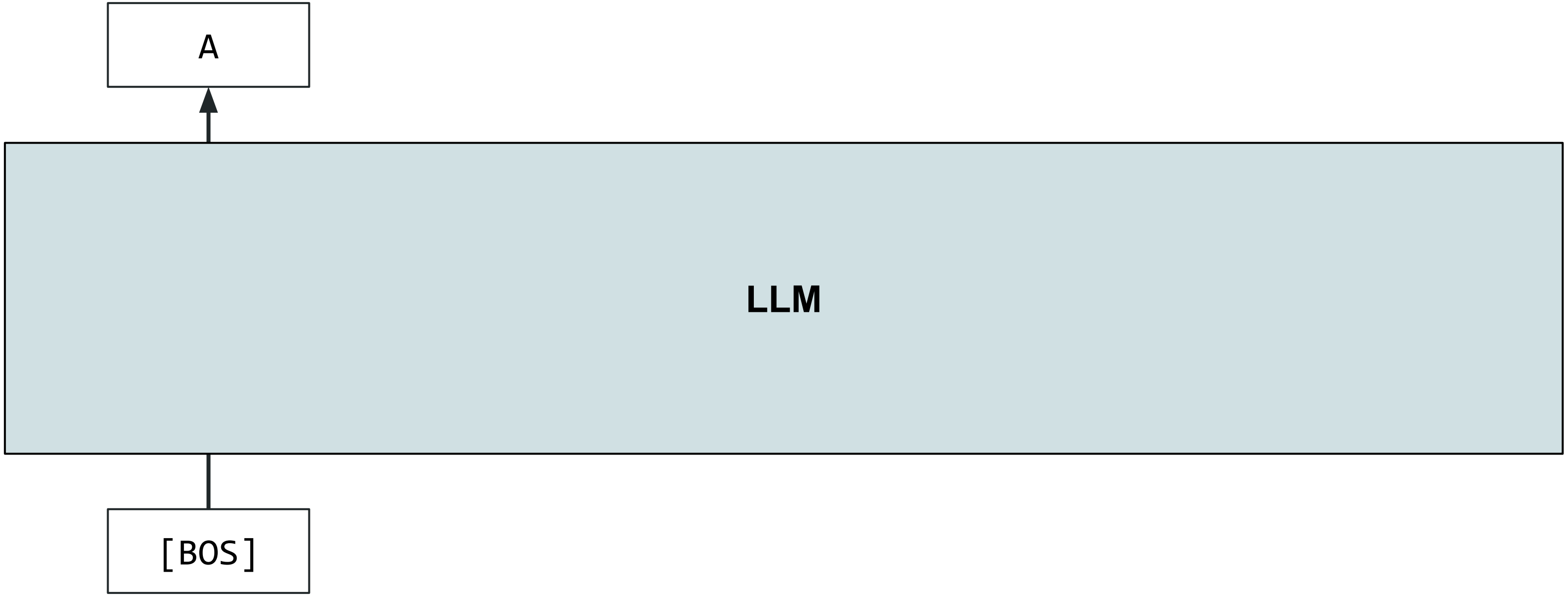
Next token prediction



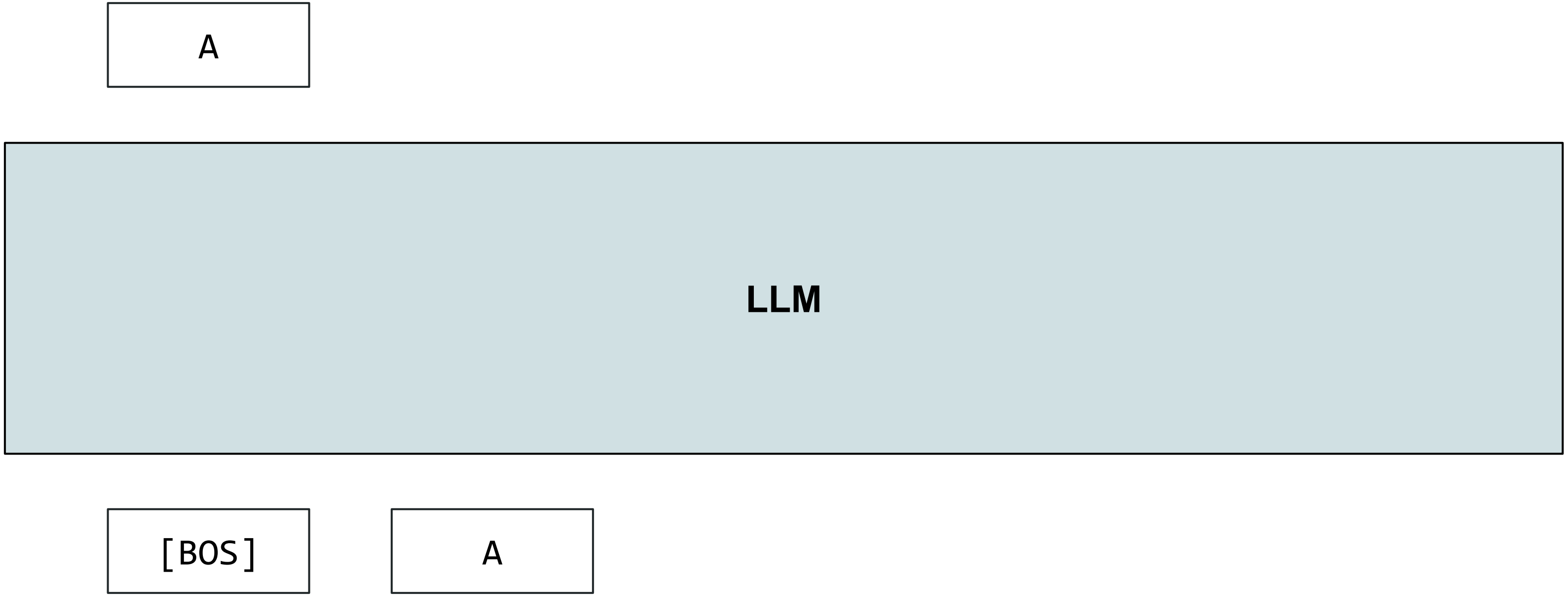
Next token prediction



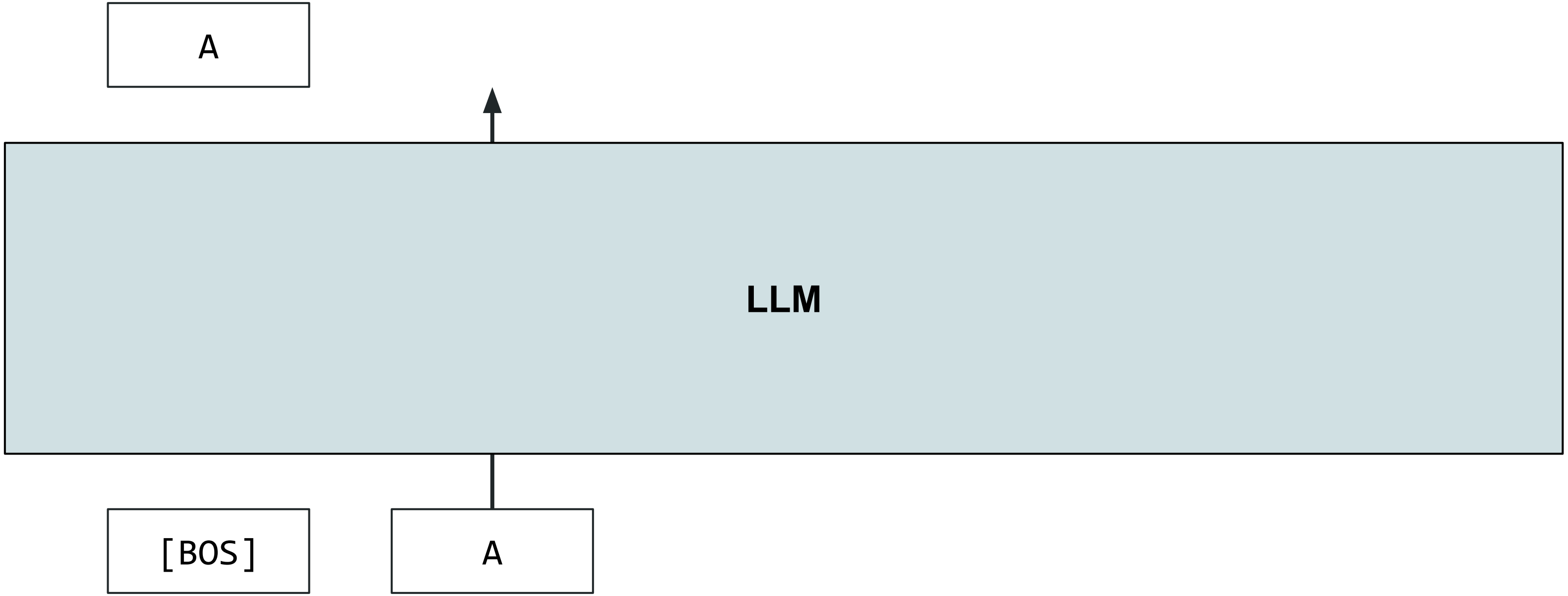
Next token prediction



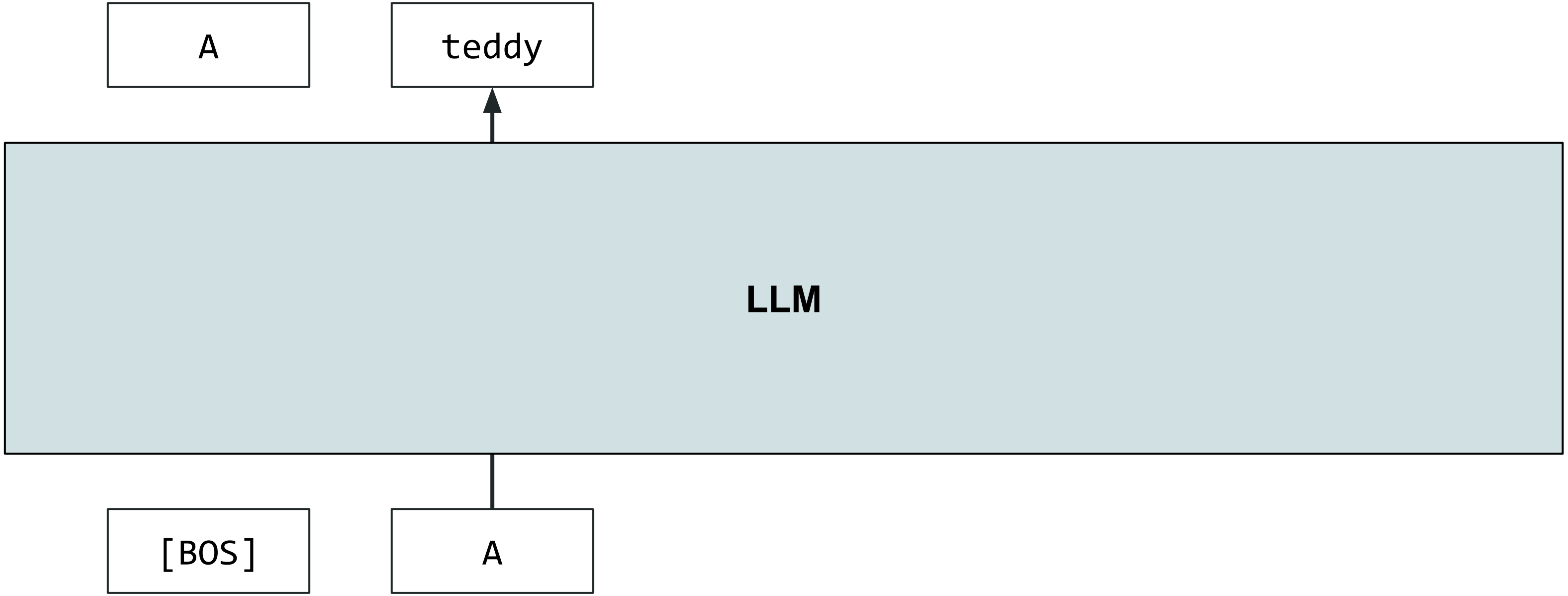
Next token prediction



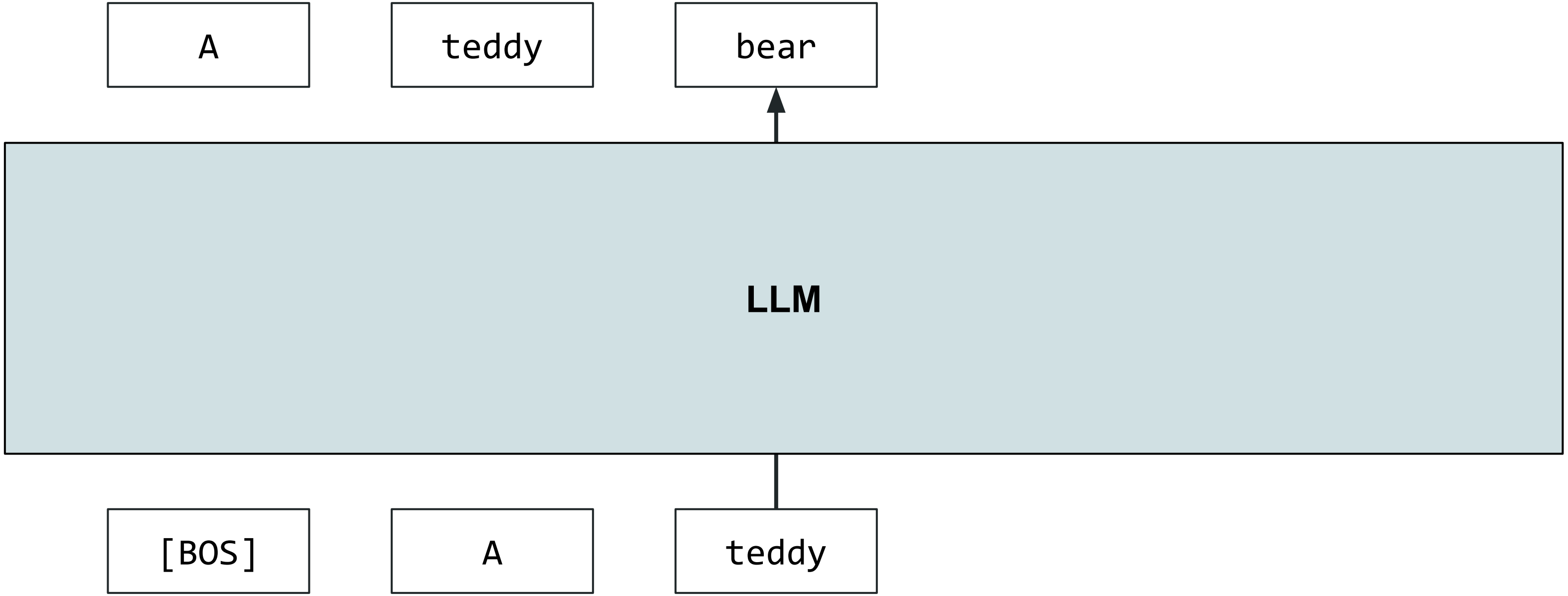
Next token prediction



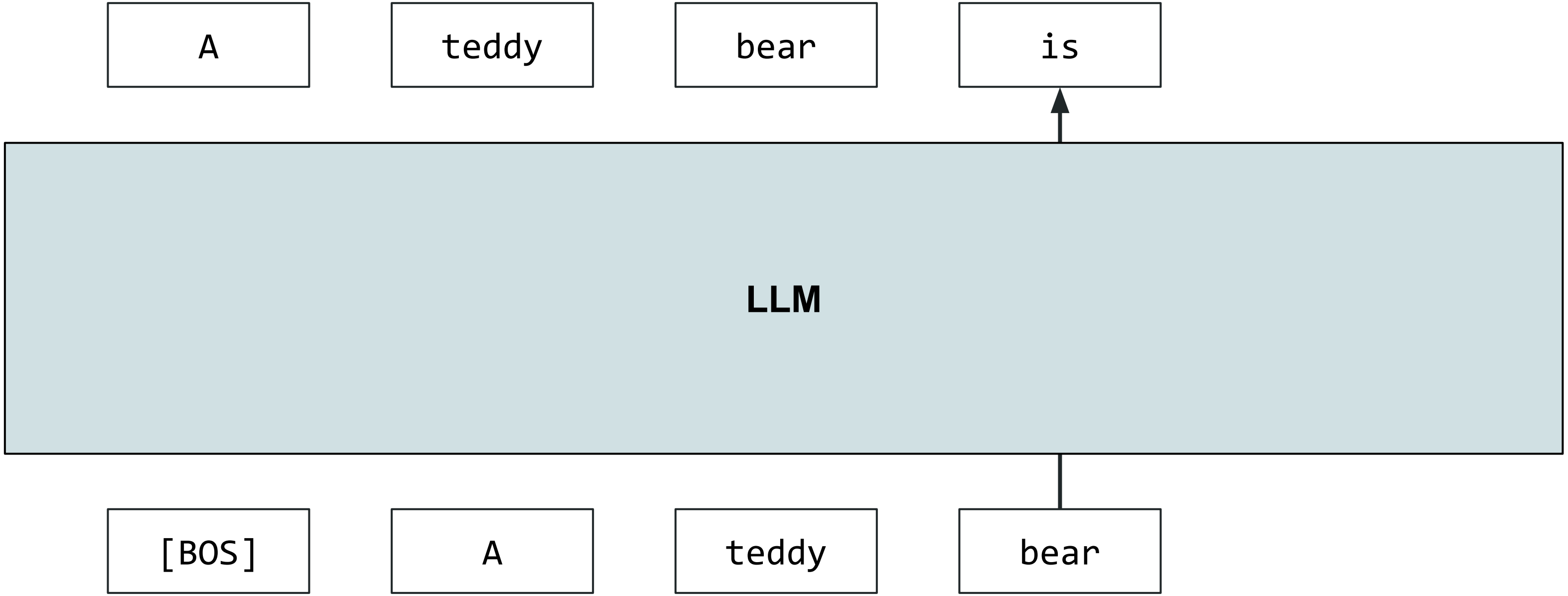
Next token prediction



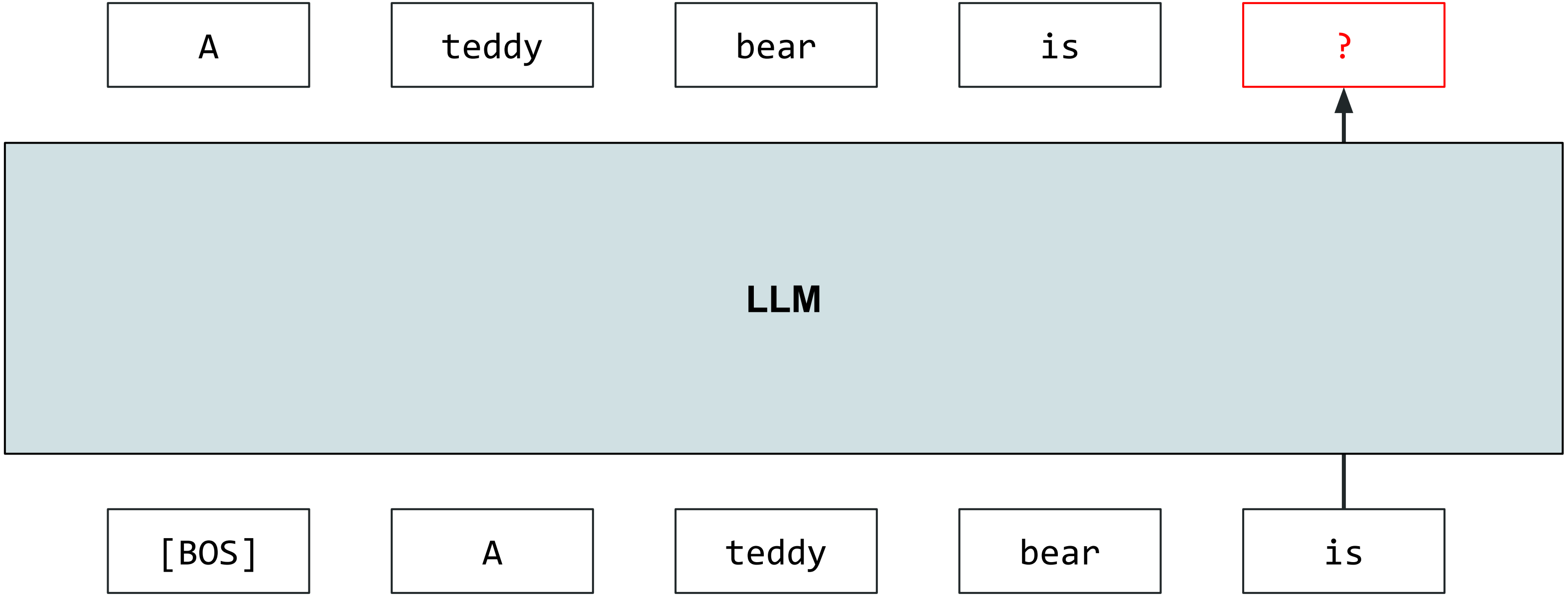
Next token prediction



Next token prediction



Next token prediction



Predicting next token

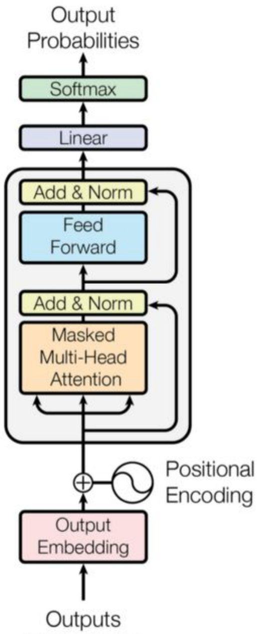


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token

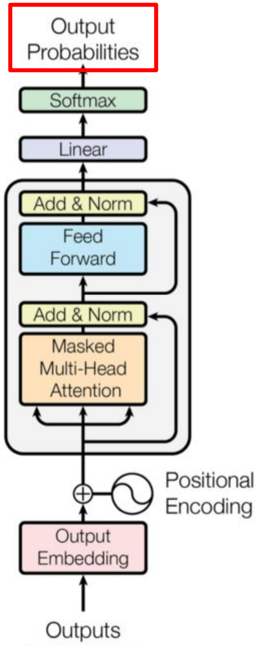
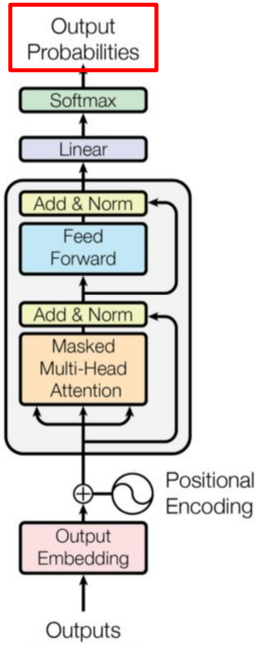


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token



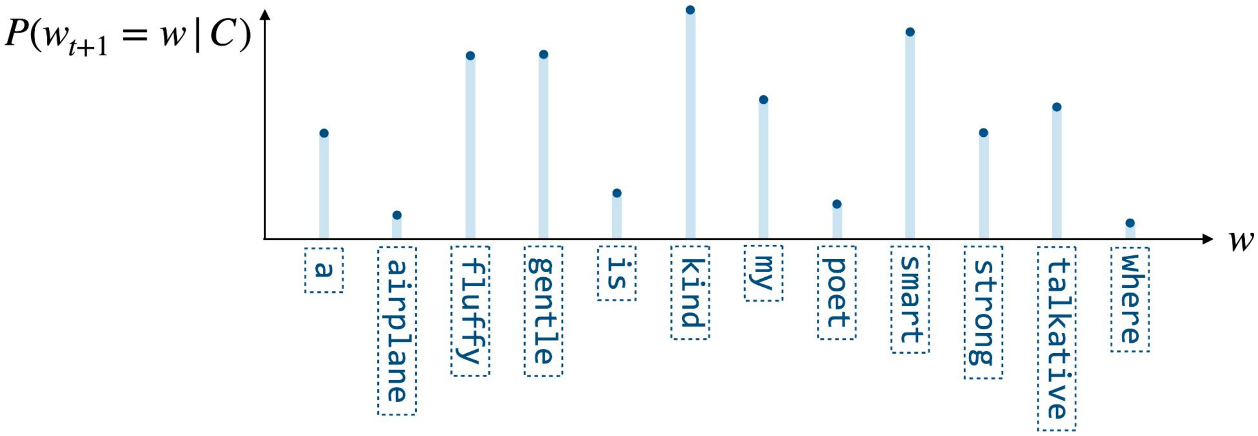
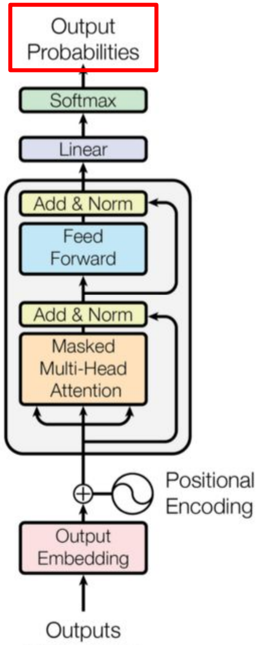
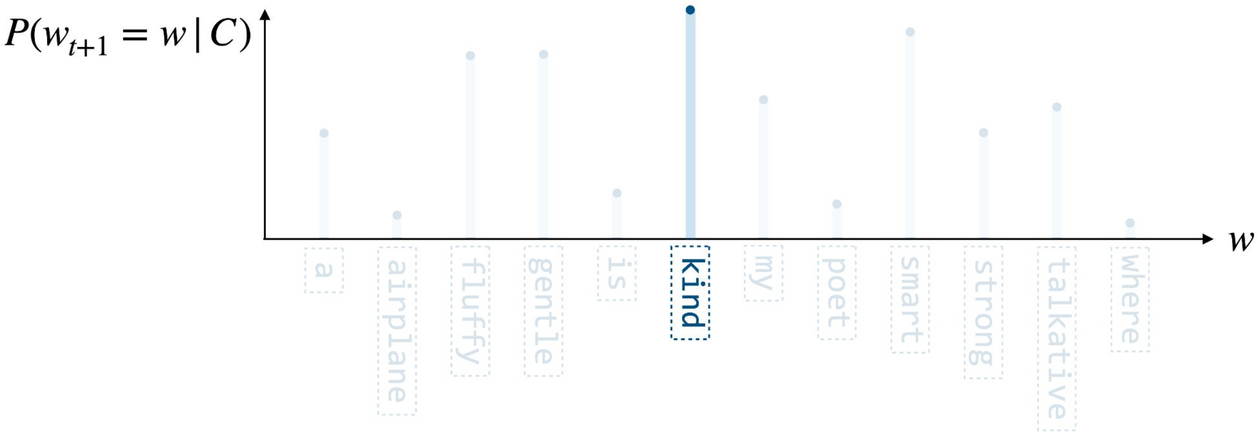


Figure adapted from "Attention is All You Need", Vaswani et al., 2017. "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

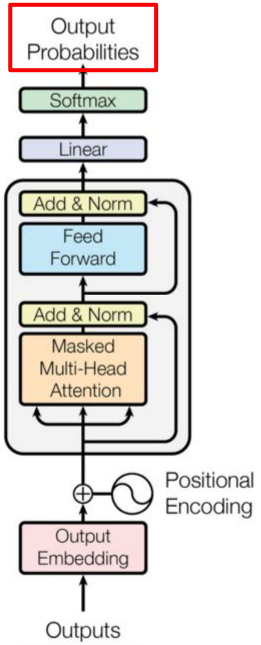
Predicting next token with greedy decoding



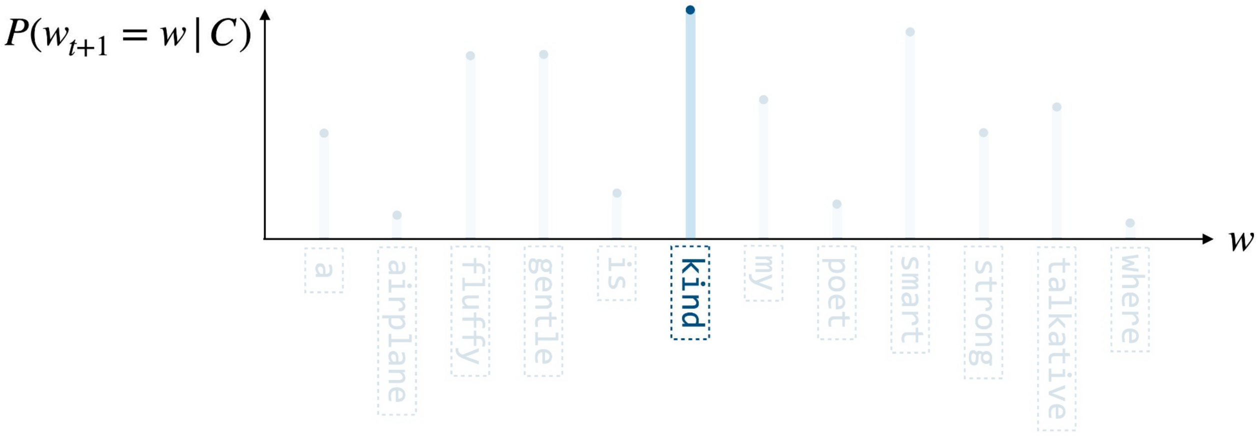
1st idea. Take token with highest predicted probability



Predicting next token with greedy decoding



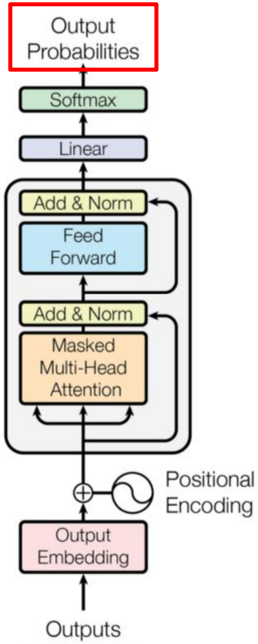
1st idea. Take token with highest predicted probability



Limitations. Output not optimal, natural and/or diverse

Figure adapted from "Attention is All You Need", Vaswani et al., 2017. "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Predicting next token with beam search

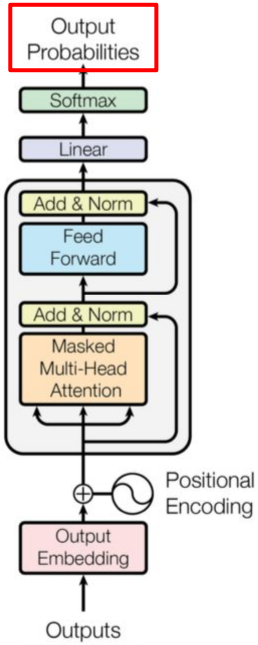


2nd idea. Keep k paths that are the most likely

[BOS]

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



2nd idea. Keep k paths that are the most likely

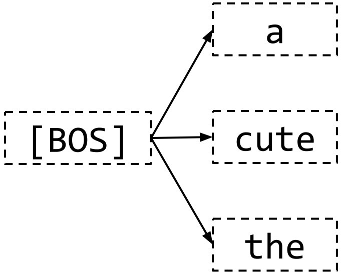
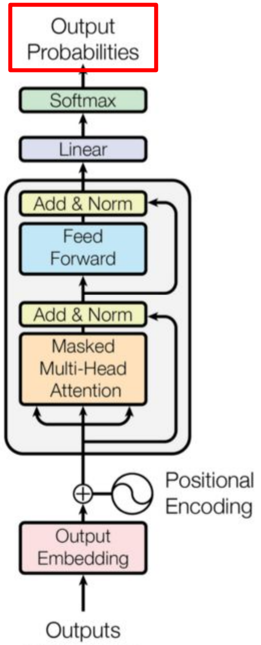


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



2nd idea. Keep k paths that are the most likely

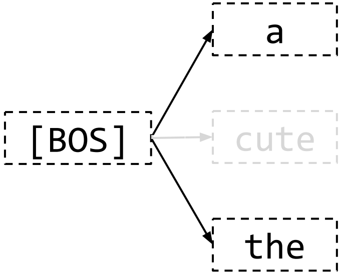
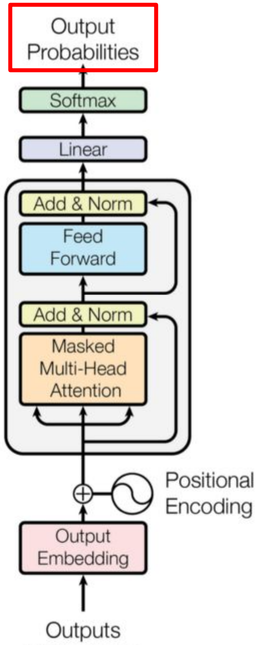


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



2nd idea. Keep k paths that are the most likely

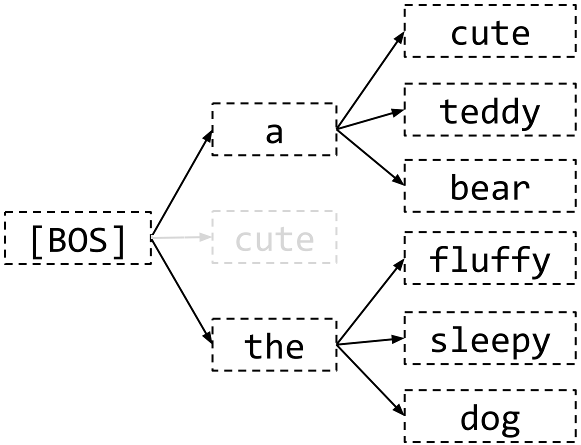
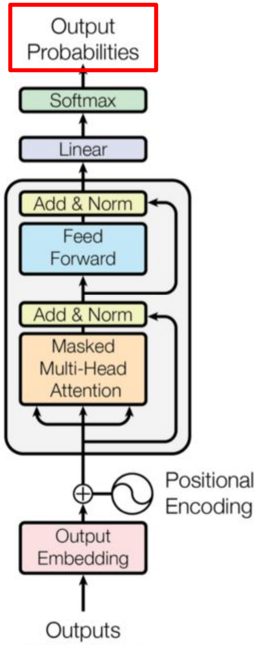


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



2nd idea. Keep k paths that are the most likely

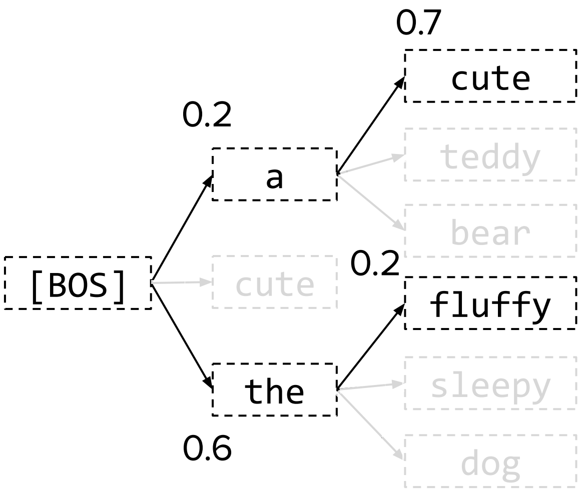
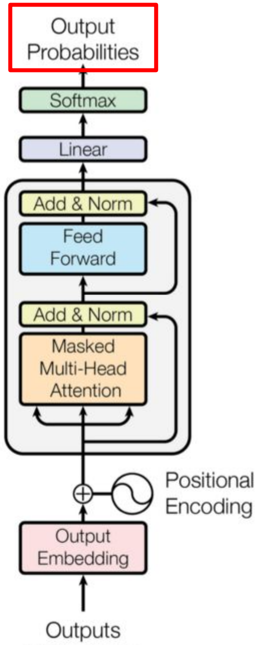


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



2nd idea. Keep k paths that are the most likely

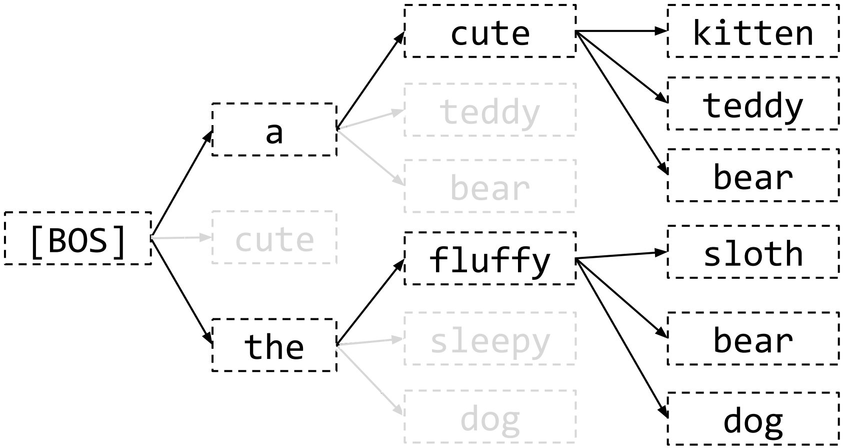
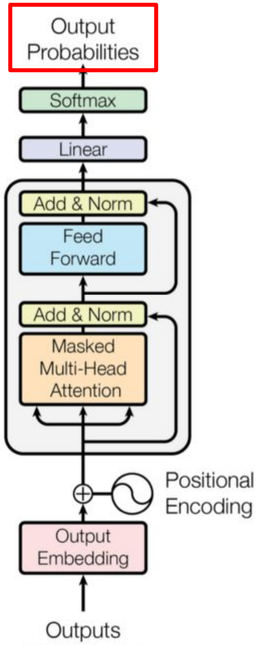


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



2nd idea. Keep k paths that are the most likely

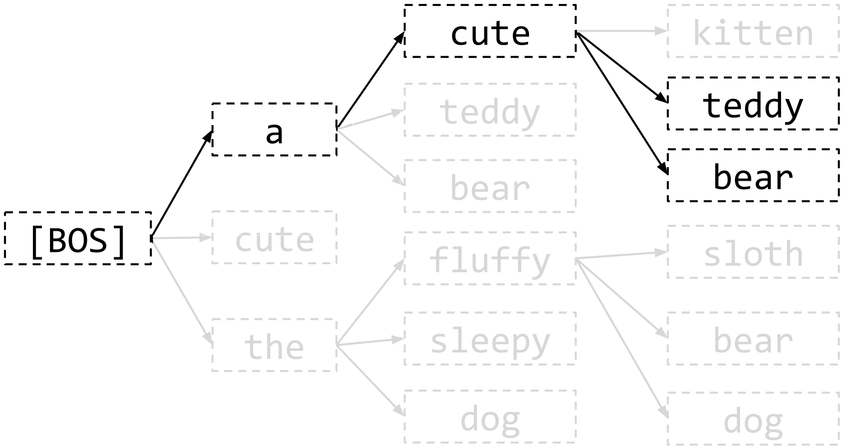
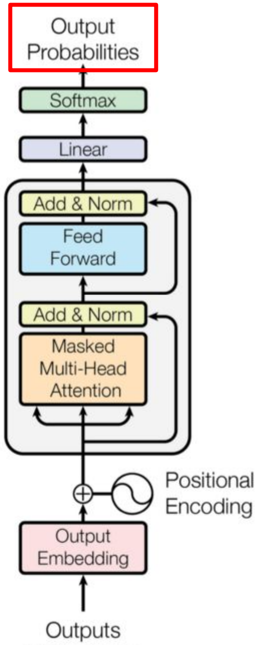
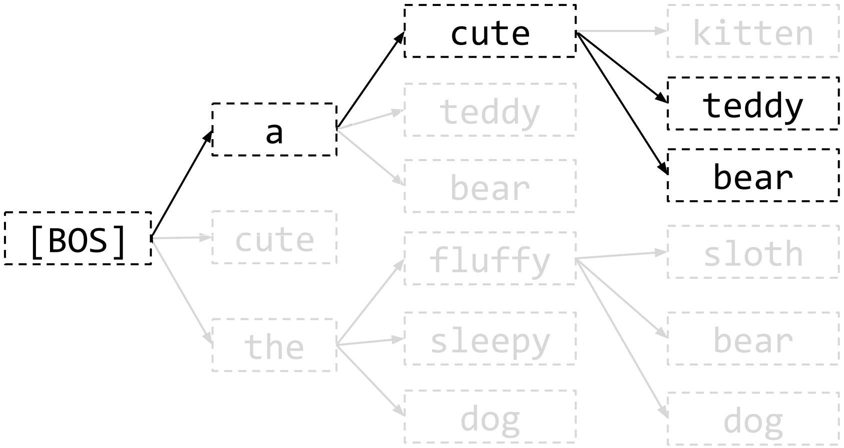


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



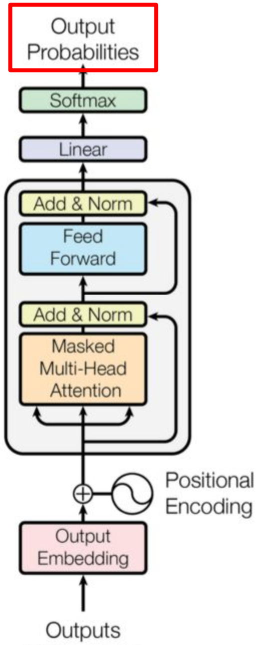
2nd idea. Keep k paths that are the most likely



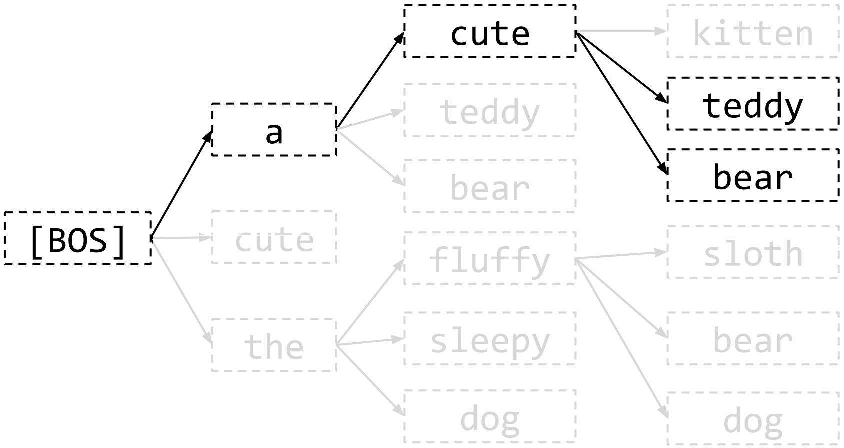
Ends at [EOS]

Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



2nd idea. Keep k paths that are the most likely

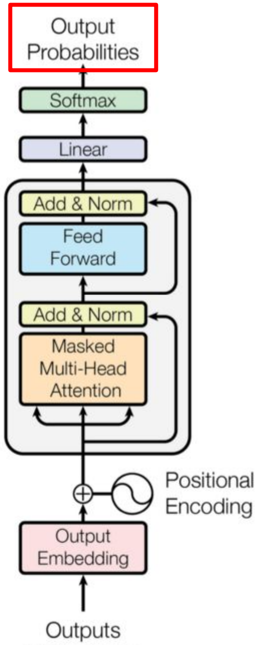


Ends at [EOS]

Limitations. Needs computations + lacks diversity/creativity

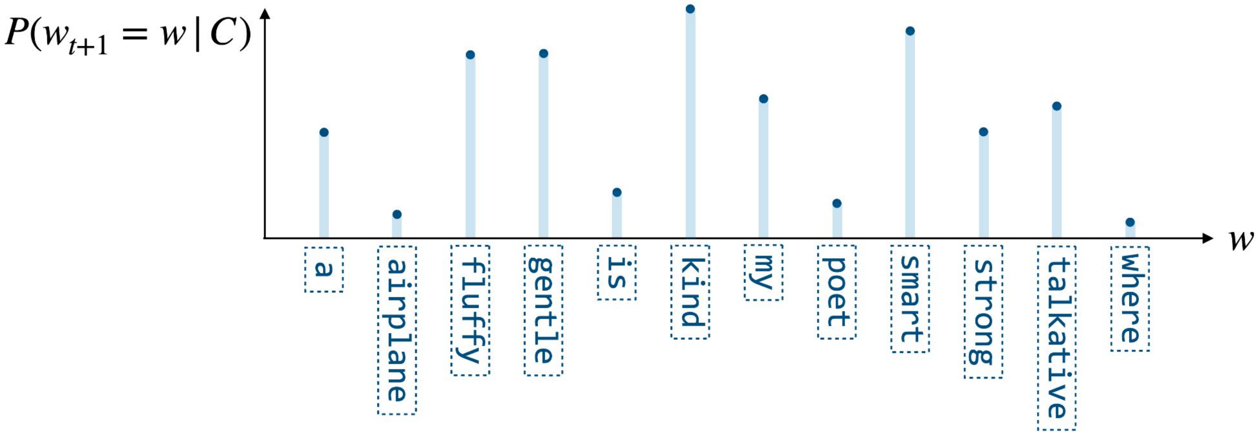
Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with sampling

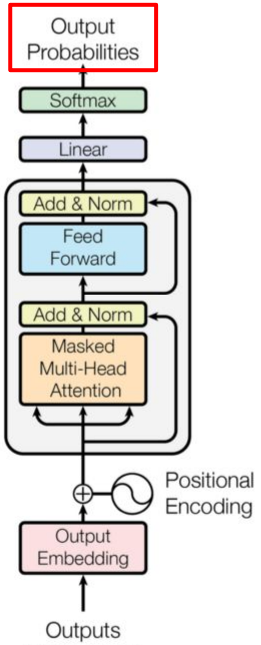


3rd idea. Sample next token from probability distribution:

$$\boxed{\widehat{w}_{t+1} \sim P(w_{t+1}|C)}$$

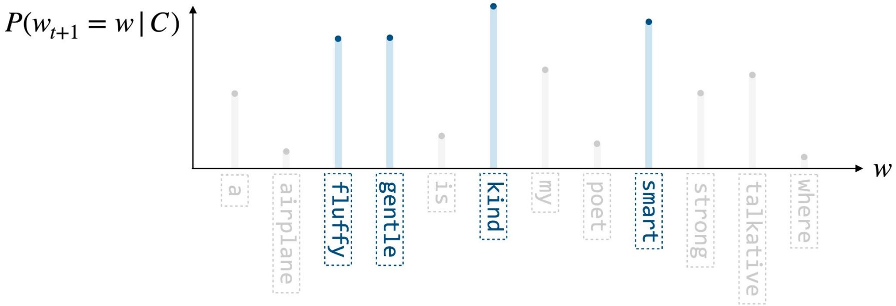


Sampling strategies

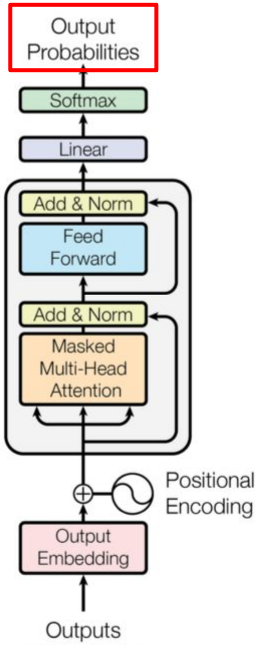


- **Top-k**: Sample among top k most probable tokens

k = 4



Sampling strategies

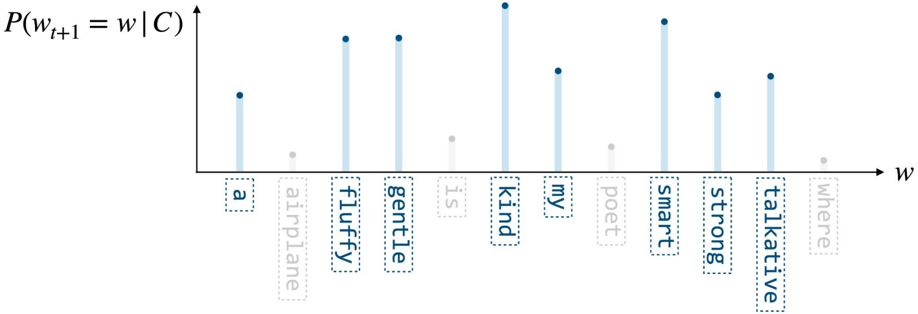


- **Top-k**: Sample among top k most probable tokens

k = 4

- **Top-p**: Random sample among smallest set of tokens with cumulative probability >= p

p = 90%

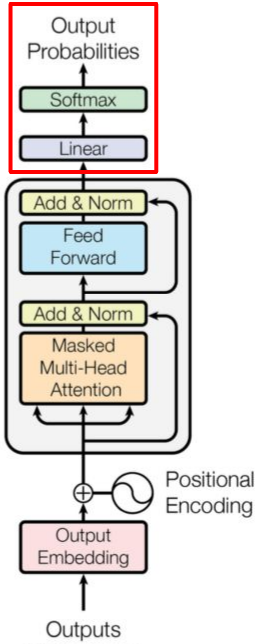


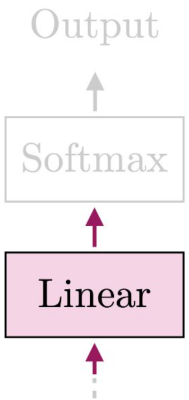
"*Super Study Guide: Transformers and Large Language Models*", *Amidi et al., 2024.*

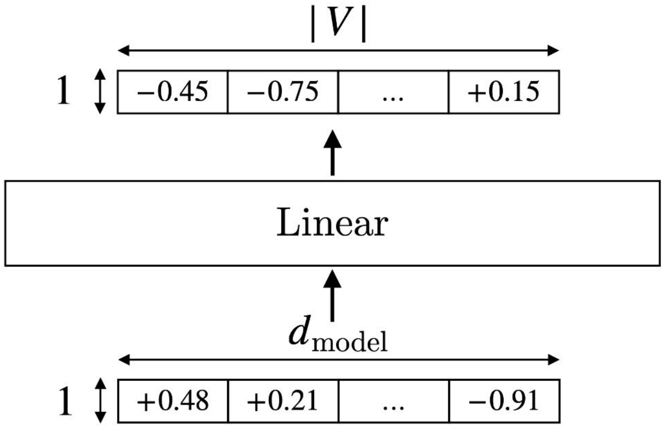
Next token prediction

But **how** are probabilities **obtained**?

Probability computation

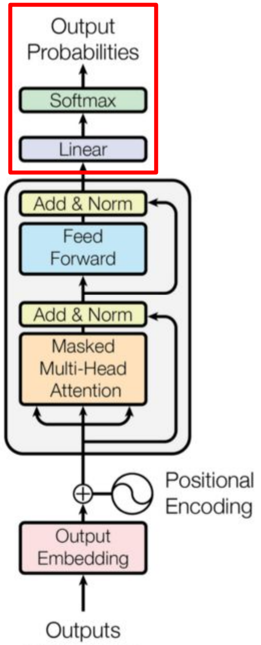


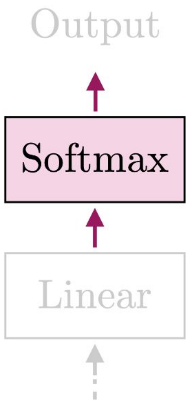


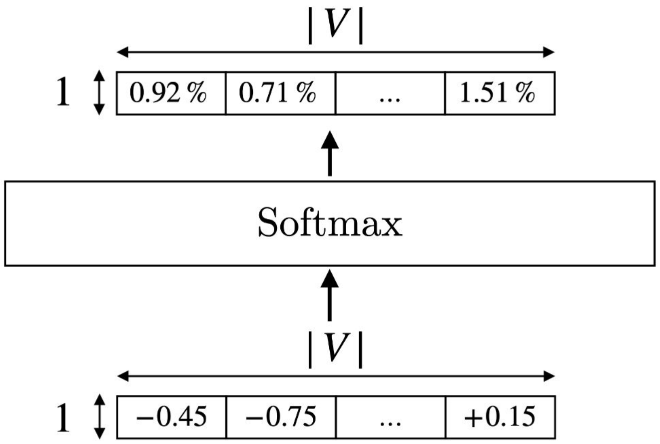


"*Super Study Guide: Transformers and Large Language Models*", *Amidi et al., 2024.*

Probability computation

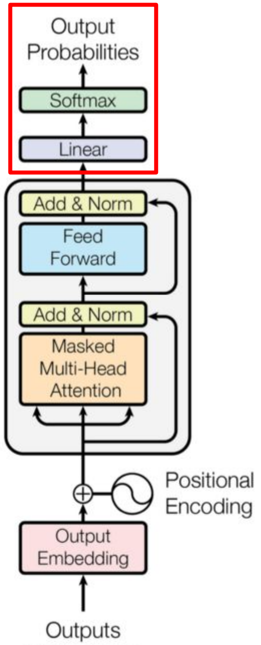


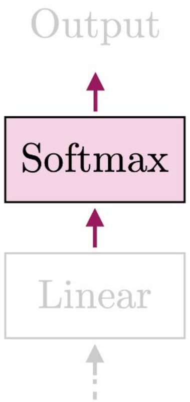




"*Super Study Guide: Transformers and Large Language Models*", *Amidi et al., 2024.*

Temperature allows to tweak output probabilities





$$P_{\text{adj}}(w_{t+1} = w_i | C) = \frac{\exp\left(\frac{x_i}{T}\right)}{\sum_{j=1}^{n} \exp\left(\frac{x_j}{T}\right)}$$

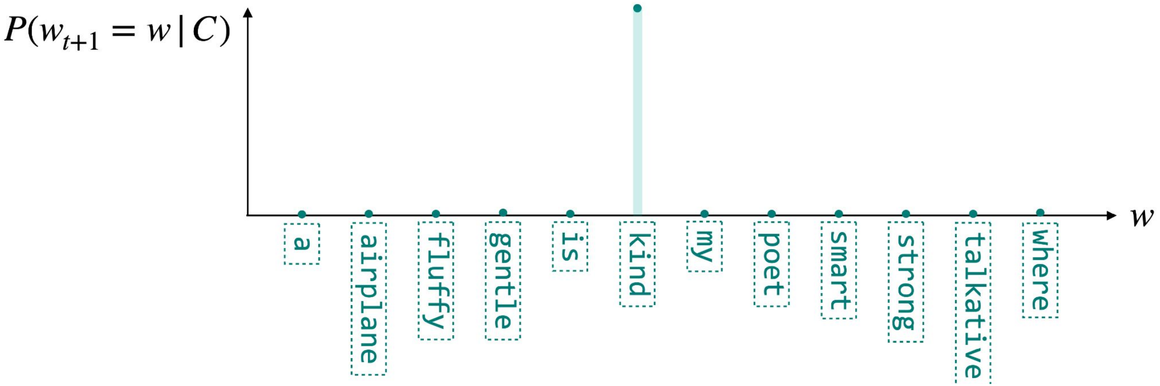
Impact of temperature on probabilities

Small T ?

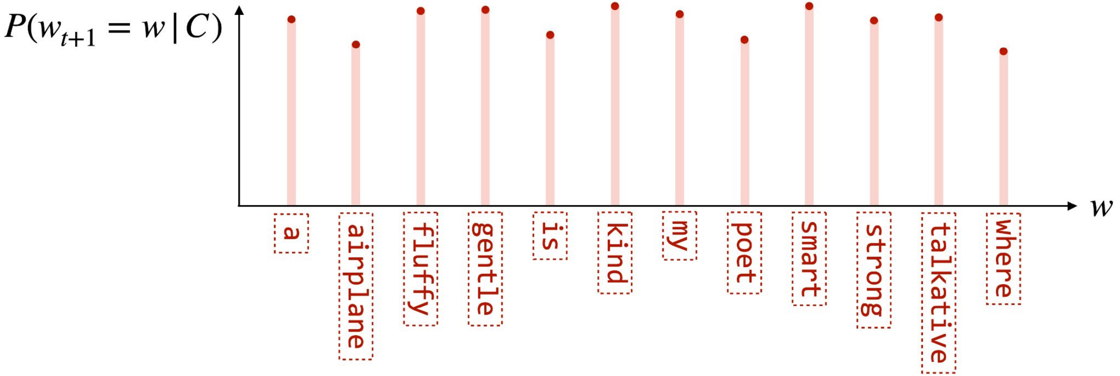
High T ?

Impact of temperature on probabilities

Small T



High T



"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Suggested reading: "Defeating Nondeterminism in LLM Inference", He et al., 2025.

Constraining the output via guided decoding

Motivation. Generate output in a specific format

Input prompt
Generate a description of my 33-year old teddy bear who likes reading. *Do this in JSON format.*

Desired output (JSON)

```json
{
    "first_name": "teddy",
    "last_name": "bear",
    "age": 33,
    "hobby": "reading"
}
```

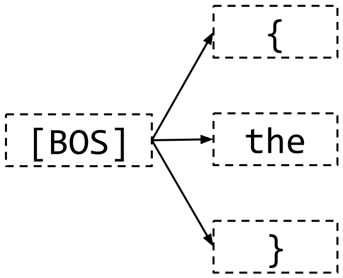
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens

[BOS]

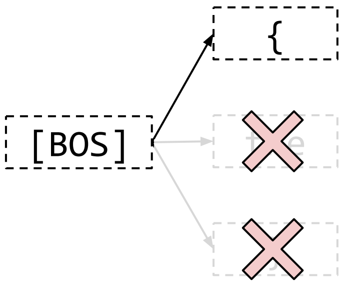
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



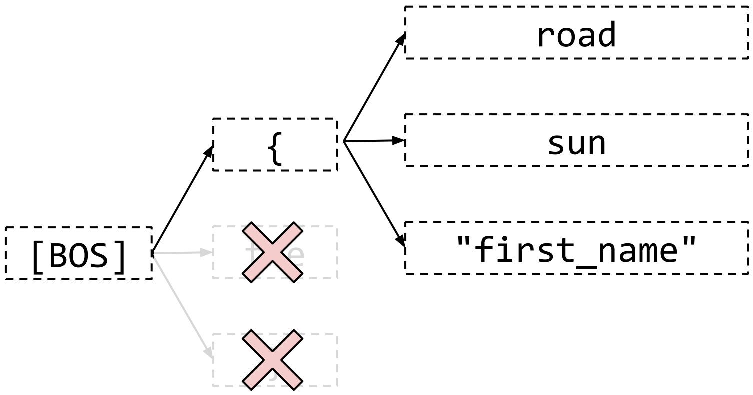
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



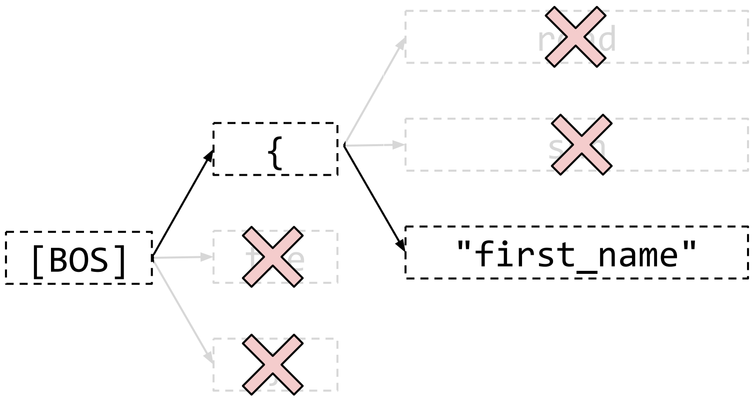
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



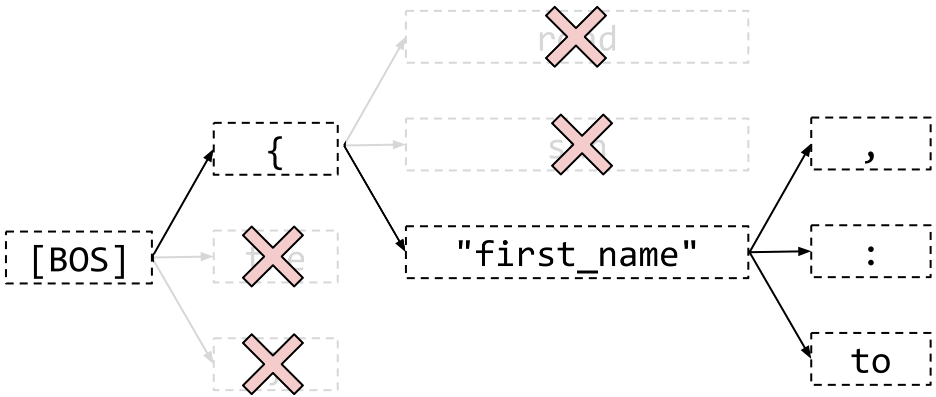
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



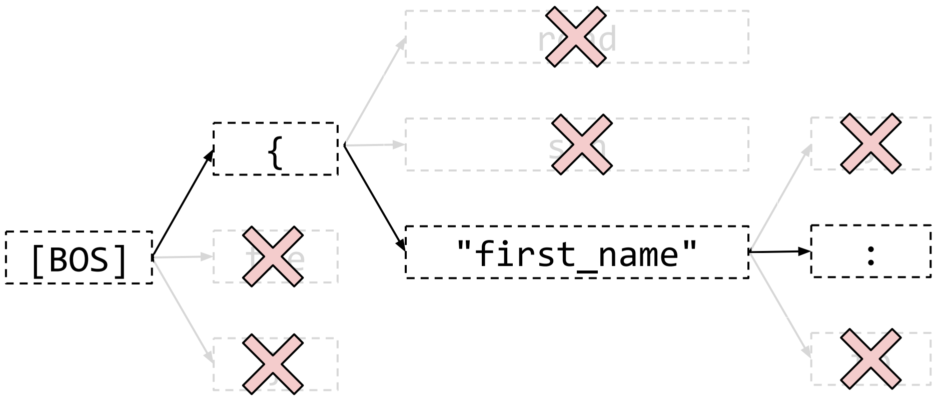
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



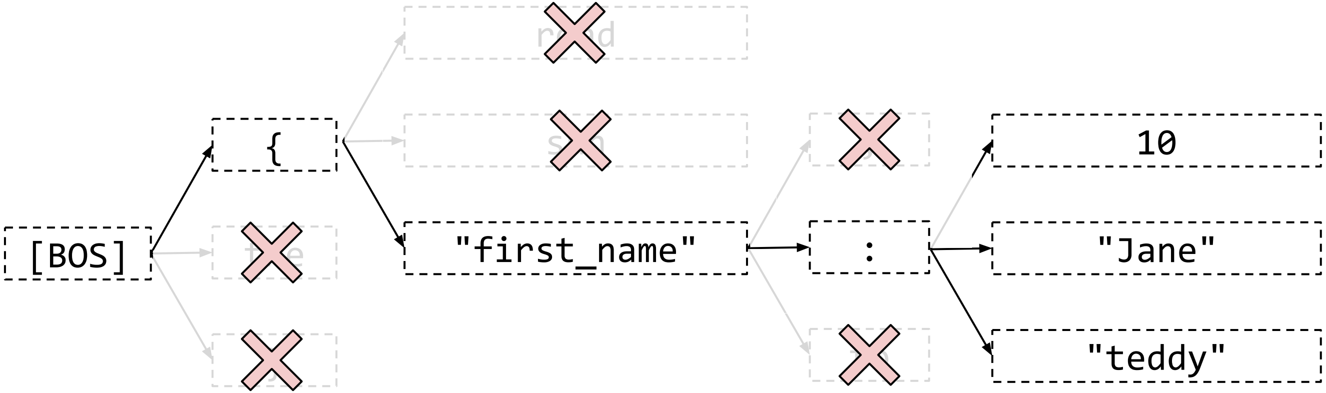
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



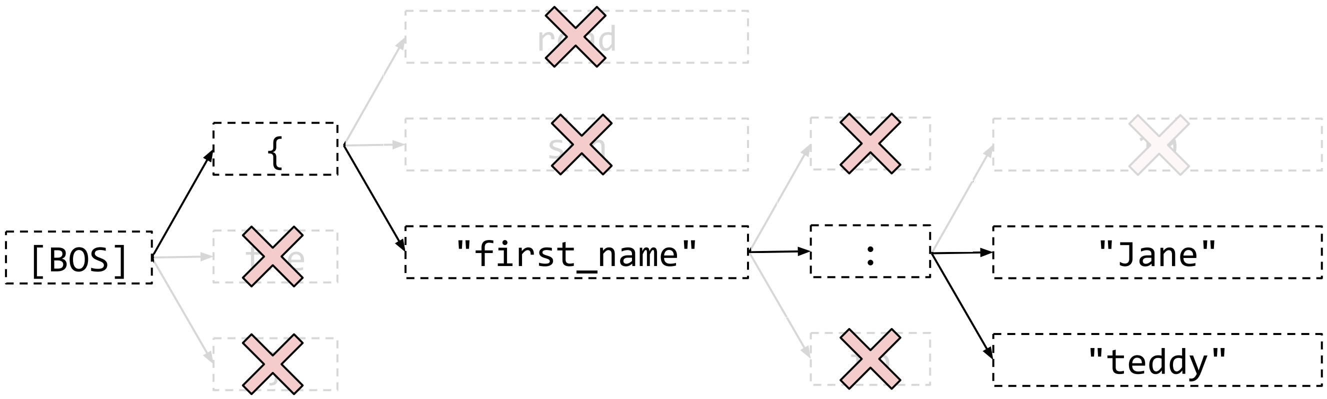
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



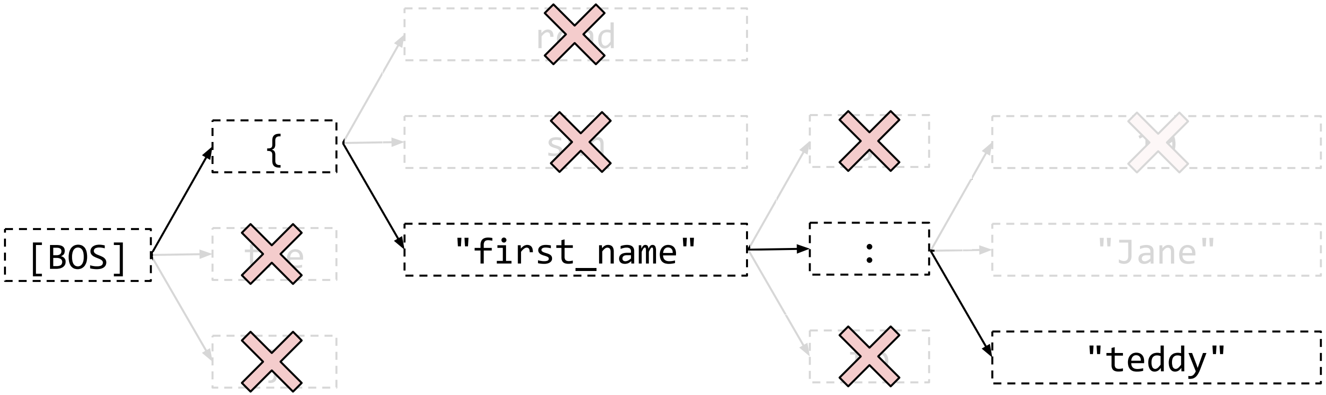
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



Transformers & Large Language Models

LLM overview

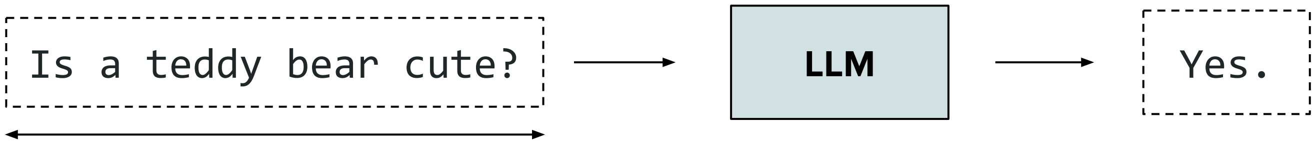
MoE-based LLMs

Response generation

Prompting strategies

Inference optimizations

Terminology



Context length

Context size

Window size

Terminology

Discussion. Orders of magnitude by:

- Input type
- Models

Terminology

Discussion. Orders of magnitude by:

- Input type
- Models

Beware of "context rot"!

"*Context Rot: How Increasing Input Tokens Impacts LLM Performance*", *Hong et al., 2025.*

Main structure

My teddy bear had a long day and needs a bedtime story. — **Context**

Generate a bedtime story that takes place in a specific location. — **Instructions**

Location: Country of teddy bears — **Input**

The story needs to be suitable for teddy bears that are tired. — **Constraints**

"*Super Study Guide: Transformers and Large Language Models*", *Amidi et al., 2024.*

In-context learning

IC**L** = **I**n-**C**ontext **L**earning

"*Language Models are Few-Shot Learners*", *Brown et al., 2020.*

In-context learning

ICL = In-Context Learning

Zero-shot learning	Few-shot learning
• Question is asked without examples • Performance heavily depends on performance of initial model	• Prompt contains examples of input / output • Typically has a better performance

"Language Models are Few-Shot Learners", Brown et al., 2020.

In-context learning

ICL = In-Context Learning

Zero-shot learning	Few-shot learning
• Question is asked without examples • Performance heavily depends on performance of initial model	• Prompt contains examples of input / output • Typically has a better performance

Discussion. Showing examples in the prompt is *generally* better but:

- Requires effort
- Increases computational complexity & cost
- Increases latency

Chain of thought

Idea. Explaining reasoning helps in improving performance.

```
Q: How old is this bear?
A: The bear was born in
2020. It is therefore 4.


Q: How old will the bear
be next year?
```

→ LLM →

```
A: It will be one
year older than
its age this year,
which was 4.
Hence, it will be
5.
```

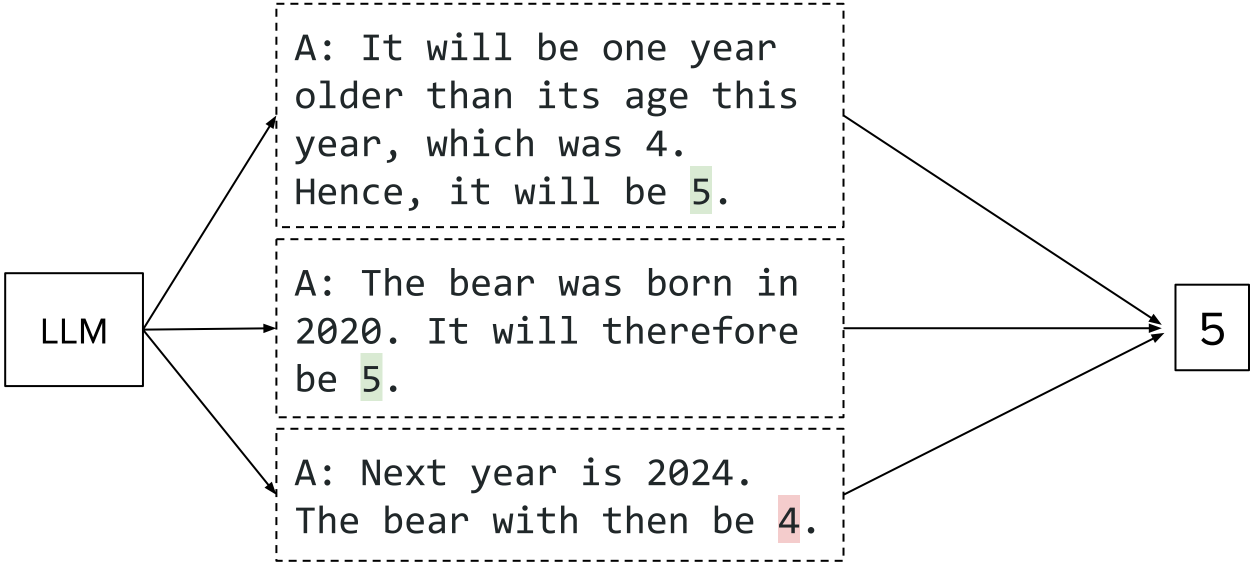
Discussion.

- Interpretability + explanation
- More tokens: higher cost and latency

"*Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*", *Wei et al., 2022.*

Self-consistency

Idea. Aggregating over reasoning paths improves performance.



Discussion. Trade-off between performance and added cost.

"*Self-Consistency Improves Chain of Thought Reasoning in Language Models*", *Wang et al., 2022.*

Transformers & Large Language Models

LLM overview

MoE-based LLMs

Response generation

Prompting strategies

Inference optimizations

Challenges

Motivation. Computations are expensive, any way to reduce complexity?

Categories. Many dimensions to optimize for.

Challenges

Motivation. Computations are expensive, any way to reduce complexity?

Categories. Many dimensions to optimize for.

"Exact" efficiency:

- Avoid redundancies
- Memory management
- Reformulate the math

Challenges

Motivation. Computations are expensive, any way to reduce complexity?

Categories. Many dimensions to optimize for.

"Exact" efficiency:

- Avoid redundancies
- Memory management
- Reformulate the math

Approximations:

- Architectural changes
- Embeddings representations
- Token prediction

Attention-based tricks

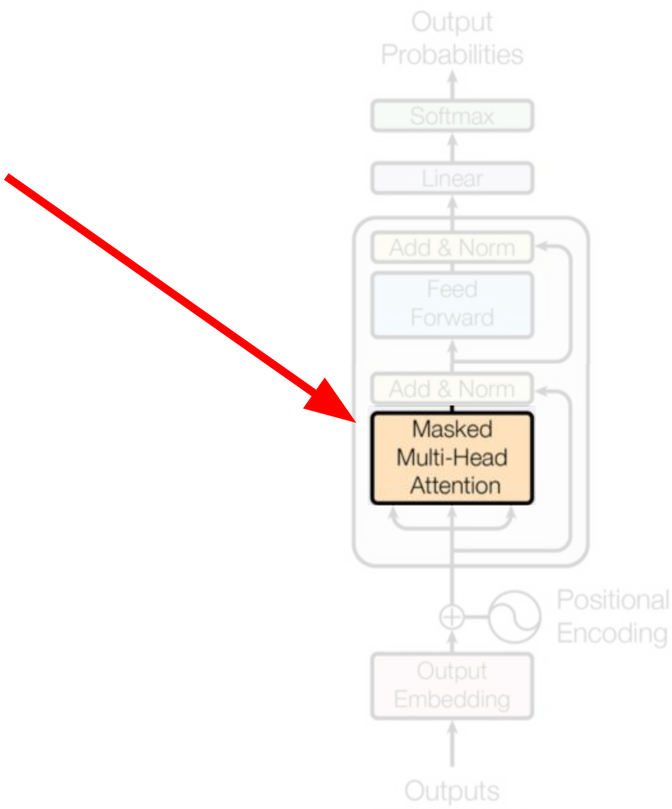


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

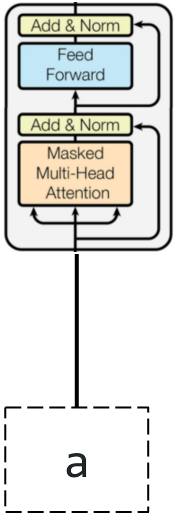


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

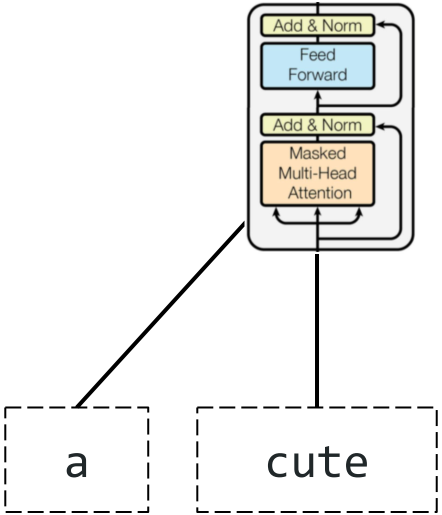


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

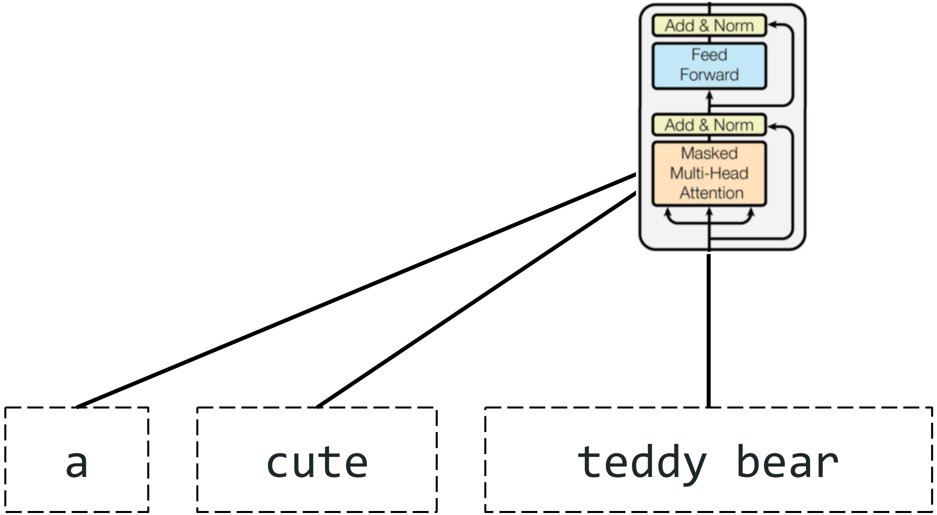


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

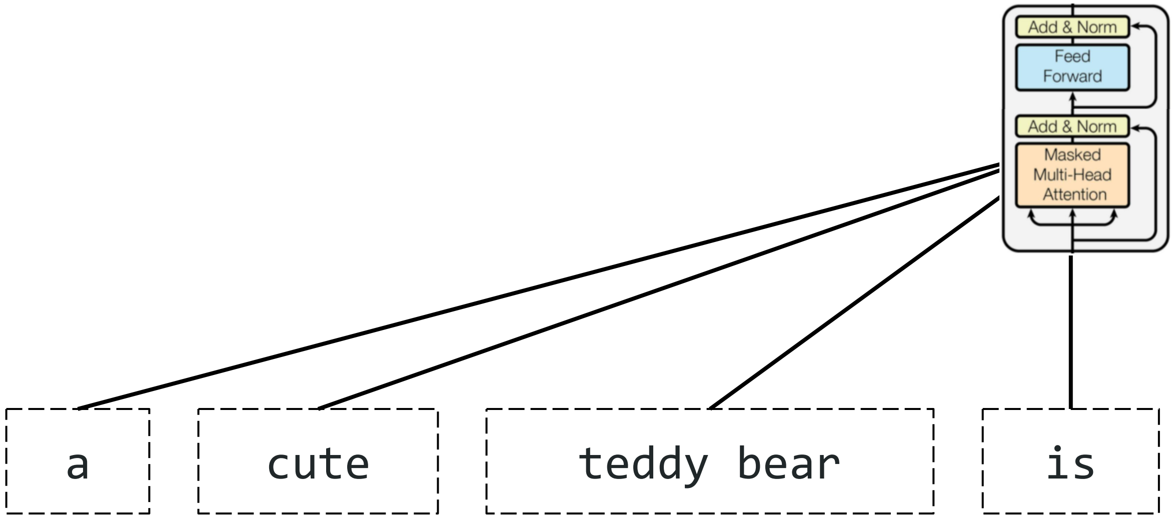


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

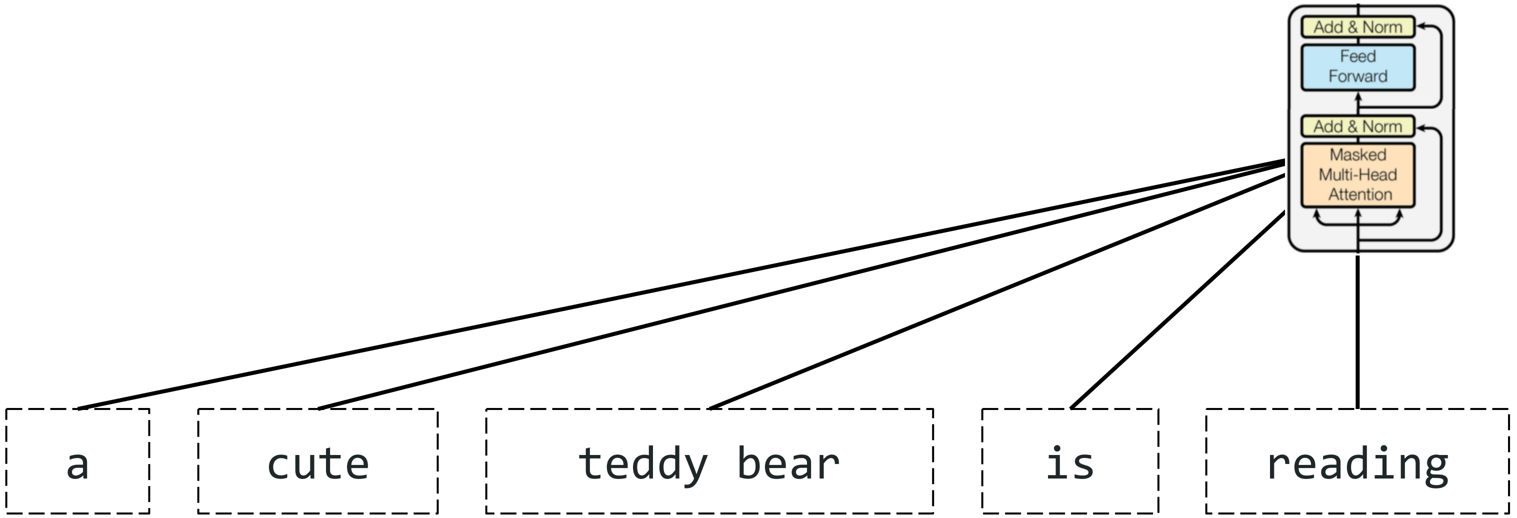


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

$$\text{softmax}\left(\frac{Q\boxed{K^T}}{\sqrt{d_k}}\right)\boxed{V}$$

Reusing computations with KV caching

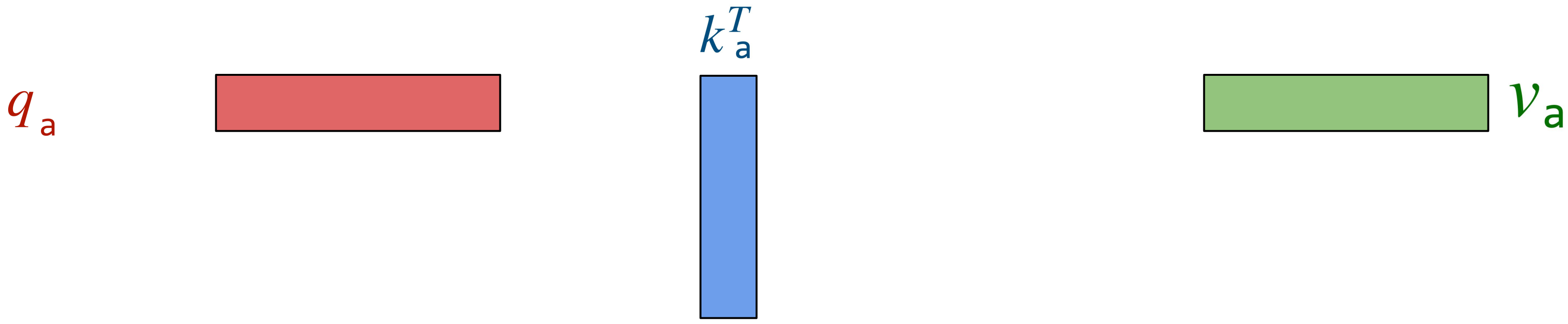
Motivation. New token needs to interact with all previous tokens.

Idea. Keep keys and values in a cache.

Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

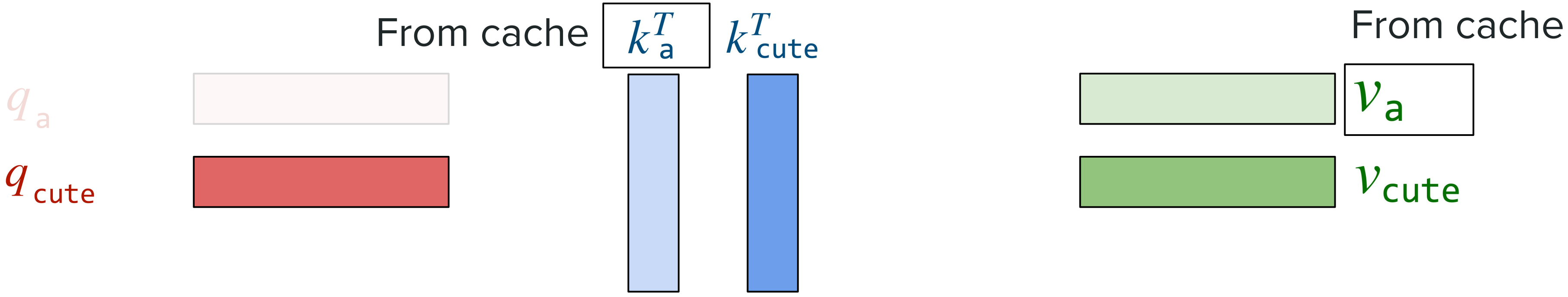
Idea. Keep keys and values in a cache.



Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

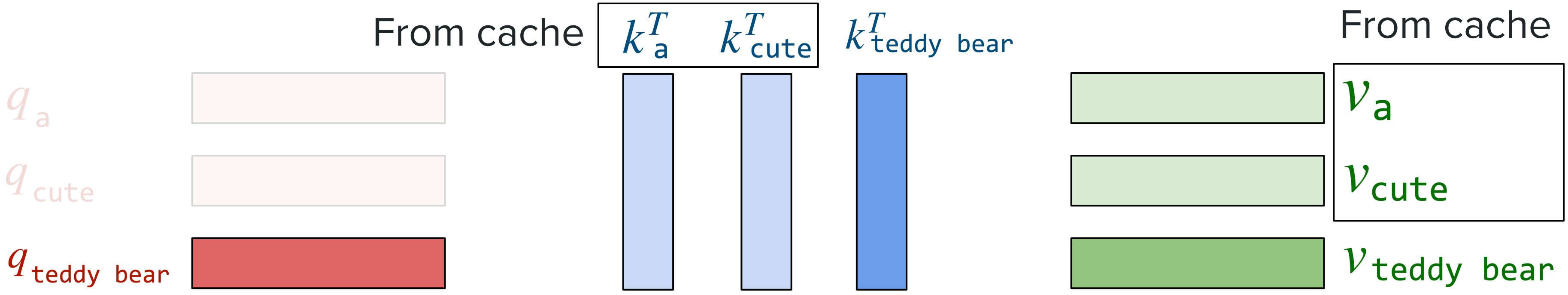
Idea. Keep keys and values in a cache.



Reusing computations with KV caching

Motivation. New token needs to interact with all previous tokens.

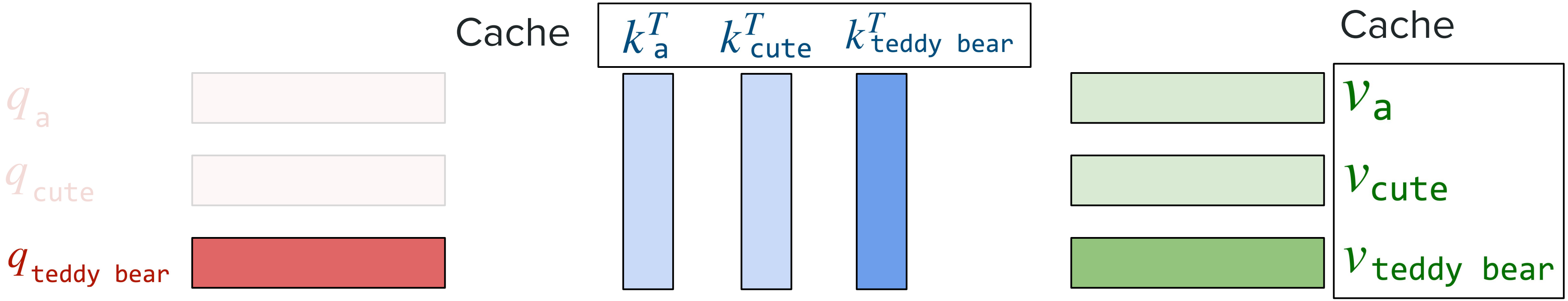
Idea. Keep keys and values in a cache.



Reusing computations with KV caching

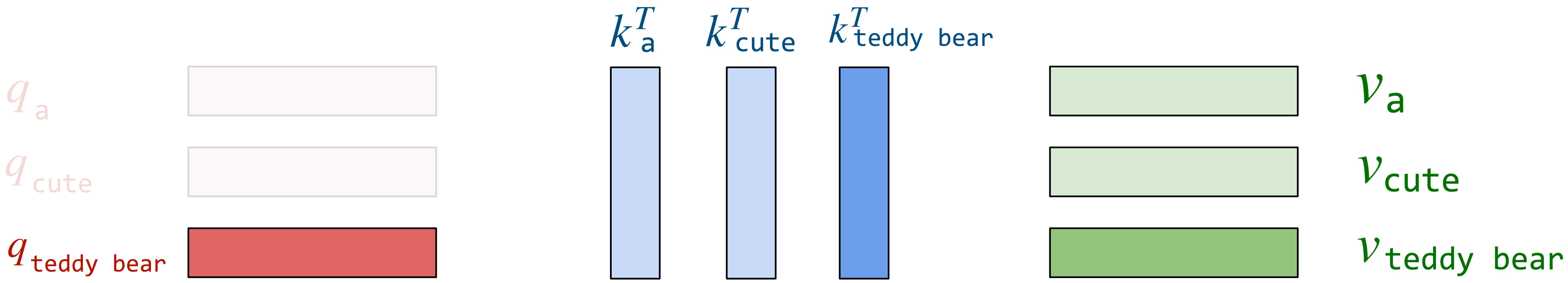
Motivation. New token needs to interact with all previous tokens.

Idea. Keep keys and values in a cache.

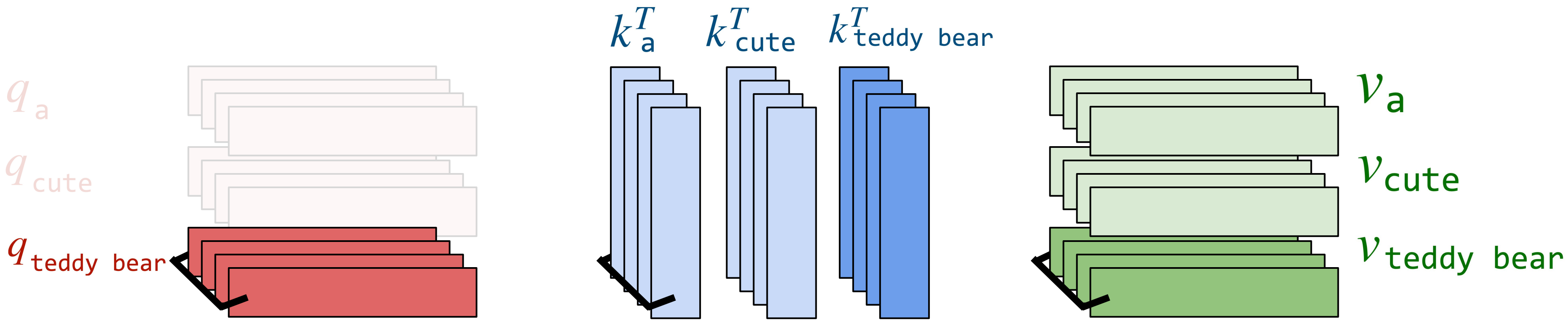


KV caching = **K**ey-**V**alue **Caching**

Sharing attention heads



Sharing attention heads



Number of **query heads**

Number of **key heads**

Number of **value heads**

In vanilla MHA, Number of **query heads** = Number of **key heads** = Number of **value heads** = h

Sharing attention heads

Idea. Share key/value attention heads within groups of queries

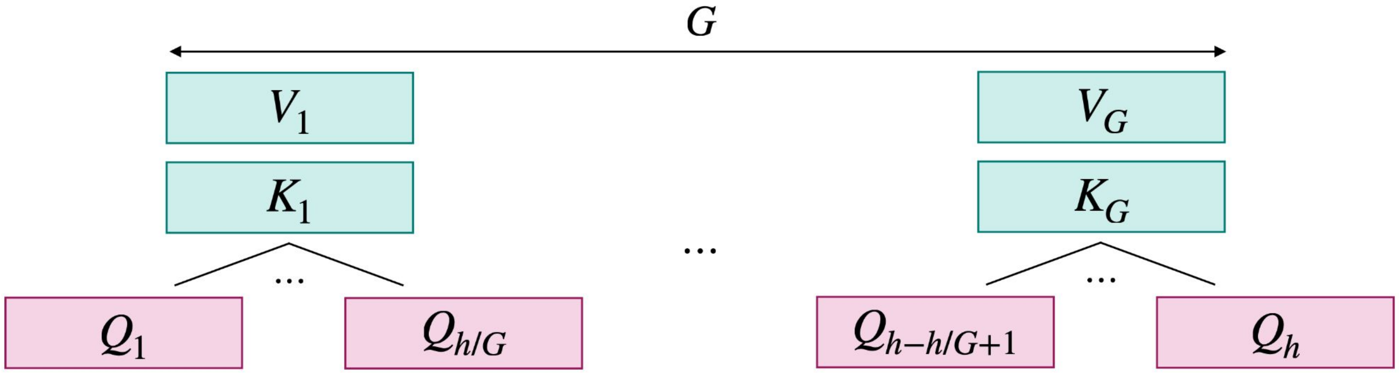


Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Sharing attention heads

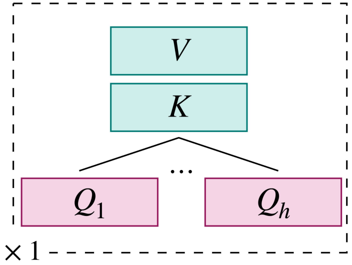
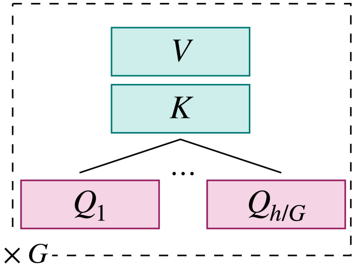
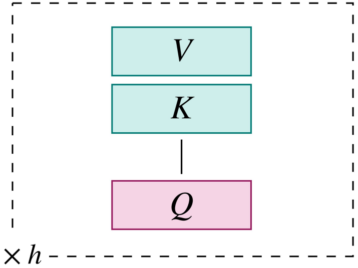
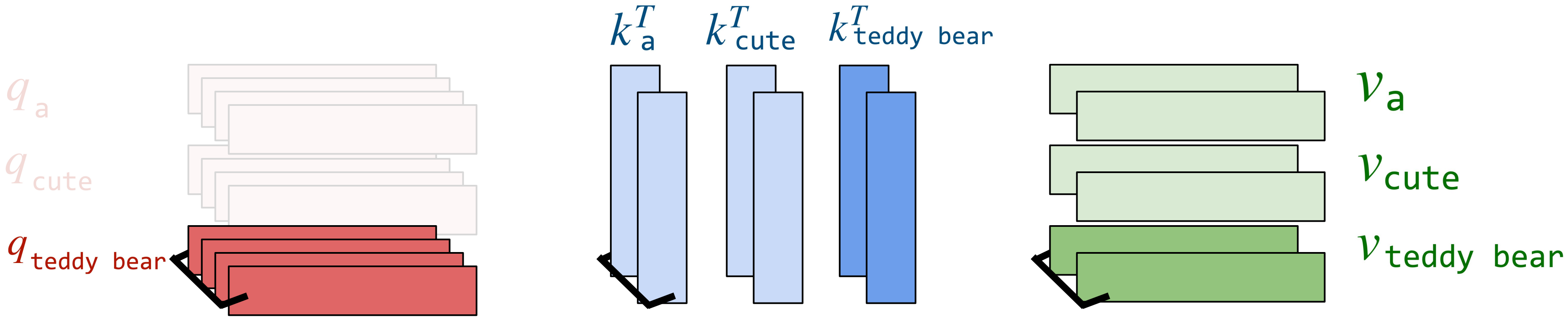
$G = 1$	**M**ulti-**Q**uery **A**ttention (MQA)	
$1 < G < h$	**G**roup-**Q**uery **A**ttention (GQA)	
$G = h$	**M**ulti-**H**ead **A**ttention (MHA)	

Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Sharing attention heads with GQA



Number of **query heads** = h

Number of **key heads** = G < h

Number of **value heads** = G < h

Manage memory with PagedAttention

Observation. Lots of memory waste when storing KV cache.

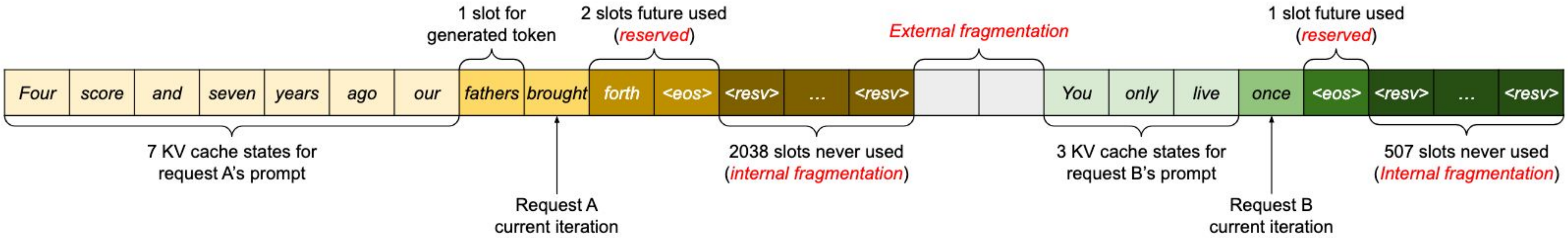


Figure from "Efficient Memory Management for Large Language Model Serving with PagedAttention", by Kwon et al., 2023.

Manage memory with PagedAttention

Idea. Store K and V in non-contiguous space to minimize wasted memory.

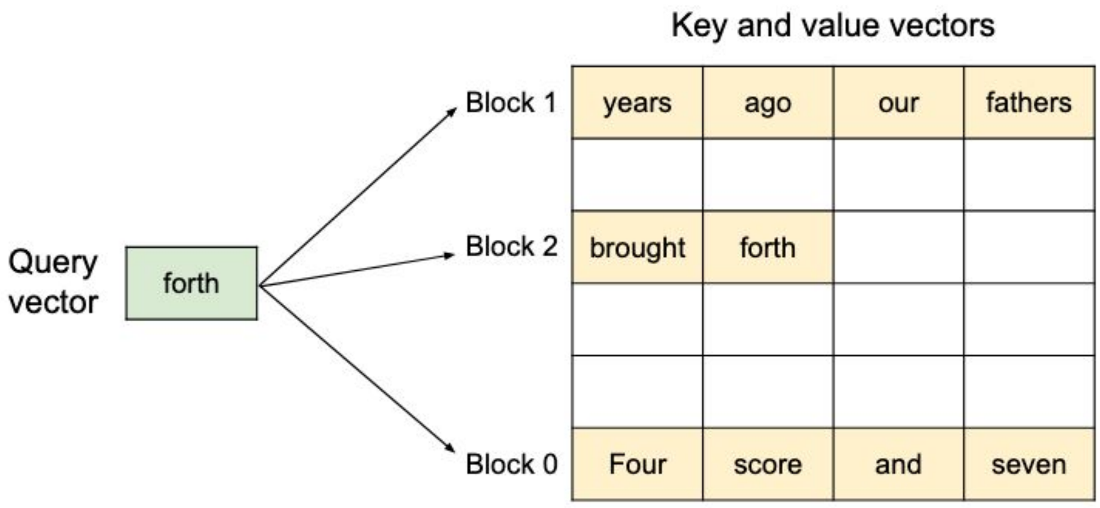
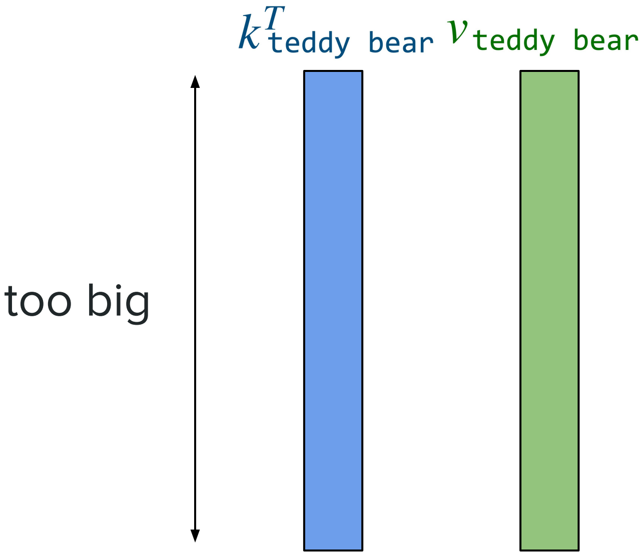


Figure from "Efficient Memory Management for Large Language Model Serving with PagedAttention", by Kwon et al., 2023.

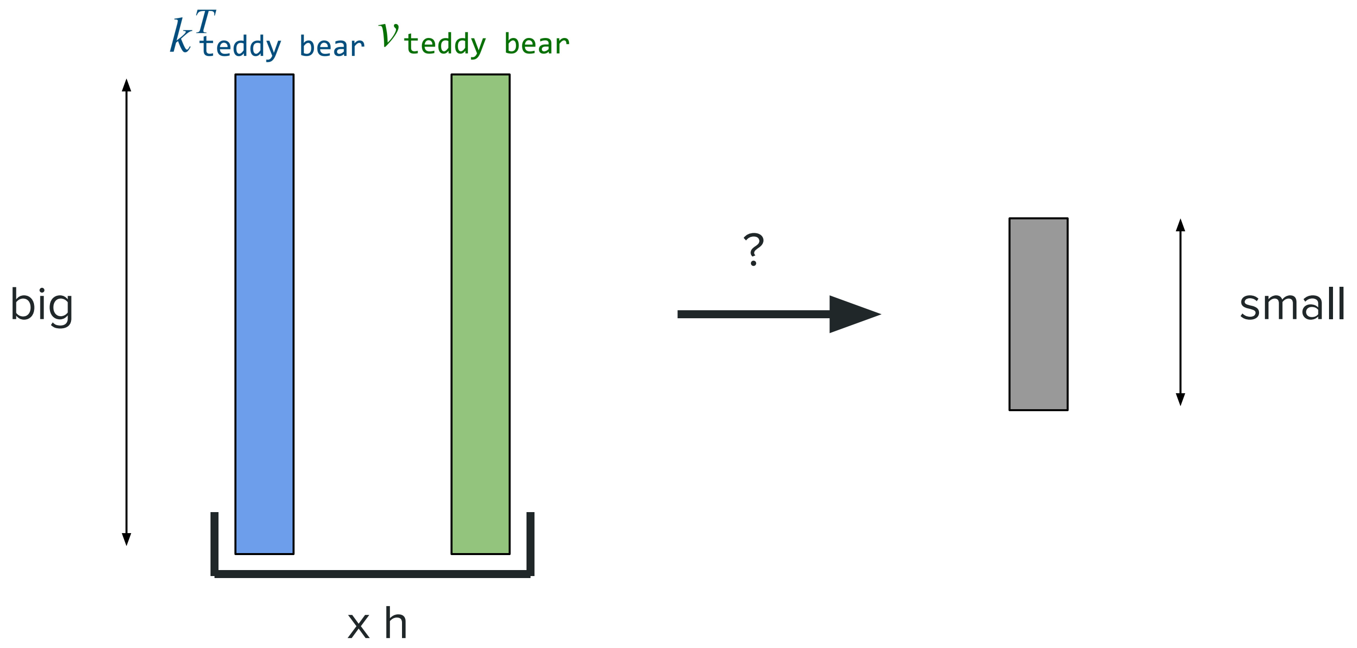
Reduce memory of KV cache with latent attention

Goal. Reduce dimension of K and V stored in memory.



Reduce memory of KV cache with latent attention

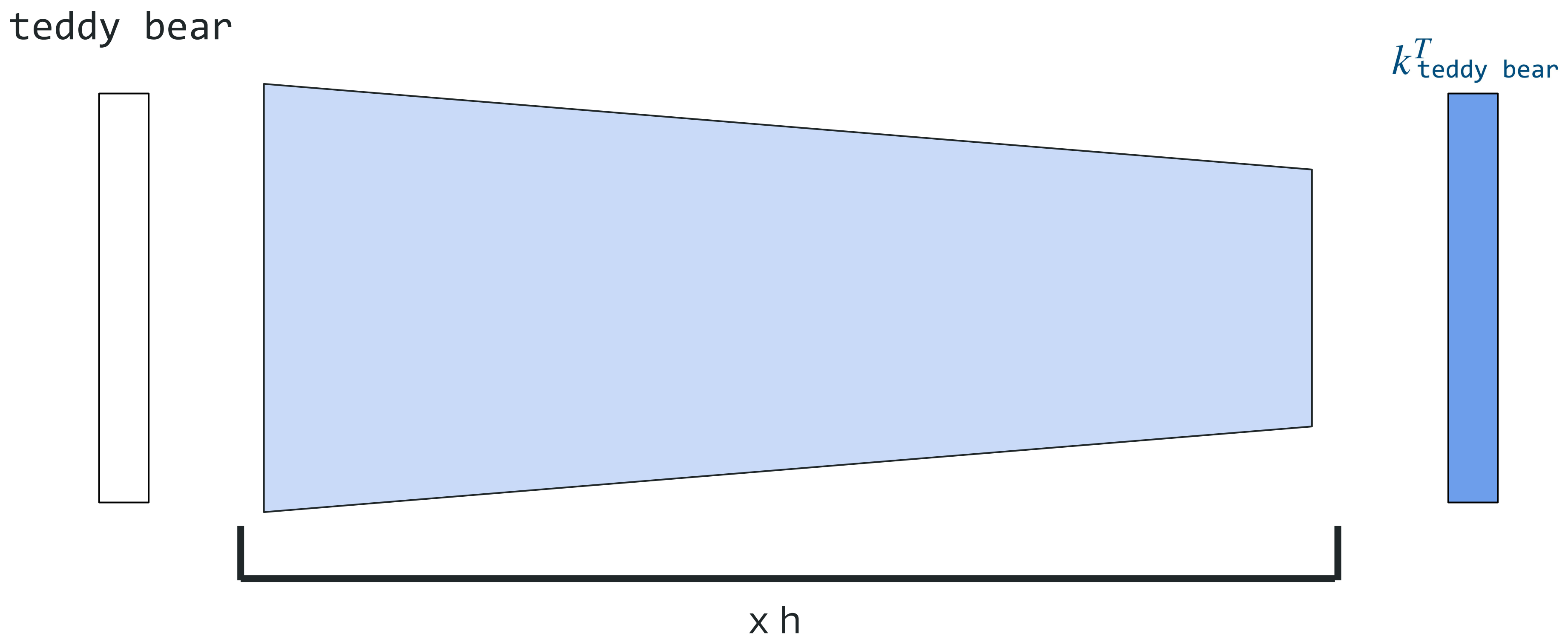
Solution. Store compressed representations instead!



"*DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model*", *by DeepSeek, 2023.*

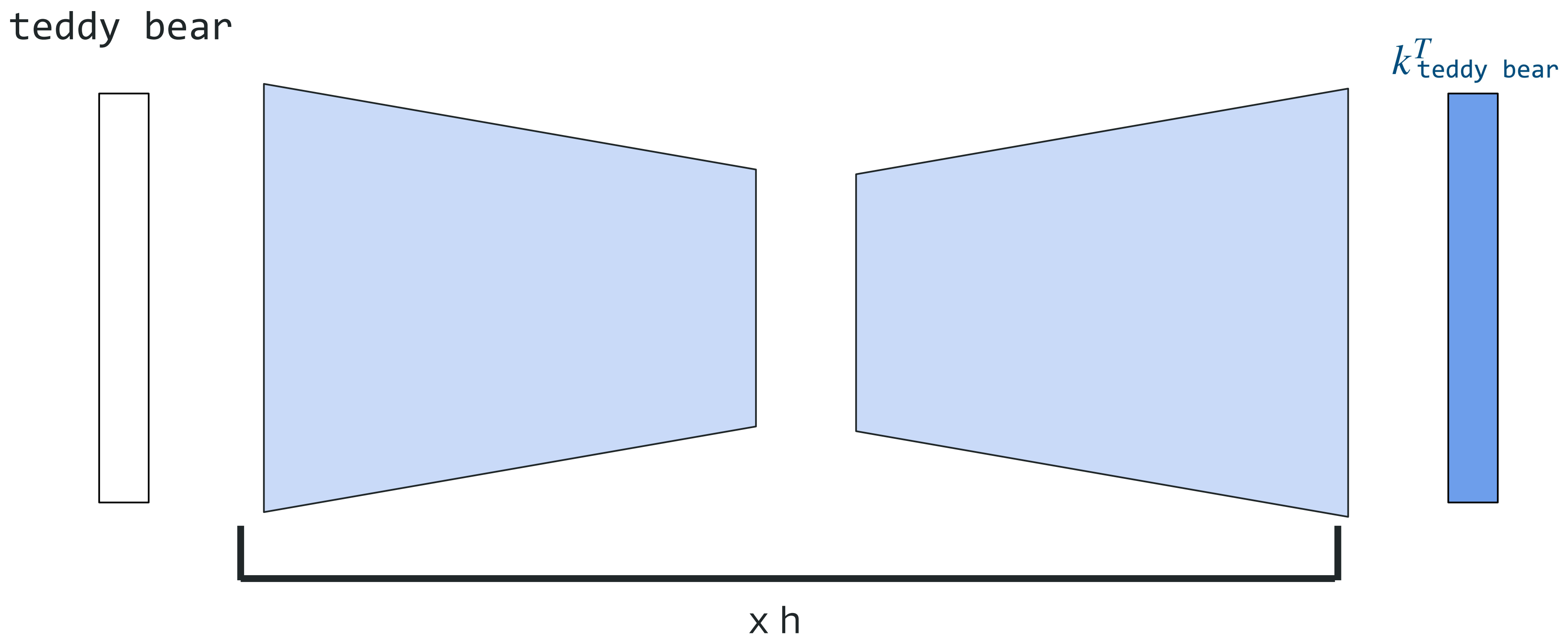
Reduce memory of KV cache with latent attention

BEFORE



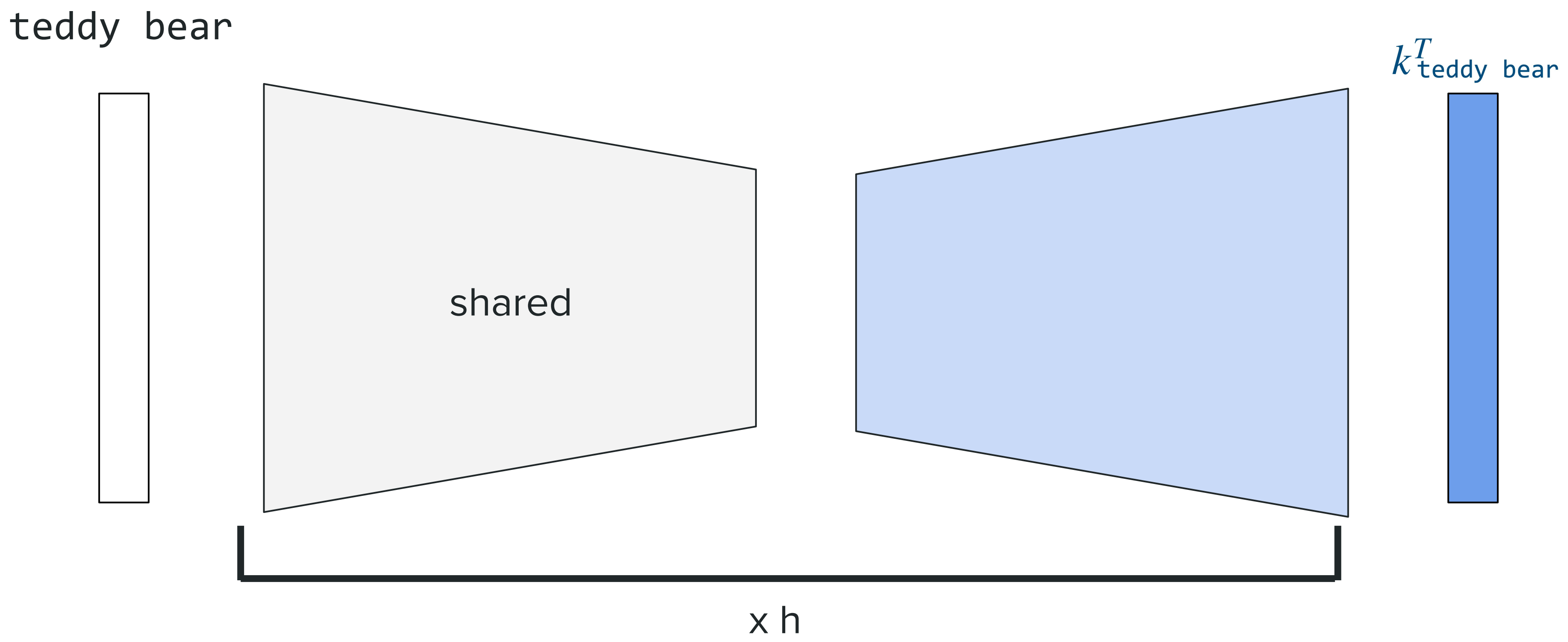
"DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model", by DeepSeek, 2023.

Reduce memory of KV cache with latent attention



"DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model", by DeepSeek, 2023.

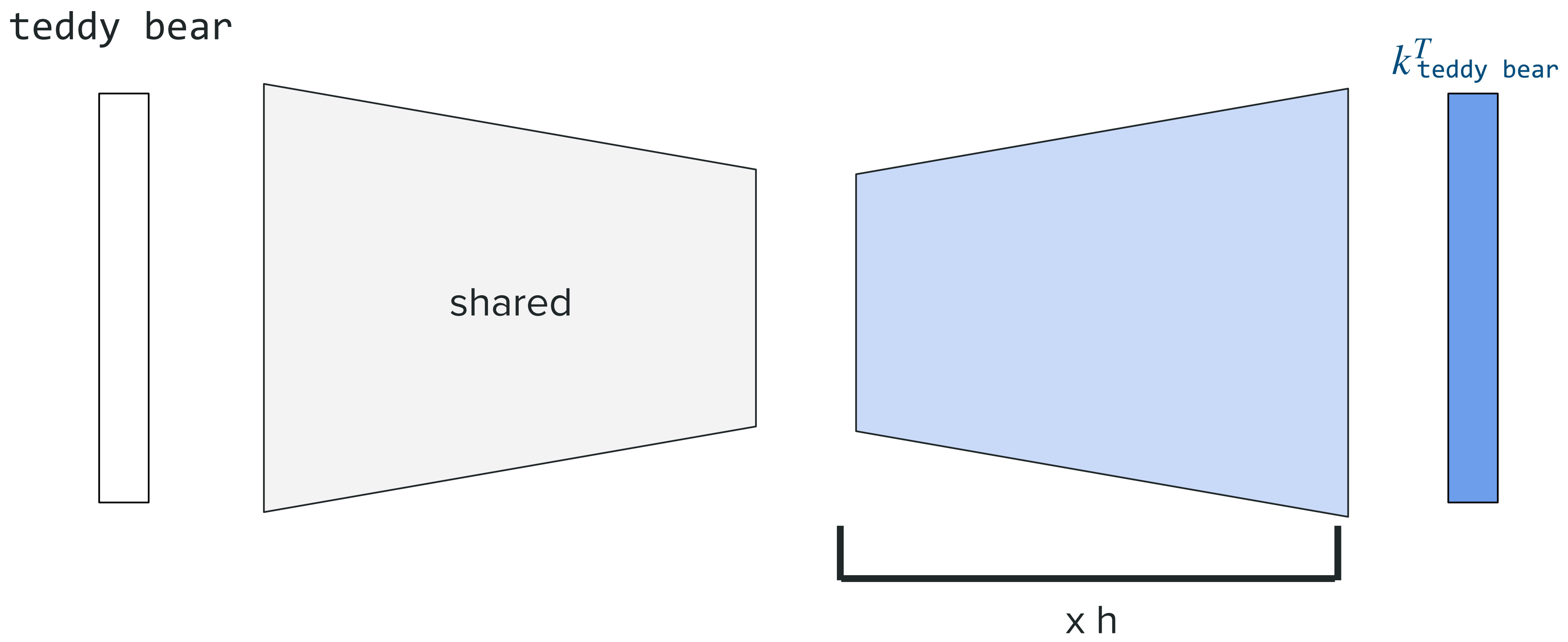
Reduce memory of KV cache with latent attention



"DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model", by DeepSeek, 2023.

Reduce memory of KV cache with latent attention

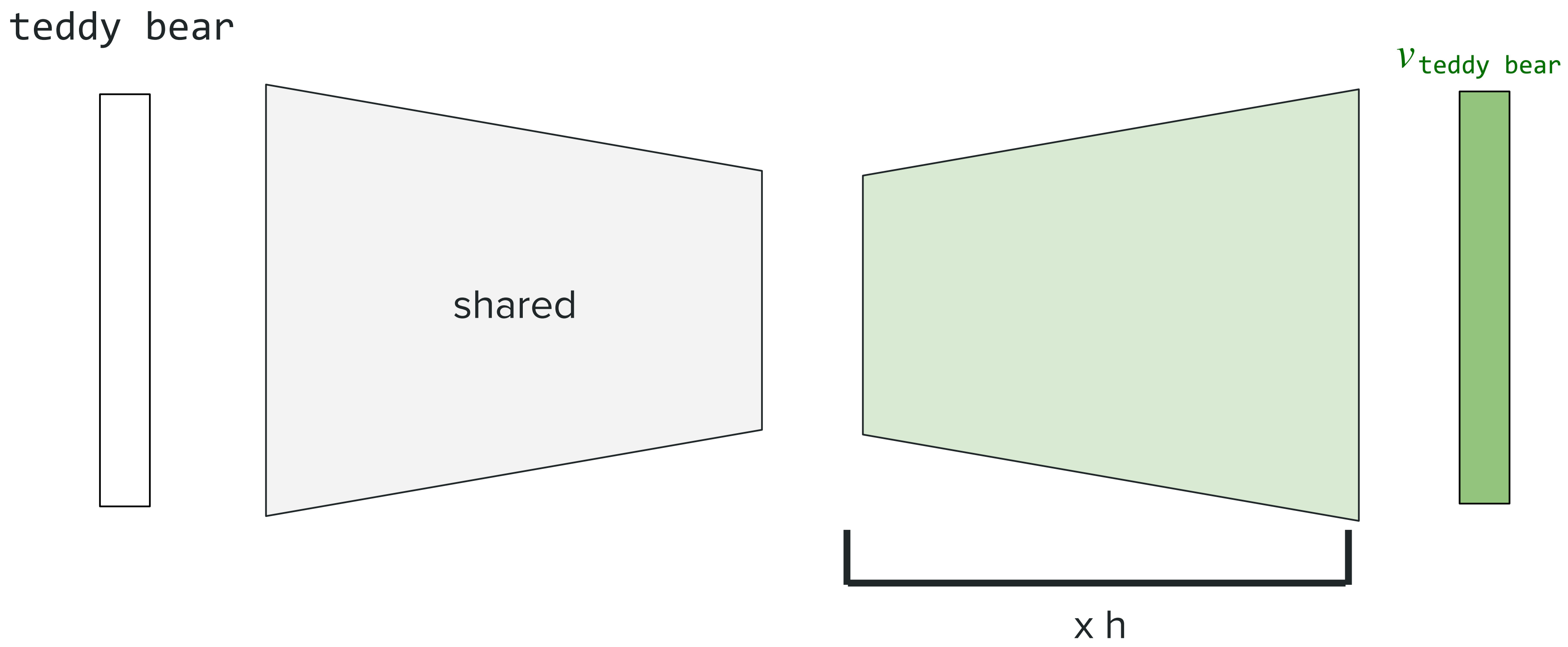
AFTER



"DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model", by DeepSeek, 2023.

Reduce memory of KV cache with latent attention

AFTER



Token generation tricks

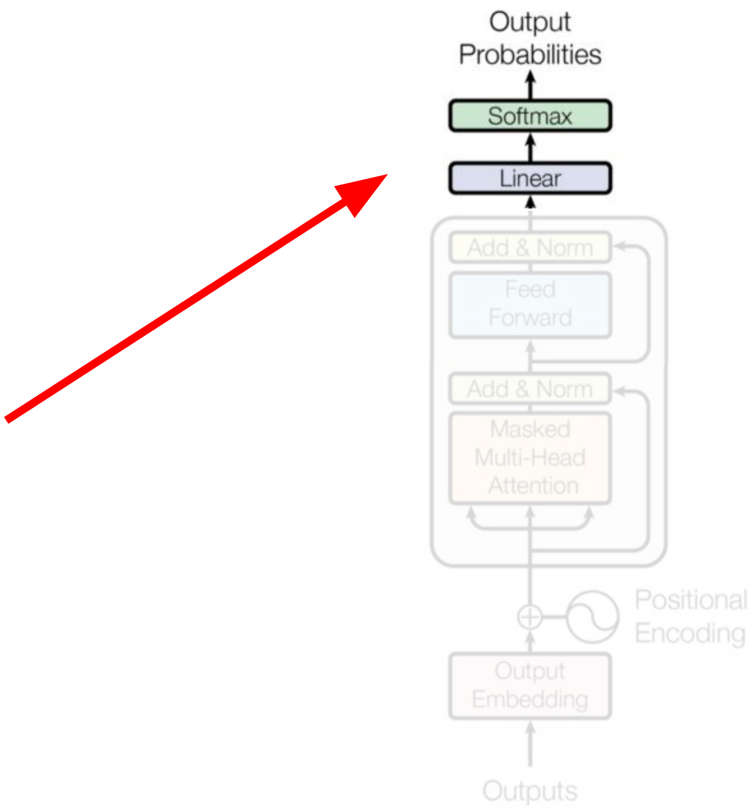


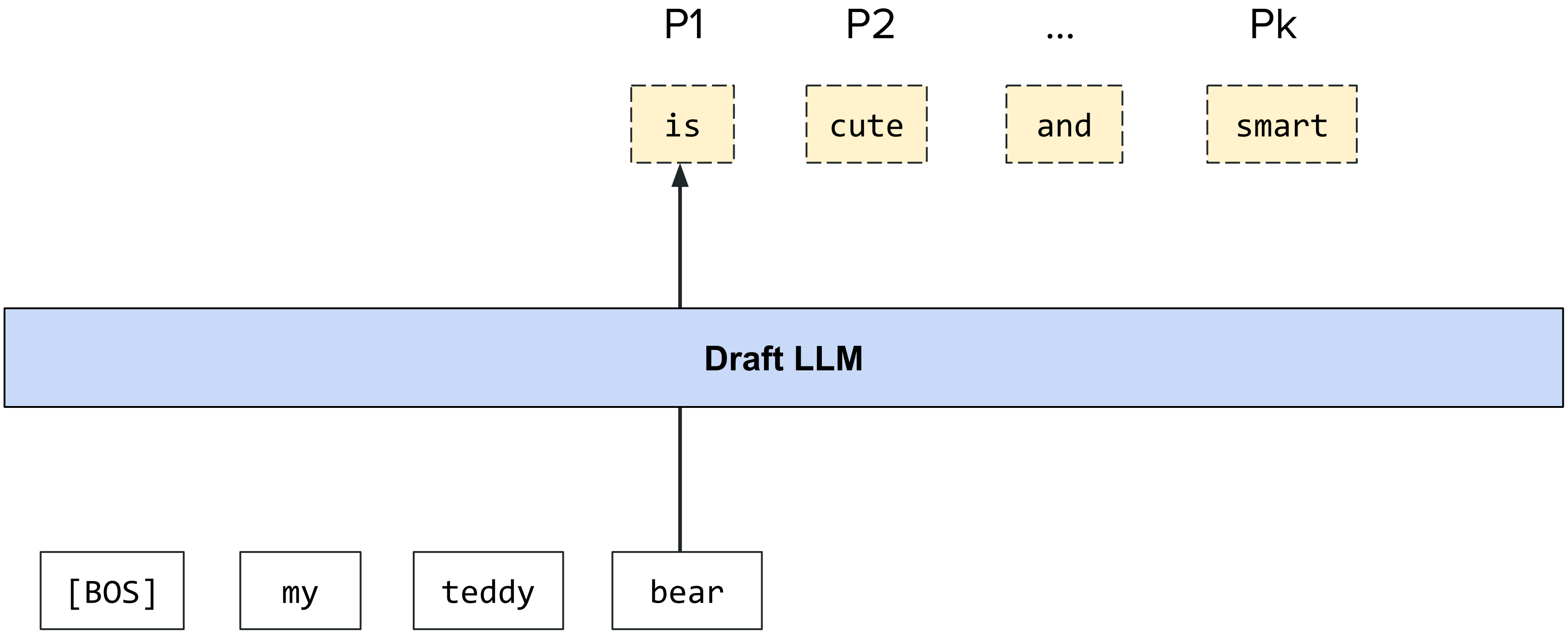
Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Speeding up decoding with speculative decoding

Idea. Use a draft (small) model to generate tokens validated by a target (big) model.

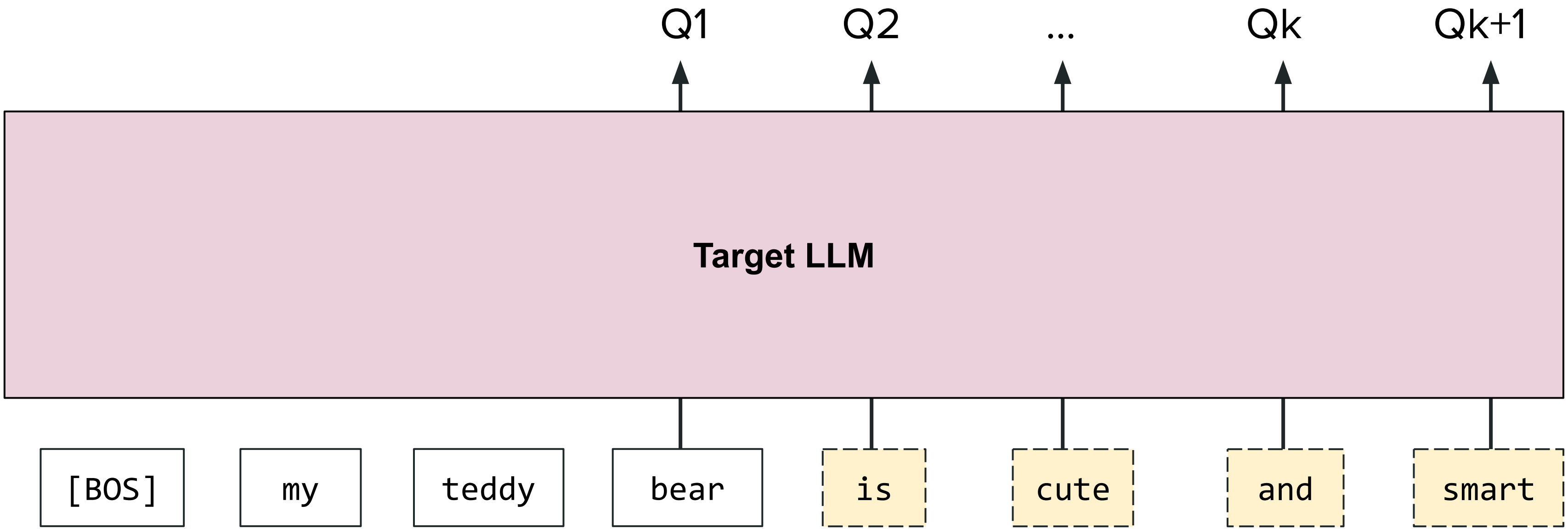
"Accelerating Large Language Model Decoding with Speculative Sampling", Chen et al., 2023.

Speeding up decoding with speculative decoding



"*Accelerating Large Language Model Decoding with Speculative Sampling*", *Chen et al., 2023.*

Speeding up decoding with speculative decoding



 "*Accelerating Large Language Model Decoding with Speculative Sampling*", *Chen et al., 2023.*

Speeding up decoding with speculative decoding

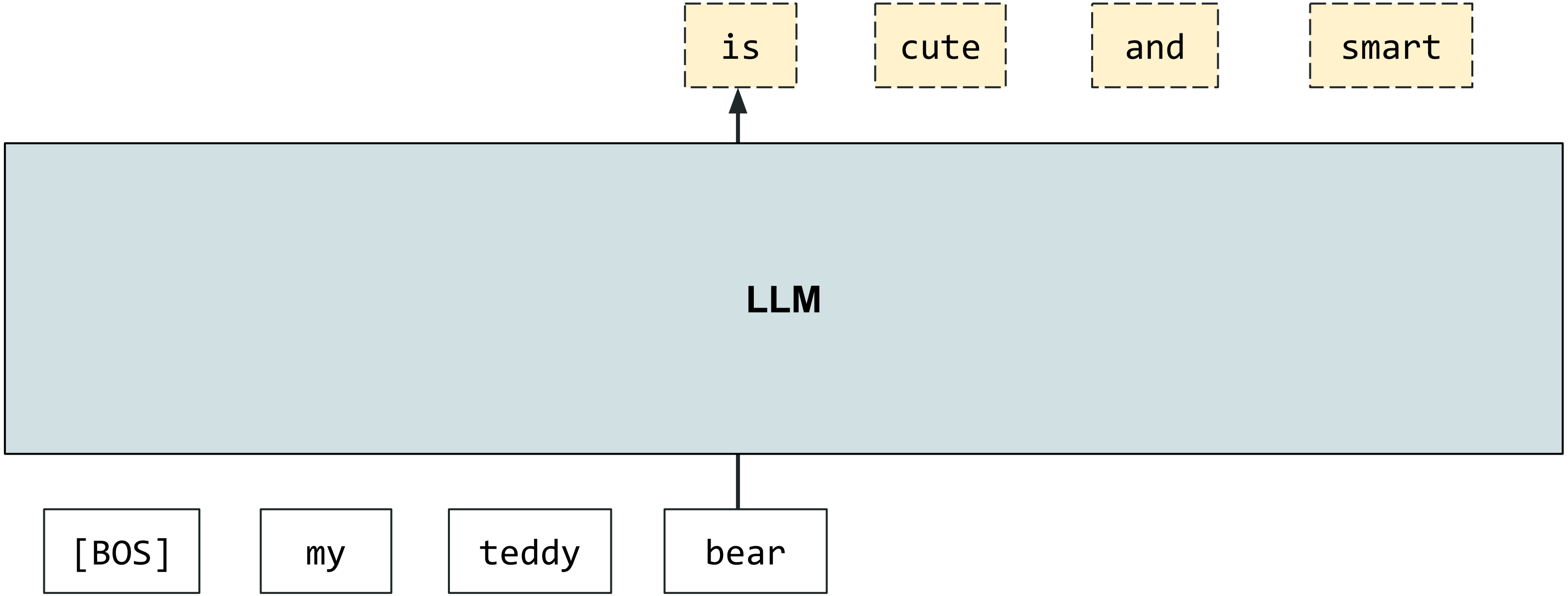
Sampling algorithm. We distinguish the following cases:

- If Qi(token) >= Pi(token) token ✅
- Otherwise
  - with probability Qi(token) / Pi(token) token ✅
  - with probability 1 - Qi(token) / Pi(token) token ❌

If a rejection happens, re-sample next token with distribution [Qi - Pi]+ and exit

Generate several tokens at once via MTP

MTP = **M**ulti-**T**oken **P**rediction



"Better & Faster Large Language Models via Multi-token Prediction", Gloeckle et al., 2024.

Generate several tokens at once via MTP

Idea. Train k prediction heads: same draft and target models!

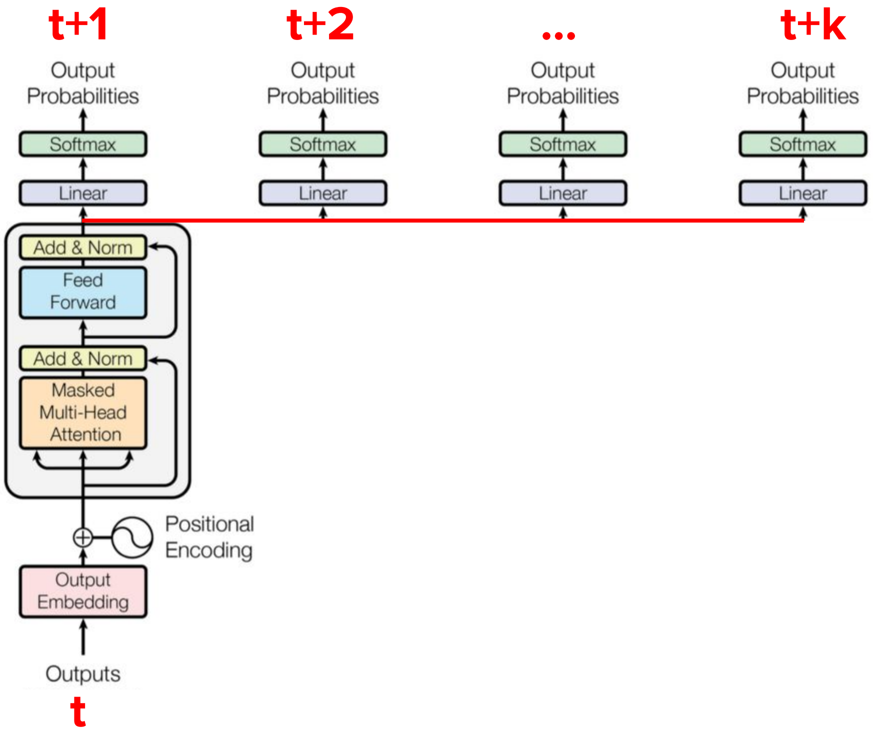


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Challenges

Categories. Many dimensions to optimize for.

"Exact" efficiency:

- Avoid redundancies
- Memory management
- Reformulate the math

Approximations:

- Architectural changes
- Embeddings representations
- Token prediction

Challenges **and some remedies**

Categories. Many dimensions to optimize for.

"Exact" efficiency:

- Avoid redundancies ← **KV cache**
- Memory management ← **PagedAttention**
- Reformulate the math ← **Speculative decoding**

Approximations:

- Architectural changes ← **Grouped query attention**
- Embeddings representations ← **Latent attention**
- Token prediction ← **Multi-token prediction**

Thank you for your attention!