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

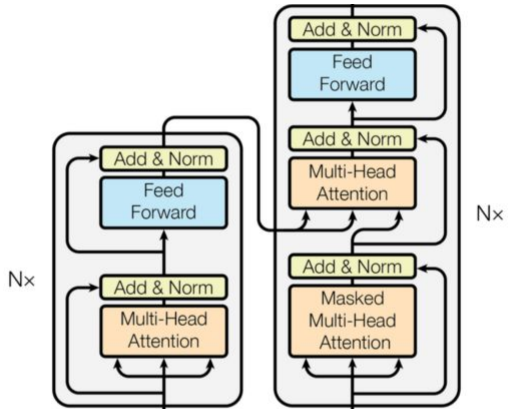


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

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

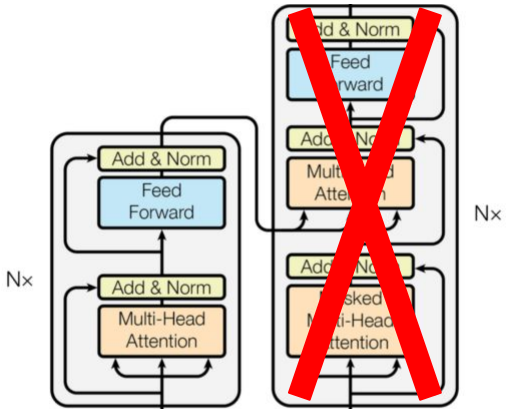


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

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

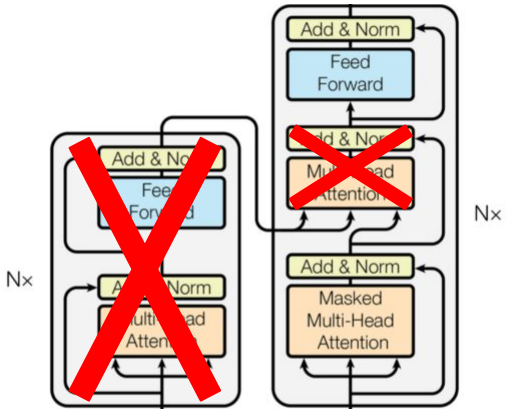


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

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

it needs atleast 100B of tokens in training dataset for benefits properly.

assigns probability to sequence of tokens, it always tries to predict the probability.

this kind of model are Prefrably trues longer planetes on training architecture and reactive huge computation of GPUs.

(but having optimized in also to train/inference in small computing device)

Terminology

LLM = Large **Language Model**

Terminology

LLM = Large **Language Model**

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

Terminology

LLM = Large Language Model

Terminology

LLM = **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

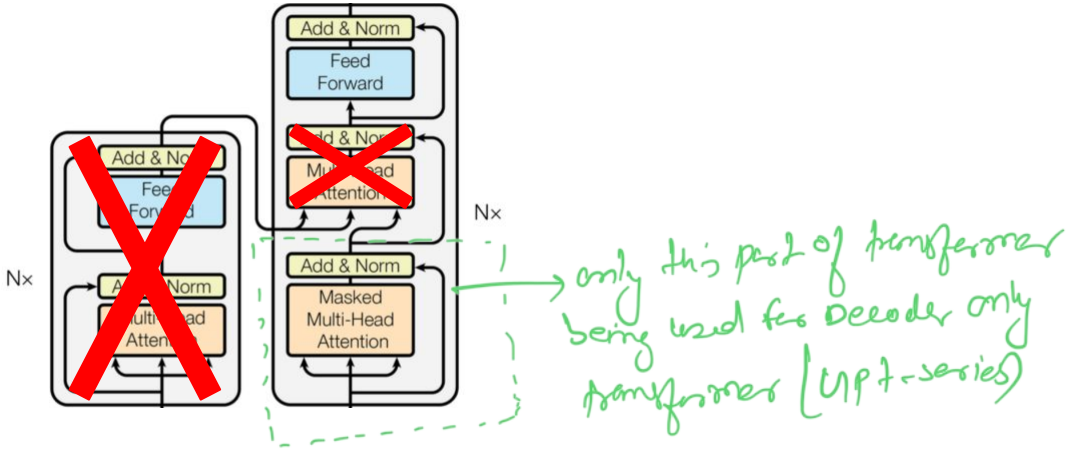


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Characteristics

Decoder-only **Transformer**-based model

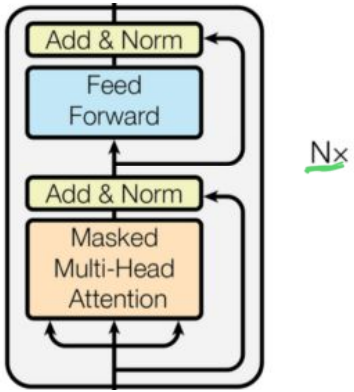
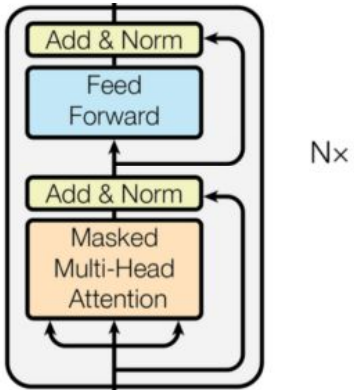


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

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

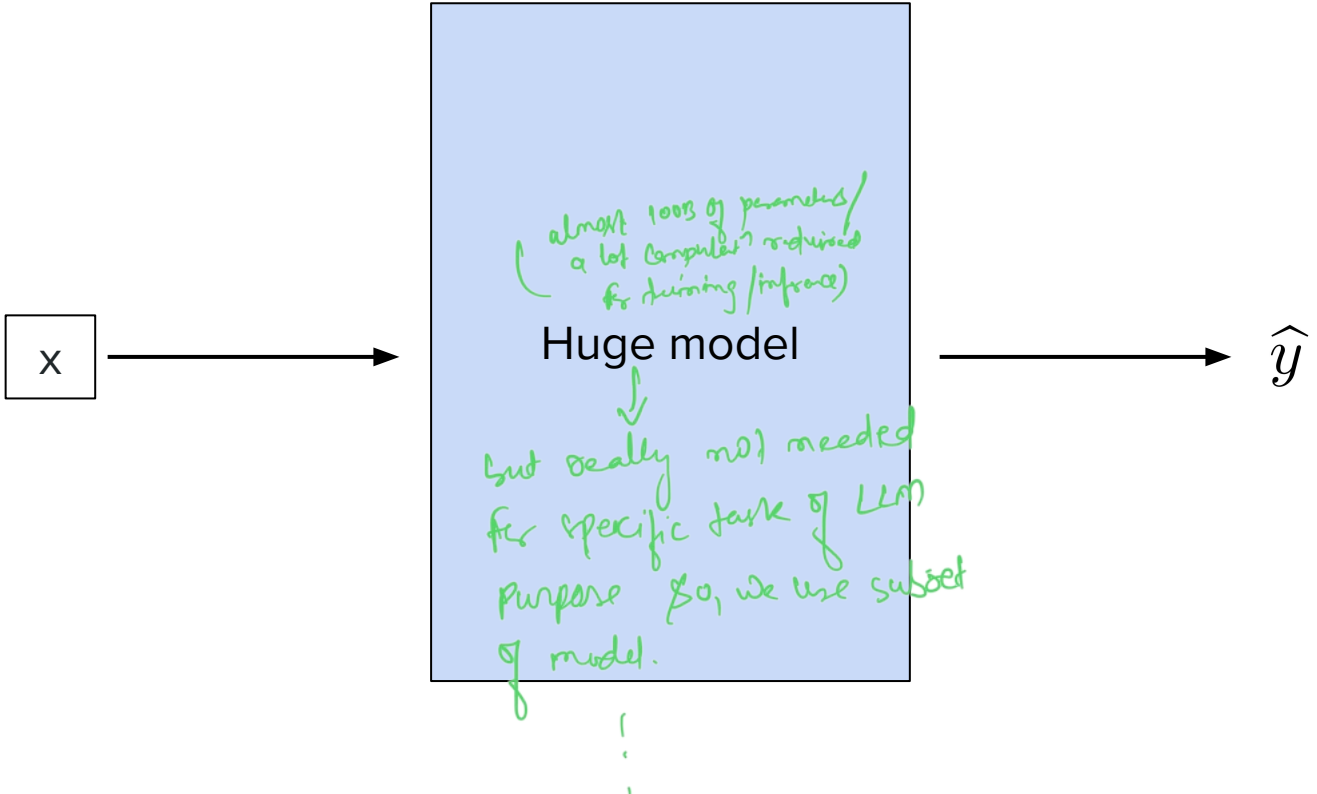
MoE-based LLMs

Response generation

Prompting strategies

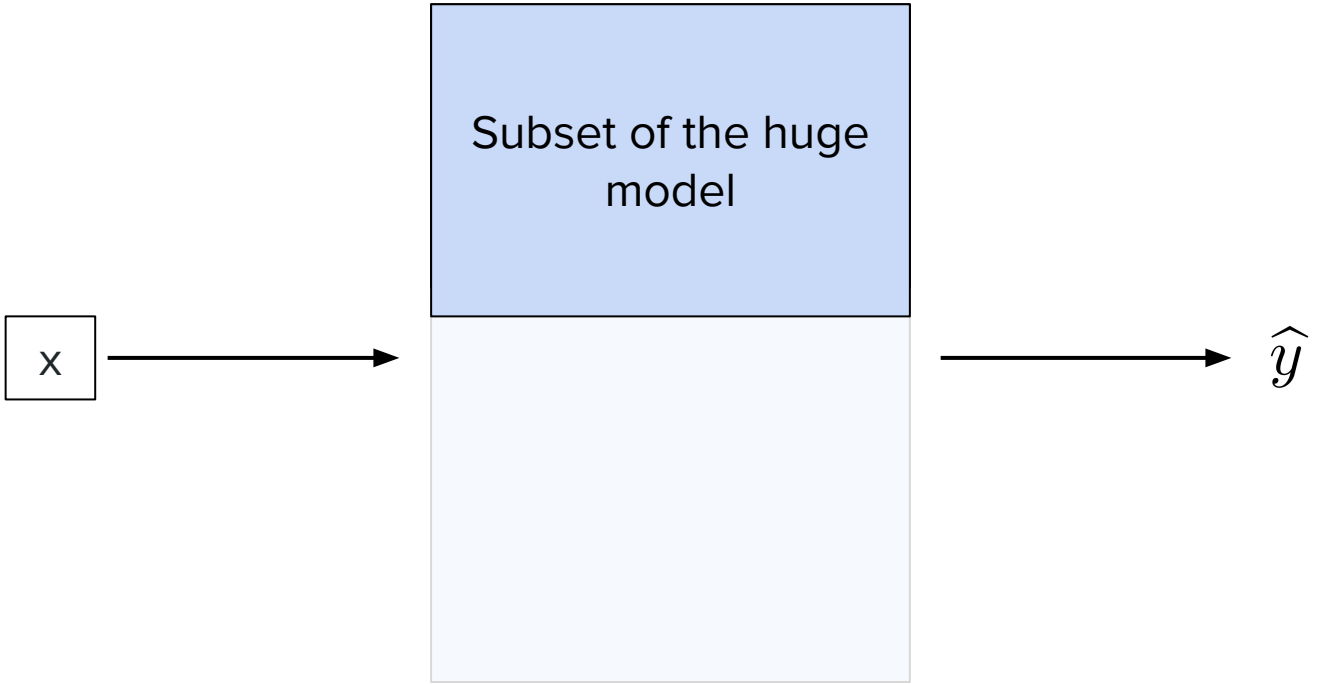
Inference optimizations

Motivation

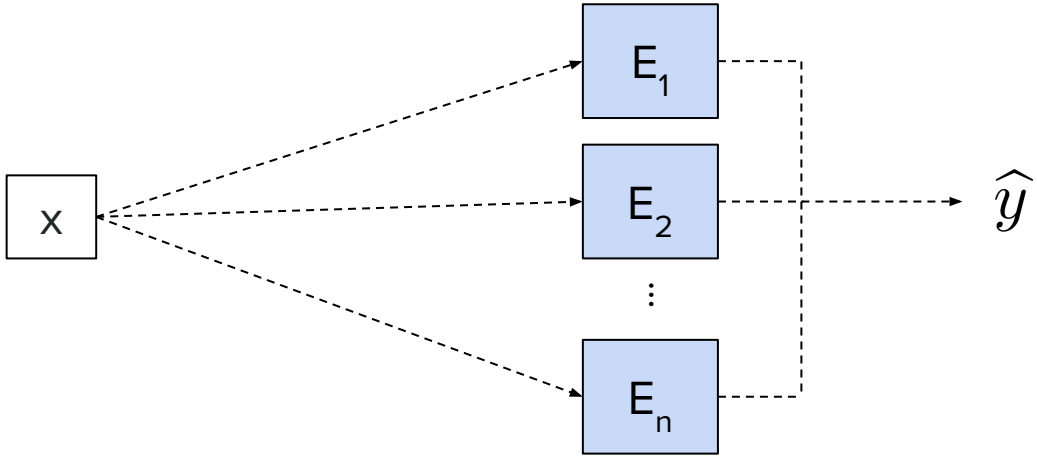


Motivation

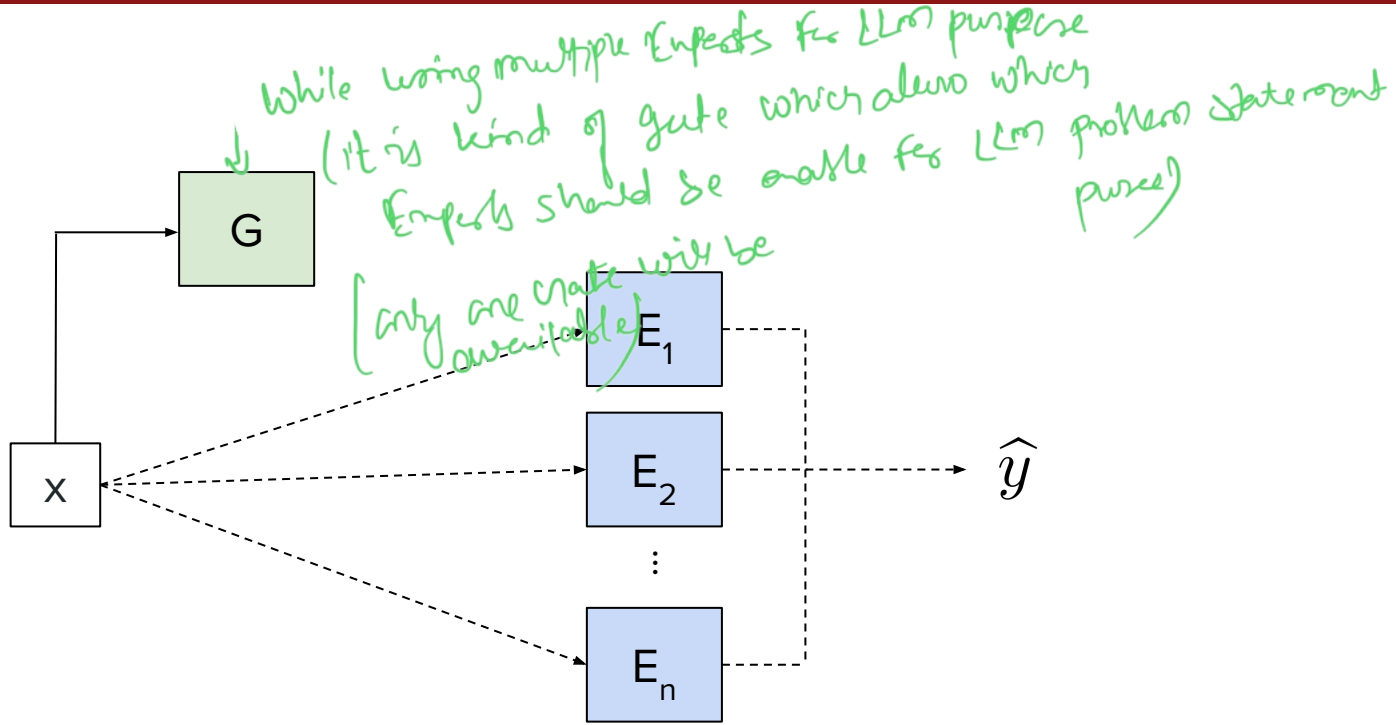
Idea. Not all weights are useful in the forward pass



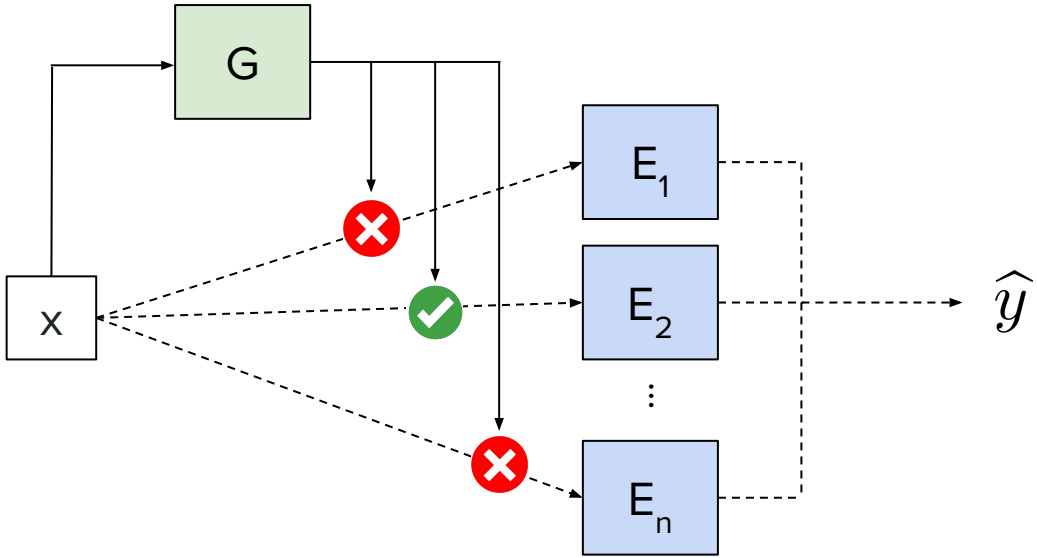
Motivation



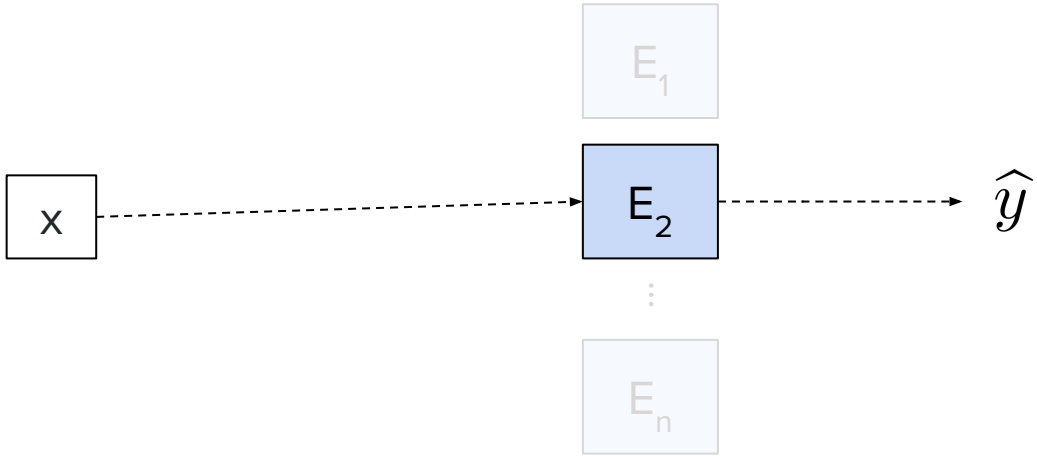
Motivation


While using multiple Experts for LLM purpose
(It is kind of gate which also which
Experts should be enable for LLM problem statement
pass)
G
[only one route will be
available]
E1
E2
En
x
ŷ

Motivation

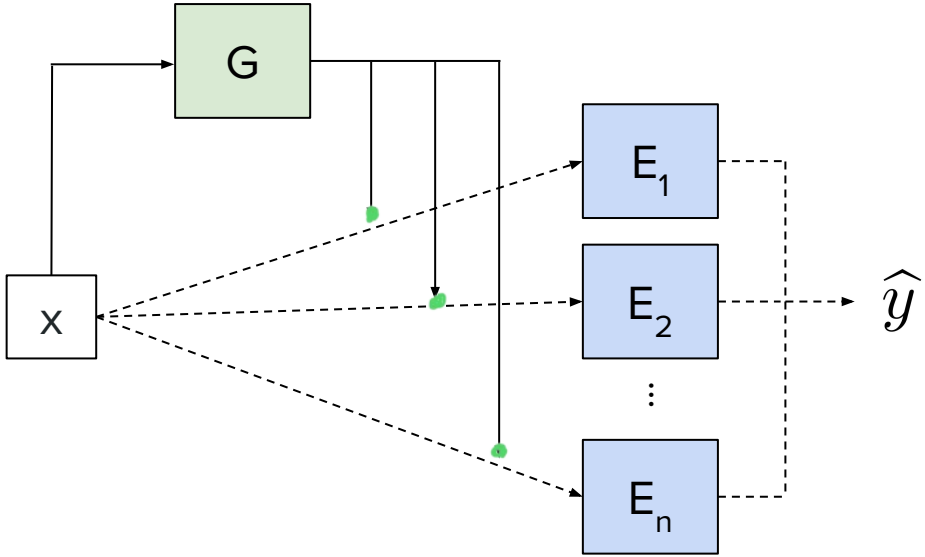


Motivation



Overview of MoEs

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



Weighted sum of Experts as per requirements for purpose

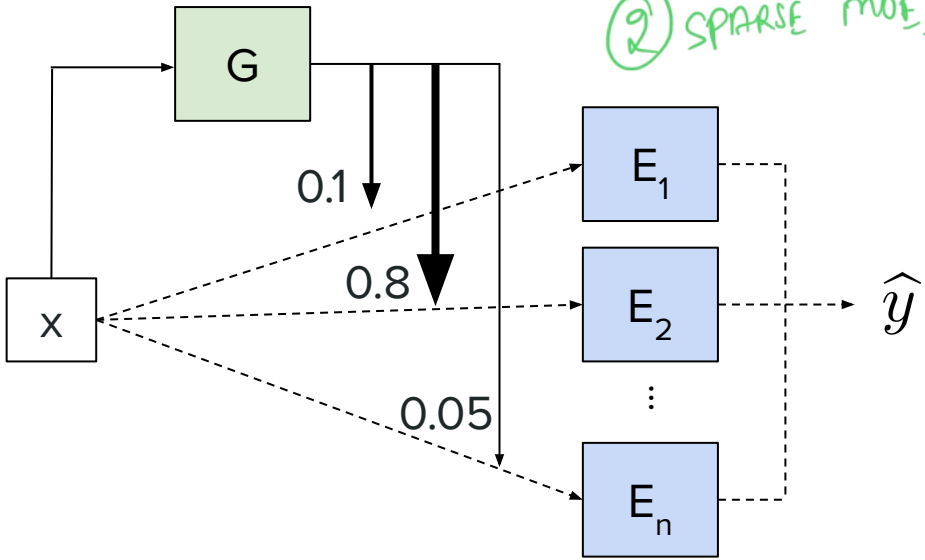
$$\hat{y} = \sum_{i=1}^{n} G(x)_i E_i(x)$$

Experts trained for (x)

Weight assign based on proficiency of Experts based on LLM requirements

Overview of MoEs

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

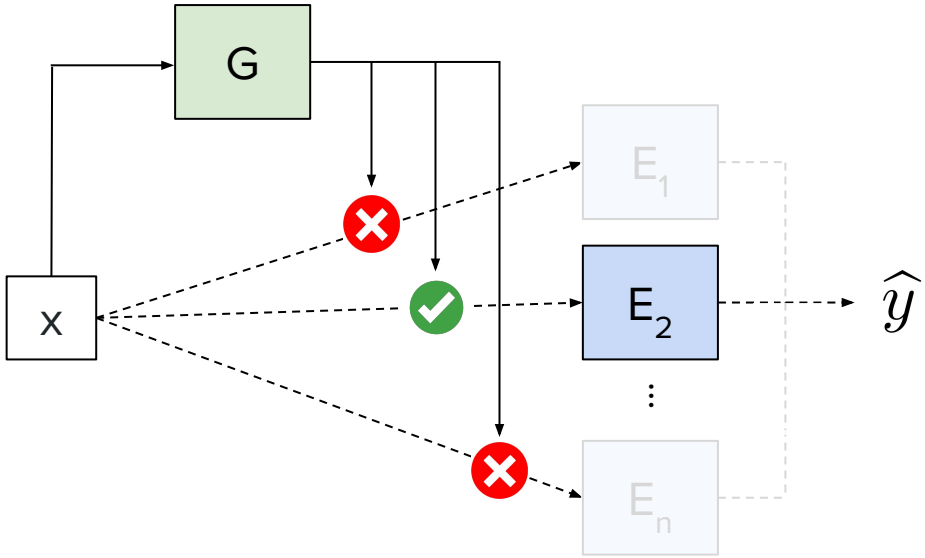


Dense MoE. Output is weighted average of **all** expert outputs.

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

Overview of MoEs

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

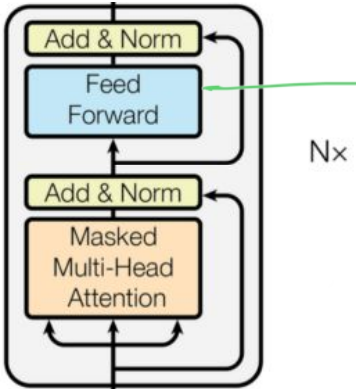


Sparse MoE. Output is weighted average of **selected** expert outputs.

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

Via **top-k** selection

MoE in Transformer-based models



responsible for combining multiple Experts

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

MoE in Transformer-based models

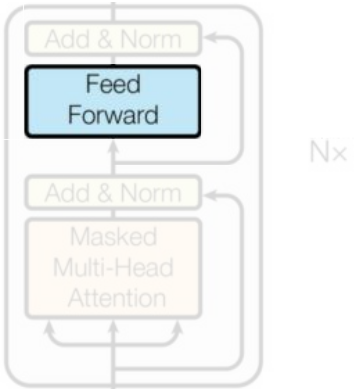
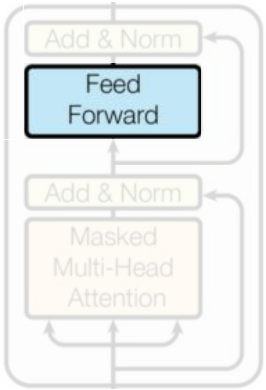


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

MoE in Transformer-based models



N×

Routing done **for each token**!

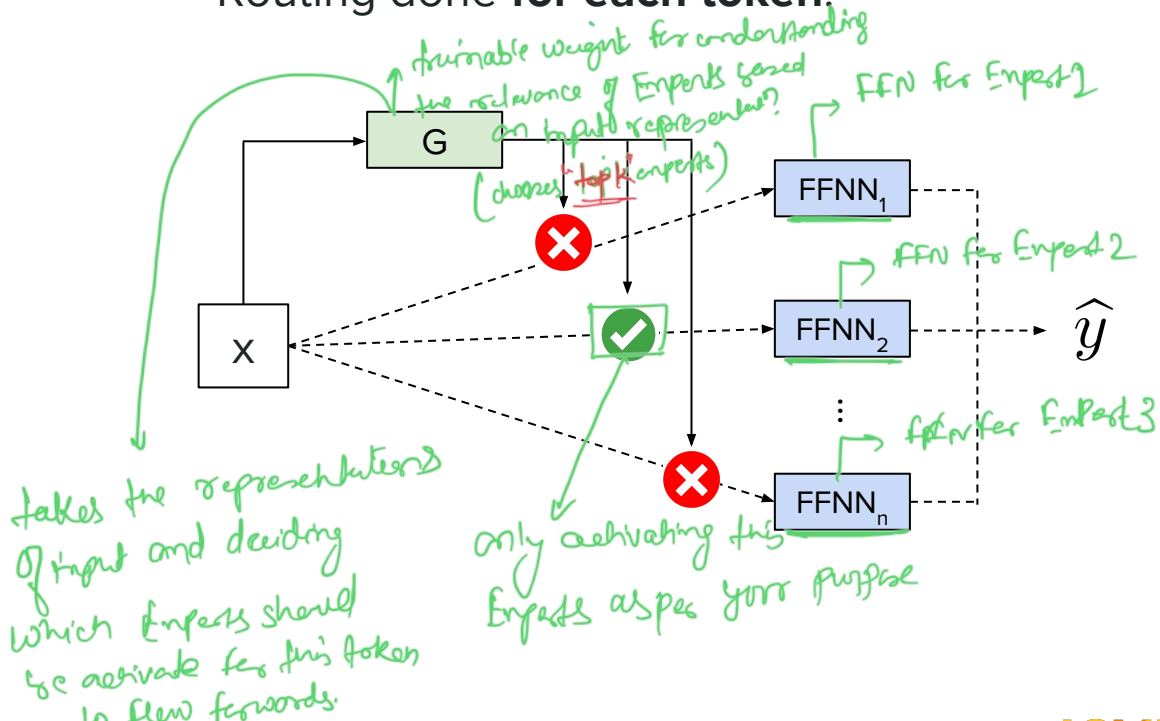


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Training challenges include routing collapse

Symptom. Same expert gets selected most of the time.

"routing collapse"

Problem of mixture of Experts.
Even Experts will have weight but as 1/2 only Experts always activated) So, if Fixes using changing the loss function and adding Extra term (α)

"*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:

total # of Experts.

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

Converging through uniform distribution across Experts.

f_i = fraction of tokens routed to expert i

P_i = average routing probability for expert i (function of gate output)

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

Interpreting experts

"Mixtral of Experts"

Layer 0

```
class MoeLayer(nn.Module):
    def __init__(self, experts: List[nn.Module],
        super().__init__()
        assert len(experts) > 0
        self.experts = nn.ModuleList(experts)
        self.gate = gate
        self.args = moe_args

    def forward(self, inputs: torch.Tensor):
        inputs_squashed = inputs.view(-1, inputs.
        gate_logits = self.gate(inputs_squashed)
        weights, selected_experts = torch.topk(
            gate_logits, self.args.num_experts_pe
        )
        weights = nn.functional.softmax(
            weights,
            dim=1,
            dtype=torch.float,
        ).type_as(inputs)
        results = torch.zeros_like(inputs_squashe
        for i, expert in enumerate(self.experts):
            batch_idx, nth_expert = torch.where(s
            results[batch_idx] += weights[batch_id
                inputs_squashed[batch_idx]
            )
        return results.view_as(inputs)
```

Each color represent an expert

(it defines which Experts is expertise on what kind of input representation)

Transformers & Large Language Models

LLM overview

MoE-based LLMs

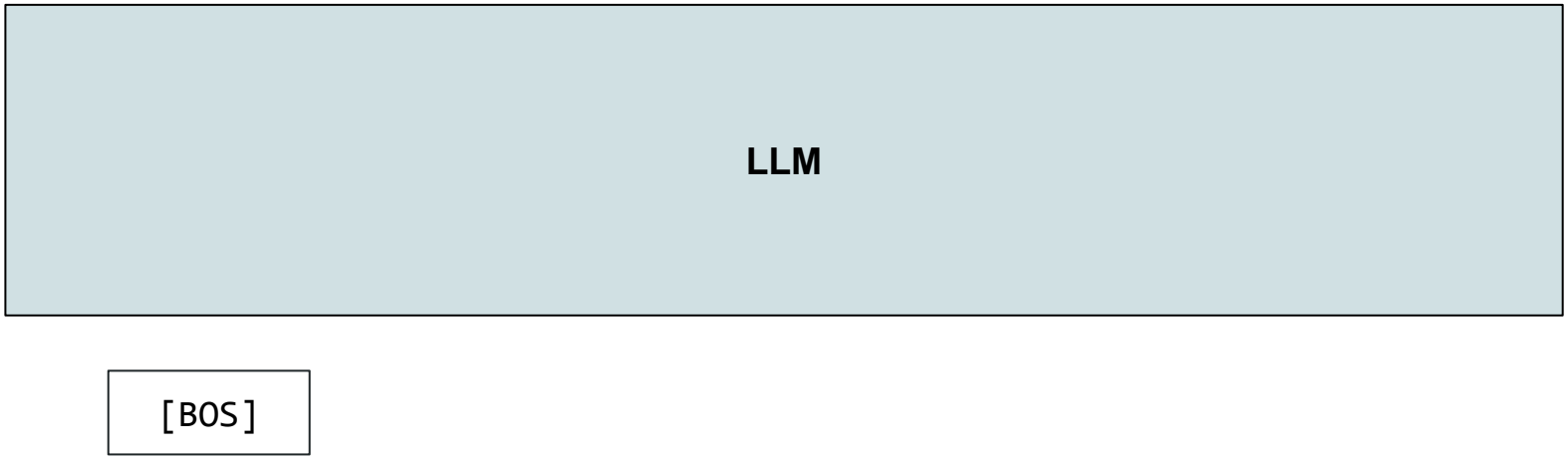
Response generation

Prompting strategies

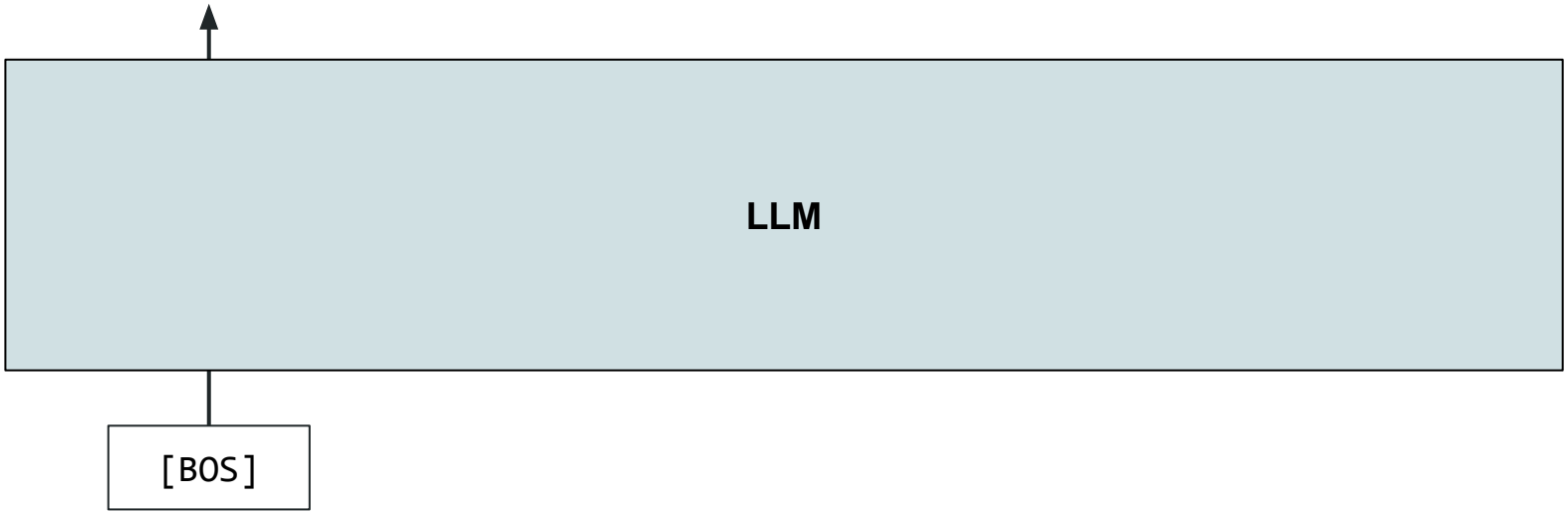
Inference optimizations

based on LLM/Experts how response is generated?

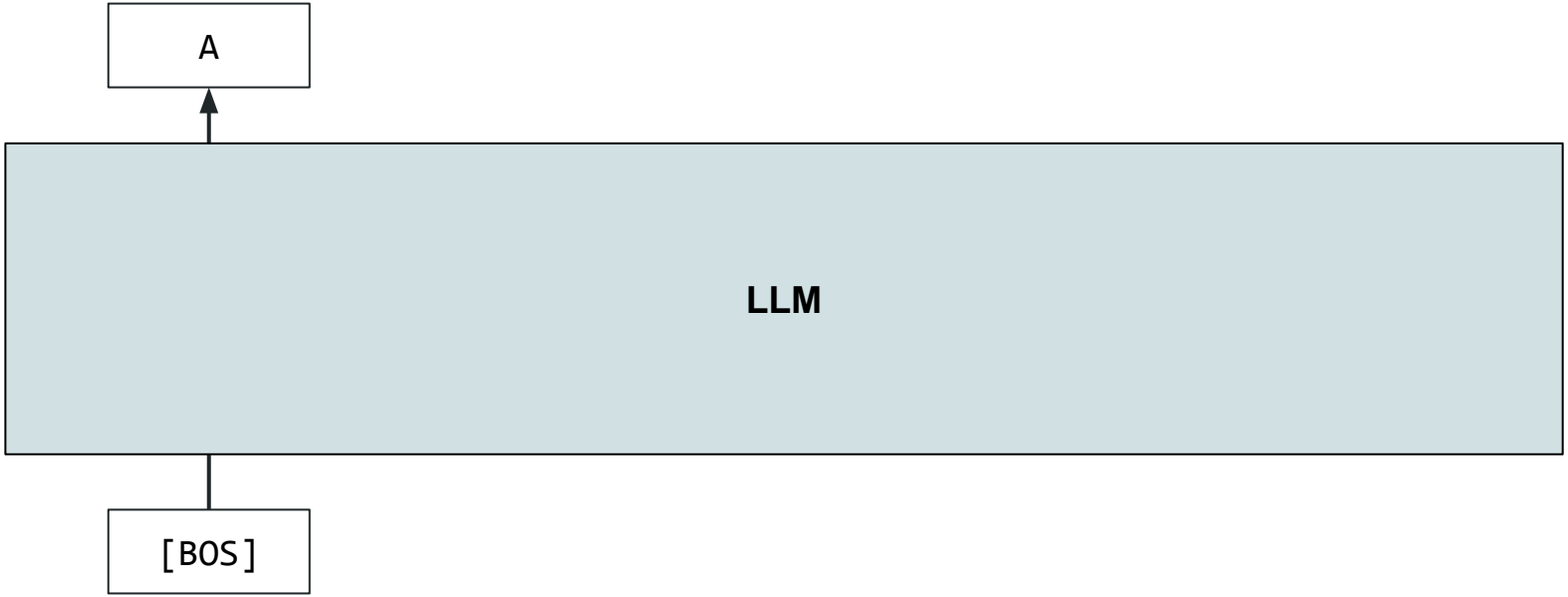
Next token prediction



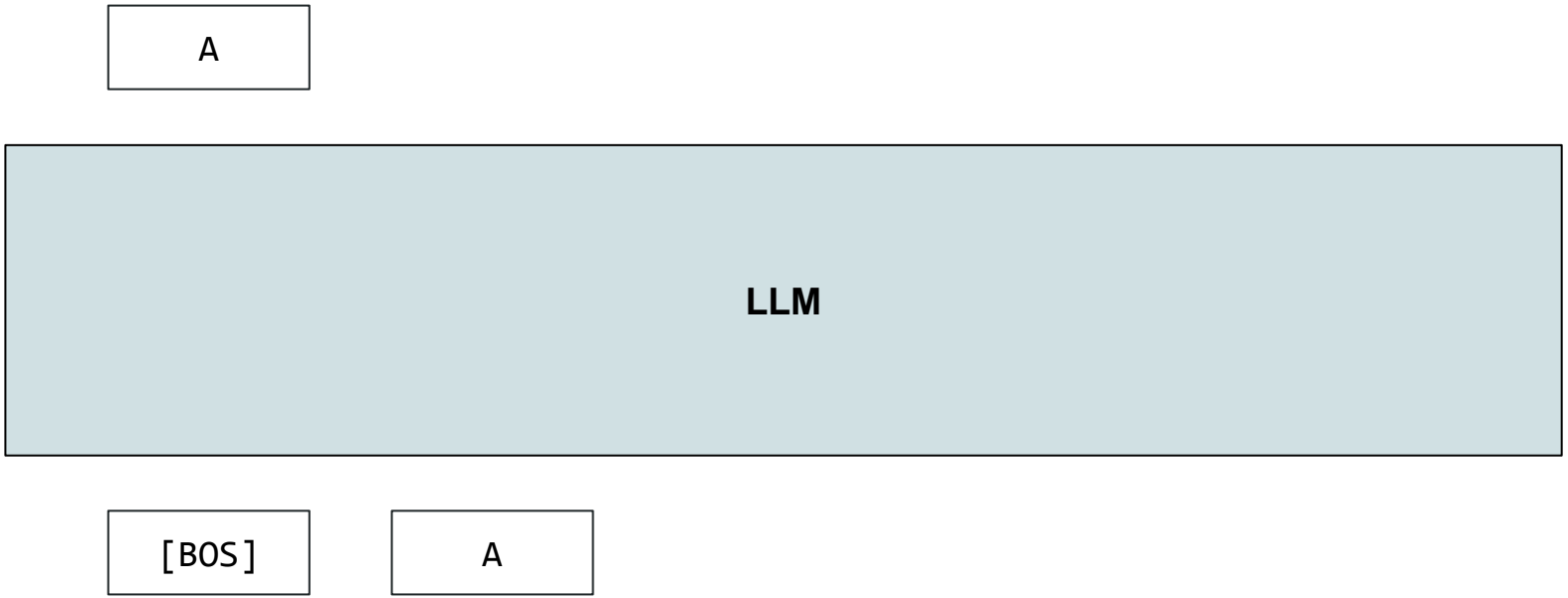
Next token prediction



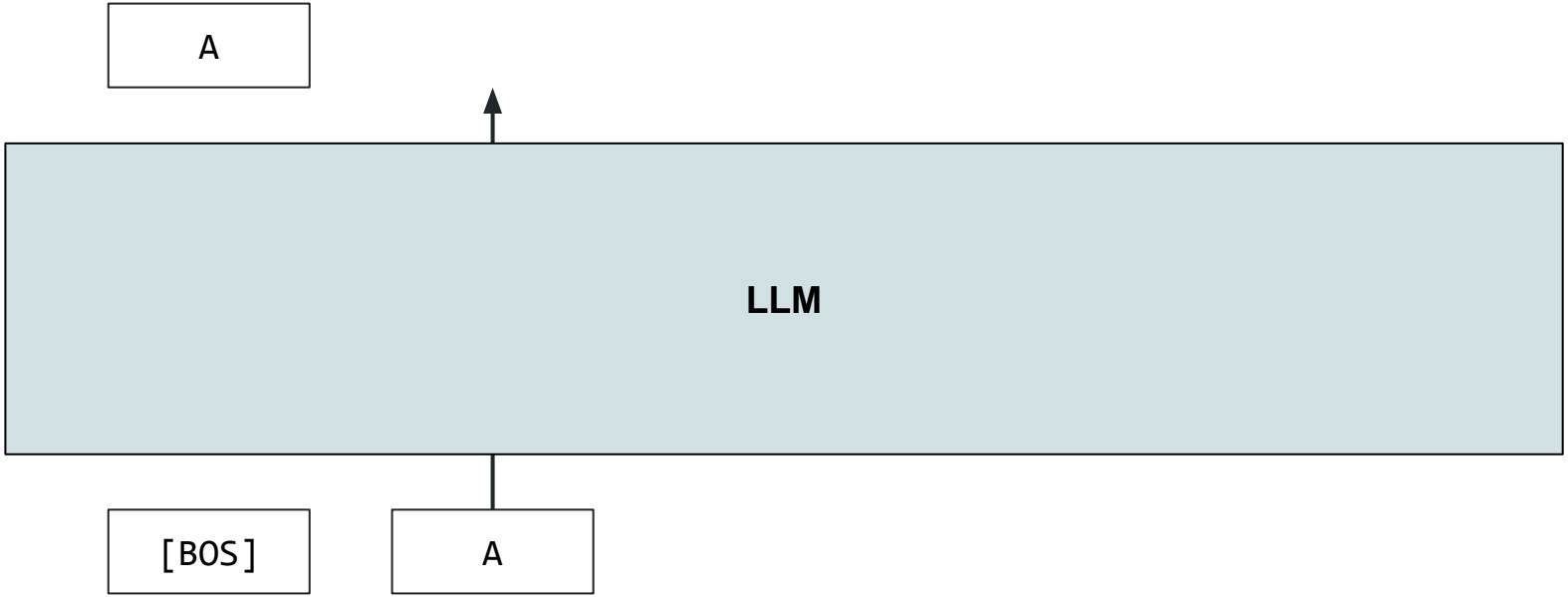
Next token prediction



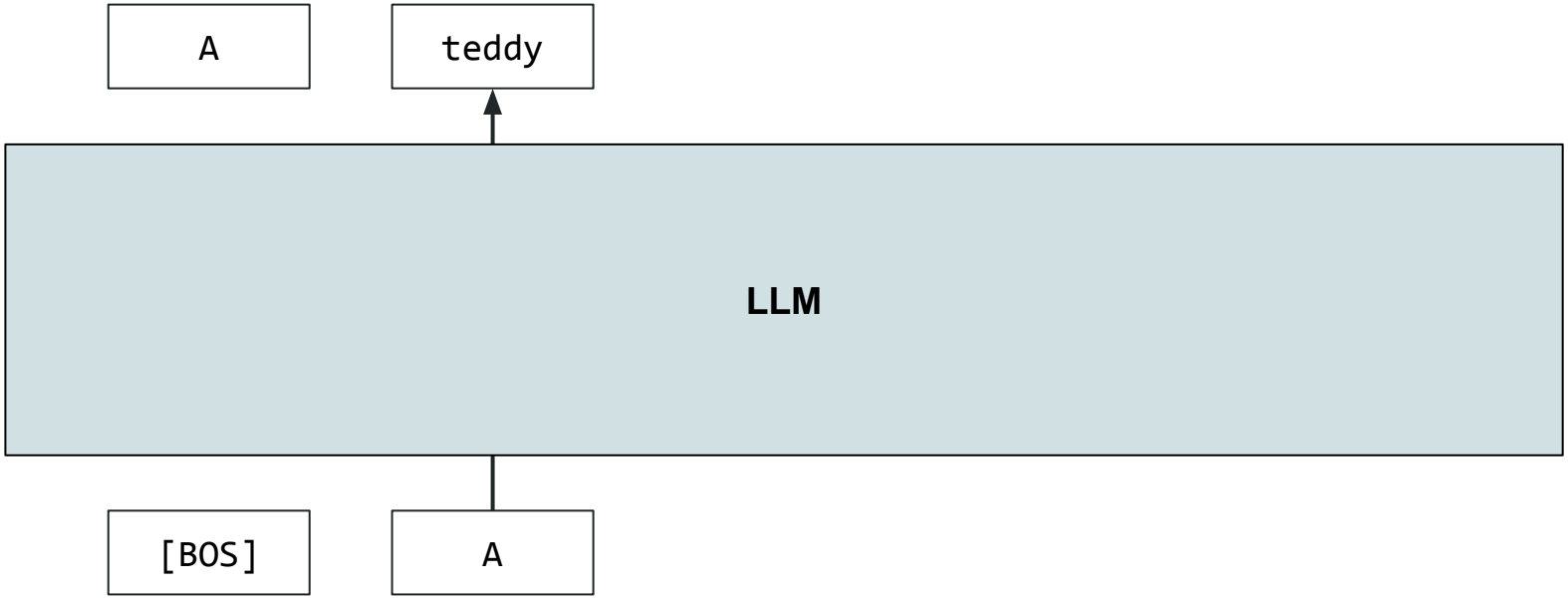
Next token prediction



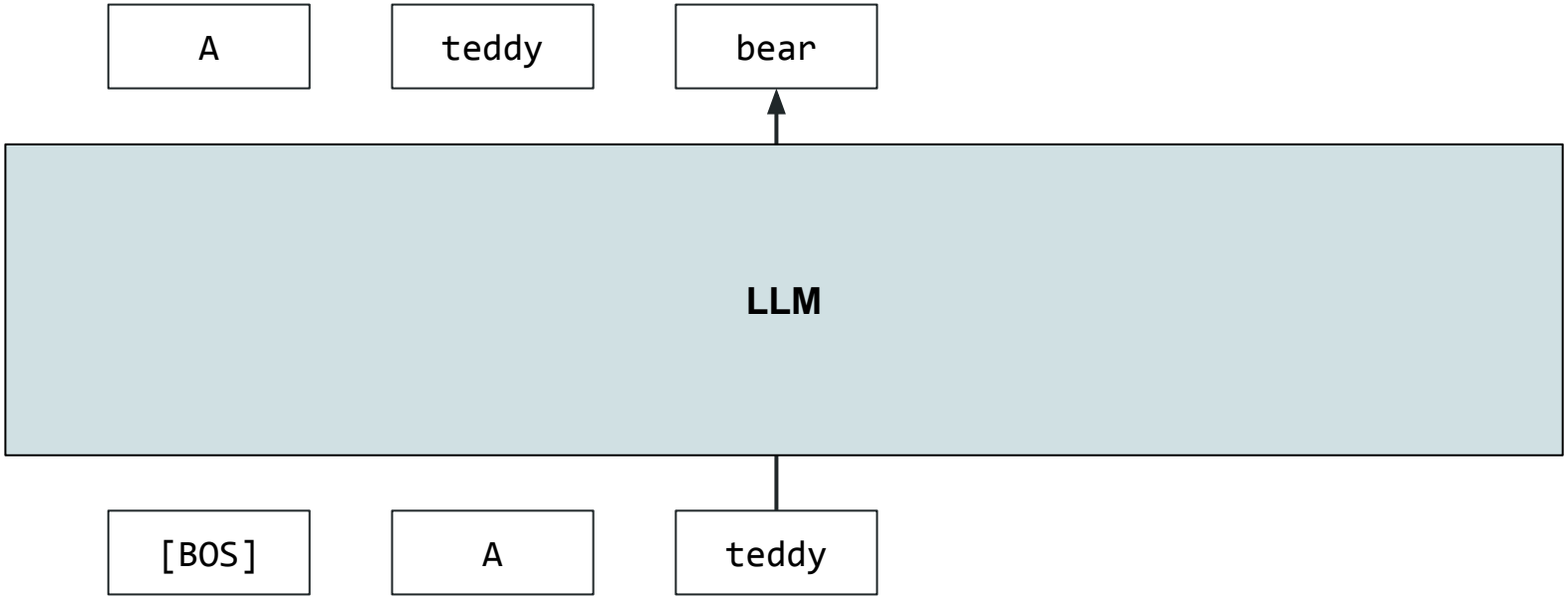
Next token prediction



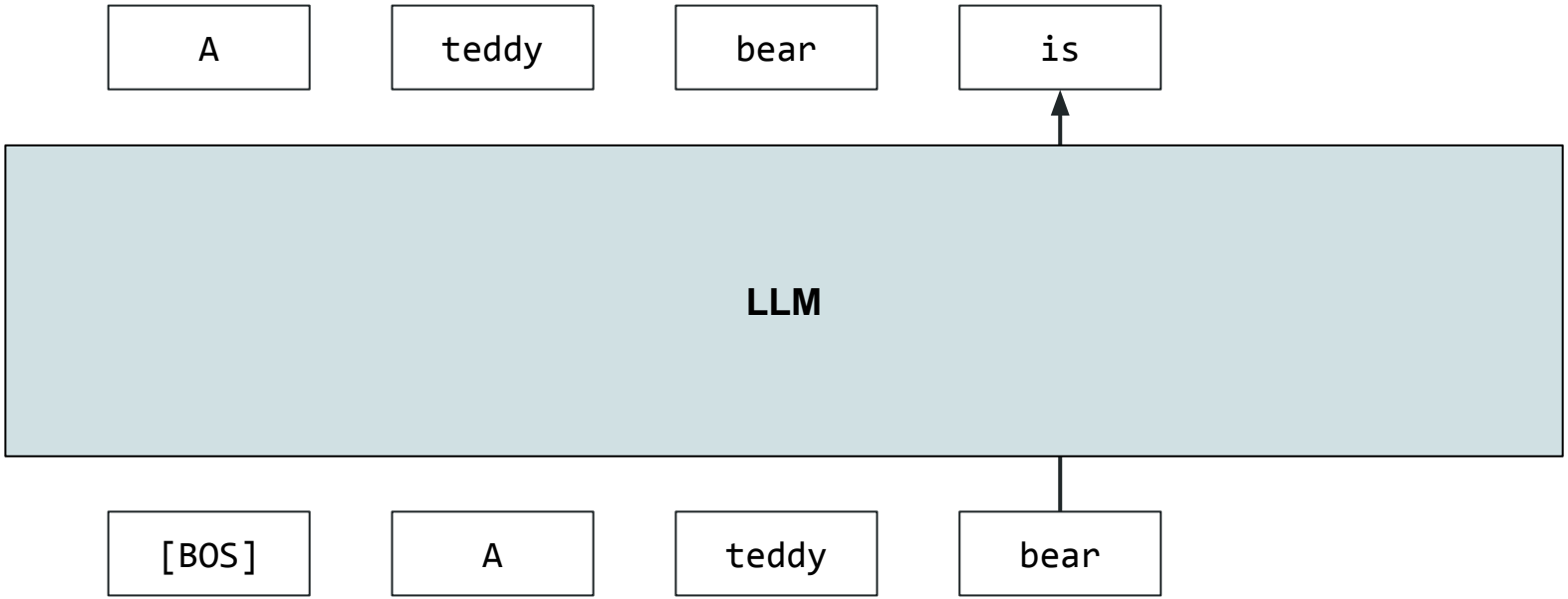
Next token prediction



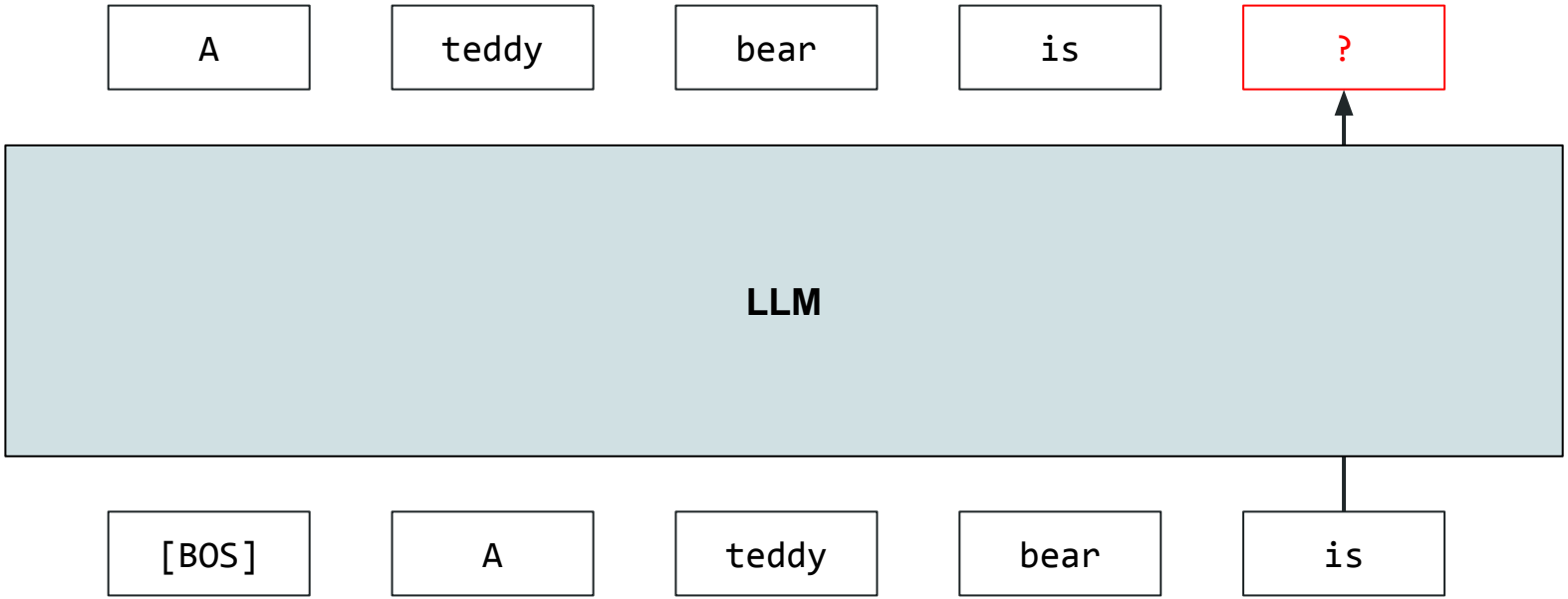
Next token prediction



Next token prediction



Next token prediction



Predicting next token

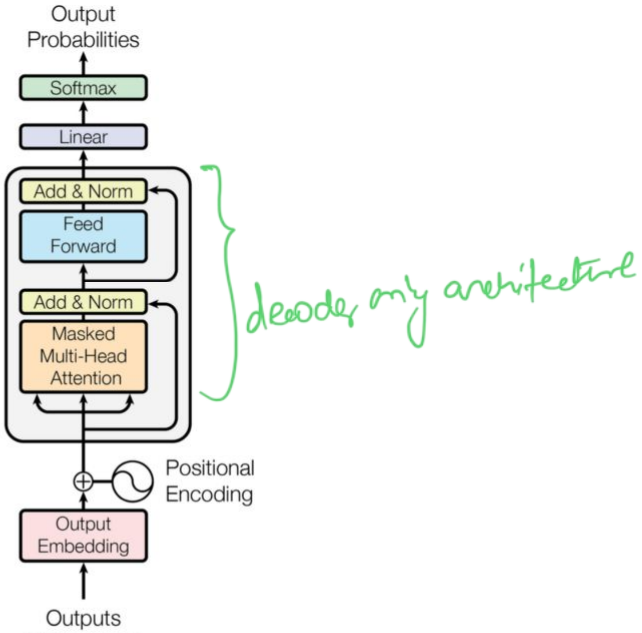


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token

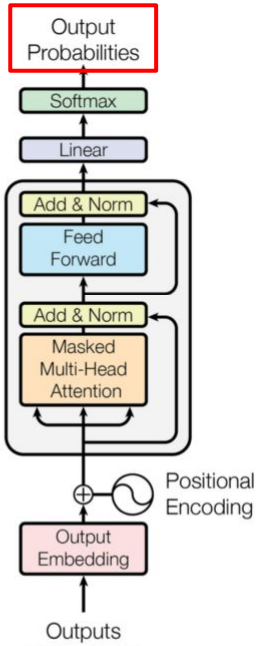
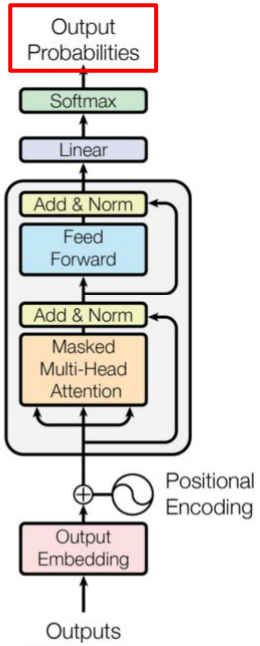


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token



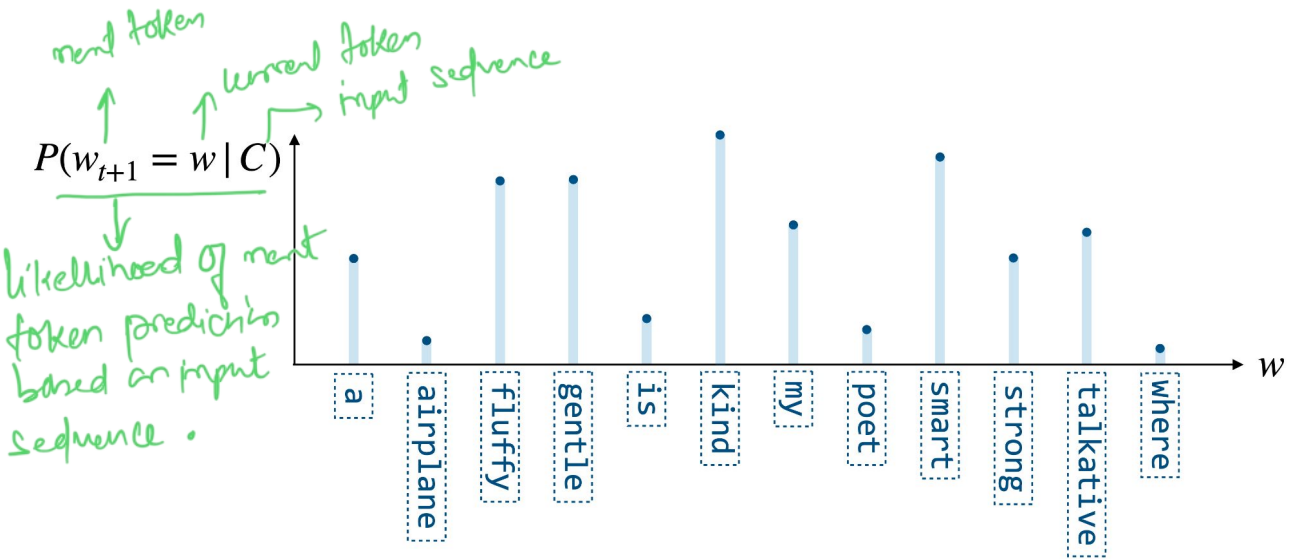
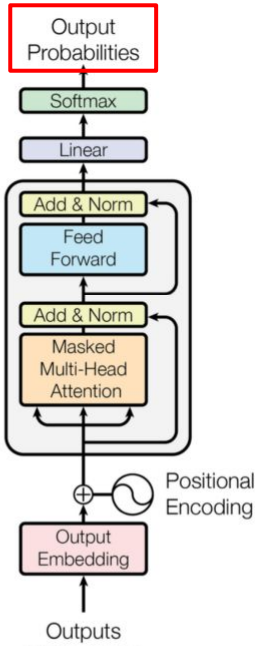


Figure adapted from "Attention is All You Need", Vaswani et al., 2017. "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Predicting next token with greedy decoding



1st idea. Take token with highest predicted probability

highest probability of this token based on input sequence / past tokens. (but it is locally optimal not globally optimal)

$P(w_{t+1} = w \mid C)$

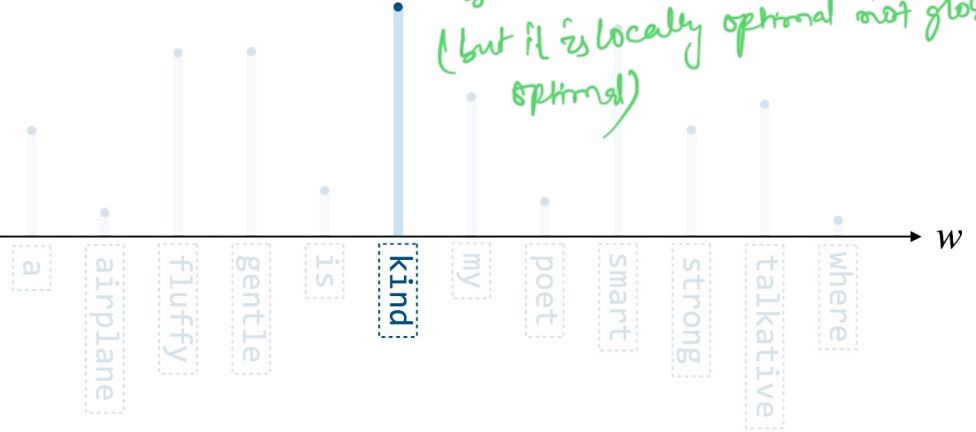
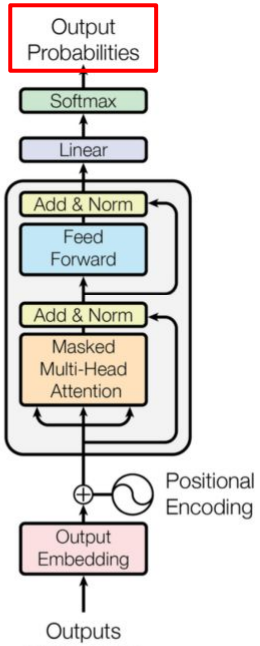
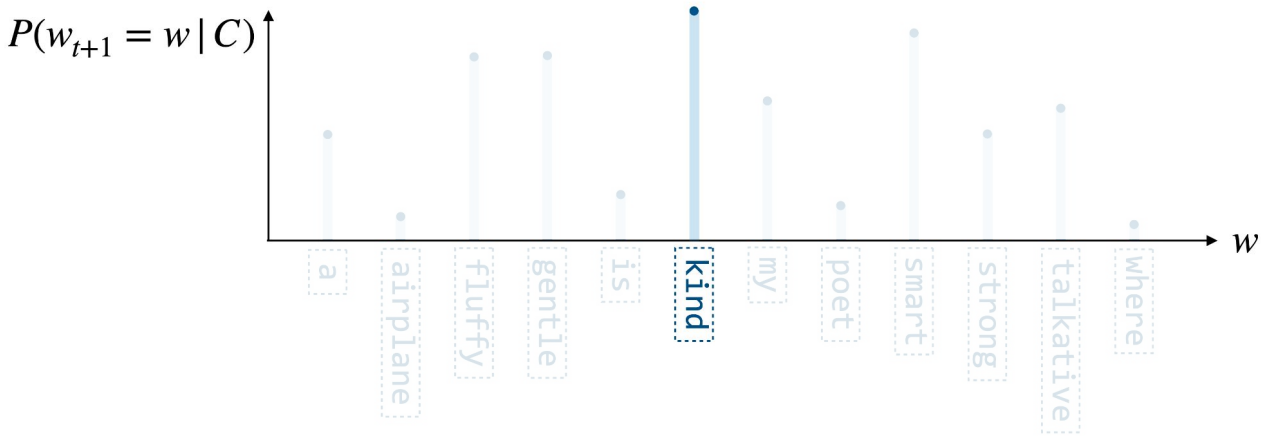


Figure adapted from "Attention is All You Need", Vaswani et al., 2017. "Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Predicting next token with greedy decoding



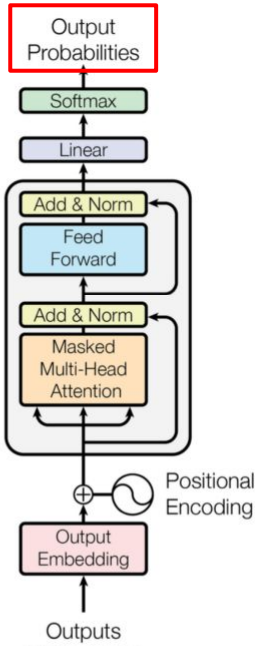
1st idea. Take token with highest predicted probability



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



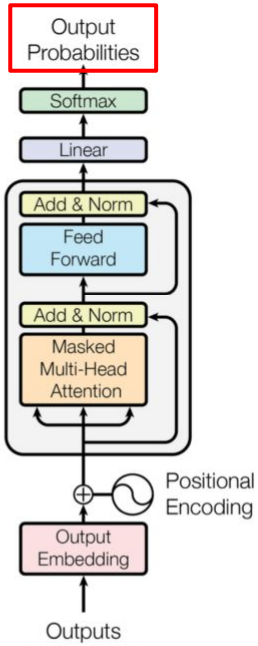
2nd idea. Keep k paths that are the most likely

↓ it is greedily optimal for choosing the next token based on keeping k paths that are most likely based on input sequence

[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

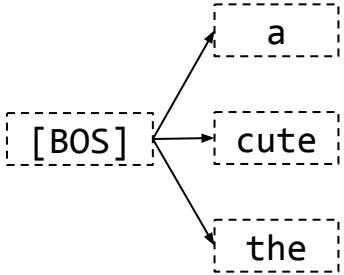
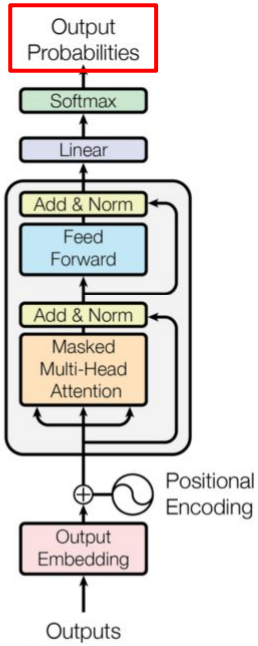


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



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

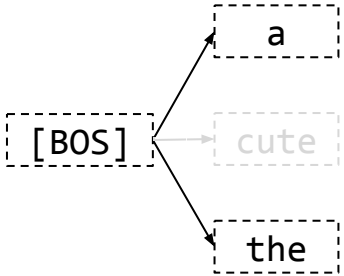
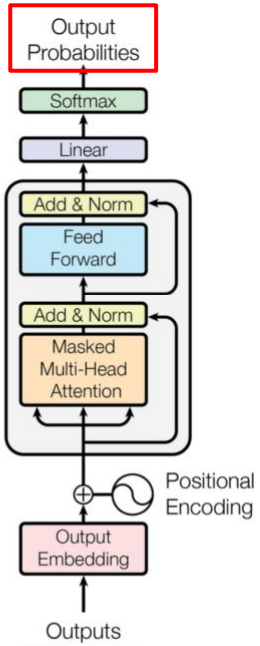


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



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

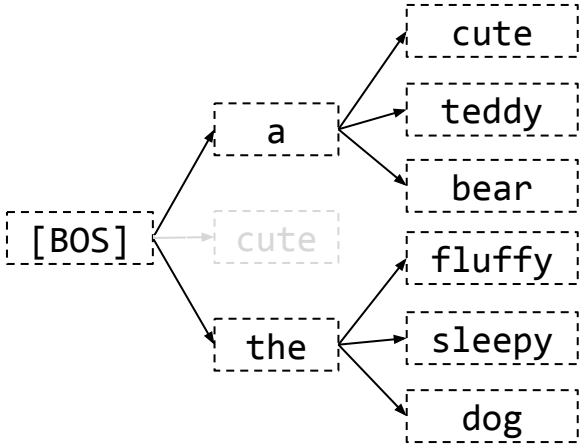
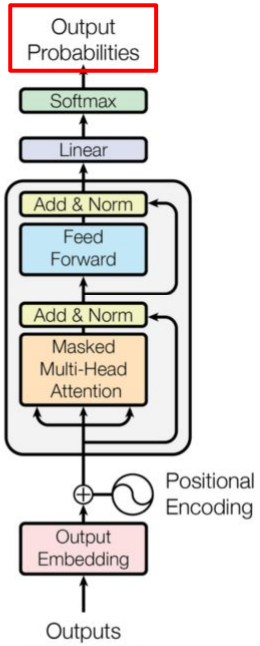


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



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

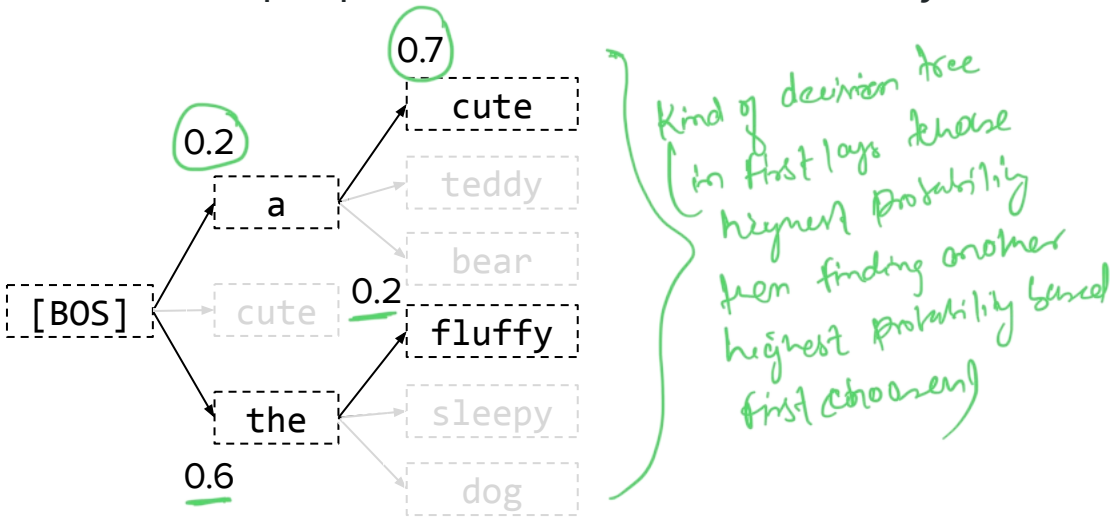
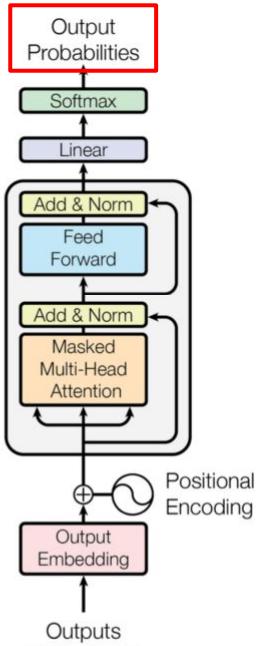


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



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

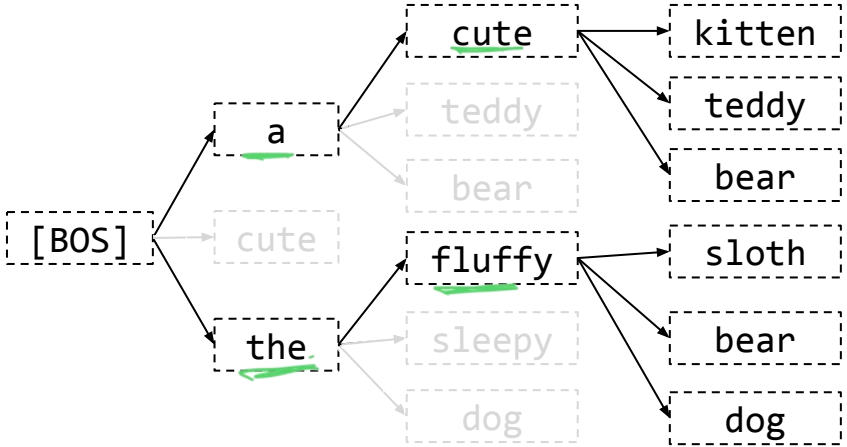
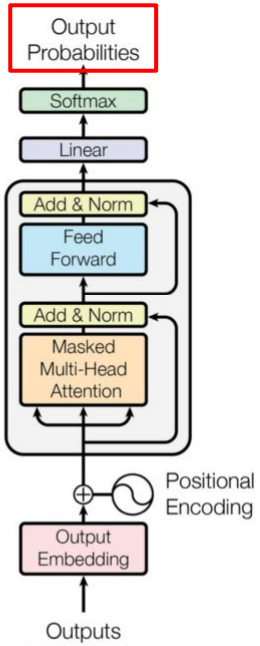


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



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

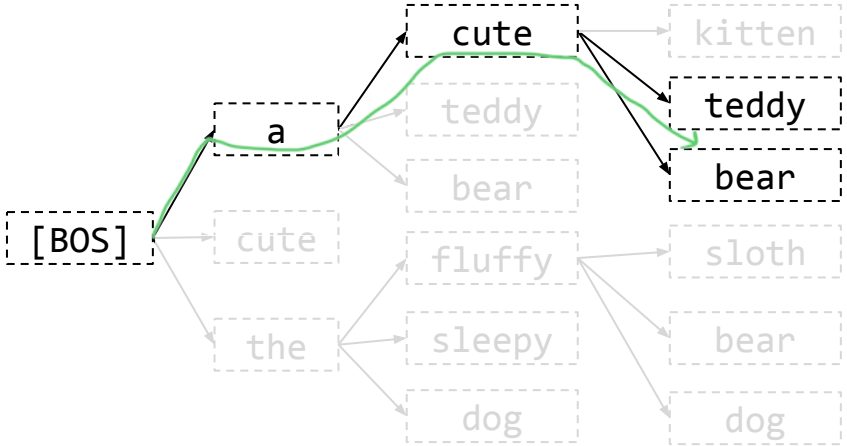
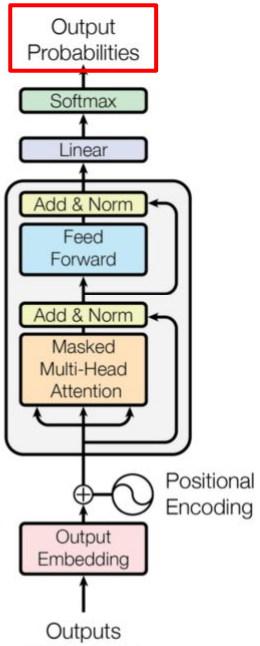
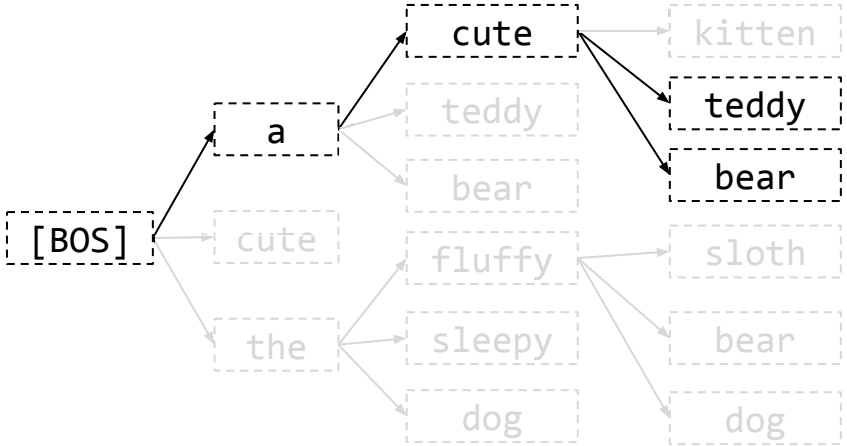


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with beam search



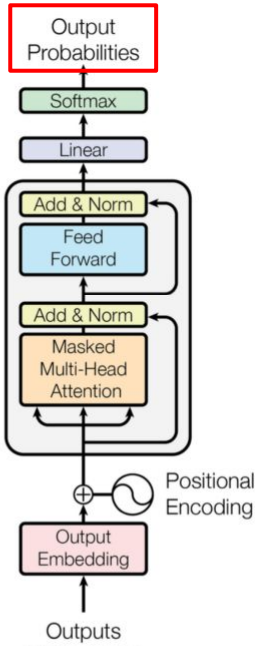
2nd idea. Keep k paths that are the most likely



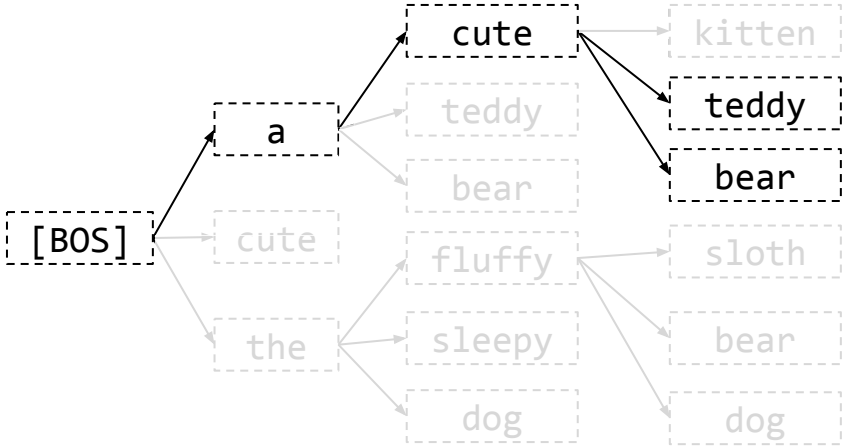
Ends at [EOS]

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



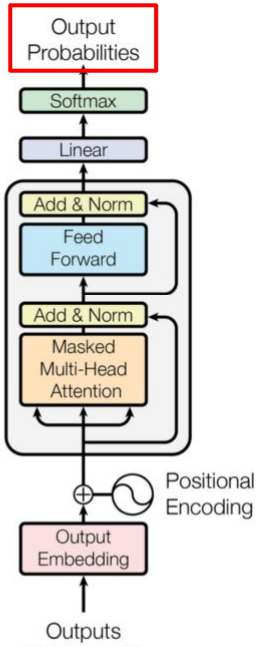
Ends at [EOS]

Limitations. Needs computations + lacks diversity/creativity

↳ bad for keeping track of n possible path requires huge computation/memory [Beam Search]

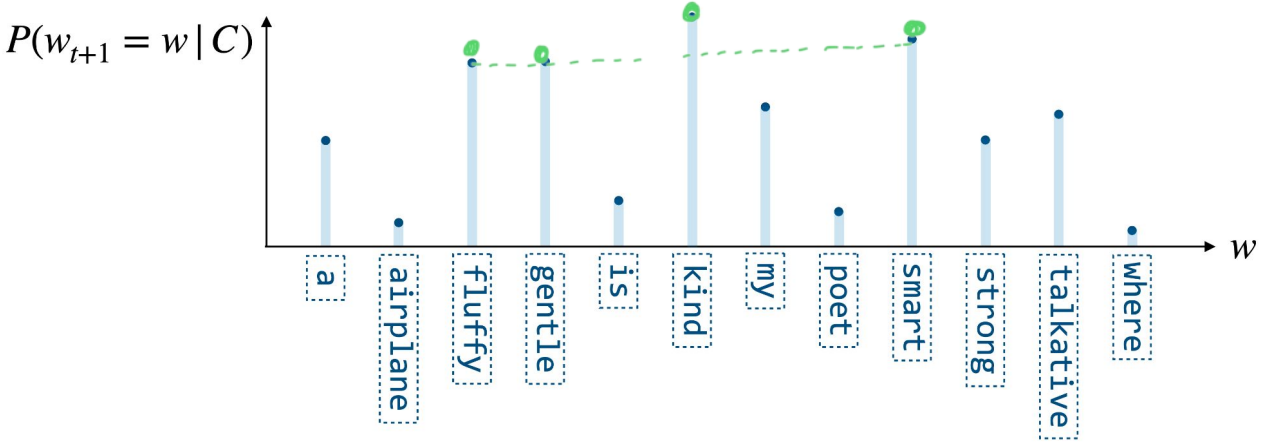
Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Predicting next token with sampling



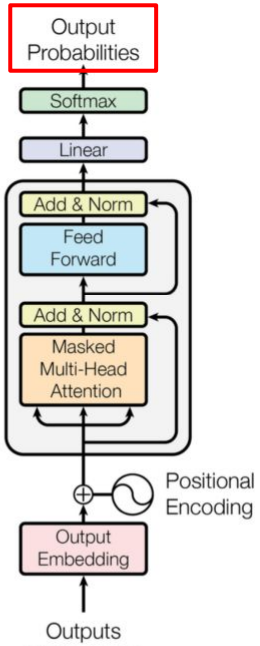
3rd idea. Sample next token from probability distribution:

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



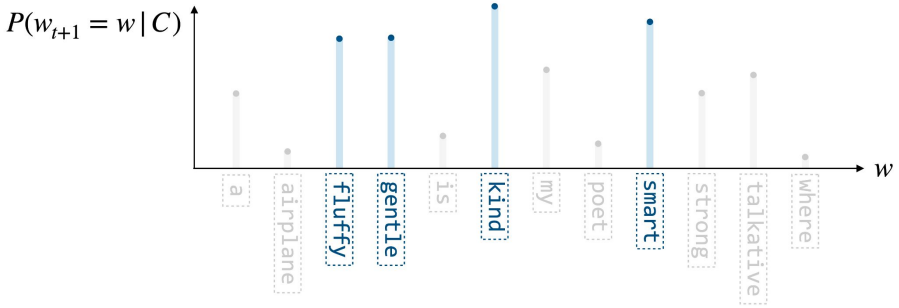
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Sampling strategies



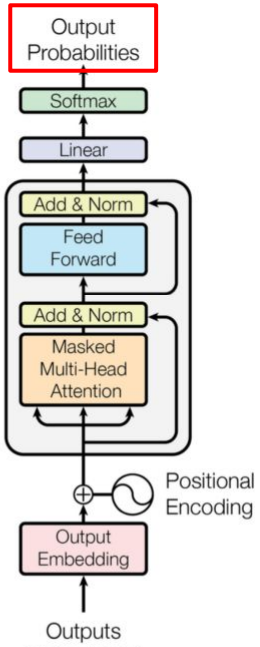
- **Top-k**: Sample among top k most probable tokens

k = 4

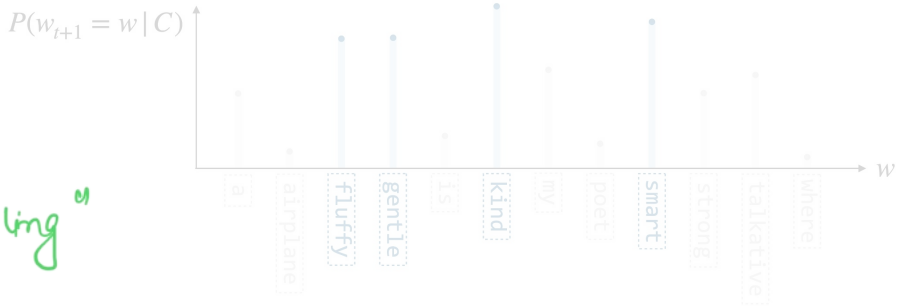


"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

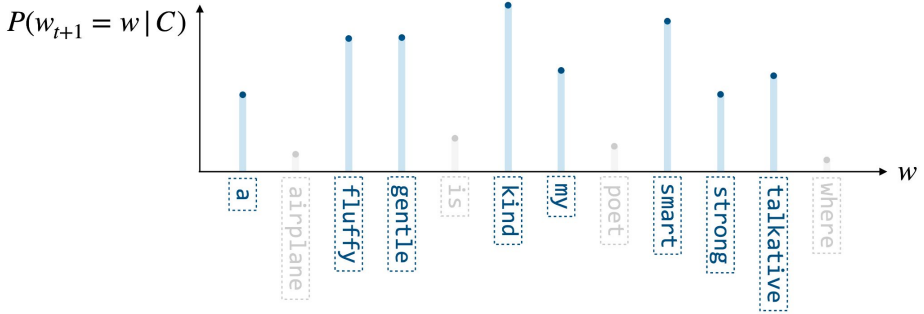
Sampling strategies



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



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

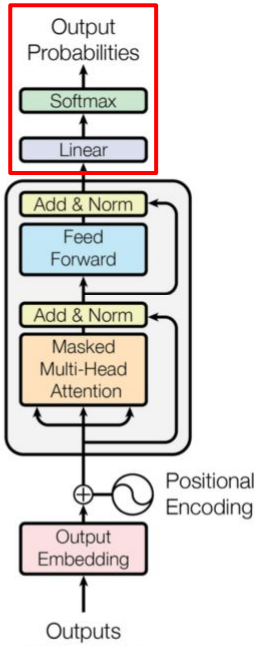


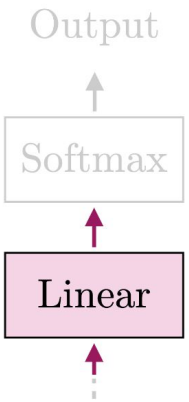
"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

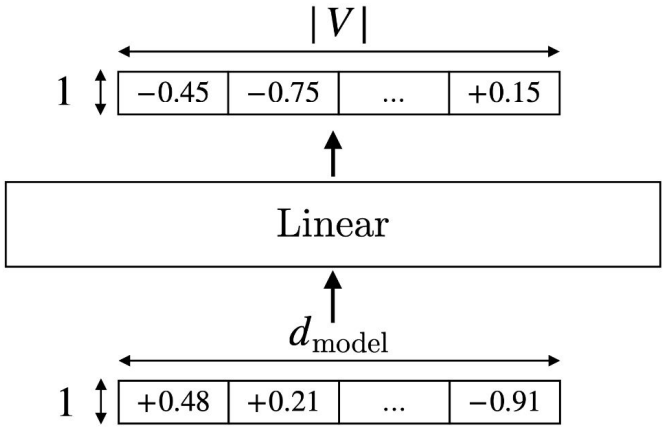
Next token prediction

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

Probability computation

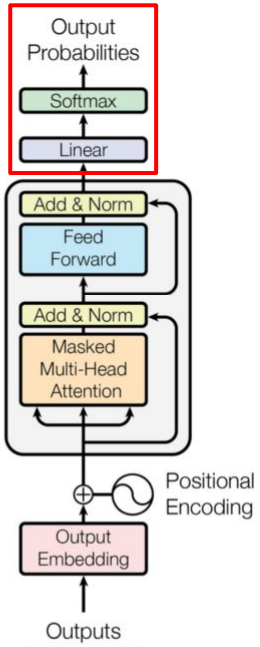


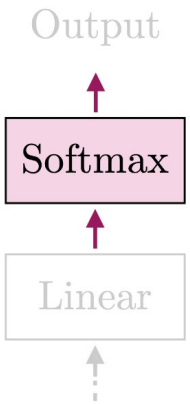


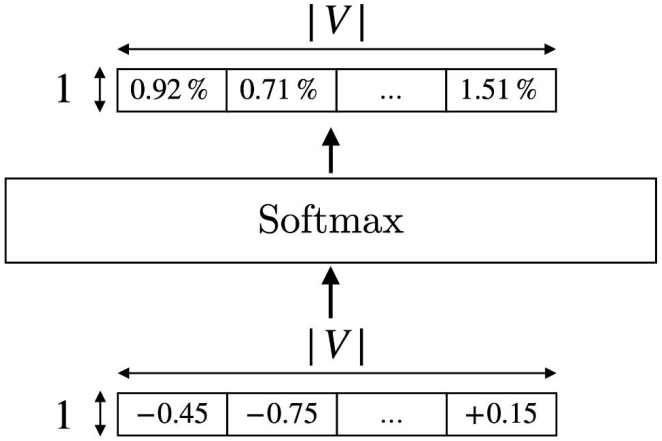


"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Probability computation

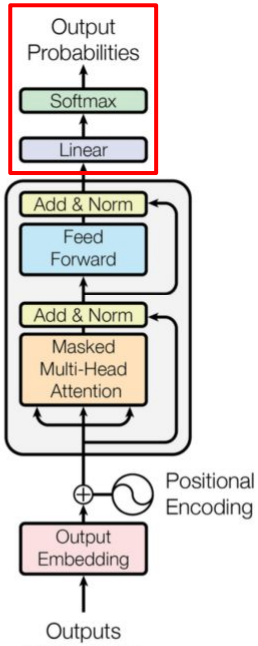


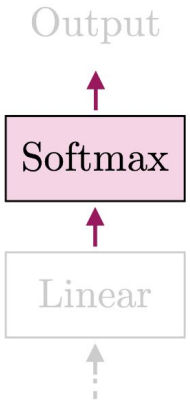




"Super Study Guide: Transformers and Large Language Models", Amidi et al., 2024.

Temperature allows to tweak output probabilities





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

Spike distribution

temperature : ignorance

small temperature : high ignorance

high temperature : Low ignorance

Uniform distribution

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

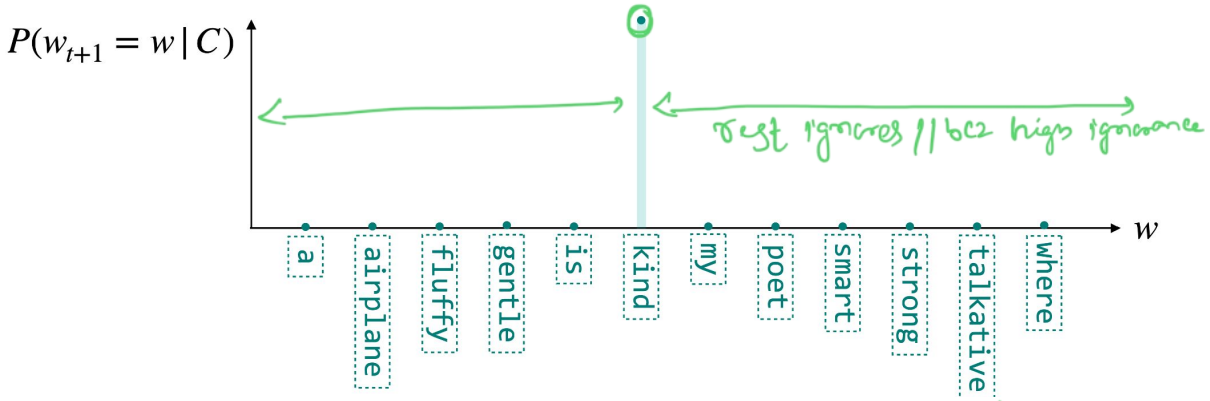
Impact of temperature on probabilities

Small T — Encourage the next token could be highest probable ?

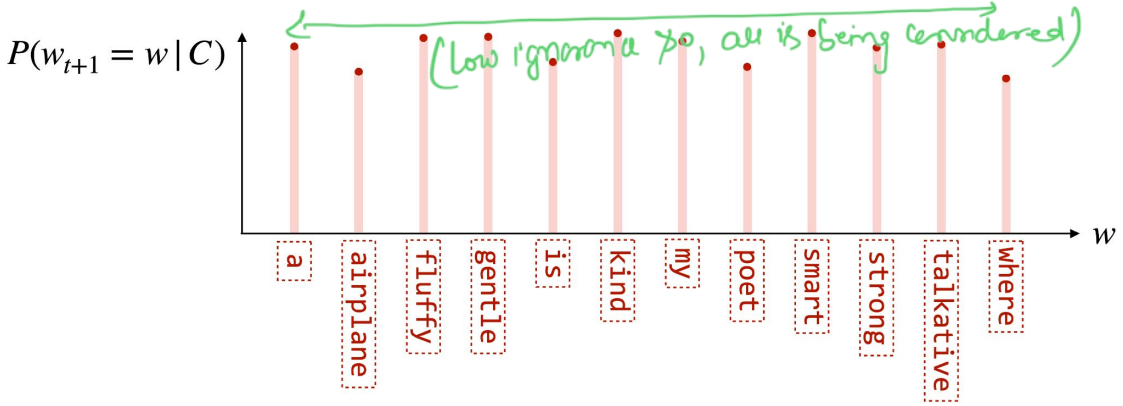
High T → Encourage the next token could be Uniform distribution of next prediction of token / multiple possibility of probable token and more creating tokens. ?

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)

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

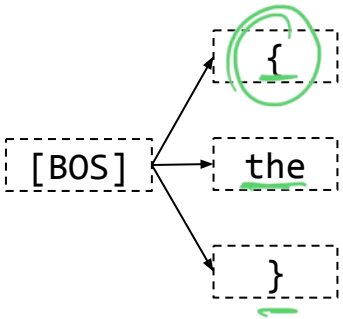
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens

`[BOS]`

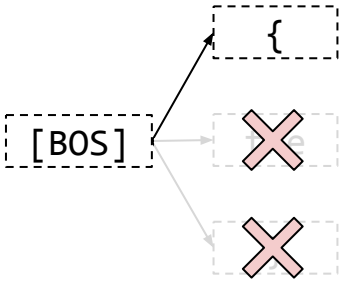
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



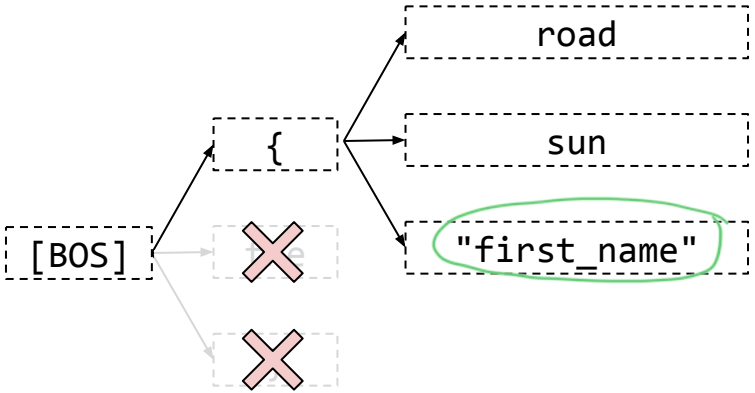
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



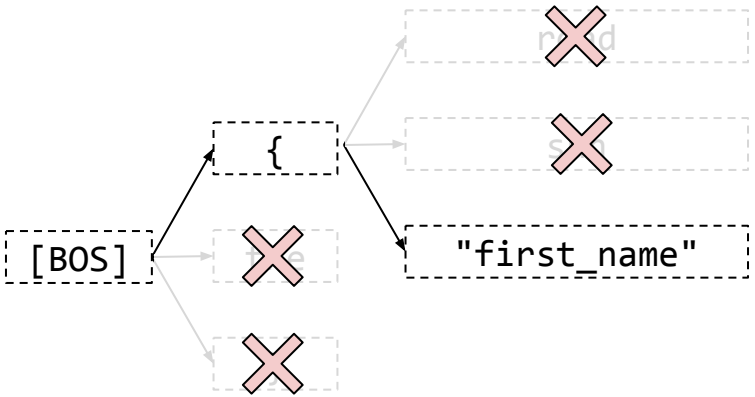
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



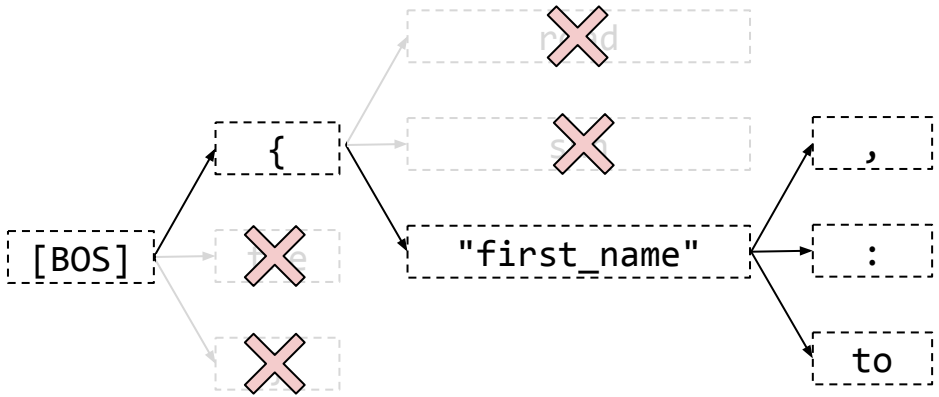
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



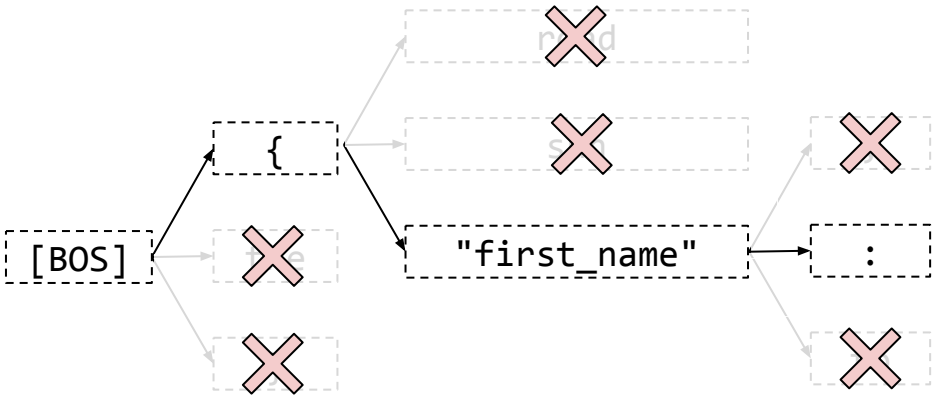
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



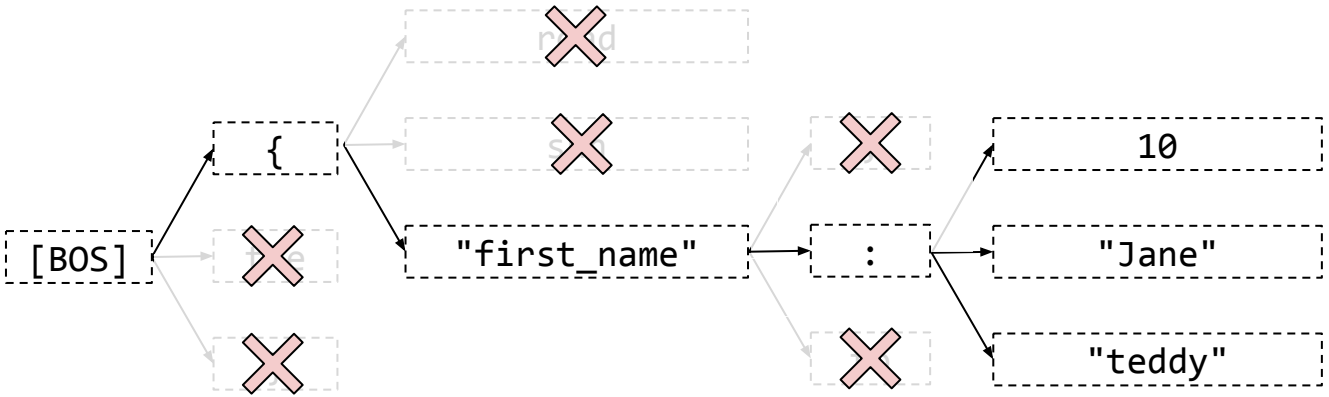
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



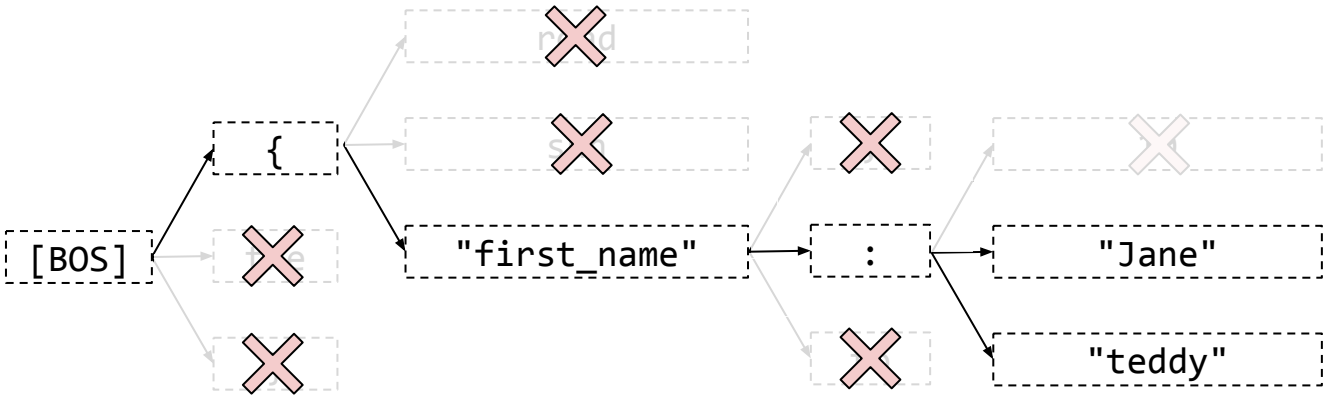
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



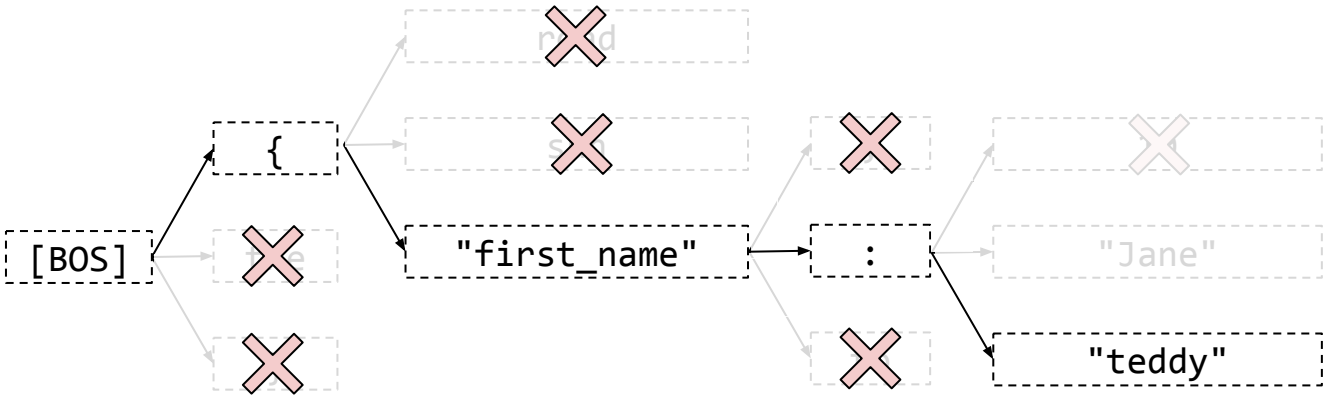
Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



Constraining the output via guided decoding

Idea. Only allow "valid" next tokens



Transformers & Large Language Models

LLM overview

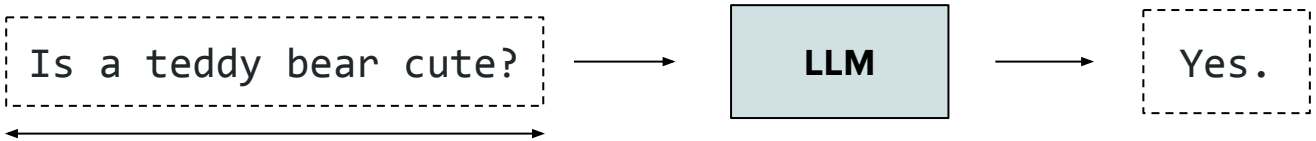
MoE-based LLMs

Response generation

Prompting strategies

Inference optimizations

Terminology ← prompting strategies



Context length

Context size

Window size

↓ length of input

Terminology

Discussion. Orders of magnitude by:

- Input type
- Models

Terminology

Discussion. Orders of magnitude by:

- Input type
- Models

Beware of "context rot"!

Experimenting the capacity of model with different length of context.
a needle in a haystack" → Experimentation"

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

Main structure

At increasing of content length; ability to process the information with the answer i.e ↓ on content decrease

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

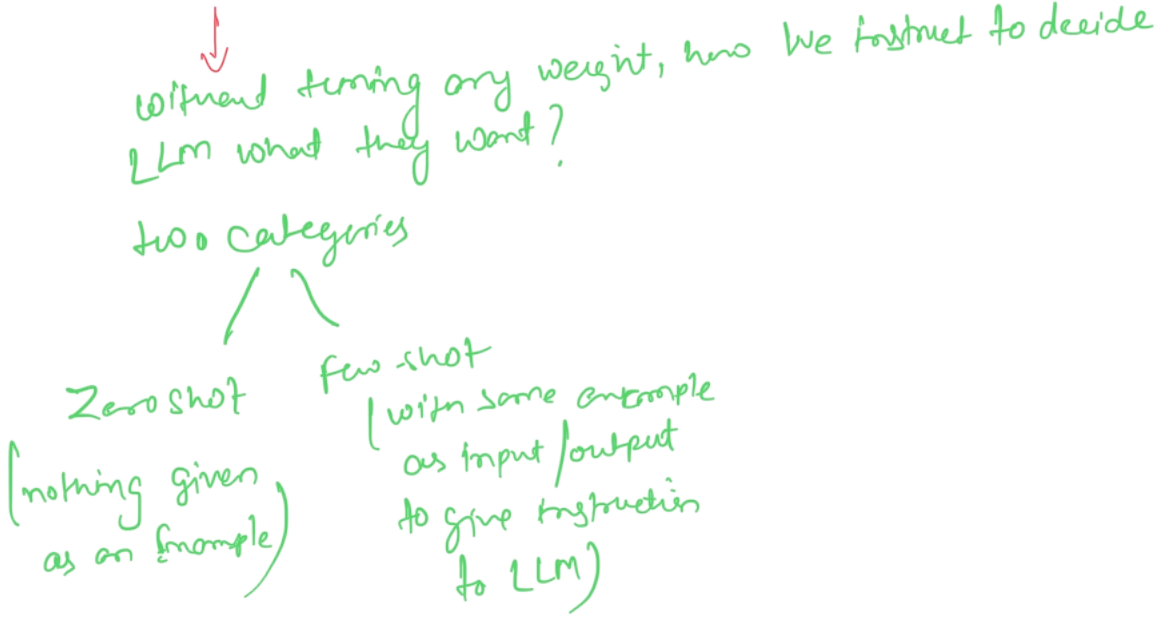
Generate a bedtime story that takes place in a specific location. — **Instructions**

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

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

In-context learning

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



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

In-context learning

ICL = In-**C**ontext **L**earning

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

(if we provide example, we kind of giving instruction to make the connection b/w input and output for understanding the connection which literally improves the performance in case of prediction)

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

In-context learning

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

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

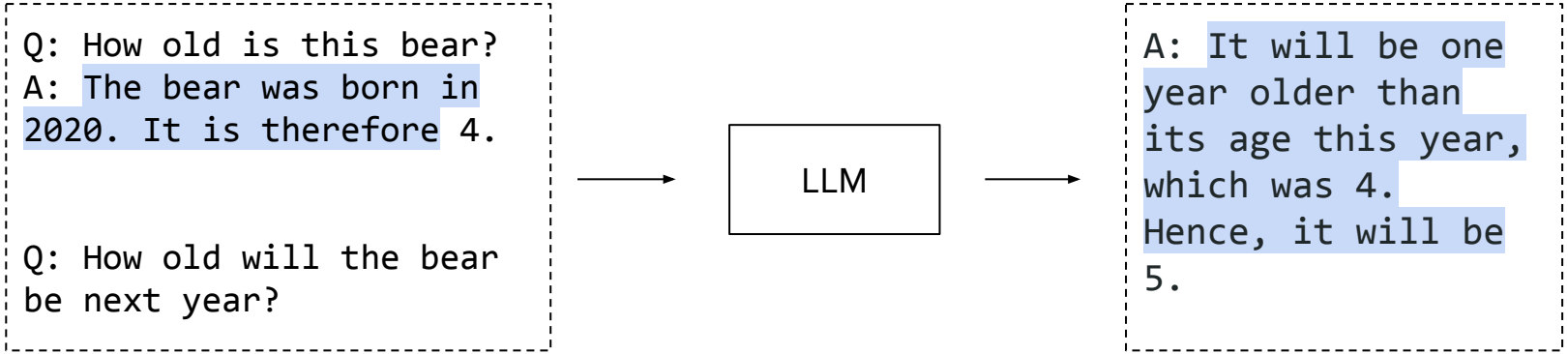
Discussion. Showing examples in the prompt is *generally* better but:

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

Chain of thought – if we force the model to come with some example as q/A it improves the performance.

↳ root causing will be easy for improved results

Idea. Explaining reasoning helps in improving performance.



Discussion.

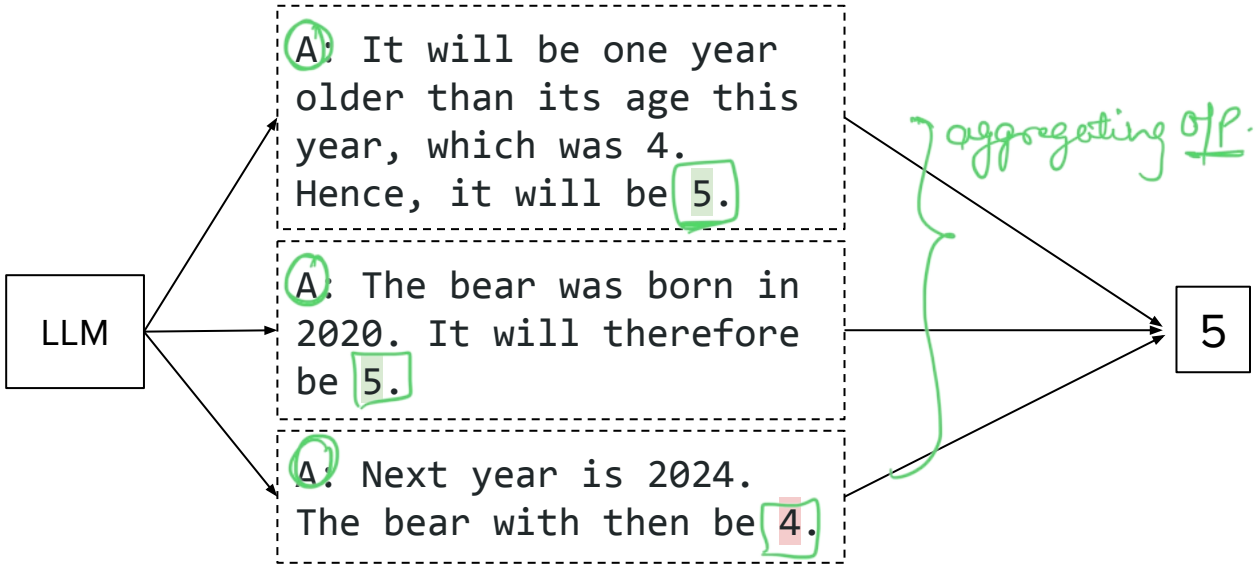
- Interpretability + explanation
- More tokens: higher cost and latency

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

Self-consistency

Samples the model with different response and aggregating it for better response

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:

1. Avoid redundancies
2. Memory management
3. Reformulate the math

} reducing complexity but better performance

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:

3. Architectural changes
4. Embeddings representations
5. Token prediction

lesser cost, but better o/p at prediction

Attention-based tricks

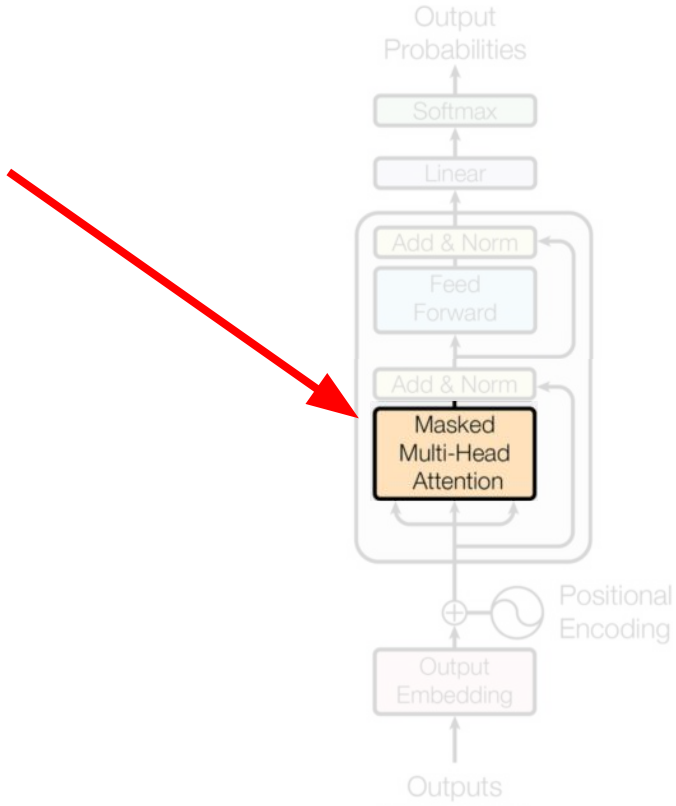


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

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

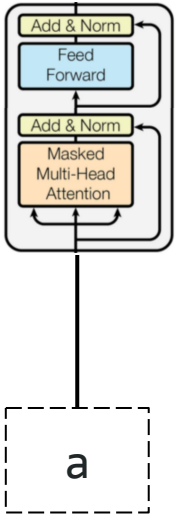


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

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

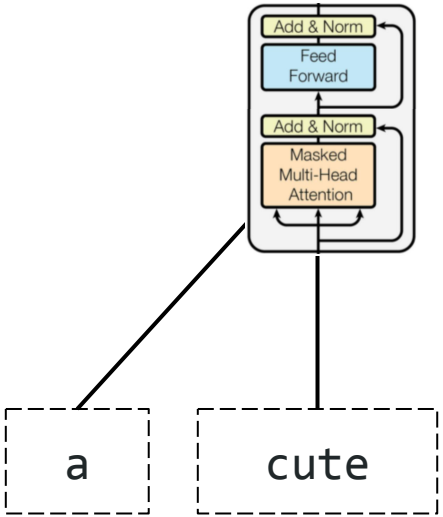


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

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

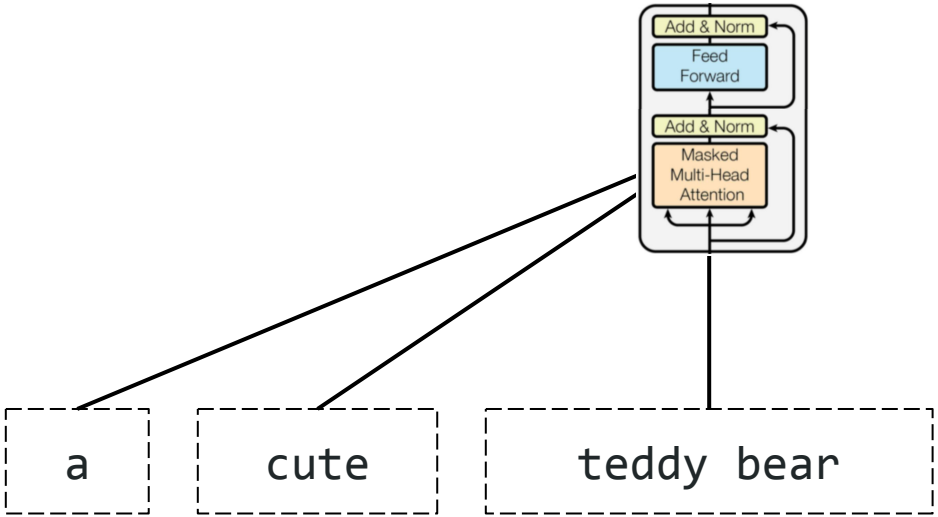


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

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

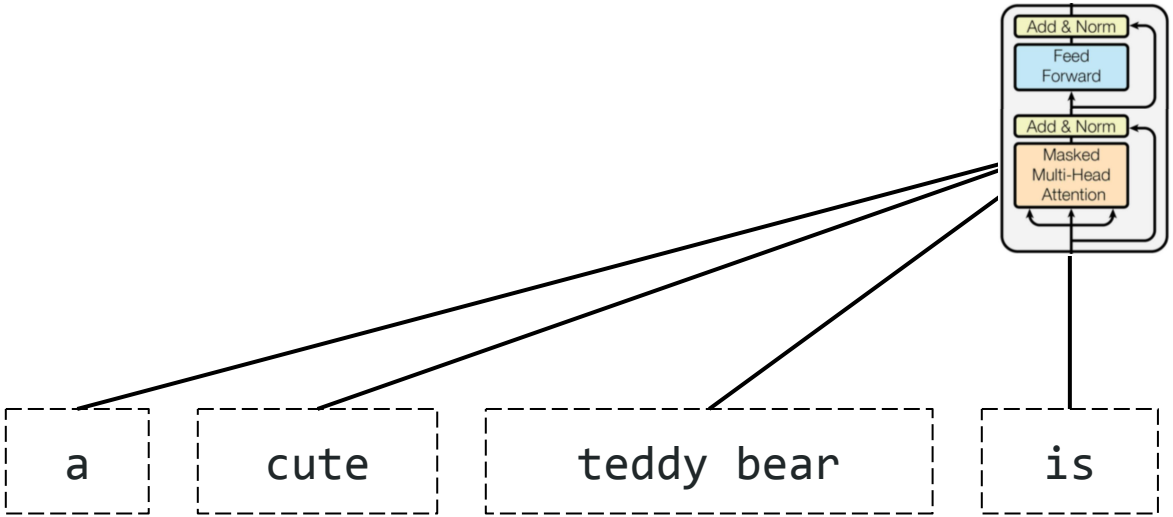
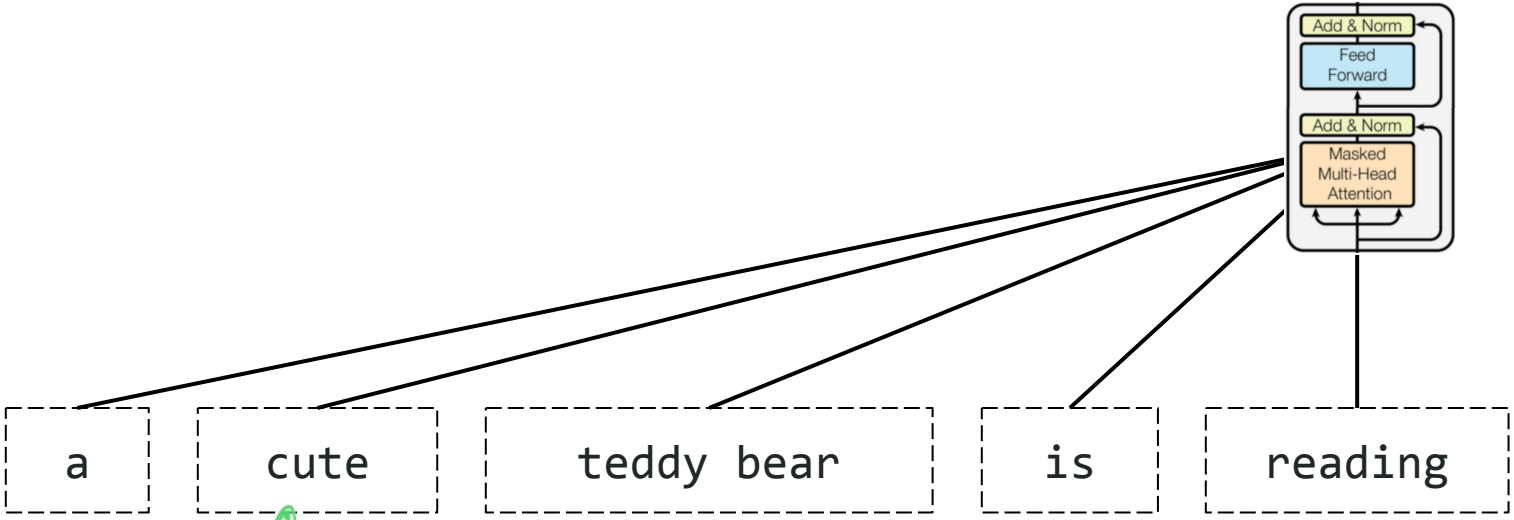


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Reusing computations with KV caching

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



Each current token are predicted based on previous tokens.

So, "cute" is attending 'a' and teddy bear is attending "a" and "cute"

So we need to keep the predicted already previous token for further as it is repeating so, it has

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.

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

Reusing computations with KV caching

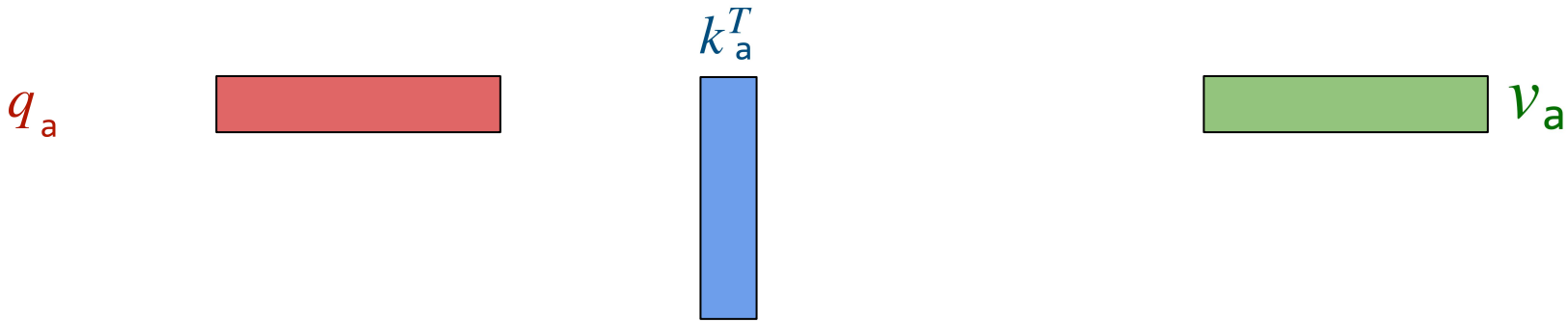
Motivation. New token needs to interact with all previous tokens.

Idea. Keep keys and values in a cache.

Reusing computations with KV caching

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

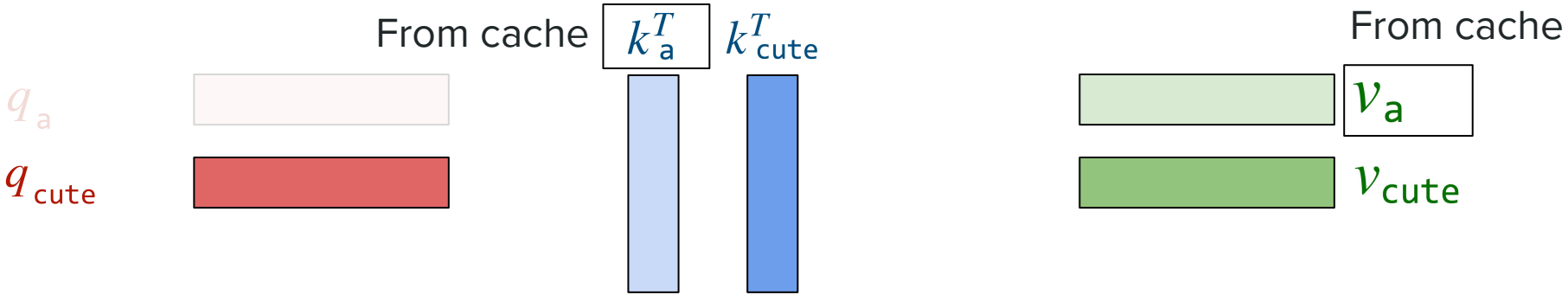
Idea. Keep keys and values in a cache.



Reusing computations with KV caching

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

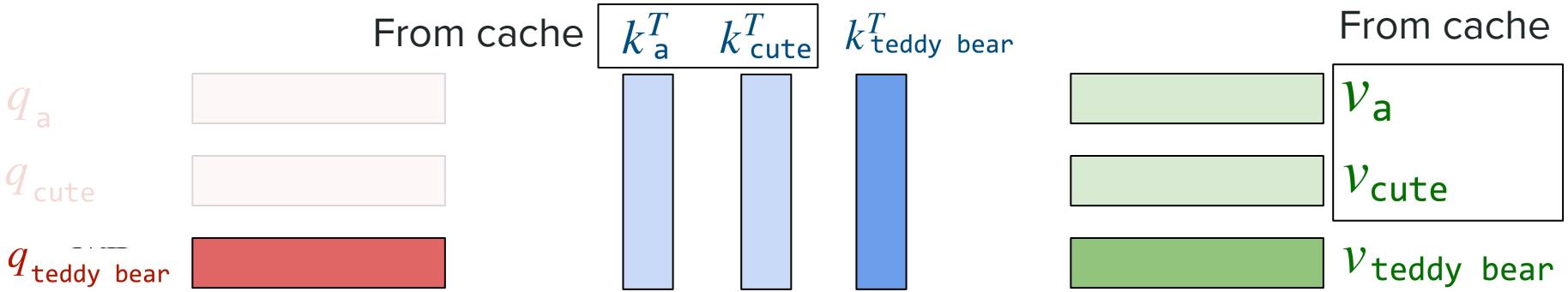
Idea. Keep keys and values in a cache.



Reusing computations with KV caching

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

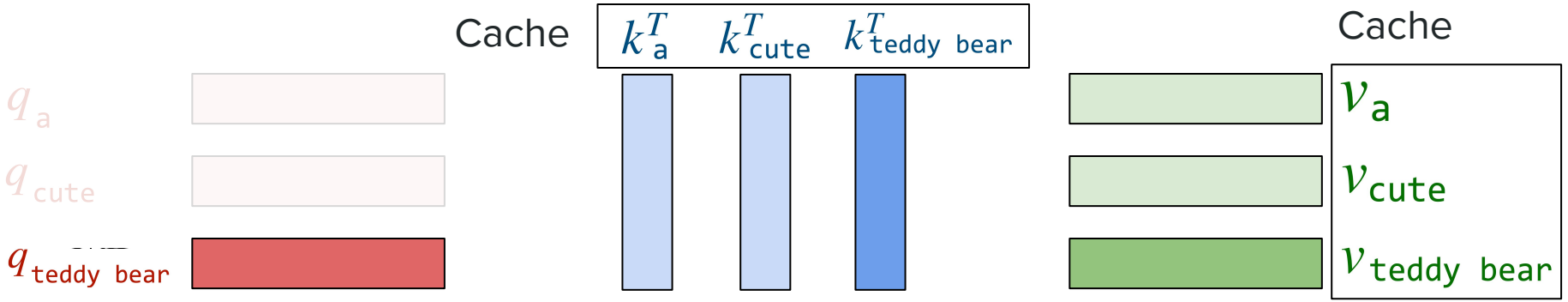
Idea. Keep keys and values in a cache.



Reusing computations with KV caching

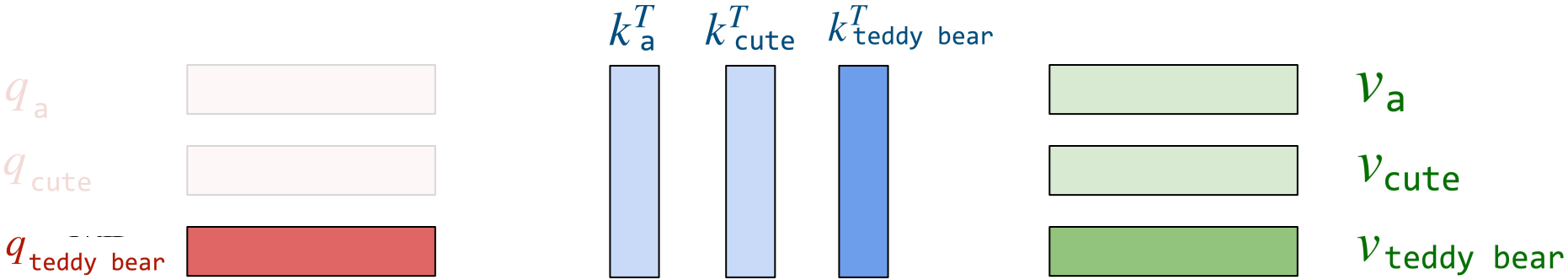
Motivation. New token needs to interact with all previous tokens.

Idea. Keep keys and values in a cache.

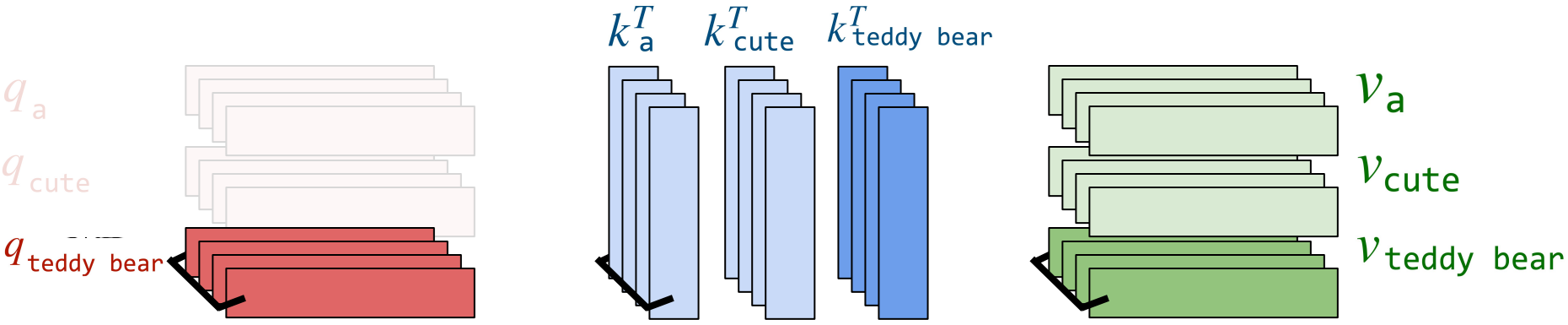


KV caching = **K**ey-**V**alue **Caching**

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

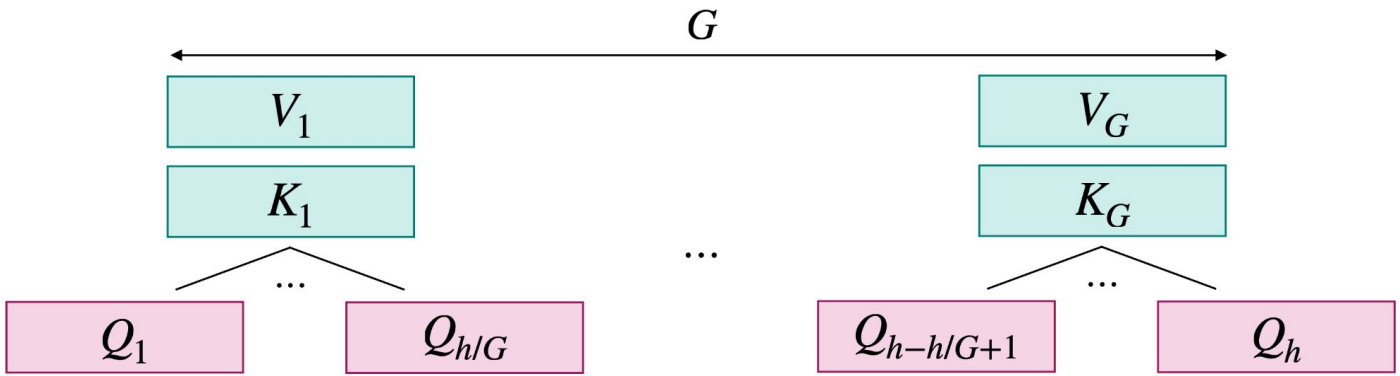


Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Sharing attention heads

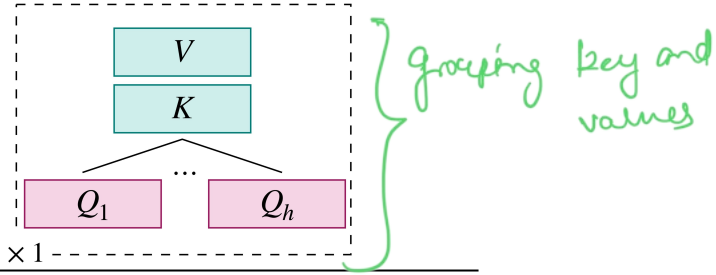
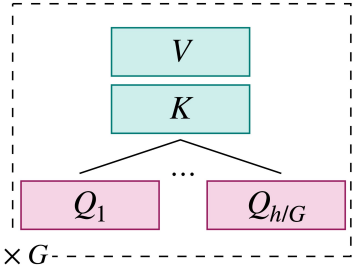
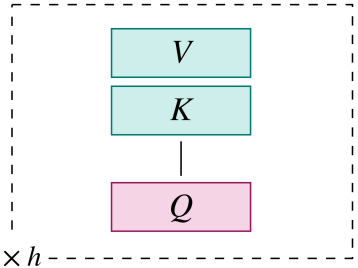
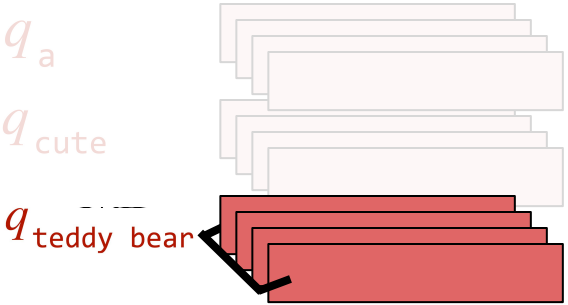
$G = 1$	**M**ulti-**Q**uery **A**ttention (MQA)	
$1 < G < h$	**G**roup-**Q**uery **A**ttention (GQA)	
$G = h$	**M**ulti-**H**ead **A**ttention (MHA)	

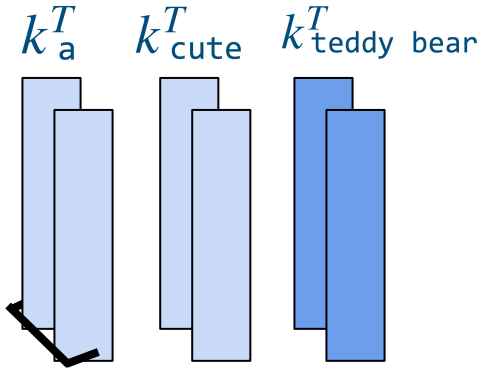
Figure from "Super Study Guide: Transformers & Large Language Models", Amidi et al., 2024.

Sharing attention heads with GQA



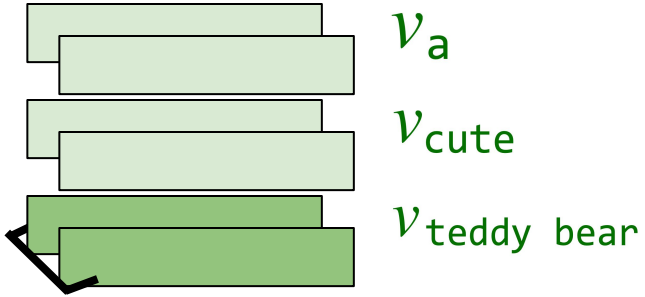
Number of
query heads

= 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.

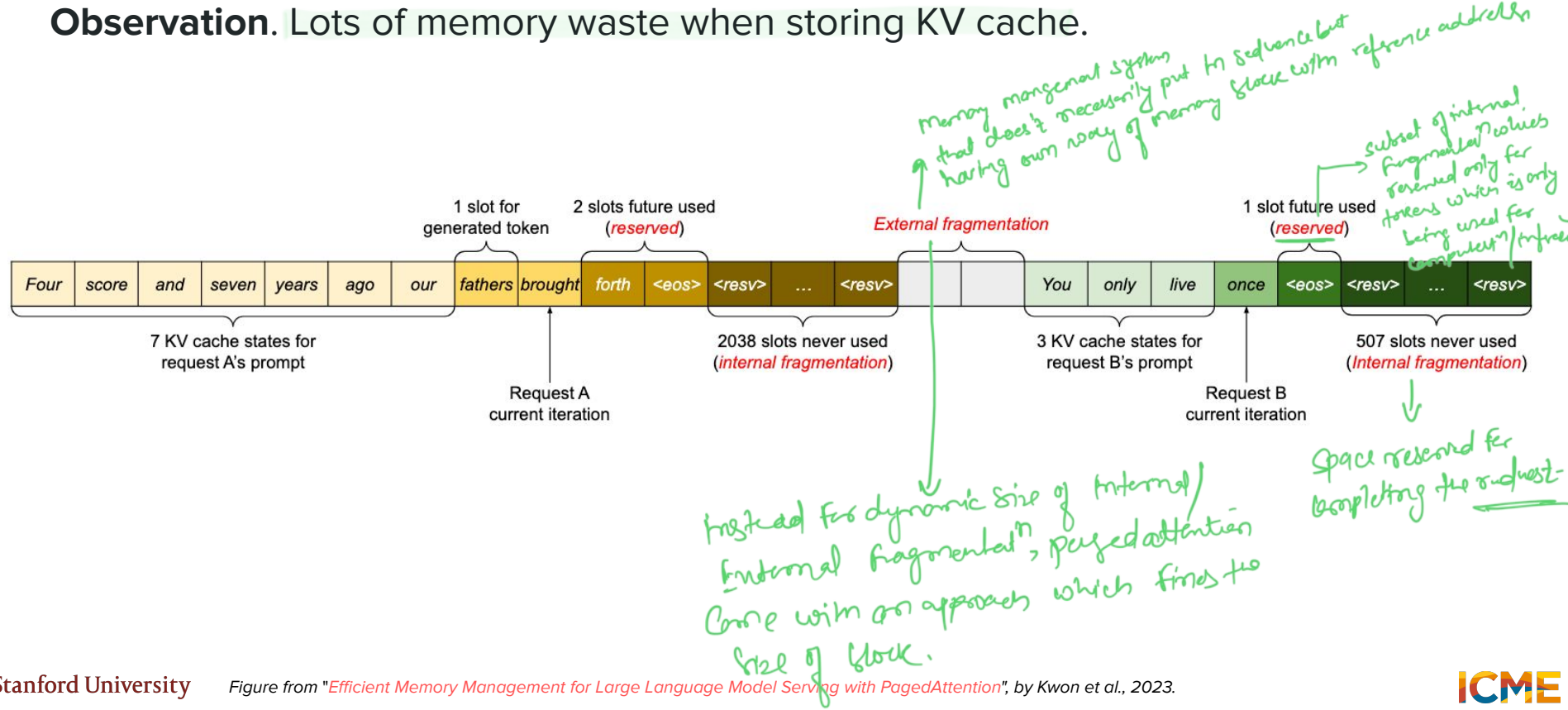


Figure from "Efficient Memory Management for Large Language Model Serving with PagedAttention", by Kwon et al., 2023.

Manage memory with PagedAttention

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

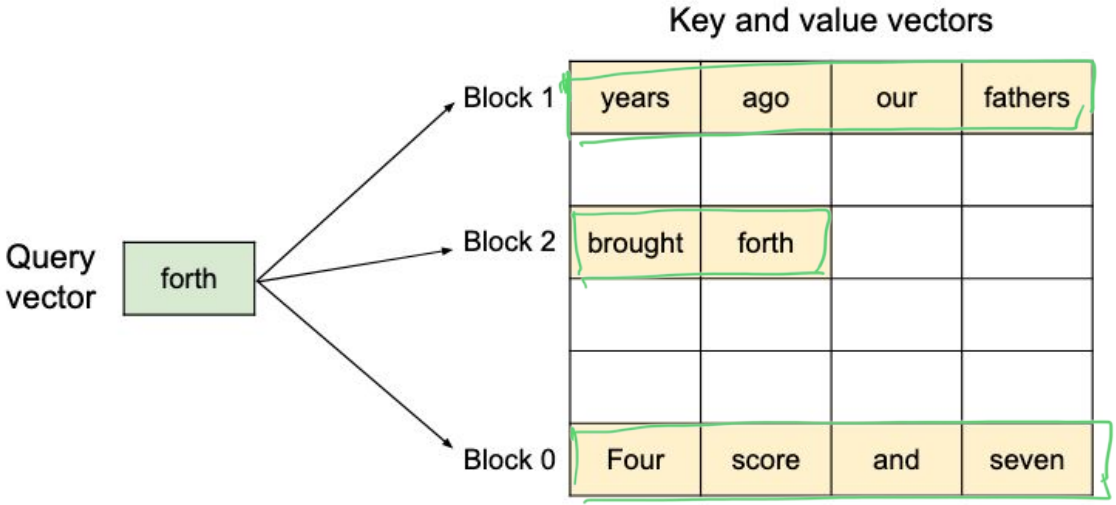
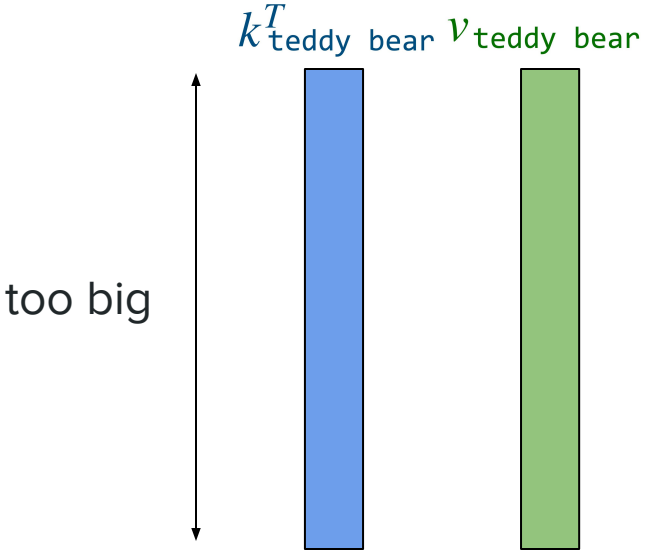


Figure from "Efficient Memory Management for Large Language Model Serving with PagedAttention", by Kwon et al., 2023.

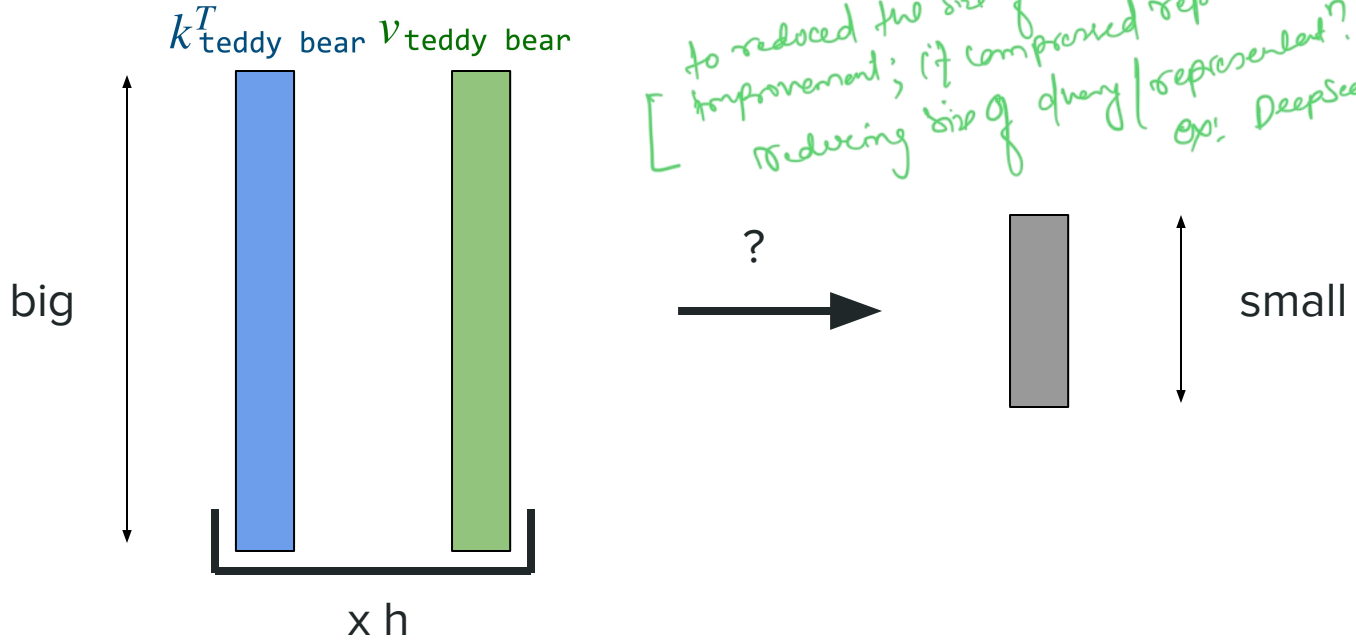
Reduce memory of KV cache with latent attention

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



Reduce memory of KV cache with latent attention

Solution. Store compressed representations instead!

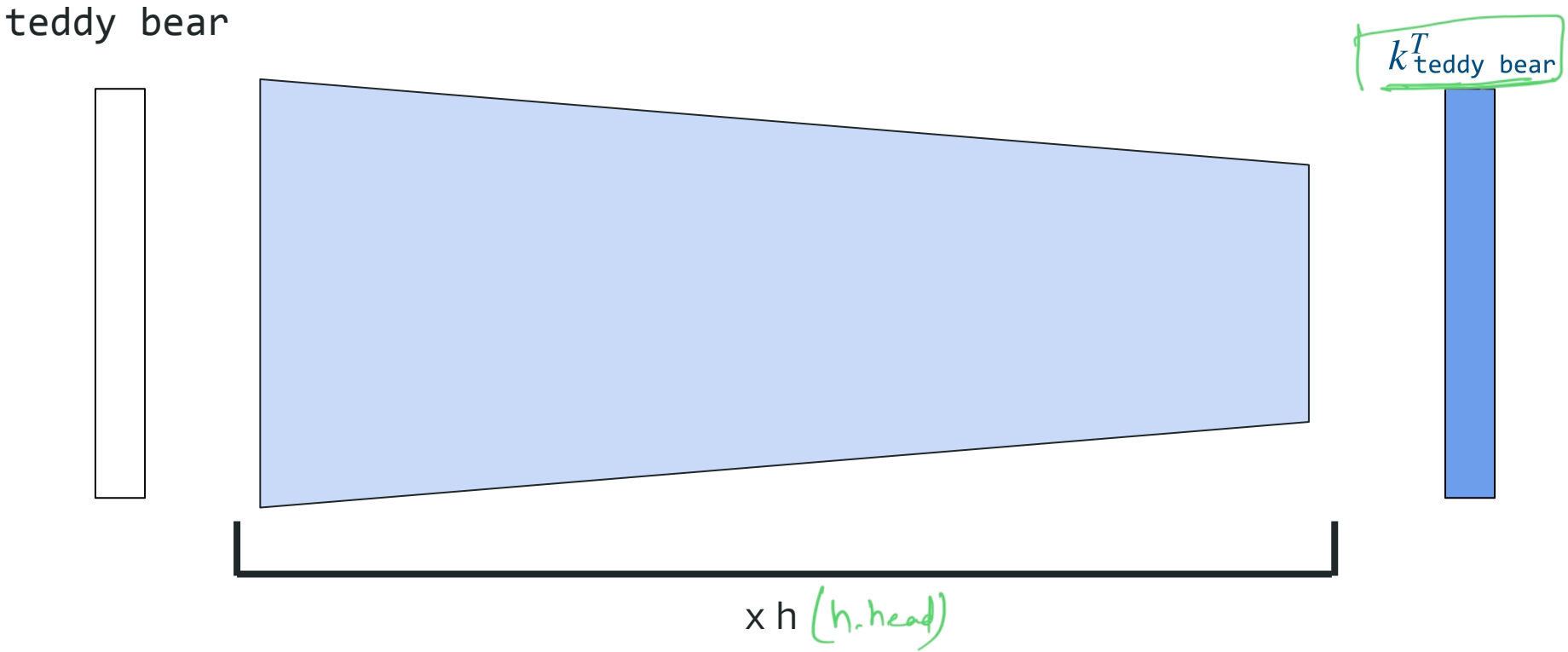


[to reduced the size of kv cache for performance improvement; if compressed representation and reducing size of query representation?
eg: DeepSeek

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

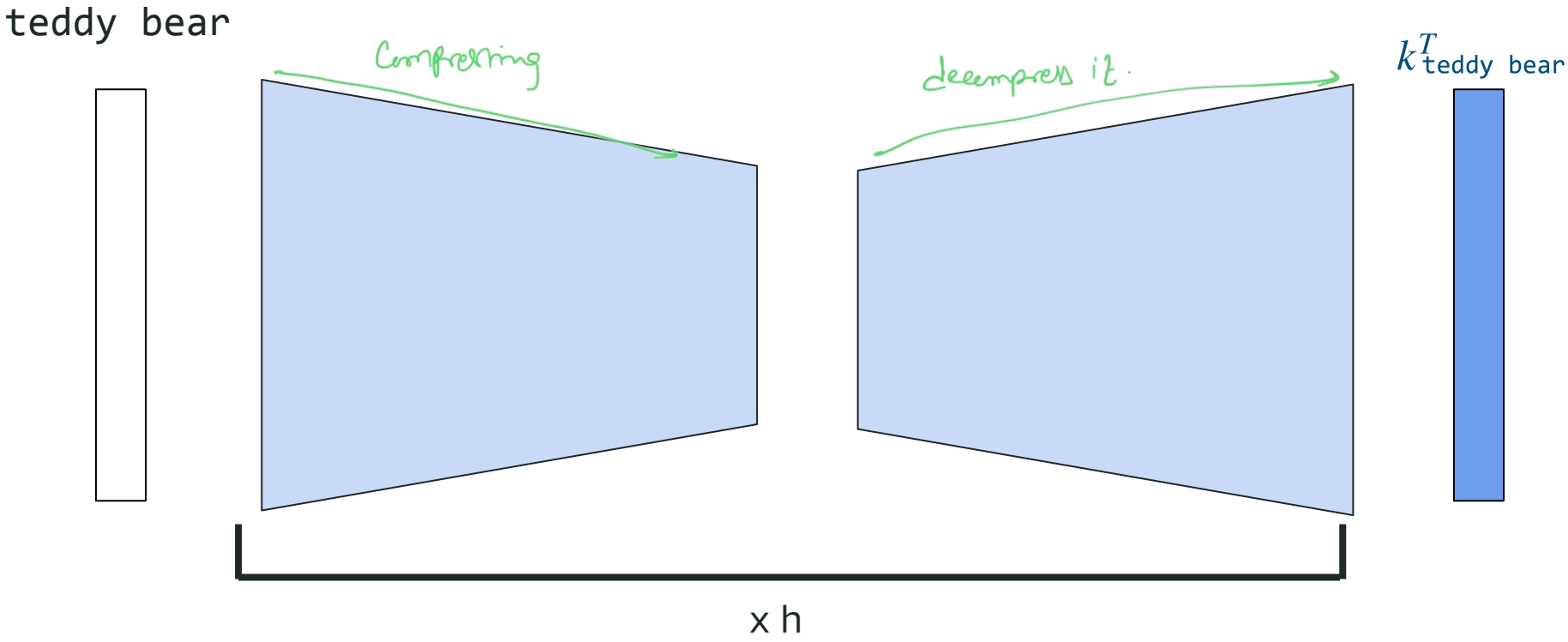
Reduce memory of KV cache with latent attention

BEFORE



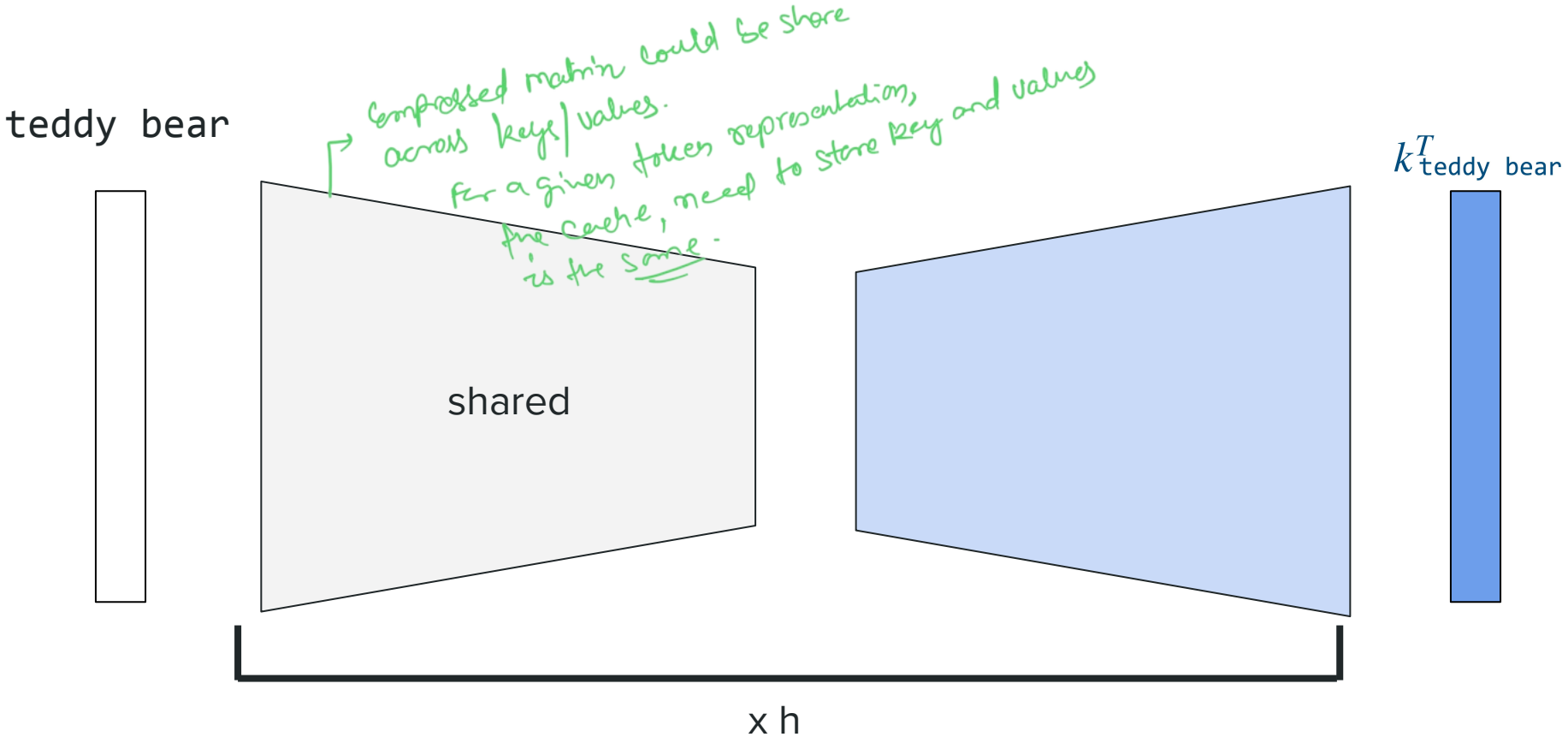
"DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model", by DeepSeek, 2023.

Reduce memory of KV cache with latent attention



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

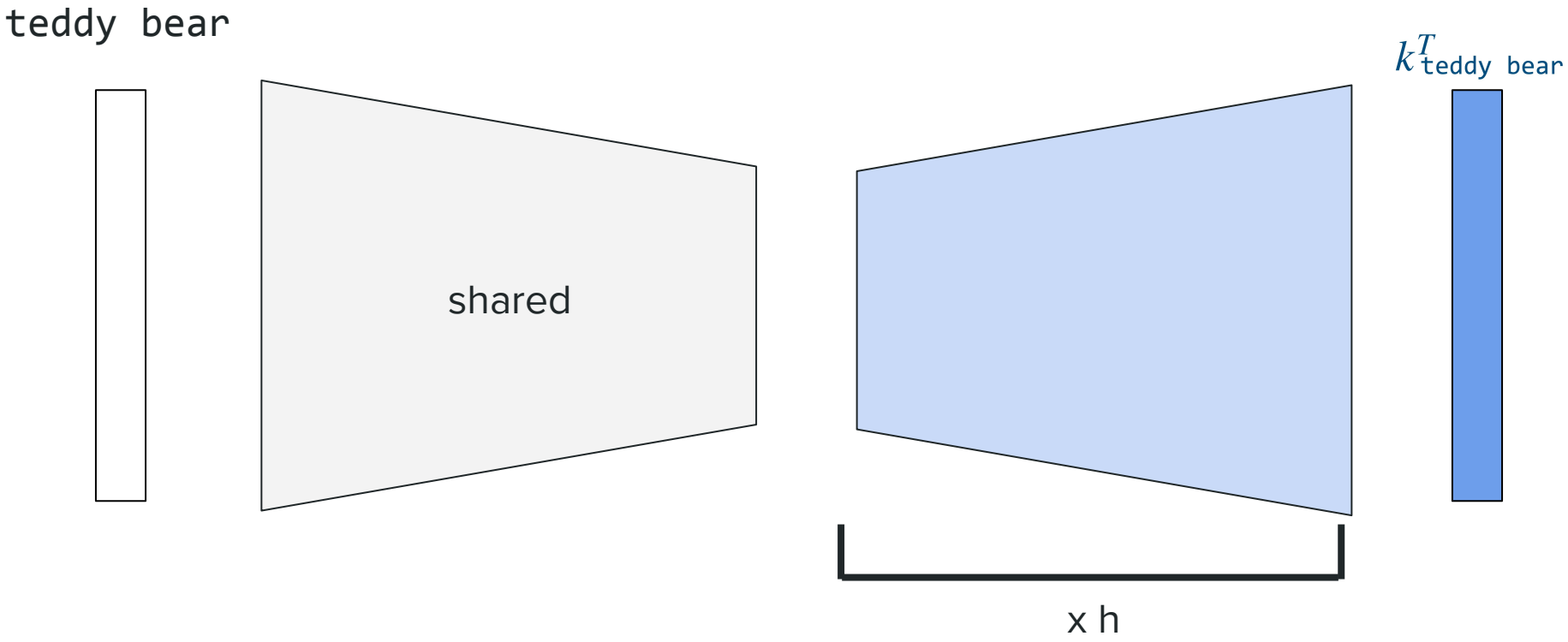
Reduce memory of KV cache with latent attention



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

Reduce memory of KV cache with latent attention

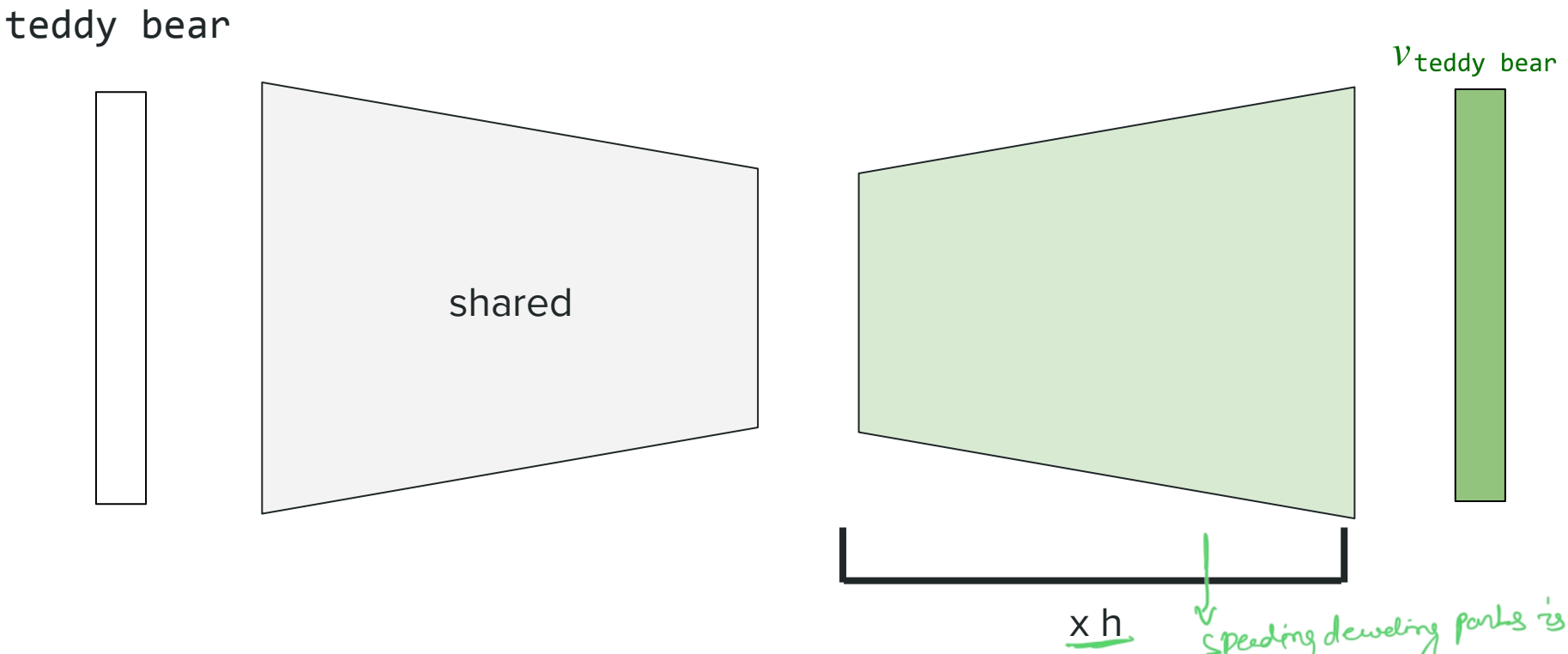
AFTER



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

Reduce memory of KV cache with latent attention

AFTER



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

Token generation tricks

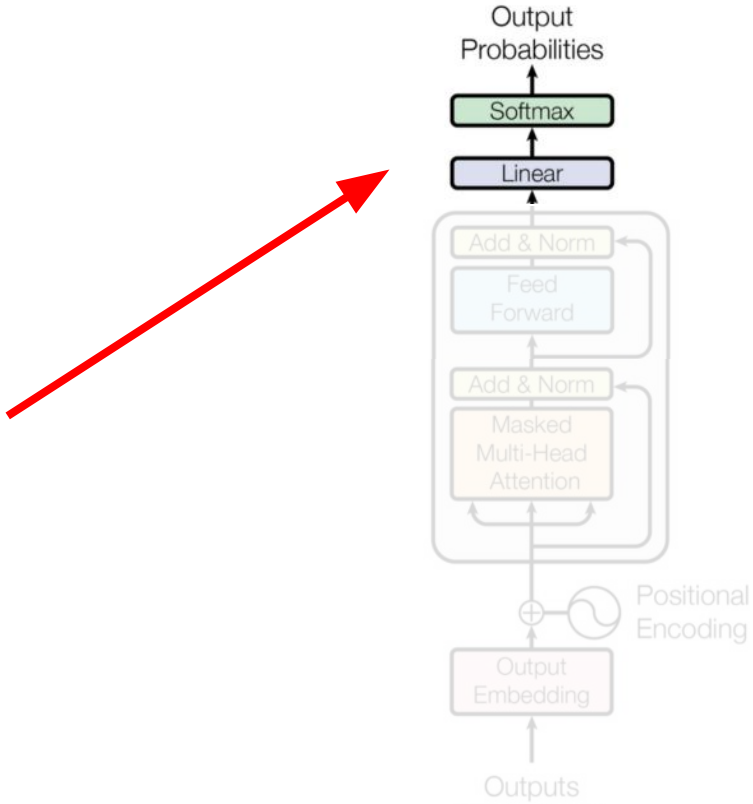


Figure adapted from "Attention is All You Need", Vaswani et al., 2017.

Speeding up decoding with speculative decoding

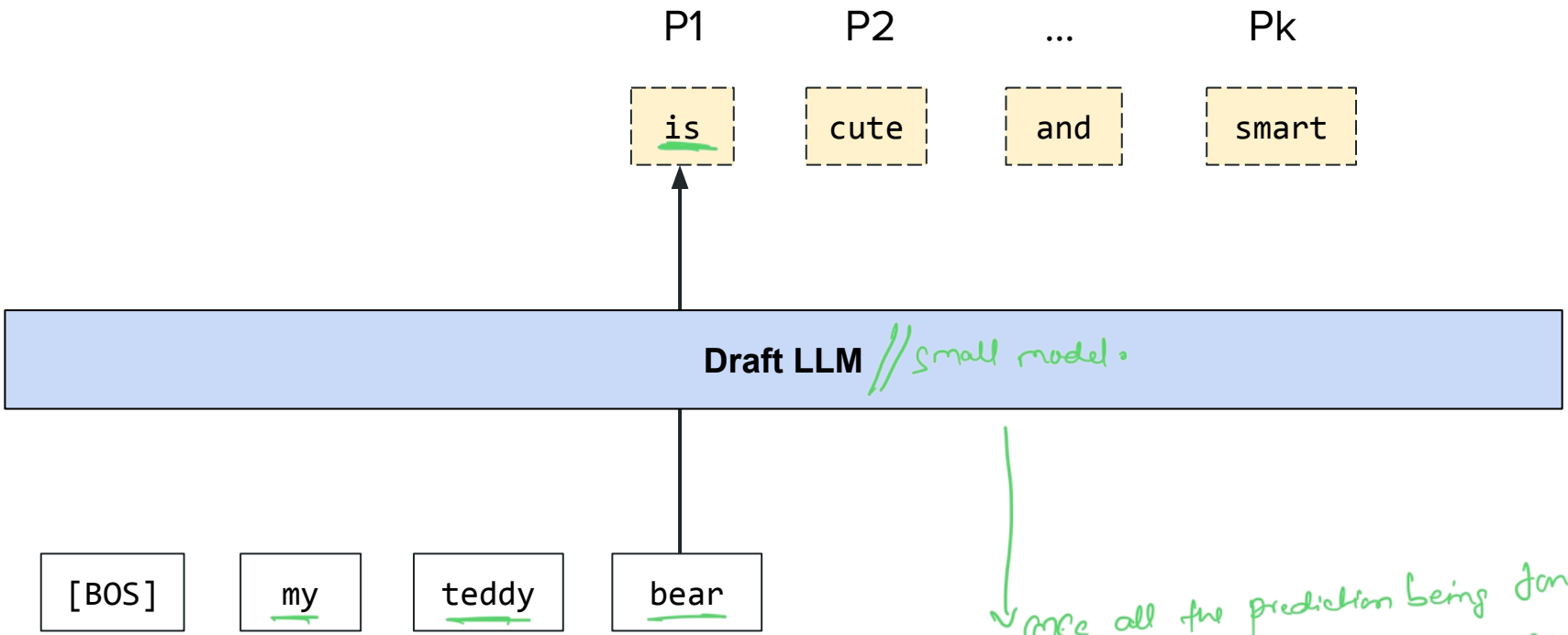
Idea. Use a draft (small) model to generate tokens validated by a target (big) model.

Speculative decoding technique which uses "smaller model" to help to generation to bigger model and uses some scheme that make distributions outputs match one of the target one.

(uses smaller model but with the validation by big model)

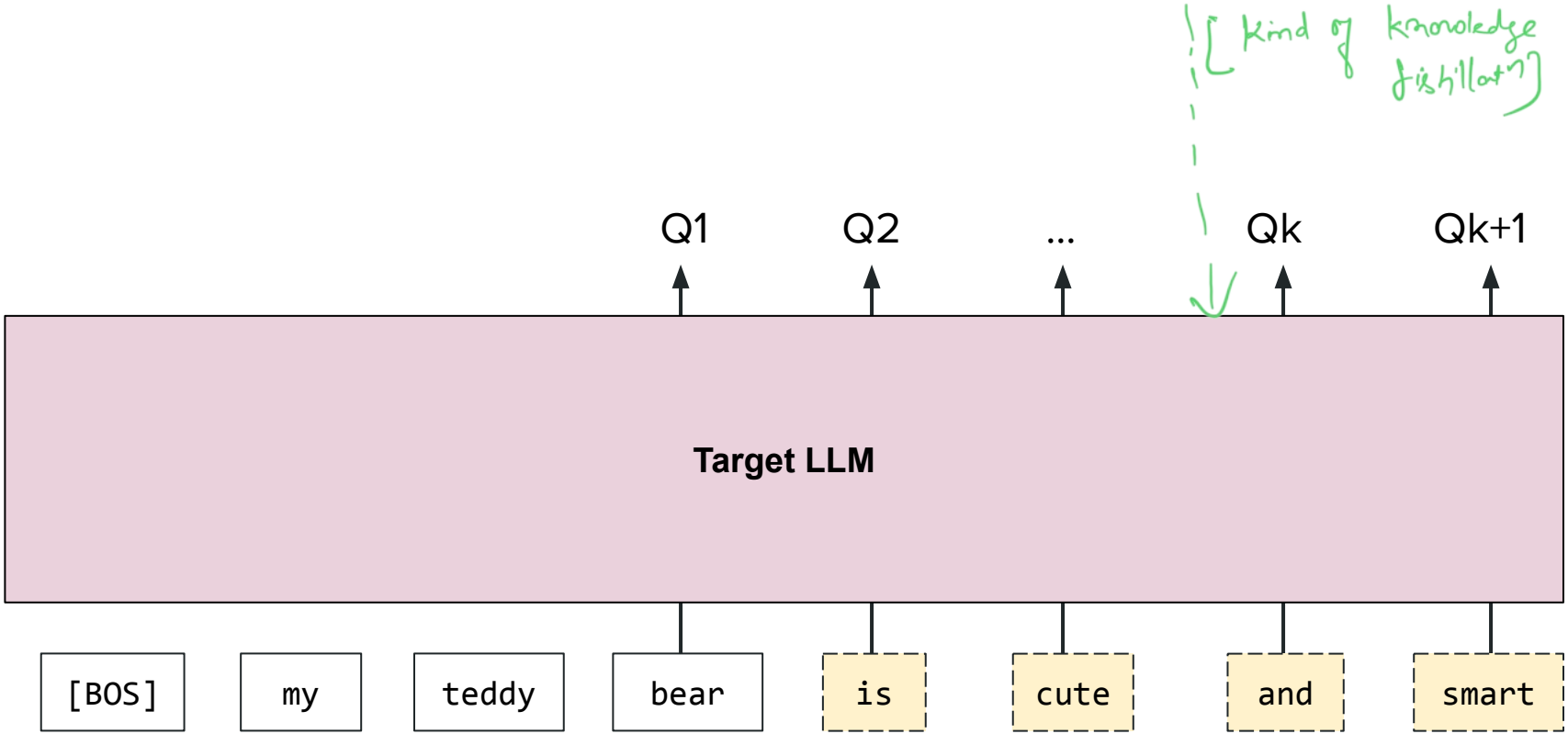
"*Accelerating Large Language Model Decoding with Speculative Sampling*", Chen et al., 2023.

Speeding up decoding with speculative decoding



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

Speeding up decoding with speculative decoding



Speeding up decoding with speculative decoding

Sampling algorithm. We distinguish the following cases:

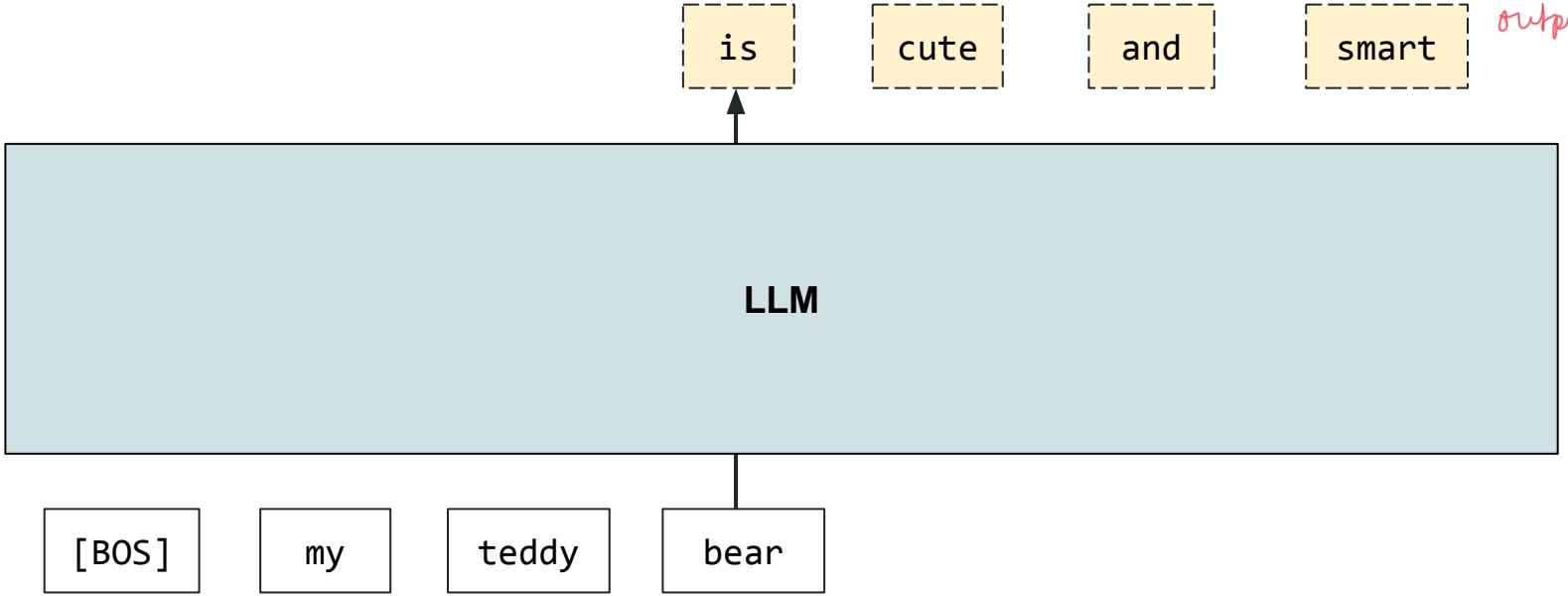
- If Qi(token) >= Pi(token) → token ✅
  (longer model prediction probability of token → smaller model prediction probability then accepted the token (predicted))
- Otherwise
  - with probability Qi(token) / Pi(token) → token ✅
  - with probability 1 - Qi(token) / Pi(token) → token ❌

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

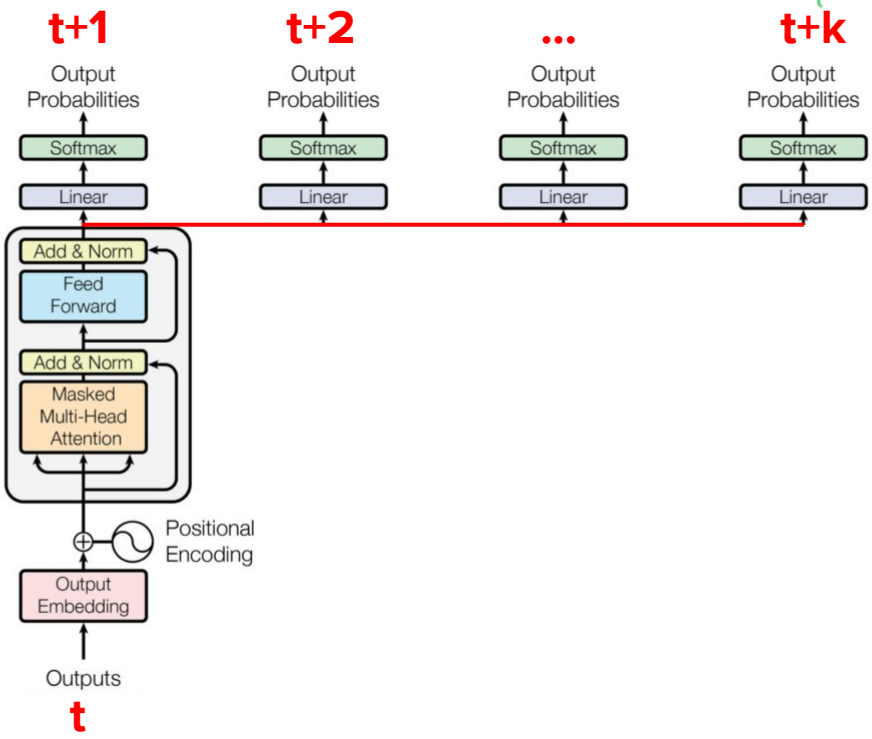
calculate multiple token and makes with longer model output.



"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!



↳ attends multiple head.
(multiple head prediction)
and learn which head works well in inference)

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** (storing in fixed block size in memory)
- ✓ Reformulate the math ← **Speculative decoding** (small & large model based learning)

Approximations:

- ✓ Architectural changes ← **Grouped query attention** (change architecture for grouping)
- ✓ Embeddings representations ← **Latent attention**
- ✓ Token prediction ← **Multi-token prediction** [multiple head into architecture while training]

Thank you for your attention!